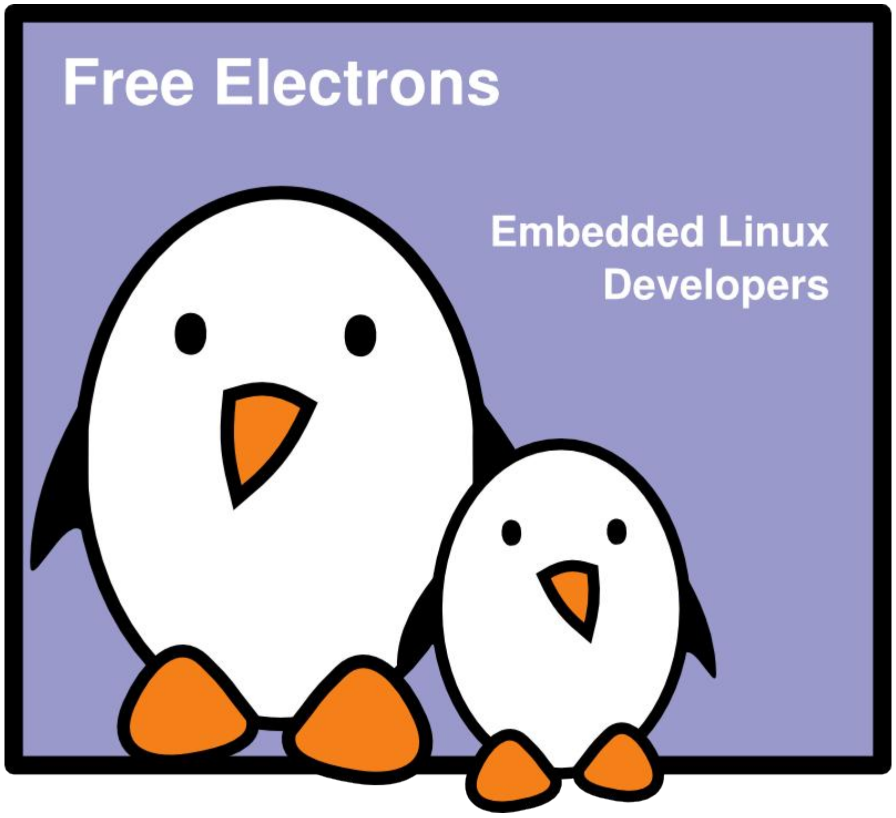
Linux kernel introduction

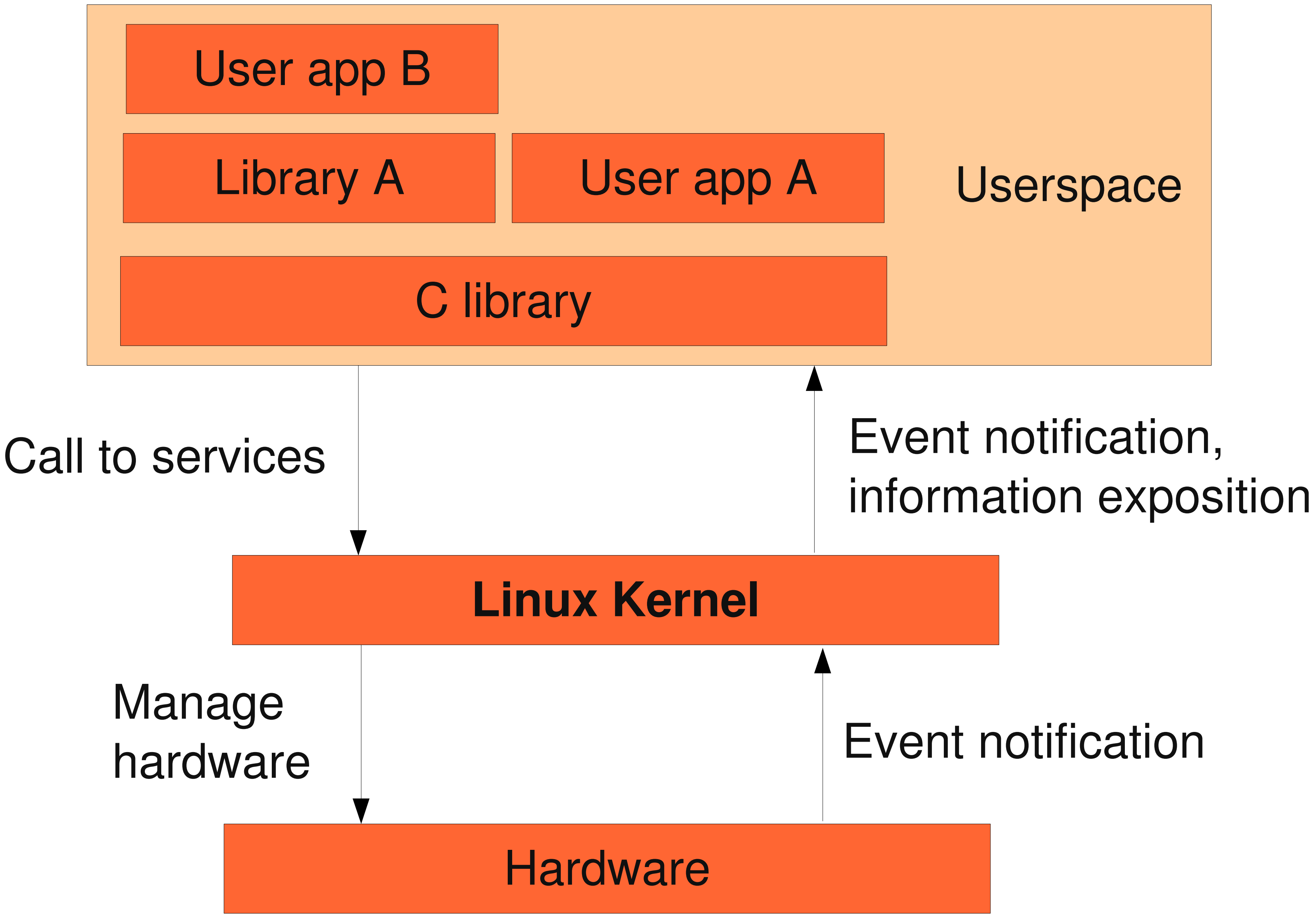
Michael Opdenacker
Thomas Petazzoni
Free Electrons



Kernel overview

Linux features

Linux kernel in the system



History

- The Linux kernel is one component of a system, which also requires libraries and applications to provide features to end users.
- The Linux kernel was created as a hobby in 1991
by a Finnish student, Linus Torvalds.
  - Linux quickly started to be used as the kernel for free software operating systems
- Linus Torvalds has been able to create a large and dynamic developer and user community around Linux.
- Nowadays, hundreds of people contribute to each kernel release, individuals or companies big and small.

Linux license

- The whole Linux sources are Free Software released under the GNU General Public License version 2 (GPL v2).
- For the Linux kernel, this basically implies that:
  - When you receive or buy a device with Linux on it, you should receive the Linux sources, with the right to study, modify and redistribute them.
  - When you produce Linux based devices, you must release the sources to the recipient, with the same rights, with no restriction..

Linux kernel key features

- Portability and hardware support
  Runs on most architectures.
- Scalability
  Can run on super computers as well as on tiny devices (4 MB of RAM is enough).
- Compliance to standards and interoperability.
- Exhaustive networking support.
