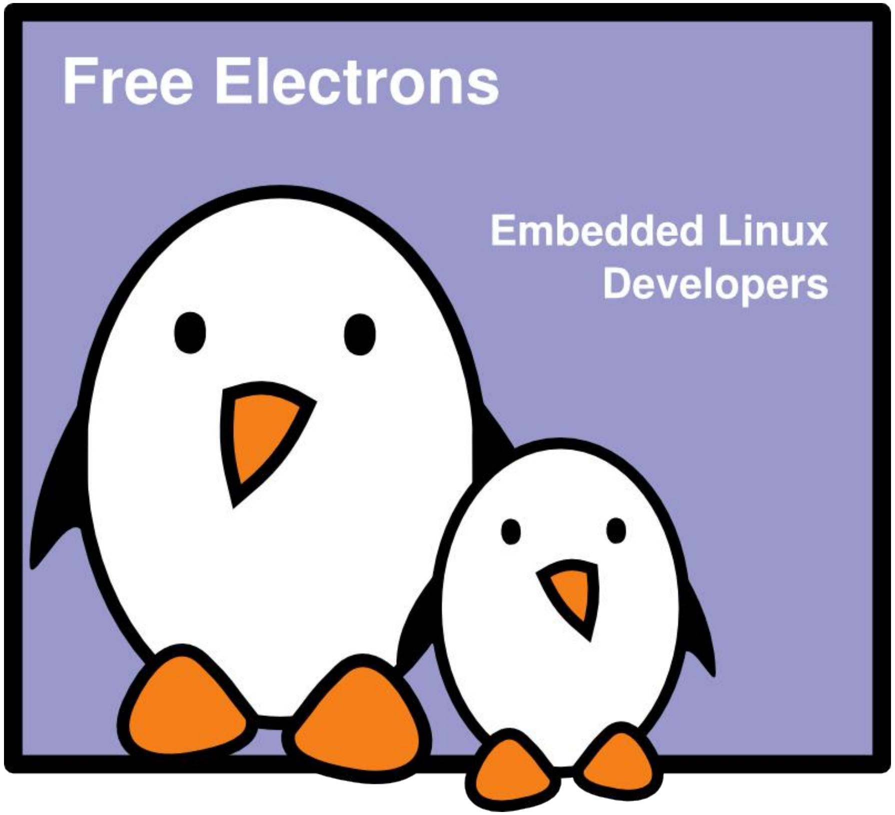
Bootloaders

Thomas Petazzoni
Michael Opdenacker
Free Electrons



Latest update: Jan 19, 2011,
Document sources, updates and translations:
http://free-electrons.com/docs/bootloaders
Corrections, suggestions, contributions and translations are welcome!

Bootloaders

- The bootloader is a piece of code responsible for
  - Basic hardware initialization
  - Loading of an application binary, usually an operating system kernel, from flash storage, from the network, or from another type of non-volatile storage.
  - Possibly uncompression of the application binary
  - Execution of the application
- Besides these basic functions, most bootloaders provide a shell with various commands implementing different operations.
  - Loading of data from storage or network, memory inspection, hardware diagnostics and testing, etc.

Bootloaders on x86 (1)

- The x86 processors are typically bundled on a board with a non-volatile memory containing a program, the BIOS.
- This program gets executed by the CPU after reset, and is responsible for basic hardware initialization and loading of a small piece of code from non-volatile storage.
  - This piece of code is usually the first 512 bytes of an hard disk
- This piece of code is usually a 1st stage bootloader, which will load the full bootloader itself.
- The bootloader can then offer all its features. It typically understands filesystem formats so that the kernel file can be loaded directly from a normal filesystem.

Bootloaders on x86 (2)

- GRUB, Grand Unified Bootloader, the most powerful one.
  http://www.gnu.org/software/grub/
  - Can read many filesystem formats to load the kernel image and the configuration, provides a powerful shell with various commands, can load kernel images over the network, etc.
  - See our dedicated presentation for details:
    http://free-electrons.com/docs/grub/
- LILO, the original Linux Loader
  http://freshmeat.net/projects/lilo/
- Syslinux, for network and removable media booting
  http://syslinux.zytor.com

Bootloaders on embedded architectures (1)

- On embedded architectures, the low-level booting process is very CPU and board dependent
  - Some boards have a NOR flash from which the CPU starts executing instructions after reset. In that case, the bootloader must directly be flashed inside the NOR at the proper location
  - Some CPUs have an integrated bootcode in ROM that automatically loads a small portion of a DataFlash or NAND flash, usually to a static RAM. In that case, a minimal first stage bootloader is required, that will load the main bootloader (BootROM on AT91SAM CPUs, Steppingstone on S3C24xx CPUs, etc.).
- The bootloader on embedded architectures starts right after CPU reset, so it must initialize all the devices, including the memory controller in order to access the DRAM.
- As the boot process is very CPU and board dependent,
  refer to the vendor documentation.

