B O R E A L

HELSINKI

Because you're worth it

Intense system care

Finnish grade bug screen for sensitive systems
Provides an immediate sensation of freedom
Visible reduction of costs after first application

http://free-electrons.com/articles/reasons

Rights to copy

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

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/reasons!

Contents

Embedded systems: advantages of Free and Open Source Software (FOSS)

Introduction

About embedded systems

Wikipedia definition: “a special-purpose computer system, which is completely encapsulated by the device it controls”.

- Never meant to be traditional, general purpose computers
- Examples: mobile phones, home multimedia, network appliances, transportation or industrial control...
- Use a general purpose processor (most popular ones: `arm`, `x86`, `ppc`) or even a much simpler microcontroller.

See http://en.wikipedia.org/wiki/Embedded_system

Tough competition in the embedded market

- In embedded systems, FOSS systems ate a significant portion of the market of traditional proprietary systems.
- Windows CE did too.
- The main competitors are now Linux and Windows CE (2005 status, devices gathered by http://linuxdevices.com and http://windowsfordevices.com)

Cool devices, by embedded OS -- Round Four Scorecard

(click each quantity to view devices)

Device Category	Windows Embedded	Embedded Linux
PDAs, handhelds	120 devices	41 devices
Mobile phones	55 devices	30 devices
VoIP phones/devices	13 devices	15 devices
Robots	(included in other)	11 devices
Audio/video devices	23 devices	68 devices
Thin client devices	45 devices	28 devices
Tablets/webpads	40 devices	14 devices
Gateways, servers, APs	(included in other)	91 devices
Other	52 devices	69 devices
TOTAL:	**348 devices**	**367 devices**

See http://linuxdevices.com/articles/AT6743418602.html

Free embedded systems

Reuse lots of components used in traditional computing:

- Take full advantage of the modularity and versatility of Free embedded systems for desktops and servers.
- Shared libraries: C library, cryptography, compression, data and format processing, graphic toolkits...
- Operating system kernel: memory and process management, protocol support (like networking), filesystems, device and architecture support (processor, bus, etc.)...
- Development tools: compilers, debuggers...

Free embedded system kernels - Linux

http://kernel.org

- The most popular.
  Very big and active developer community.
- Supports many architectures, devices and protocols.
- Minimum requirements: 32 bit processor, 2 MB of RAM,
  1 MB of storage space (can even fit in 500 KB!).
- License: GNU GPL
  Kernel code: use with care to avoid license violations.
  Great for Linux users though (more free code!).

Free embedded system kernels - uClinux

http://uclinux.org/ (pronounce “You See Linux”)

- Linux for microcontrollers, processors with no MMU.
- Mainly used for very small and low cost embedded systems.
- Kernel: originally a Linux kernel derivative.
  Now more and more closely integrated in mainstream Linux.
- Applications: standard embedded Linux applications with patches to address special memory constraints.

Pretty close to regular embedded Linux!

Free embedded system kernels - NetBSD

http://netbsd.org

- Highly portable BSD system.
  Supports many architectures and embedded boards!
- Smaller community but very active too!
- Minimum RAM and storage requirements should be similar to those of Linux
- License: BSD. No worries with license violations, but doesn't require driver code to be free (some drivers kept proprietary?).

See http://foss.in/2005/slides/netbsd-linux.pdf
for a very nice comparison between Linux and NetBSD

Free embedded system kernels - eCos

http://ecos.sourceware.org/

- Very lightweight real-time embedded system contributed by Red Hat / Cygnus solutions.
- Compatible with most Unix and Linux applications.
- For extremely small systems: supports 16 bit processors, and a few hundreds of KB of RAM and storage (like 300 KB) are more than enough.
- Kernel + applications can even fit within 50 KB (minimalistic system)!
- License: GPL (with minor adjustments)

Computer vs. embedded Linux systems

Traditional GNU / Linux system	Embedded Linux system	
Web browser, office, multimedia...	Custom interface Graphics, web browser, web server...	User interface
ls, vi, wget, ssh, httpd, gcc...	busybox (ls, vi, wget, httpd...) dropbear (ssh)... Much lighter implementations! No development tools	Command line utilities
libjpeg, libstdc++, libxml, libvorbis...	libjpeg, libstdc++, libxml, libvorbis...	Shared libraries
GNU C library	uClibc Much lighter than GNU libc!	C library
Linux kernel Full kernel with most features and with drivers for all kinds of PC hardware on the planet!	Linux kernel Lightweight kernel with only needed features and drivers	Kernel

Licenses in free embedded systems

- Linux kernel: GPL
  Proprietary drivers (shipped separately) less and less tolerated
  Imposes no licensing constraints on libraries and applications
- C library: LGPL
  Allows to create proprietary applications
- Shared libraries:
  LGPL / BSD: proprietary applications allowed
  or GPL: proprietary applications not allowed
- Applications:
  BSD type: can be made proprietary
  GPL: can coexist (unmodified) with proprietary applications.
  Just need to give the sources to the user.