- Security
  It can't hide its flaws. Its code is reviewed by many experts.
- Stability and reliability.
- Modularity
  Can include only what a system needs even at run time.
- Easy to program
  You can learn from existing code. Many useful resources on the net.

Supported hardware architectures

2.6.31 status

- See the `arch/` directory in the kernel sources
- Minimum: 32 bit processors, with or without MMU, and gcc support
- 32 bit architectures (`arch/` subdirectories)
  `arm`, `avr32`, `blackfin`, `cris`, `frv`, `h8300`, `m32r`, `m68k`, `m68knommu`, `microblaze`, `mips`, `mn10300`, `parisc`, `s390`, `sparc`, `um`, `xtensa`
- 64 bit architectures:
  `alpha`, `ia64`, `sparc64`
- `32/64 bit architectures`
  `powerpc`, `x86`, `sh`
- Find details in kernel sources: `arch/<arch>/Kconfig`, `arch/<arch>/README`, or `Documentation/<arch>/`

System calls

- The main interface between the kernel and userspace is the set of system calls
- About ~300 system calls that provides the main kernel services
  - File and device operations, networking operations, inter-process communication, process management, memory mapping, timers, threads, synchronization primitives, etc.
- This interface is stable over time: only new system calls can be added by the kernel developers
- This system call interface is wrapped by the C library, and userspace applications usually never make a system call directly but rather use the corresponding C library function

