How to do Embedded Linux [not] right

by

Harald Welte <laforge@gnumonks.org>

netfilter.org / openmoko.org / openpcd.org
gpl-violations.org / openezx.org
hmw-consulting.de / viatech.com

Introduction

Who is speaking to you?

- an independent Free Software developer, consultant and trainer
- 13 years experience using/deploying and developing for Linux on server and workstation
- 10 years professional experience doing Linux system + kernel level development
- strong focus on network security and embedded
- expert in Free and Open Source Software (FOSS) copyright and licensing
- digital board-level hardware design, esp. embedded systems
- active developer and contributor to many FOSS projects
- thus, a techie, who will therefore not have fancy animated slides ;)

Introduction

Why am I qualified?

- The 'Linux community' POV
  - Former kernel subsystem maintainer (netfilter/iptables)
  - Initiator of OpenEZX project
  - Author of various drivers for embedded hardware
- The 'embedded Linux done the right way' POV
  - Lead System Architect (SW+HW) at Openmoko, Inc.
  - Co-creator of Open Hardware + Software for RFID
    - OpenPCD, OpenPICC, librfid, libmrtd
- The 'chip manufacturer' POV
  - Open Source Liaison at VIA Technologies, Inc.
- The 'customer of consumer-grade embedded Linux' POV
  - Done reverse-engineering on hundreds of devices for gpl-violations.org

Linux is everywhere!

Linux is everywhere

Linux is everywhere!

- Linux mobile phones (Motorola, Openmoko)
- In-flight entertainment systems
- Embedded networking gear
  - DSLAMs
  - rack monitoring
- Public payphones
- ATM's / PoS / vending machines
- Now even Fitness gear (Daum Ergometer)

Strengths of FOSS [0/4]

What are the true strengths of FOSS

- Innovative and creative development
- Security due to code review / bugreport / patches
- Long-term maintainable code
- Stable and reliable systems

Strengths of FOSS [1/4]

Innovative and creative development

- easy-to-read existing codebase
- standard (FOSS) development tools
- thus, easy to modify and add features
- community will build around great new features/apps

Strengths of FOSS [2/4]

Security due to code review / bugreport / patches

- all the code is out there
- many people are familiar with existing architecture
- code quality requirements usually very high
- community process allows quick and fast integration of bugfix

Strengths of FOSS [3/4]

Long-term maintainable code, because

- there's a lot of attention on good software architecture
- many developers are familiar with the shared/common API's
- code quality requirements usually very high
- all code in mainline gets maintained and updated

Strengths of FOSS [4/4]

Stable and reliable systems, because

- code quality requirements usually very high
- kernel releases are quite frequent
- all mainline code is automatically ported to new releases

Reality Check

- So we should have the perfect world
  - tons of embedded Linux products
  - all of them maintainable, secure, stable
  - encouraging lots of creative work on top of their codebase
- What is the reality
  - tons of embedded Linux products
  - none of the strengths of FOSS present in 99% of them

Differences to PC Linux

Differences to Linux on a PC

- In the PC world, I can
  - download the latest kernel from kernel.org
  - compile + install it on almost every current+old platform
  - have an almost 100% chance that it will boot and support all the major peripherals
  - only some more obscure hardware might not have drivers yet
  - update at any time to the latest 2.6.x.y maintenance release
  - update at any time to the next 2.6.(x+1) release

Differences to PC Linux

Differences to Linux on a PC

- In the PC world, I can
  - take about any major Linux distribution, based on my own preference
  - install and run that very distribution on about any hardware
  - distribution kernels are very close to mainline these days
  - benefit of regular security updates by distributions

Differences to PC Linux

Differences to Linux on a PC

- In the Embedded world
  - every CPU/SoC maker runs their own kernel tree
  - often one kernel tree per product, based on different mainline versions
  - ages-old base revisions
  - a never-ending security nightmare
  - no benefit from recent new features in mainline
  - non-standard subsystems (e.g. different USB device or SDIO stack)
  - proprietary drivers cause lock-in to old kernels

Differences to PC Linux

Differences to Linux on a PC

- In the Embedded world
  - I often do not have a choice of which distro to run
  - There might actually be no distribution
  - No regular security updates
  - Often no package management for deploying those updates
  - If there are distributions, they either need to use the kernel from the BSP (which is ages old) or create yet another custom off-mainline branch/port

Differences to PC Linux

THIS SUCKS!

What does the vendor get

So what do the embedded vendors get?

- unstable software
- security nightmares
- unmaintainable code
- no innovation
- no user-contributed bug fixes

What does the vendor get

unstable software

- because the code really sucks in many cases
- because they patch around problems rather than solving them

What does the vendor get

security nightmares

- because they use stone-age forks of the kernel
- because they never contributed their code mainline
- because those forks can never be merged back with mainline again

What does the vendor get

unmaintainable code

- because they have one fork of the code per device (product)
- because their code quality sucks

What does the vendor get

no innovation

- because they try to hide their code (gpl-violations.org)
- because their R&D environment is non-standard
  - weird cross-toolchains that nobody has seen before
  - weird filesystems with custom patches that nobody knows
- because they add proprietary components to lock developers from adding features
  - e.g. the entire web-based UI for embedded networking gear
  - binary-only kernel modules that force people to use old kernels with no interesting new features
- because it is, overall, way too hard to develop on/for their platform
- because they don't disclose serial console and/or JTAG access

Reality Check

So why do they still do it?

