Java in Embedded Linux Systems

Thomas Petazzoni / Michael Opdenacker
Free Electrons
http://free-electrons.com/

Created with OpenOffice.org 2.x

Rights to copy

creative commons
COMMONS DEED

Attribution – ShareAlike 2.5

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/2.5/legalcode

Document sources, updates and translations:
http://free-electrons.com/articles/java

Corrections, suggestions, contributions and translations are welcome!

Best viewed with...

This document is best viewed with a recent PDF reader or with OpenOffice.org itself!

- Take advantage of internal or external hyperlinks. So, don't hesitate to click on them!
- Find pages quickly thanks to automatic search
- Use thumbnails to navigate in the document in a quick way

If you're reading a paper or HTML copy, you should get your copy in PDF or OpenOffice.org format on http://free-electrons.com/articles/java!

Contents

Java history (1)

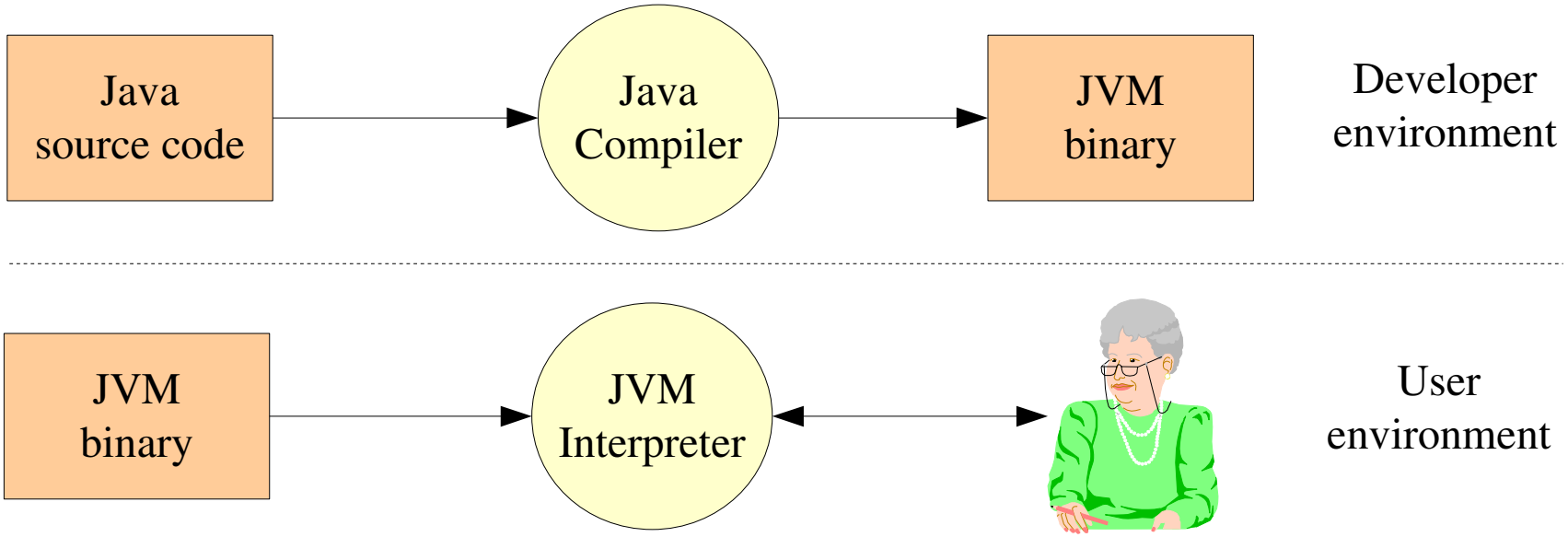
- 1991: Sun started working on a project to anticipate the next wave of computing: the convergence of digitally controlled consumer devices and computers.
- 1992: Implemented home entertainment control demos with a processor independent language: Oak.
- 1992-1994: focus moving from digital cable television network to the Internet as a way to publish and share content.
- 1994: Oak renamed as Java.
- 1994: HotJava web browser with dynamic, executable content.
- 1995: First public release of Java. Great success.