Bootloaders on embedded architectures (2)

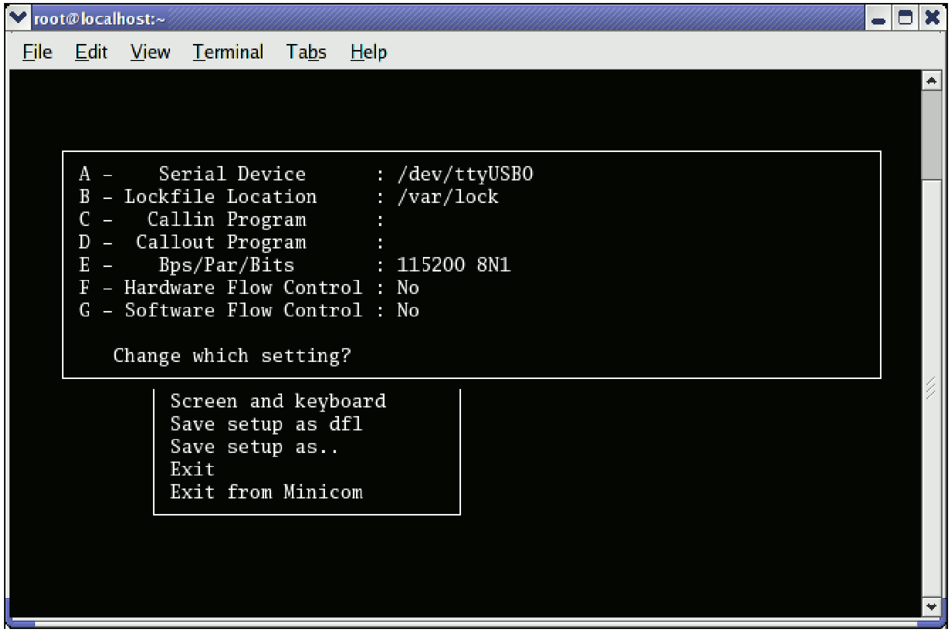
- We will focus on the generic part, the main bootloader, offering the most important features.
- There are several open-source generic bootloaders.
  Here are the most popular ones:
  - U-Boot, the universal bootloader by Denx
    The most used on ARM, also used on PPC, MIPS, x86, m68k, NIOS, etc.
    The de-facto standard nowadays. We will study it in detail.
    http://www.denx.de/wiki/U-Boot
  - Barebox, a new architecture-neutral bootloader, written as a successor of U-Boot. Better design, better code, active development, but doesn't yet have as much hardware support as U-Boot.
    http://www.barebox.org
- There are also a lot of other open-source or proprietary bootloaders, often architecture-specific
  - RedBoot, Yaboot, PMON, etc.

Accessing a serial console

Minicom (1)

- Definition: serial communication program
- Available in all GNU / Linux distributions
- Capabilities (all through a serial link):
  - Serial console to a remote Unix system
  - File transfer
  - Modem control and dial-up
  - Serial port configuration

Minicom (2)



```
A -    Serial Device      : /dev/ttyUSB0
B - Lockfile Location     : /var/lock
C -   Callin Program      :
D -  Callout Program      :
E -    Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

   Change which setting?

        Screen and keyboard
        Save setup as dfl
        Save setup as..
        Exit
        Exit from Minicom
```

- Start by running `minicom -s` to setup Minicom
- A bit austere at first glance, but quickly gets friendly (see the labs for details)