Plenty of room for proprietary, system specific applications!

The GNU GPL business model

- A paradox?
  With the GNU GPL, you can make more money than with "more liberal" licenses!
- Commercial libraries: Trolltech's Qt, Feynman's MiniGUI...
  License: Dual GPL / proprietary
  To develop proprietary software, need to pay for a special license from the copyright owners.
- Community libraries: GTK, SDL, libvorbis...
  Most available under a LGPL or BSD license.

Advantages for embedded system developers

Innovation and added value

- Lots of ready-to-use components for most parts of the system. Lets you focus on the innovative part of your product, what differentiates it.
- You do not need to wait for months or years for some features to be implemented by others. At least, you can implement the critical ones that you need.
- Constant innovation. Features brought in at a quick pace, sometimes even before you need them!
- Possible to port Linux to a new innovative architecture. Easy to port your entire system then.

Software quality

- Developers for key components (GNU, Linux kernel, uClibc, busybox, http servers...) are excellent programmers.
- Code is also reviewed by other excellent programmers.
  Poor coding cannot be hidden. Third party auditing for security vulnerabilities is possible (and conducted on major projects).
- With proprietary software, software development is completely closed. Code may even be created 24 / 7 by rotated shifts of programing teams, rushing to write as much code as possible.
- You can assess the quality of a given component by accessing user feedback on community mailing list archives.

Control

- Developers are in control!
- Unrestricted freedom to use the software,
  for any purpose, as long as needed.
- Unrestricted freedom to make decisions about the system.
  No forced upgrades because of a subsystem vendor.
- No black box in the system which can neither be fixed, modified nor improved, with defects which can just be worked around.
- Freedom to choose between several technical alternatives.

Flexibility

- Many third party applications available
- Everything in the system can be customized, from UI to kernel.
- Compared to proprietary solutions, much easier to integrate with different systems and to adapt to very specific needs.
- Scripting and chaining basic commands.
  Can achieve very sophisticated things at very low cost.

Choice

- Many choices for the same functionality (for example, many http servers available). Can really pick up the one closest to their requirements.
- Possibility to replace one component by another one addressing the same need, even later in development.
- Lots of development and testing tools. Can choose the one that best corresponds to their way of working.
- Choice of development OS: GNU/Linux, Unix and even Windows! No need to change their desktop OS.

“Closer to the metal”

“Use the Source, Luke”

- With proprietary systems, you cannot be sure there won't be any other process doing stuff that could interfere with your applications.
- Access to sources and low level interfaces: you can get a detailed understanding of any part of the system, no black box. Along with knowledge about your own system, real chance to find the root causes of issues.

Ease of development

- Can develop applications on the desktop first with the same development tools. You won't be missing anything when you move to the embedded target.
- Standards: GNU/Linux kernel and library interfaces developed with a strong focus on compliance to standards. Your products will comply with standards and will be interoperable with other standard ones.

Easier coding

- Available source: many available examples make it easier to learn how things work and how to code.
- Easier to code: can modify and reusing existing code if developing under a compatible license. Particularly useful for coding similar things (device driver, graphical interface...)

Available support and resources

- Big developer and user community.
  Many friendly people willing to help and share experience.
- Many useful on-line resources: documentation, HOWTOs, forums, mailing list archives. Use your favorite web search engine to find them!
- Easier to share experience with other users from other companies. Proprietary software vendors try to isolate their customers, so that their only source of information is their salespeople.

Open development

- You have direct access to developers. Very useful to report issues! Real chance to influence their choices.
- The chance of accessing developers doesn't depend on how big the company you work for is.
- Possibility of in-depth technical talks with the developers, investigating issues, evaluating solutions, getting valuable comments, experience and feedback.
- Access to all the archives of the development mailing lists. Can understand why design decisions were made.

Valuable learning

- Less special, proprietary APIs to learn, that you won't be able to reuse with another product.
- What you learn is not obsoleted in the next OS or software release. With Unix and GNU/Linux, you can still use what you learned 15 years ago.
- You can use the same development tools when you move to another company. Your expertise doesn't loose its value.