Java history (2)

- 1995: Netscape announced support for Java.
- 1996: Java 1.0 release.
- 1996: Microsoft takes a Java license.
- 1997-2001: Sun sues Microsoft for trying violating its license and trying to enforce its own Java standard.
- 2001: Microsoft loses the right to advertise its products as “Java Compatible”. Microsoft's licensed terminated but can continue to use its own implementations.
- 1998: Java 1.2. Renamed as “Java 2”
- 2006: Sun announces the release of Java under the GPL License. Java is Free!

Java key concepts

WORE
Write Once, Run Everywhere
Java applications are compiled in Java Virtual Machine (JVM) bytecode. Can be run on any platform with a JVM implementation.



Java features

See http://java.sun.com/docs/overviews/java/java-overview-1.html for details

- Simple
  Automatic garbage collection, small footprint
- Object oriented
  For well defined, reusable sw components
- Network-Savvy
  Support for network protocols (http, ftp...)
- Robust
  No pointers, no memory corruption, true arrays. Strict compile and run time checking.
- Secure
  Built for security in networking environments. Data protection (no pointers).
- Architecture neutral
  Code runs anywhere a VM is available
- Portable
  No architecture dependent data sizes (int: 32 bit, float: 32 bit). Portable VM source C code.
- Interpreted
  On the fly conversion to native machine code
- High performance
  VM code designed for simple machine code generation. Optimized bytecode generated by the compiler. VM interpreter optimizations.
- Multithreaded
  Multithread support in the language itself. But Real-time performance limited by OS RT perf.
- Dynamic
  Upgrading a class doesn't force to recompile the applications using it.

JNI: Java Native Interface

- Helps to understand how you can access the hardware with the JVM!
- The JNI lets code running on the JVM operate with applications and libraries written in other languages such as C, C++ or assembly.
- Needed to use system dependent routines handling hardware access.

Java's strengths in embedded systems

- Hides the specificities of the device and the embedded OS
- Provides a number of features beyond those offered by the OS
- Allows to build network interoperable embedded systems, whatever their hardware architecture.
- Simplifies product development. The software can easily be developed on workstations in parallel with the hardware.
- Makes it possible to reuse code
- Easy integration of Java Code and native code
- Smaller applications footprint: more compact bytecode and higher level class availability.

Sun Java platforms

Sun defined 3 platforms targeting different application environments

- Java Platform Micro Edition (Java ME), for environments with limited resources
- Java Platform Standard Edition (Java SE), for workstation environments. The version everyone uses on his workstation to run Java applications or applets
- Java Platform Enterprise Edition (Java EE), for large distributed enterprise or Internet environments

Java terminology

To run Java applications, a **JRE** is required

- Java Runtime Environment
- Composed of a **JVM**, Java Virtual Machine, which executes the bytecode
- And of an implementation of all standard Java classes

To develop Java applications, a **JDK** is needed

- Java Development Kit
- Composed of a **JRE** and development tools such as the Java compiler

Java ME

Java ME specifies two configurations, with different set of classes and APIs

- *Connected Limited Device Configuration*, the smallest possible Java configuration for embedded devices such as phones.
- *Connected Device Configuration*, a much more featureful configuration for larger embedded devices such as high-end PDAs, set-top boxes, etc.
- Each configuration exists in different profiles defining exactly what features are available.
- Sun provides a reference implementation, but third parties often have their own implementation.

Implementations from Sun

Implementations originating from Sun

Sun's implementation of Java SE

Sun implementation of Java used to be closed-source

- Official reason was the fear of incompatible forks that would break the uniformity of the Java platform.

In 2006, Sun opened the source code of its Java SE implementation under GPL version 2.

- The version available today for Linux, Windows and Solaris, both x86 and x86_64.
- Some pieces of code could not be released under the GPL, because Sun was not the full owner of the copyright. Sun started the OpenJDK project to replace these parts with open source versions.