Virtual filesystems

- Linux makes system and kernel information available in user-space through virtual filesystems (virtual files not existing on any real storage). No need to know kernel programming to access such information!
- Mounting `/proc`:

```
sudo mount -t proc none /proc
```

- Mounting `/sys`:

```
sudo mount -t sysfs none /sys
```

Filesystem type

Raw device
or filesystem image
In the case of virtual
filesystems, any string is fine

Mount point

/proc details

A few examples:

- `/proc/cpuinfo`: processor information
- `/proc/meminfo`: memory status
- `/proc/version`: kernel version and build information
- `/proc/cmdline`: kernel command line
- `/proc/<pid>/environ`: calling environment
- `/proc/<pid>/cmdline`: process command line

... and many more! See by yourself!

Lots of details about the `/proc` interface are available in `Documentation/filesystems/proc.txt` (almost 2000 lines) in the kernel sources.

Embedded Linux usage

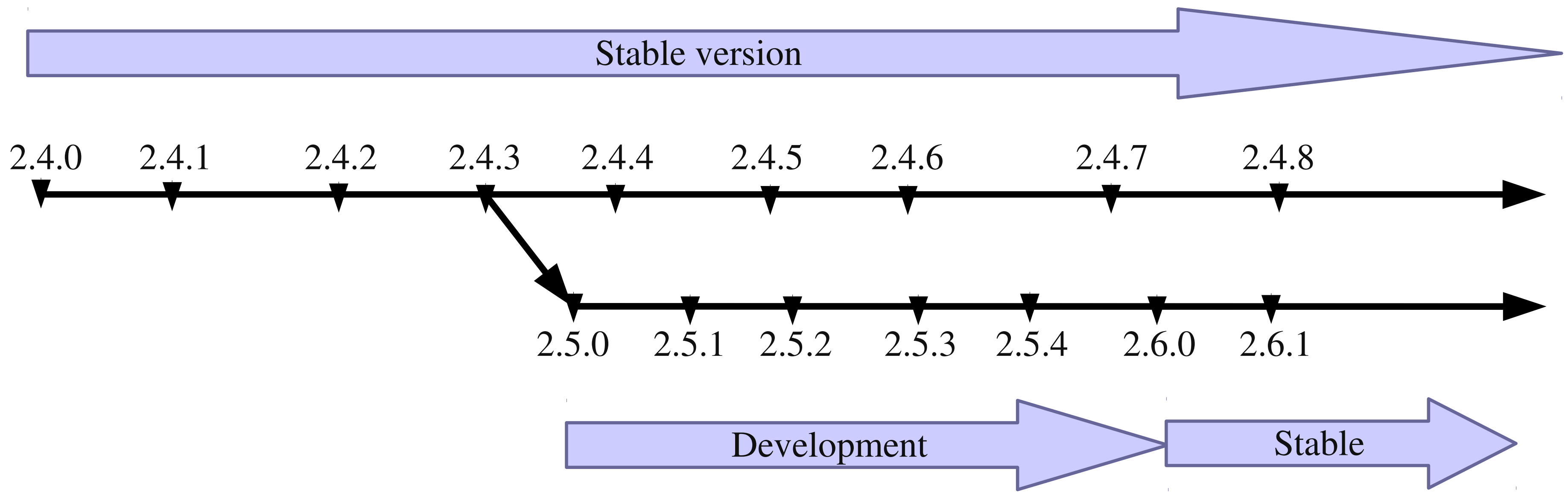
Kernel overview

Linux versioning scheme and development process

Until 2.6 (1)

- One stable major branch every 2 or 3 years
  - Identified by an even middle number
  - Examples: `1.0, 2.0, 2.2, 2.4`
- One development branch to integrate new functionalities and major changes
  - Identified by an odd middle number
  - Examples: `2.1, 2.3, 2.5`
  - After some time, a development version becomes the new base version for the stable branch
- Minor releases once in while: `2.2.23, 2.5.12`, etc.