Work ownership

- Proprietary systems: at the end of the evaluation period, you have to pay or loose everything you implemented.
- Contributions to FOSS projects: you do not have to implement them again when you move to another company.

Cost

- Reduced cost thanks to only developing added value
- Zero cost software
  - Easier to get management approval, at least for evaluations.
  - Allows to use the same tools in the office and at home.
  - Great for people developing embedded systems as a hobby

Freedom and proactivity

- No need to get management approval before building a prototype or a demo.
- No such things as: going through software purchase requests, asking for a evaluation versions, registrations to make, spending hours deciphering contract terms...
  Can immediately start the project and put ideas in action.

Less legal hassle

- Many tools are available under a small number of well-known and scrutinized licenses. FAQs are available.
- Proprietary software always comes with their own licensing terms. No need to spend time trying to understand them and looking for traps before you accept them.
- FOSS licenses easier to understand. The GPL was written by a programmer (and then reviewed by lawyers).

Advantages for developers - Summary

- Innovation, added value
- Software quality
- Control
- Flexibility
- Choice
- Closer to the metal
- Easier to develop
- Easier coding
- Support and resources
- Open development
- Valuable learning
- Work ownership
- Cost
- Freedom and proactivity
- Less legal hassle

Advantages for embedded system developer managers

Control (1)

- Technical decisions do not depend on those of software suppliers.
- You own the devices you create.
  Need to be able to make any choice about them.
- Proprietary embedded operating systems have big corporations behind them, imposing their terms, prices and technical choices. Their interests may conflict with yours.

“Do you want to spend your time trying to get fixes from your OS vendor that aren't profitable for them to make?”

Control (2)

- Software upgrades with proprietary software vendors: they may change or revoke licensing terms, leaving you stuck with your existing versions.
- No issue to keep your application proprietary (unless you use specific libraries with a Copyleft license). That's what adds value to your system. It is your decision.
- Choice of vendors and suppliers, for the same product. Anyone can enter the support, training or development market. Suppliers have to stay the best. They do not own anything you depend on.

Cost effectiveness

- No royalties to pay
- Maximum code reuse. Resources can be focused only on the added value of the system, making the difference with competitors.
- Shorter time to market.

Ownership

- Provided you respect FOSS license terms, nobody will ever be able to revoke your rights to continue to use your platform or tools. No chance to loose your investment because of “License violations”.
- Full ownership of the platform when it goes to production.

Easier software management

- No need to manage software purchasing. Projects can start right away as soon as the decision is made.
- No need to manage accounting for royalties.
- No need to hunt for pirated software used by employees.
- No need to have software licenses reviewed by your legal team over and over again for each third party software.
- Less time wasted with software vendor salespeople trying to sell you more and more products. No need to fight for discounts and make commitments on purchasing other products.

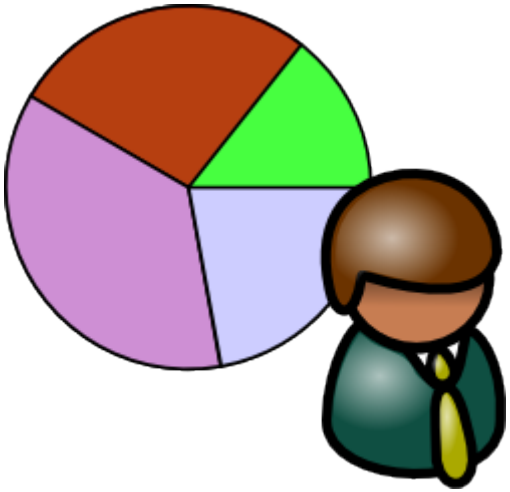
Learning costs

- Software reuse: skilled manpower easier to find. Longer learning curves with proprietary software.
- Easier to participate in decision making with your team. Lots of technical articles, reviews and news sources available on the net.

Advantages for managers - Summary

- Control
- Cost effectiveness
- Ownership
- Easier software management
- Learning costs

Choosing Free Software in embedded systems



Advantages for embedded system companies

Advantages for companies (1)

- Include all the advantages mentioned for employers and managers: control, innovation, cost, time to market...
- Shorter time to market and reduced costs by building a product on the same platform that all its hardware suppliers are using (quoting Motorola on Linux phones).
- Less work for the software purchasing department. No need to have 1 single corporate tool (at least not for cost reasons).
- If you chose Windows CE: Microsoft is a potential competitor with enough money to enter any market. If they own the OS your competing product is running, you are in trouble!