Java SE for embedded

- With the increasingly fast CPUs and growing amount of memory found in embedded systems, Sun is now adverstising Java SE for embedded systems.
- Suitable if you have more than 32 MB of memory and storage, otherwise only Java ME is suitable.
- Available for ARM, MIPS and PowerPC. Headless versions available, reduced footprint of the JRE.
- However, these versions are not released under the GPL, and subject to royalties when distributed.
- http://java.sun.com/javase/embedded/

OpenJDK (1)

http://openjdk.java.net/

- Project started by Sun after open-sourcing Java in November 2006
- Goal: « to collaborate on an open-source implementation of the Java Platform, Standard Edition and related projects »
- Encumbered parts: audio engine, cryptographic code, font rasterizer, color management, etc.
- In June 2007, RedHat started the IcedTea project to create experimental patches to replace these parts and was shipping IcedTea in its distribution.
  See http://en.wikipedia.org/wiki/IcedTea

OpenJDK (2)

- November 2007, RedHat joins OpenJDK.
- December 2007, Java source code moved to Mercurial.
- February 2008, first read/write access to the repository to non-Sun engineers.
- February 2008, OpenJDK 6 created to quickly have a fully Open Source implementation with as few differences as possible.
- This version is now shipped in Ubuntu 8.04, Fedora 9 and Red Hat Entreprise Linux 5.
- http://www.sun.com/software/opensource/java/faq.jsp

OpenJDK license

- OpenJDK is released under the GNU General Public License version 2, with Classpath exception.
- The Classpath exception has been designed for the GNU Classpath project (covered later).
- It allows to create and distribute proprietary applications on top of free implementations of Java.
- http://www.sun.com/software/opensource/java/faq.jsp#g

Phone ME

https://phoneme.dev.java.net/

- Project created after open-sourcing Sun implementation of Java ME
- Two versions
  - PhoneME feature software (CLDC), for mobile phones
  - PhoneME advanced software (CDC), for higher-end devices
- Reference implementations available for x86, ARM and MIPS. Porting to other platforms and architectures said to be easy.
- Java still sells its commercial version of Java ME

Independent J2SE implementations

Components (1)

A usable implementation of Java is made of several important components

- A Java compiler, either to machine or byte code
- A virtual machine capable of executing the Java bytecode
- The class library implementing all standard Java classes and methods

Components (2)

- Developing a virtual machine is interesting, a lot of innovation and research is possible
  - Lots of different projects available
- Developing the class library is not really interesting, very long and probably boring
  - Only one usable project: GNU Classpath
- The virtual machine and class library are not independent: some integration between them is needed
  - http://wiki.debian.org/Java/DevJam/2008/Fosdem?action=AttachFile&do=get&target=fosdem2008-vm-interfaces.pdf

GNU Classpath

http://www.gnu.org/software/classpath/

- Java classes implementation from the GNU project
- License: GNU GPL, which an exception for linking with this library. So, no license issue with proprietary Java programs.
- Exhaustive Java API implementation, able to run applications such as Eclipse.
- Graphical applications rely on GTK or Qt.
- Active project.

GNU compiler for Java

http://gcc.gnu.org/java/

`gcj` can compile:

- Java source code directly to native machine code
- Java source code to Java bytecode (class files)
- Java bytecode to native machine code.

`gcj` can also be configured as a cross-compiler!

`gdb` support

`libgcj`: runtime for `gcj` compiled applications

- core class libraries
- garbage collector
- bytecode interpreter

`libgcj` is now merged with GNU Classpath, so the support is very wide.

`gij`: bytecode interpreter

Using gcj: sample program

```
class Hello {
        public static void main(String args[]) {
                System.out.println("Hello World");
        }
};
```


Using gcj: native code

```
$ gcj Hello.java --main=Hello -o hello
$ file hello
hello: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.6.8, dynamically linked (uses shared libs), not stripped
$ ./hello
Hello World

$ ldd hello
[...]
 libgcj.so.90 => /usr/lib/libgcj.so.90 (0xb5f5c000)
[...]
```

On a Debian system, `libgcj.so` takes ~32 MB.

Using gcj: bytecode

Compiling

```
$ gcj Hello.java -C
$ ls Hello.*
Hello.class Hello.java
```

Executing with gij and Sun's JVM

```
$ gij Hello
Hello World
$ java Hello
Hello World
```


Other compilers

- Jikes
  - Initially created by IBM, now a community project
  - http://jikes.sourceforge.net
- Eclipse compiler
  - Incremental compiler used in Eclipse
  - Now used for parsing by `gcj`, since gcc 4.3 (March 2008), in order to support the latest additions of Java 1.5 such as generics
  - http://www.eclipse.org/jdt/core/