Until 2.6 (2)



Note: in reality, many more minor versions exist inside the stable and development branches

Changes since Linux 2.6 (1)

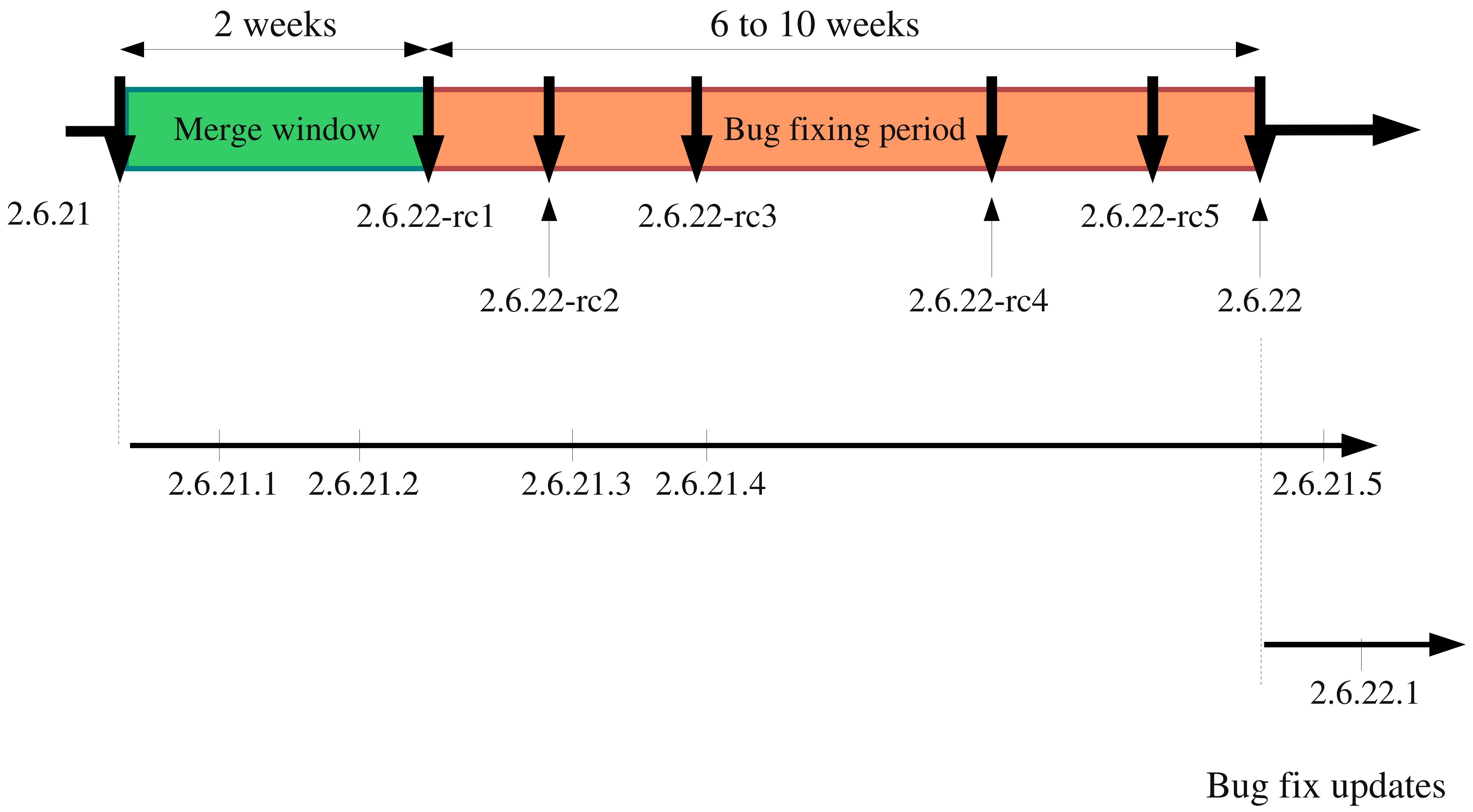
- Since `2.6.0`, kernel developers have been able to introduce lots of new features one by one on a steady pace, without having to make major changes in existing subsystems.
- Opening a new Linux `2.7` (or `2.9`) development branch will be required only when Linux `2.6` is no longer able to accommodate key features without undergoing traumatic changes.
- Thanks to this, more features are released to users at a faster pace.

