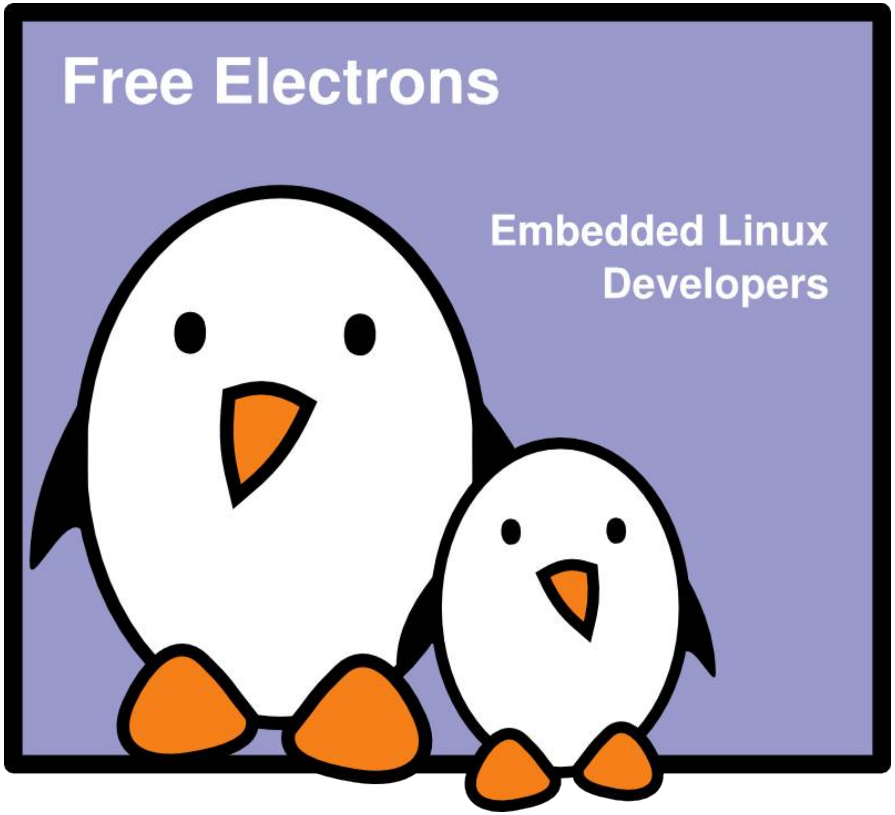
Introduction to embedded Linux

Michael Opdenacker
Thomas Petazzoni
Free Electrons



© Copyright 2009, Free Electrons.
Creative Commons BY-SA 3.0 license
Latest update: May 26, 2010,
Document sources, updates and translations:
http://free-electrons.com/docs/embedded-linux-intro
Corrections, suggestions, contributions and translations are welcome!

Embedded system?

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems control many of the common devices in use today.

Wikipedia, http://en.wikipedia.org/wiki/Embedded_system

Many different systems

A very generic definition

- Covers very different types of systems
- Fuzzy border with “standard” systems.

Consumer electronics (CE) products

- Home routers, DVD players, TV sets, digital cameras, GPS, camcorders, mobile phones, microwave ovens...

Industrial products

- Machine control, alarms, surveillance systems, automotive, rail, aircraft, satellite...