Kaffe

http://www.kaffe.org/

- Clean-room JVM implementation plus associated class libraries needed by a Java runtime environment, taken from the GNU Classpath project
- Not an officially licensed JVM implementation.
- Not fully compatible with Java specs and lacks some key features (security, etc.)
- Supported in several platforms: x86, arm, mips, m68k, ppc...
- Great for VM education or research, or in a Free Software Java distribution.
- Until version 1.1.9, Kaffe has additional graphical backends over Classpath but they will be dropped in future versions
- License: GNU General Public License

Jikes RVM

http://oss.software.ibm.com/developerworks/opensource/jikesrvm/

- RVM: Research Virtual Machine
- Open Source (Common Public License) implementation from IBM
- Implemented on Java, running on itself without requiring a second virtual machine
- Supported Linux Platforms:
  x86 and PowerPC
- Relies on GNU Classpath.

SableVM

http://sablevm.org/

"A robust, clean, easy to maintain and extend, extremely portable, efficient, and specification-compliant Java virtual machine"

- Clean room JVM implementation
- Roughly supported on most architectures supported by Linux (ARM in particular)
- Compatible with JVM and Java Language specifications
- License: GNU Lesser General Public License
- Uses GNU Classpath (GPL with an exception for linking)

JamVM

http://jamvm.sourceforge.net/

A compact Virtual Machine

- Very compact: ~200 KB on PowerPC, 180 KB on x86
- Full-featured, conforming to the specifications, contrary to other lightweight JVMs
- Actively developed
- Available for Linux on PowerPC, x86, ARM, x86_64 and MIPS
- Designed to work with GNU Classpath

Cacao

http://www.cacaojvm.org/

Another Virtual Machine using JIT

- Works on ARM, MIPS, PowerPC, x86, x86_64
- Can use either GNU Classpath, OpenJDK or PhoneME as a Java runtime library.
- Packages now available in Debian unstable.

IBM's Java on Linux

http://www-106.ibm.com/developerworks/java/jdk/linux140/

- Complete implementation from IBM:
  J2SE, J2EE, J2ME, including IBM's VM.
- Supported Linux platforms (only in 32 bit mode)
  IA32, AMD64, Itanium, PowerPC 64 bit, zSeries
- IBM's implementations
  Re-engineered Java Machine, IBM's Just In Time compiler...
- Java is at the core of IBM's strategy (WebSphere in particular)

Independent J2ME implementations

J2ME implementations

MIDPath

- Java library that provides a MIDP2 implementation, alternative to the reference implementation of PhoneME. Can be used together with the CLDC version of Cacao VM.
- MIDP2 is a specific profile of CLDC found on many phones
- http://midpath.thenesis.org/

Microemu

- J2ME emulator that runs on top of J2SE
- Purely implemented in Java
- http://www.microemu.org/

Java hardware acceleration in Linux

Some ARM cores directly support Java bytecode execution (Jazelle technology): http://www.arm.com/products/multimedia/java/jazelle.html

- The processor can directly execute ~120 Java bytecode instructions, and generate an exception for the rest so that the execution of the bytecode can be emulated.
- Unfortunately, ARM doesn't publish the specifications of this technology
  - Only licensees can get information about Jazelle
  - Sun's JVM are Jazelle aware, but through a binary-only component
- Some reverse engineering is in progress.

Conclusion

- Java has great competitive advantages for developing applications for operating systems.
- Free Software implementations for GNU / Linux are available now and satisfy most needs.

Useful reading

- Embedded Java+Linux reference:
  http://www.linuxdevices.com/articles/AT8918758707.html
- Java overview and history:
  http://en.wikipedia.org/wiki/Java_programming_language

Related documents

All the technical presentations and training materials created and used by Free Electrons, available under a free documentation license (more than 1500 pages!).

http://free-electrons.com/training

- Introduction to Unix and GNU/Linux
- Embedded Linux kernel and driver development
- Free Software tools for embedded Linux systems
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems

http://free-electrons.com/articles

- Advantages of Free Software in embedded systems
- Embedded Linux optimizations
- Embedded Linux from Scratch... in 40 min!
- Linux USB drivers
- Real-time in embedded Linux systems
- Introduction to uClinux
- Linux on TI OMAP processors
- Free Software development tools
- Java in embedded Linux systems
- Introduction to GNU/Linux and Free Software
- Linux and ecology
- What's new in Linux 2.6?
- How to port Linux on a new PDA

How to help

If you support this work, you can help ...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order training sessions performed by the author of these documents (see http://free-electrons.com/training)
- By speaking about it to your friends, colleagues and local Free Software community.
- By adding links to our on-line materials on your website, to increase their visibility in search engine results.