- there can only be one conclusion:
  - they never understood the real potential of FOSS!
  - all they do is try to compete with what proprietary competitors do
  - they never think about creating platforms, every product is distinct/separate
  - they have no interest in improving their products

What does the community get

So what does the community get?

- products that run some crappy fork of Linux somewhere under the hood
  - but which we cannot really touch/modify without a lot of effort
- we face opposition from the product maker if we want to help him to improve

TODO (device maker)

What should the device vendor do?

- stop thinking in terms of selling black boxes
- defining products that take advantage of the true strength of FOSS
- compete against the proprietary competition on a level that they can't match
- give up the idea of defining all aspects of an appliance
- rather think of building an extensible platforms and let community innovate

A note to chip vendors

There are two types of customers

- The Linux-aware customer
  - understands FOSS much better than you do
  - will share the criticism of this talk
  - will likely go to a competitor who understands Linux better
- The Linux-unaware customer
  - who just uses Linux to save per-unit OS royalties
  - who doesn't really care about most issues presented here
  - who will create inferior products
- Linux-awareness is increasing, not decreasing
  - now is already late, but if you don't have proper FOSS support on your agenda now, you will likely loose the "openness competition"

TODO (chip maker)

What should the chip vendors do?

- engage in "sustainable development"
  - develop against latest mainline
  - make your development trees public (use git!), don't just release stable snapshots as BSP to your tier-one customers
  - actively interact with the community
  - learn how to play by the rules (coding style, use common interfaces, no proprietary drivers)
  - don't just do big code drops every now and then

TODO (chip maker)

What should the chip vendors do?

- Externally
  - Don't mistake FOSS as just a technology. It is a R&D philosophy!
  - Provide public reference manuals with no NDA
  - If you have to resort to NDA, make sure they are FOSS friendly
  - Ensure you don't license IP cores that conflict with FOSS
- Internally
  - Draft a proper in-house FOSS strategy with clear goals
  - Don't expect your product mangers or engineers to know everything about FOSS without proper training
  - Hire people with strong community background into your R&D and management to facilitate the know-how transfer

Lessons Learnt (chip maker)

- Chip Product Managers need to learn
  - There's more Linux use with their chips than they ever learn
  - Their customers are not just the tier-one customers
  - If you want to support Linux, do it the mainline way. If you support only N number of distributions, your 'N' will be growing and you'll be wasting R&D resources to support each one of them
  - That there is no single 'contact window' / entity for Linux
  - The big change is the FOSS development model, not the Linux API's
  - Linux-aware customers care not only about performance+price, but also about the quality of the Linux port code

Lessons Learnt (chip maker)

- BSP R&D Managers need to learn
  - Linux is not just a set of API's for their developers to learn
  - Linux is FOSS. The FOSS R&D model is different
  - Their engineers need to be encouraged to communicate
    - and thus, need real Internet (git, mailing lists, ..)
    - have to get clear indication what is confidential and what not
  - FOSS is about cooperation. You don't need to reinvent the wheel
    - don't include your own 802.11 stack in your wifi driver
  - If you do not merge your code mainline, you end up in a maintenance nightmare

Lessons Learnt (chip maker)

- BSP Software Engineers need to learn
  - Linux is not just a set of API's
  - How and where to ask the right kind of questions
  - How and where to look for the latest code
  - Code is written to be read by other people, not just to execute
  -

TODO (community)

What should the community do

- Provide non-partisan documentation
  - on FOSS advantages, proper FOSS development
  - hardware companies are interested to learn, but don't know who to ask. If they ask a commercial distributor, then they get in bed with them, which is not the same as working with the mainline development community
- Something like a mentoring program
  - take software/driver R&D by their hand and walk them through the mainline merge
- Don't scare them away
  - They have to be taught about the communication / review culture
  - Your valid criticism to patches has to be explained

Problems (chipmaker)

- Patent licensing schemes
  - e.g. MPEG patent license doesn't at all work in a FOSS model
- Technology licensing schemes
  - e.g. Sony Memorystick. Impossible for a chip maker to provide FOSS driver
- Everyone in the industry has those very same problems
  - Chip makers should cooperate to present their case together to the respective licensors

Problems (chipmaker)

- Patent trolls
  - openly accessible documentation invites patent trolls
- Why?
  - patent trolls rarely read+understand FOSS code
  - thus, open documentation increases the risk to be hit
- However
  - this is a by-product of how the patent system currently works

Outlook

Outlook

- We will see even more embedded Linux, e.g. Mobile phones
- We will see even more restricted devices (Tivo-ization, DRM for code)
  - which go on _very_ thin ice even with GPLv2
  - which almost completely remove all freedoms of FOSS
- We have some faint dim light at the end of a very far tunnel
  - e.g. projects like Openmoko, who truly see openness as a feature
  - e.g. chip makers who slowly open up a bit more
    - on the PC side, Intel definitely is setting the best overall example
    - on the Embedded side, I see some movement in major players (TI, Marvell, Samsung, Infineon, ...)

Thanks

- Thanks for your attention.
- Some Time left for Q&A
- Enjoy the FOSS.in / 2008