Changes since Linux 2.6 (2)

Since 2.6.14, the kernel developers agreed on the following development model:

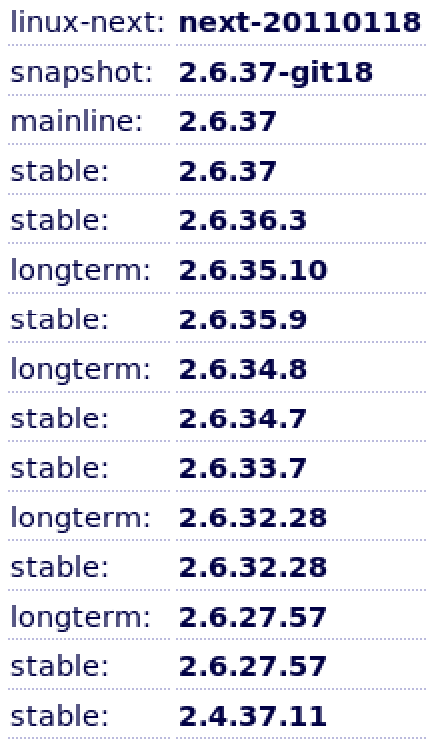
- After the release of a `2.6.x` version, a two-weeks merge window opens, during which major additions are merged.
- The merge window is closed by the release of test version `2.6.(x+1)-rc1`
- The bug fixing period opens, for 6 to 10 weeks.
- At regular intervals during the bug fixing period, `2.6.(x+1)-rcY` test versions are released.
- When considered sufficiently stable, kernel `2.6.(x+1)` is released, and the process starts again.

Merge and bug fixing windows



More stability for the 2.6 kernel tree

- Issue: bug and security fixes only released for most recent stable kernel versions.
- Some people need to have a recent kernel, but with long term support for security updates.
- You could get long term support from a commercial embedded Linux provider.
- You could reuse sources for the kernel used in Ubuntu Long Term Support releases (5 years of free security updates).
- You could choose one of the versions advertised as “long term” in the kernel.org front page. They will be maintained longer (2 or 3 years), unlike other versions.



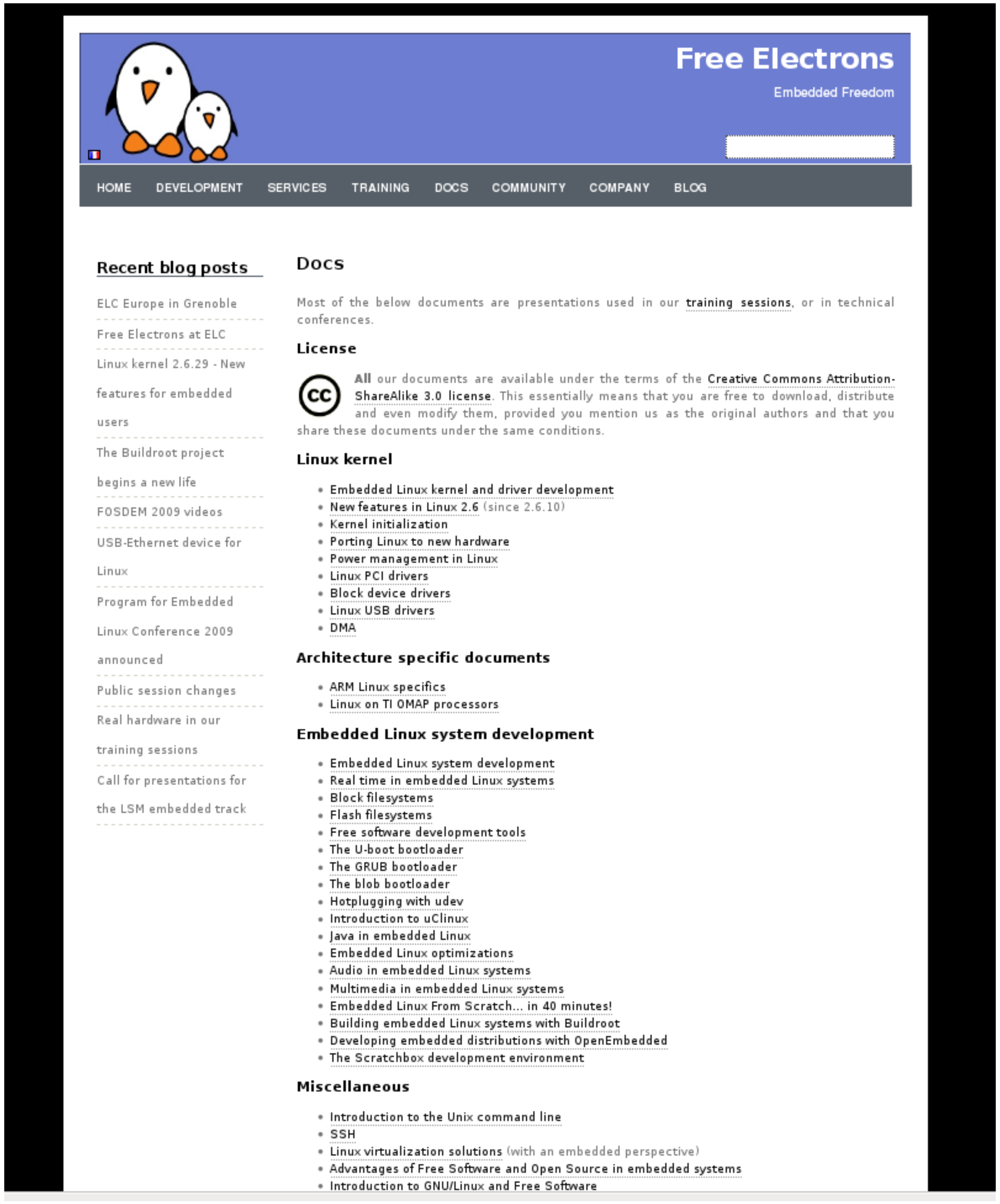
linux-next:	**next-20110118**
snapshot:	**2.6.37-git18**
mainline:	**2.6.37**
stable:	**2.6.37**
stable:	**2.6.36.3**
longterm:	**2.6.35.10**
stable:	**2.6.35.9**
longterm:	**2.6.34.8**
stable:	**2.6.34.7**
stable:	**2.6.33.7**
longterm:	**2.6.32.28**
stable:	**2.6.32.28**
longterm:	**2.6.27.57**
stable:	**2.6.27.57**
stable:	**2.4.37.11**