Free Electrons services

Embedded Linux Training

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

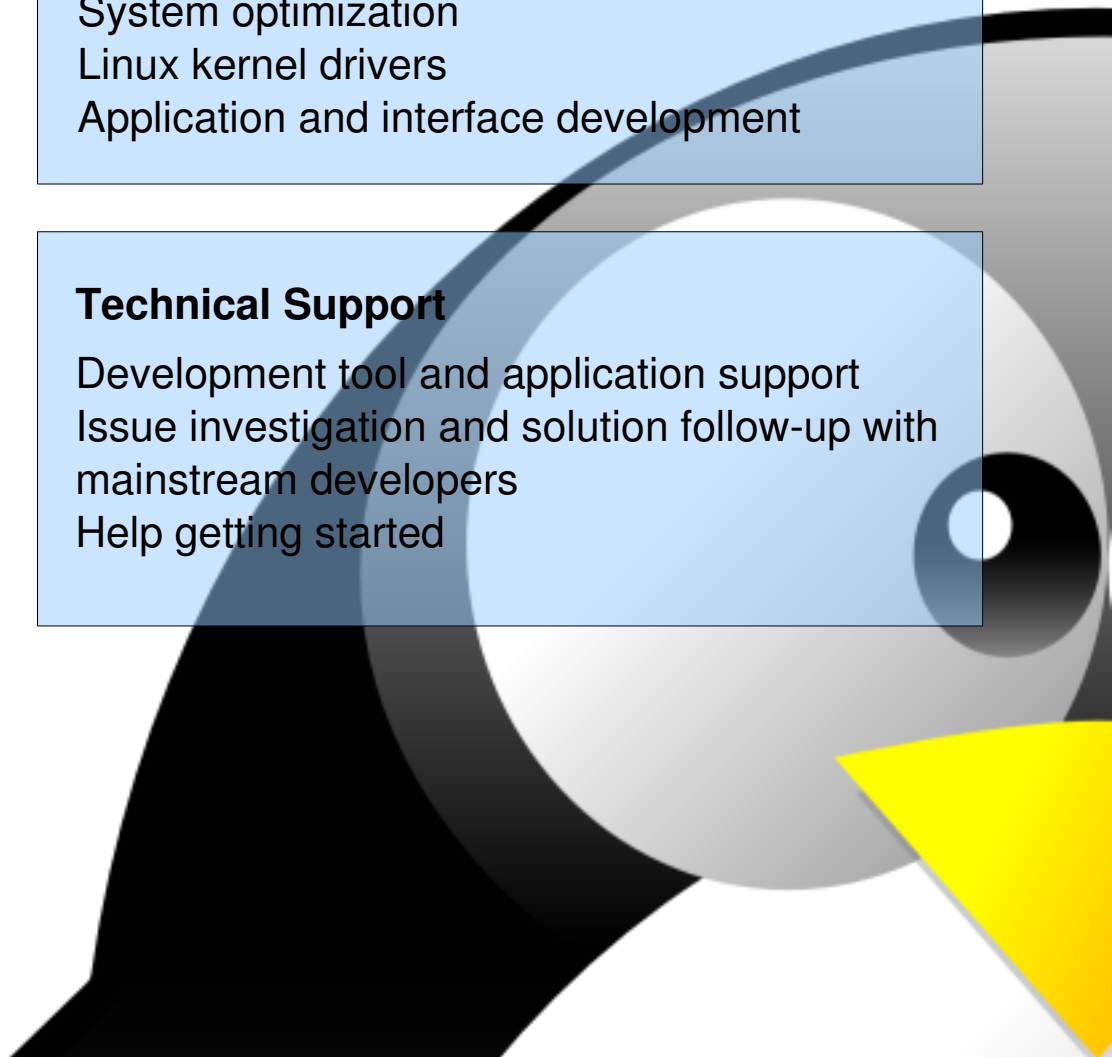
System integration
Embedded Linux demos and prototypes
System optimization
Linux kernel drivers
Application and interface development

Consulting

Help in decision making
System architecture
Identification of suitable technologies
Managing licensing requirements
System design and performance review