Many different products

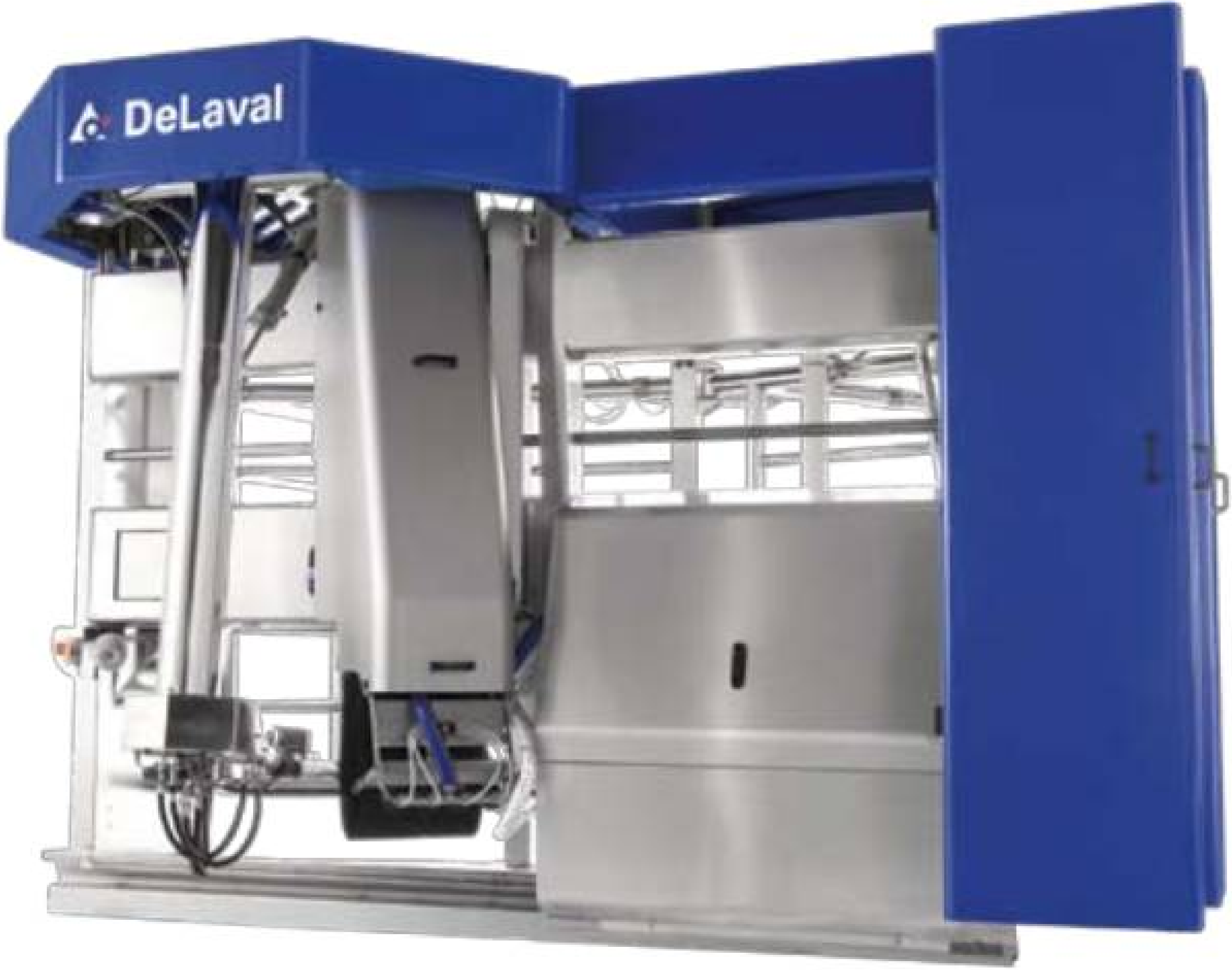
Embedded Linux

- The Free Software and Open Source world offers a broad range of tools to develop embedded systems.
- Advantages
  - Reuse of existing components for the base system. Allows to focus on the added value of the product.
  - High quality, proven components (Linux kernel, C libraries...)
  - Complete control on the choice of components. Modifications possible without external constraints.
  - Community support: tutorials, mailing lists...
  - Low cost, in particular no per-unit royalties.
  - Potentially less legal issues.
  - Easier access to software and tools.

Device examples

- GPS: TomTom and Garmin
- Home network routers: Linksys, Netgear
- PDA: Zaurus, Nokia N8x0
- TVs, camcorders, DVD players: Sony, Philips
- Mobile phones: Motorola, Android, OpenMoko
- Industrial machinery
- And many other products you don't even imagine...