Other terminal emulators

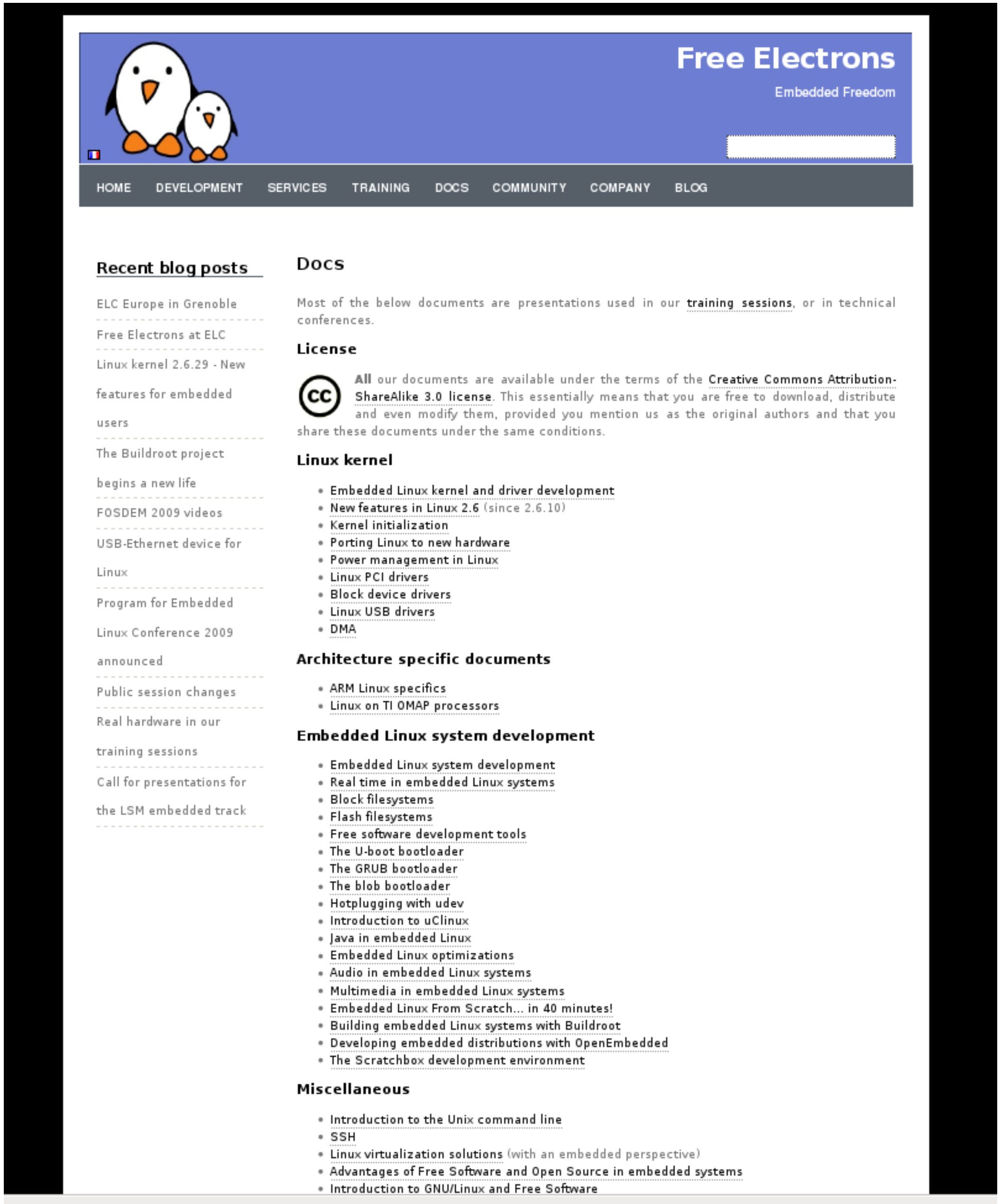
- GTKTerm: http://www.jls-info.com/julien/linux/
  Graphical. Less powerful than Minicom, but with a simpler and more attractive interface. Available in recent distros.
- CuteCom: http://cutecom.sourceforge.net/
  Another graphical and user-friendly terminal emulator. Available in recent distros.
- picocom: http://freshmeat.net/projects/picocom/
  Tiny terminal emulator (20K), can be used in embedded systems.
- GNU Screen: can also be used on a serial console:

  ```
  screen <device> <baudrate>
  ```

  Example:

  ```
  screen /dev/ttyS0 115200
  ```

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

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

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