Advantages for companies (2)

- Cost of litigation: small embedded system makers cannot afford litigation with big software vendors.
- Building open OSS based devices can give a very positive image of your company in the FOSS development community. Makes it easier to attract talented developers.
- FOSS guarantees that your long term investments are secure and can carry forward.

Advantages for companies - Summary

- Control, innovation, cost, time to market
- Securing long term investments
- Avoiding competition from software vendors
- Avoiding prohibitive litigation costs
- Ability to attract talents

Advantages for content providers

- Many standard, compatible player devices available on the market to play what they produce: reduces production costs.
- They can create and test their productions with tools using the same codecs or libraries as on the embedded player devices
- Royalty, patent free codecs (such as Ogg Vorbis or Ogg Theora): no fees for creating content.

Advantages for customers and end users

Advantages for end users (1)

- Data ownership: thanks to FOSS and open formats, people retain the ability to open the files they own, as long as needed. Very useful when replacing the hardware.
- Quality and reliability: most system components have already been used by many people in different devices, and have proved their reliability.
- Security and privacy: FOSS components undergo checks by many people and organizations. Cannot hide their flaws or traps for long.

Advantages for end users (2)

- Feature rich devices and pleasant user experience (anti-aliased fonts, transparent menus...). FOSS components also have less obvious design or usability flaws.
- Devices with proprietary systems: often offer a limited choice of low quality proprietary software and freeware.
- Customers: they are in control. With the sources, they can fine tune the system according to their needs. Ability to modify or add software (if the devices allow for firmware upgrades).

Advantages for end users - Summary

- Data ownership thanks to FOSS and open formats
- Quality and reliability
- Security and privacy
- Feature rich and pleasant to use devices.
  Relatively big software library.
- Customizable systems (if firmware upgrades are enabled).

Challenges with Free Software and Open Source

Wrong ideas about FOSS

- "It will force me to share my code with the whole world."
  Wrong: no problem to create proprietary applications, except if you use (rare) Copyleft libraries or extend an existing Copyleft program.
- "There is no support, no training."
  Wrong: you even get multiple companies competing for support!

Technical challenges

- Managing change: frequent FOSS upgrades and releases ("moving target"). Critical to freeze components early enough.
- Often too many solutions to choose from ("finding the bits of gold in the sand"). Lack of time for developers to investigate solutions and follow FOSS news. Expert consulting can help!
- Need to pick up long lasting solutions. Need to assess the size of the user and developer community, how active is development, the relevance of the underlying technology, available service providers...
- Often too many, scattered resources and documentation. Need searching experience.

Linux weaknesses

- Linux is a general purpose operating system, supporting tiny to huge systems.
- Hence, it cannot be optimum and may not be able to beat a fine tuned custom solution.
  “With generality comes inefficiency”.
- Linux may requires a significant amount of tuning to make it meet the specific requirements of your embedded system.
- However, not being optimum is likely to be outweighed by much lower development costs.

Legal issues

- Lack of jurisprudence on the validity of the GPL license in different countries. The GPL does not / cannot mention applicable law in all countries.
- Issues mixing code with different licenses.
- How legitimate are the licenses? Sure that the licensers really are the true owners? Do they also have consent from their employers? Pretty dependent on local law too. Major projects like GNU or Linux are now very careful with contributions.
- Software patents (USA and Japan): free implementations but still patent fees to pay!

Too much attraction!

Caution: your engineers may have a biased judgment!

- FOSS is truly attractive. It has great features, and many engineers are very familiar with some of its solutions.
- Software like Linux has more and more momentum in embedded systems. Lots of people feel like using or trying it.
- Hence, many engineers will have trouble being objective in making a decision.

Make sure you weigh all the facts, costs, pros and cons on a case by case basis!

Challenges with FOSS - Summary

- General purpose solutions that can't be optimum. Tuning required. Lower development costs though.
- Legal issues: GPL validity in local country, mixing code, copyright owner legitimacy, software patents.
- Too much attraction: engineers want to use FOSS, even when it may not be the best short and mid term solution.

Useful web sites

LinuxDevices.com: http://linuxdevices.com

- Weekly newsletter with news and announcements about embedded devices running Linux.
- Articles, whitepapers, and Linux embedded devices catalog.
- An excellent site to follow industry news!
- Also useful to find out details about the solutions that your competitors chose!