Quiz

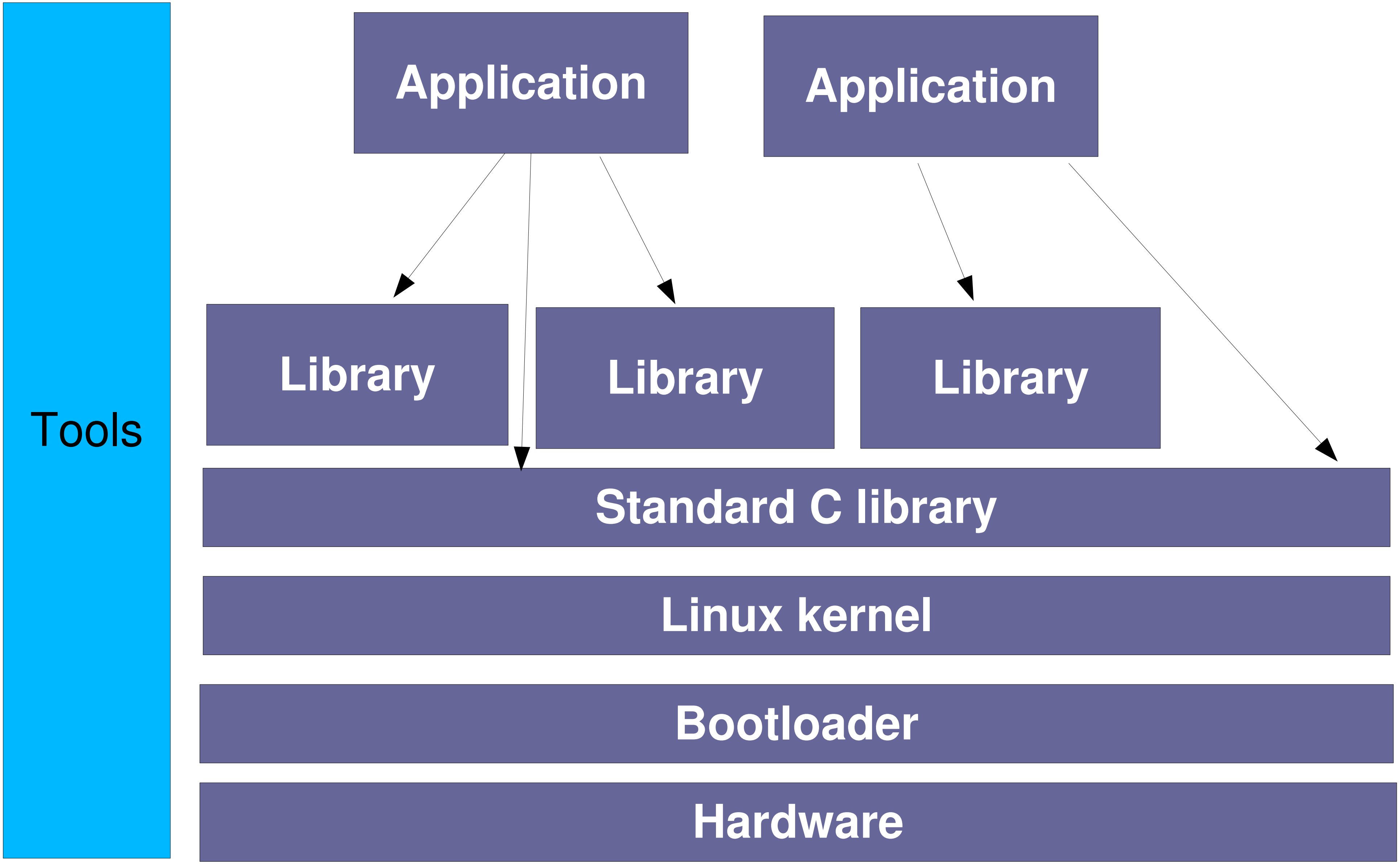


It works with Linux, but what is it for?

Answer

To milk cows!

Global architecture



Tools

Application

Application

Library

Library

Library

Standard C library

Linux kernel

Bootloader

Hardware

Embedded hardware

- Hardware for embedded systems is often different from hardware for classical systems.
  - Often a different CPU architecture:
    often ARM, MIPS or PowerPC. x86 is also used.
  - Storage on flash storage, NOR or NAND type, often with limited capacity (from a few MB to hundreds of MB)
  - Limited RAM capacity (from a few MB to several tens of MB)
  - Many interconnect bus not often found on the desktop: I2C, SPI, SSP, CAN, etc.
- Development boards starting
  from a few hundreds of EUR / USD
  - Often used as a basis for the final board design.

Examples

Picotux 100

- ARM7 55 MHz, Netsilicon NS7520
- 2 MB of flash
- 8 MB of RAM
- Ethernet
- 5 GPIOs
- Serial

OpenMoko

- ARM 920T 400 MHz, Samsung 2442B
- 2 MB of NOR flash
- 128 MB of RAM
- 256 MB of NAND flash
- 640x480 touchscreen , Bluetooth, GSM, serial, GPS, sound, 2 buttons, Wifi, USB, etc.

Minimum requirements

- A CPU supported by gcc and the Linux kernel
  - 32 bit CPU
  - MMU-less CPUs are also supported, through the uClinux project.
- A few MB of RAM, from 4 MB.
  8 MB are needed to do really do something.
- A few MB of storage, from 2 MB.
  4 MB to really do something.
- Linux isn't designed for small microcontrollers that just have a few tens or hundreds of KB of flash and RAM.
  - Base metal, no OS
  - Reduced systems, such as FreeRTOS

Software components

- Cross-compilation toolchain
  - Compiler that runs on the development machine, but generates code for the target
- Bootloader
  - Started by the hardware, responsible for basic initialization, loading and executing the kernel
- Linux Kernel
  - Contains the process and memory management, network stack, device drivers and provides services to userspace applications
- C library
  - The interface between the kernel and the userspace applications
- Libraries and applications
  - Third-party or in-house

Embedded Linux work

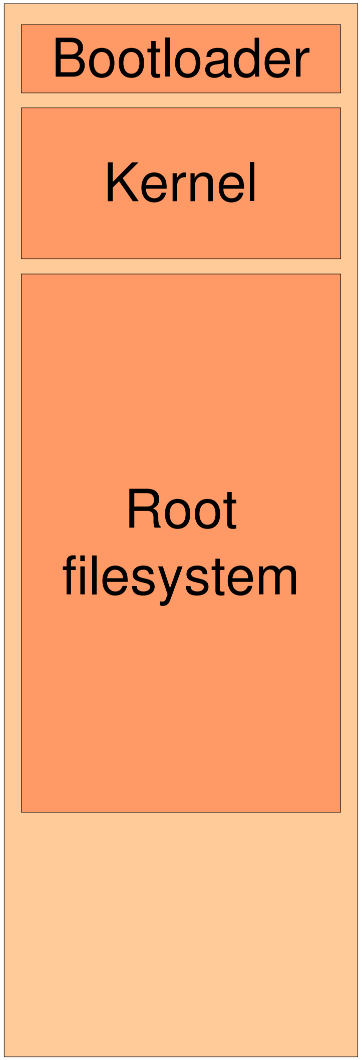
Several distinct tasks are needed when deploying embedded Linux in a product

- **Board Support Package development**
  - A BSP contains a bootloader and kernel with the suitable device drivers for the targeted hardware
  - Purpose of our « Kernel Development » training
- **System integration**
  - Integrate all the components, bootloader, kernel, third-party libraries and applications and in-house applications into a working system
  - Purpose of this training
- **Development of applications**
  - Normal Linux applications, but using specifically chosen libraries

Root filesystem

- In a Linux system, several filesystems are mounted and create a global hierarchy of files and directories
- A particular filesystem, the root filesystem, is mounted as /
- On embedded systems, this root filesystem contains all the libraries, applications and data of the system
- Therefore, building the root filesystem is one of the main tasks of integrating embedded Linux components into a device
- The kernel is usually kept separate

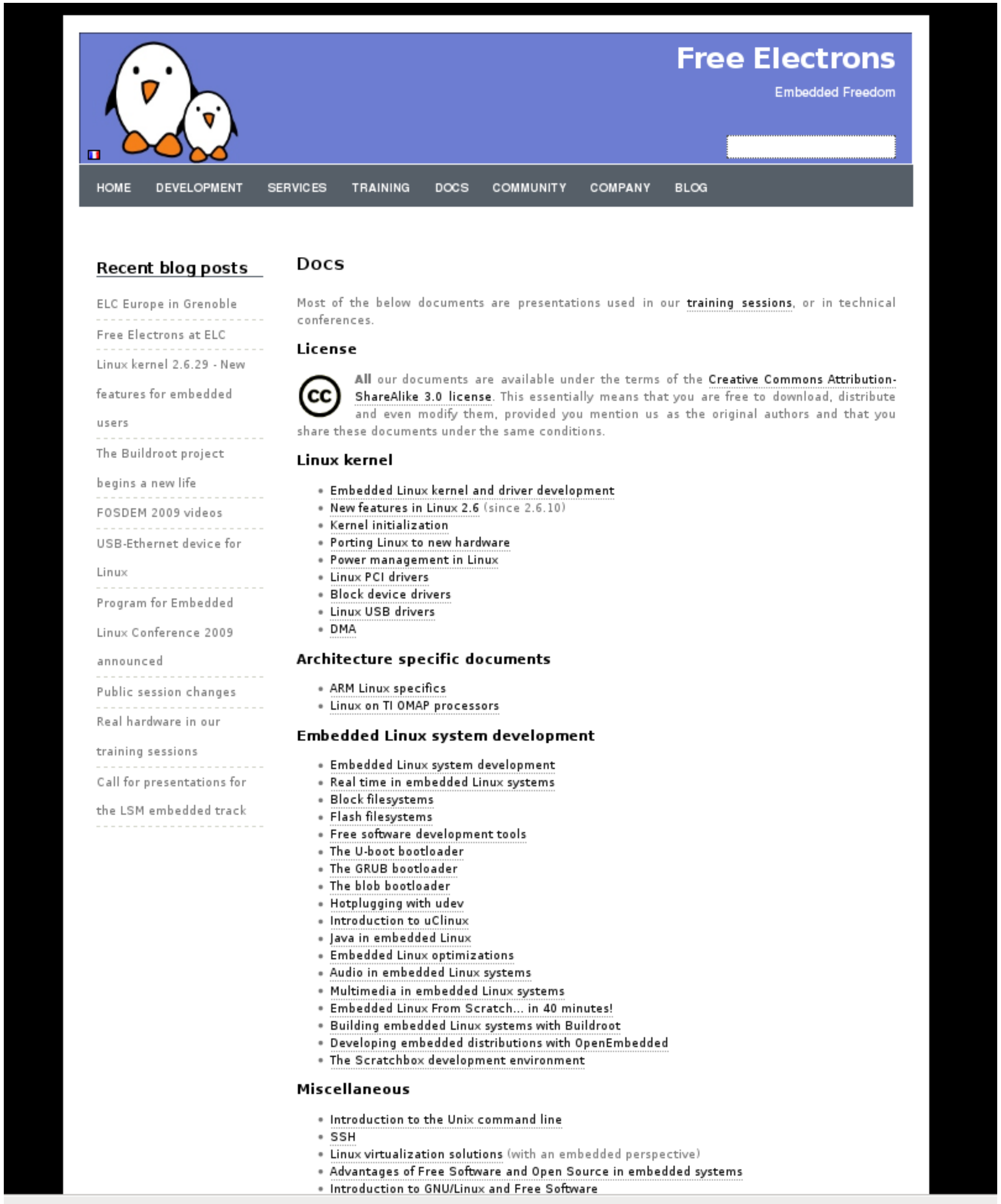
Flash contents



Development environment

- Two ways to switch to embedded Linux
  - Use solutions provided and supported by vendors like MontaVista, Wind River or TimeSys. These solutions come with their own development tools and environment
  - Use community solutions
- In Free Electrons trainings, we do not promote a particular vendor, and therefore use community solutions
  - However, knowing the concepts, switching to vendor solutions will be easy
- Doing embedded Linux development **requires** Linux on the desktop
  - The community solutions usually only exist on Linux
  - Understanding Linux on the desktop allows you to better understand Linux on the device

Related documents



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.

Free Electrons

Our services

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

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