Technical Support

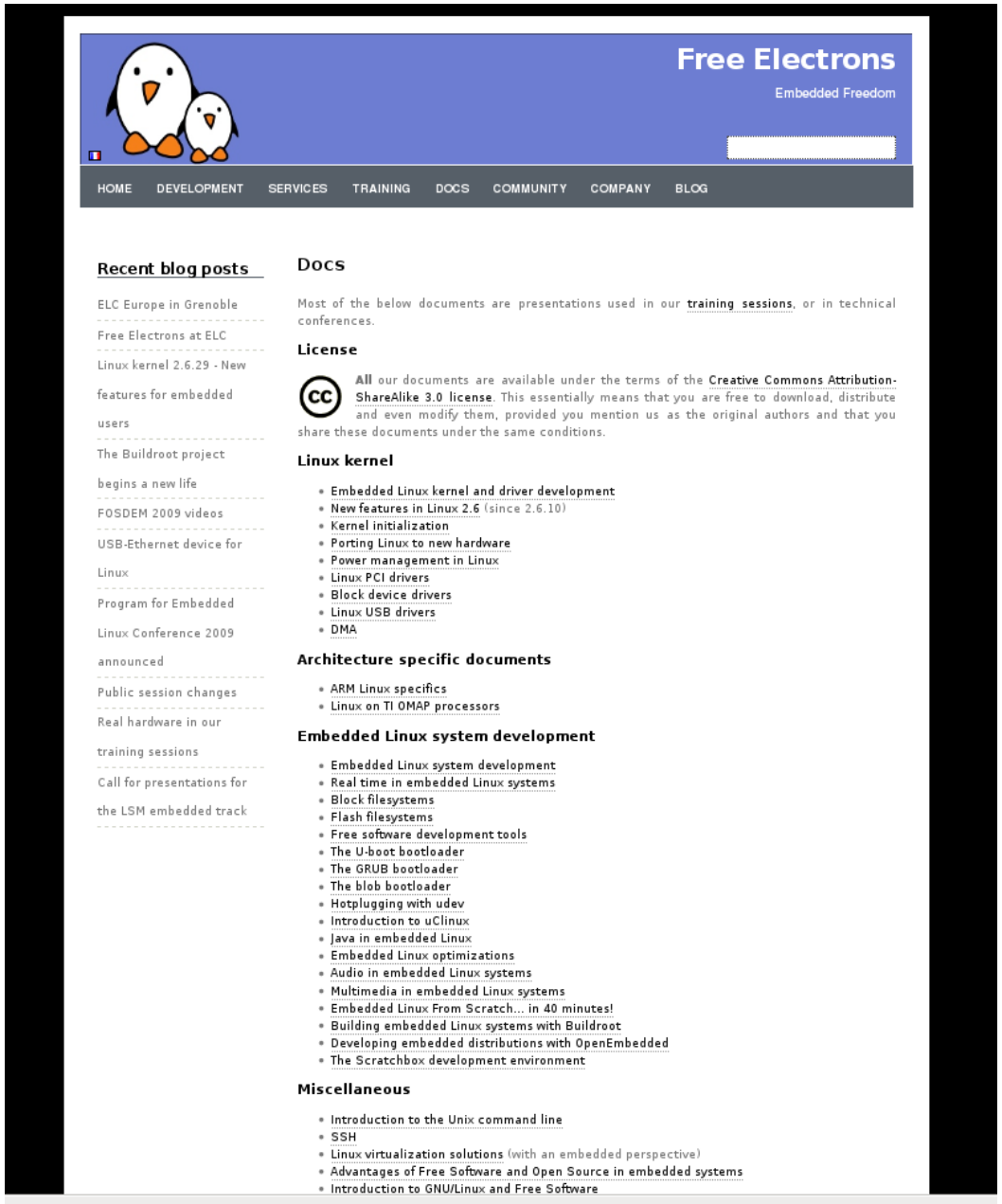
Development tool and application support
Issue investigation and solution follow-up with mainstream developers
Help getting started



http://free-electrons.com

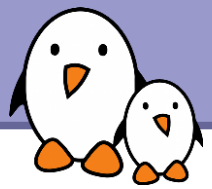
Free Electrons
Free Software for Embedded Systems

Related documents



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