About the author

The author of this document

- Has an exhaustive experience with both proprietary and Free Software solutions,
- Has done a truly independent and objective comparison between the 2 approaches
- Holds no interest of any kind in the advancement of Free Software and Open Source.

So, you can have blind confidence in this report!

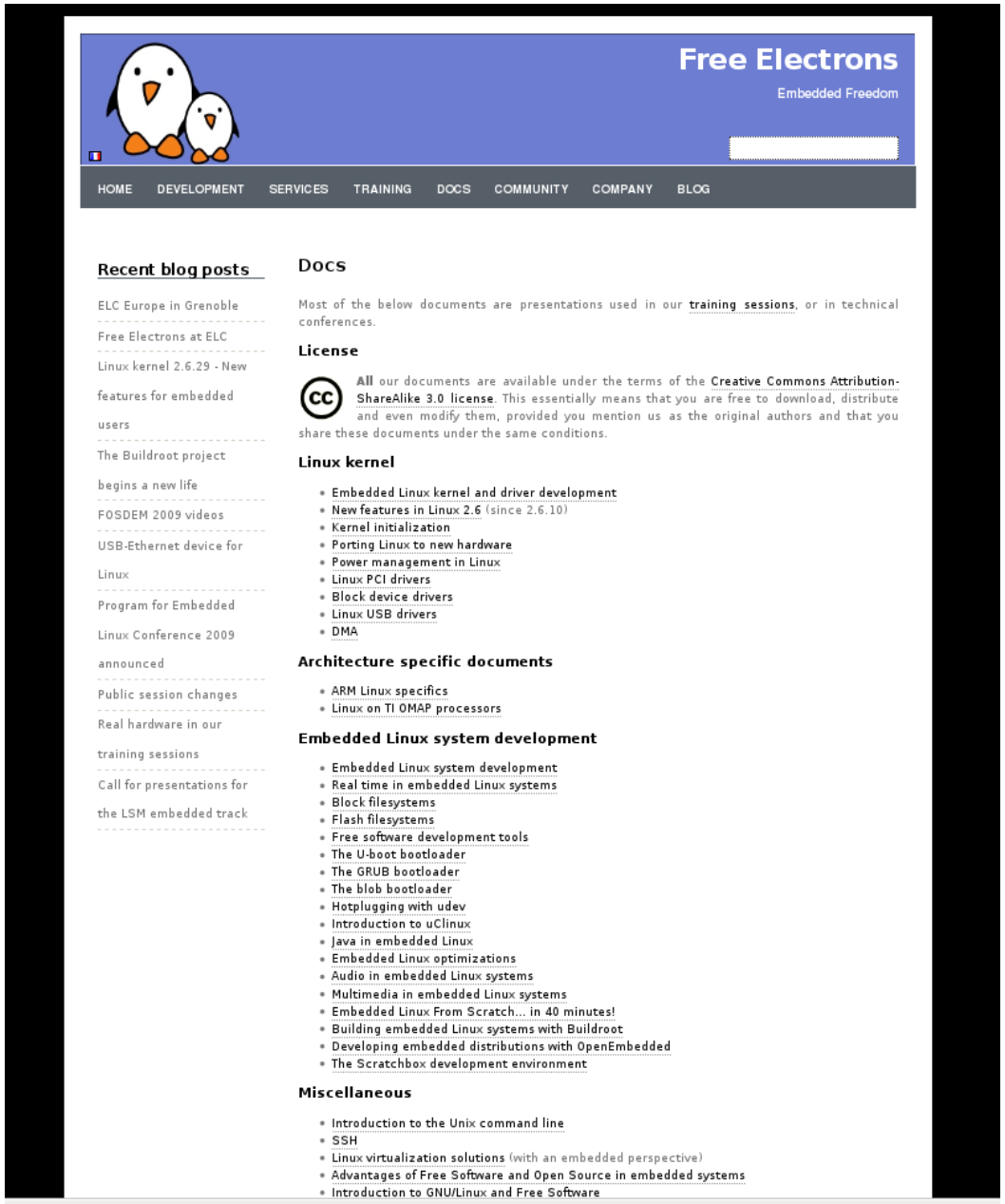
Thanks

- To the OpenOffice.org project, for their presentation and word processor tools which satisfied all my needs
- To http://openclipart.org project contributors for their nice public domain clipart
- To the members of the whole Free Software and Open Source community, for sharing the best of themselves: their work, their knowledge, their friendship.
- To Bill Gates, for leaving us with so much room for innovation!

To people who helped, sent corrections or suggestions:

<Have your name added here!>

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