What's new in each Linux release?

```
commit 3c92c2ba33cd7d666c5f83cc32aa590e794e91b0
Author: Andi Kleen <ak@suse.de>
Date:   Tue Oct 11 01:28:33 2005 +0200

    [PATCH] i386: Don't discard upper 32bits of HWCR on K8

    Need to use long long, not long when RMWing a MSR. I think
    it's harmless right now, but still should be better fixed
    if AMD adds any bits in the upper 32bit of HWCR.

    Bug was introduced with the TLB flush filter fix for i386

    Signed-off-by: Andi Kleen <ak@suse.de>
    Signed-off-by: Linus Torvalds <torvalds@osdl.org>
...
```

- The official list of changes for each Linux release is just a huge list of individual patches!
- Very difficult to find out the key changes and to get the global picture out of individual changes.
- Fortunately, a summary of key changes with enough details is available on http://wiki.kernelnewbies.org/LinuxChanges

Related documents


Free Electrons
Embedded Freedom
HOME DEVELOPMENT SERVICES TRAINING DOCS COMMUNITY COMPANY BLOG

Recent blog posts

- ELC Europe in Grenoble
- Free Electrons at ELC
- Linux kernel 2.6.29 - New features for embedded users
- The Buildroot project begins a new life
- FOSDEM 2009 videos
- USB-Ethernet device for Linux
- Program for Embedded Linux Conference 2009 announced
- Public session changes
- Real hardware in our training sessions
- Call for presentations for the LSM embedded track

Docs

Most of the below documents are presentations used in our training sessions, or in technical conferences.

License

All our documents are available under the terms of the Creative Commons Attribution-ShareAlike 3.0 license. This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

Linux kernel

- Embedded Linux kernel and driver development
- New features in Linux 2.6 (since 2.6.10)
- Kernel initialization
- Porting Linux to new hardware
- Power management in Linux
- Linux PCI drivers
- Block device drivers
- Linux USB drivers
- DMA

Architecture specific documents

- ARM Linux specifics
- Linux on TI OMAP processors

Embedded Linux system development

- Embedded Linux system development
- Real time in embedded Linux systems
- Block filesystems
- Flash filesystems
- Free software development tools
- The U-boot bootloader
- The GRUB bootloader
- The blob bootloader
- Hotplugging with udev
- Introduction to uClinux
- Java in embedded Linux
- Embedded Linux optimizations
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems
- Embedded Linux From Scratch... in 40 minutes!
- Building embedded Linux systems with Buildroot
- Developing embedded distributions with OpenEmbedded
- The Scratchbox development environment

Miscellaneous

- Introduction to the Unix command line
- SSH
- Linux virtualization solutions (with an embedded perspective)
- Advantages of Free Software and Open Source in embedded systems
- Introduction to GNU/Linux and Free Software

All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development

How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs