How To Get Your Code in the Kernel?

Hans Verkuil
Cisco Systems Norway

Overview

- Overview of the linux development process.
- Case study 1: out-of-tree development of the ivtv driver.
- Case study 2: getting ivtv in the kernel.
- Case study 3: the Xceive tuner support, or: how not to do it.
- Organizational Guidelines.
- Q&A.

The linux development process

- Linus Torvalds remains ultimate authority.
- Subsystem maintainers and responsible for kernel subsystems.
- Driver maintainers feed patches to subsystem maintainers.
- See MAINTAINERS file in kernel sources.
- Driven by technology, not commercial interests.
- Code reviews are an important part of the process.
- No roadmap!
- Communication through mailinglists, irc and conferences and summits.

The media (sub)maintainers

- Mauro Carvalho Chehab is the media maintainer.
- Submaintainers are:
  - Kamil Debski: codec (aka memory-to-memory) drivers
  - Hans de Goede: non-UVC USB webcam drivers
  - Mike Krufky: frontends/tuners/demodulators. In addition he is the main reviewer for DVB core patches.
  - Guennadi Liakhovetski: soc-camera drivers
  - Laurent Pinchart: sensor subdev drivers. In addition he is the main reviewer for Media Controller core patches.
  - Hans Verkuil: V4L2 drivers and video A/D and D/A subdev drivers (aka video receivers and transmitters). In addition he is the main reviewer for V4L2 core patches.

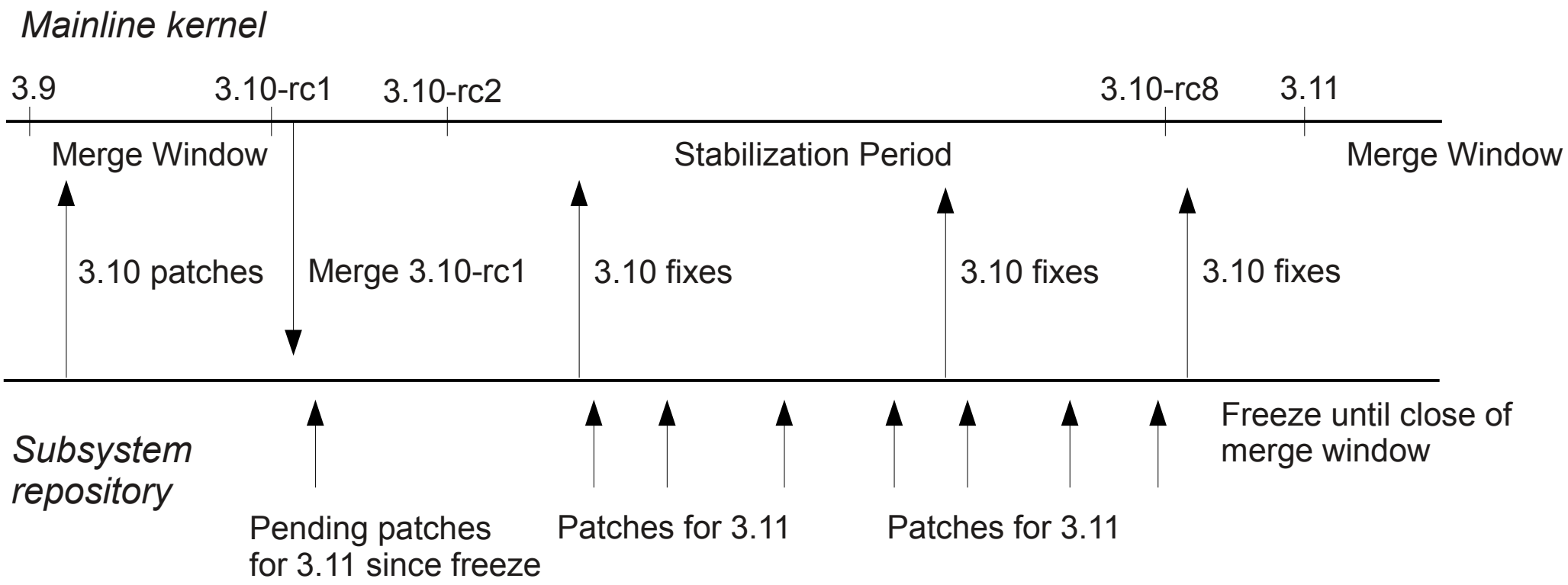
Kernel release cycle

- After a new kernel is released the two week merge window is opened for big changes (new drivers, new architectures, etc.)
- After these two weeks only bug fixes are allowed.
- Every week a new release candidate is made until the kernel stabilizes.
- After approximately 10 weeks a new kernel is released and the cycle repeats itself.
- linux-next tree to prepare for next cycle.
- drivers/staging for not-quite-prime-time drivers.

Media drivers release cycle

- Master code of the media subsystem is in the media_tree.git repository hosted on linuxtv.org.
- New features are added there, and when the two-week merge window opens all outstanding changes are merged into the kernel.
- After that only bug fixes are backported to the kernel release candidates.
- If 3.x is released, then 3.x+1 is the release candidate and v4l patches in media_tree.git are for kernel 3.x+2.

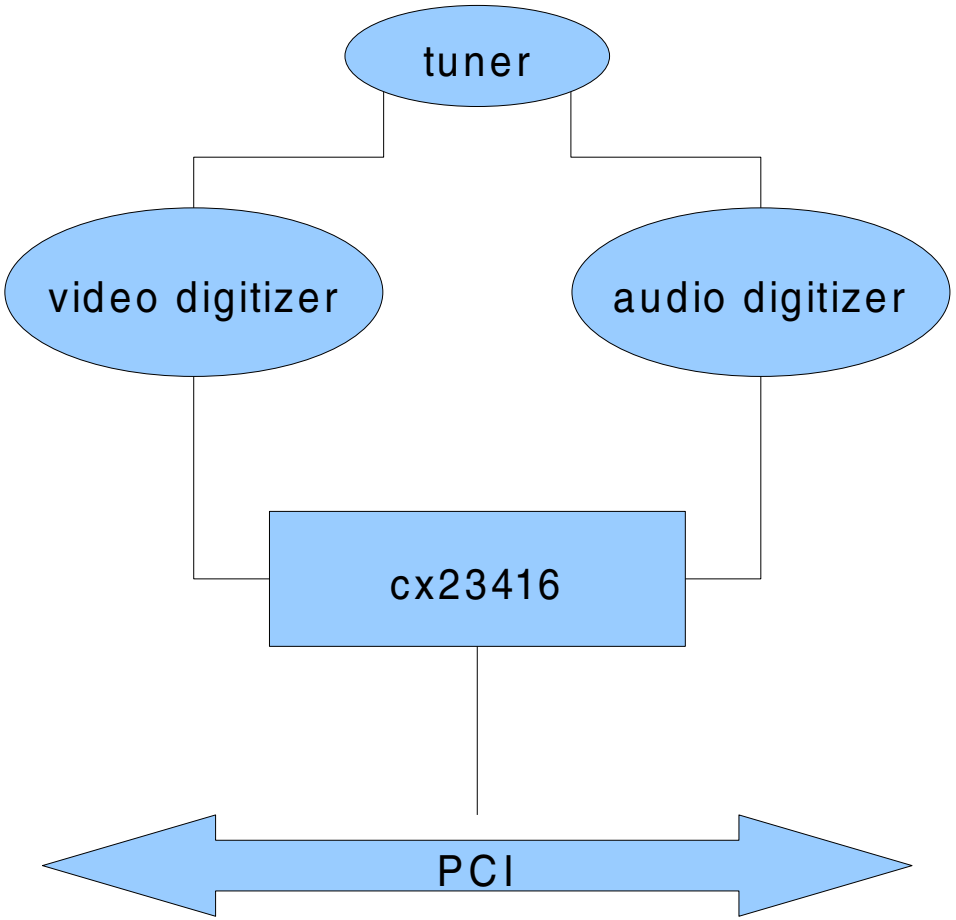
Timeline



Case Study 1

Out-of-tree development of the ivtv driver

Basic design of a cx23416 board



The ivtv driver: a history

- Started by Kevin Thayer in 2003 to support the Hauppauge PVR-350 MPEG encoder/decoder card, based on the Conexant cx23415 chip.
- Maintained by Chris Kennedy in 2004-2005, and from summer 2005 by Hans Verkuil.
- My goal was to get the out-of-tree driver into the kernel, which happened in 2.6.22, July 2007.

Out-of-tree development

Pros

- Useful for initial development and prototyping.
- No annoying code reviews and criticisms: you're the king!

Cons

- You do not exist from the point of view of the kernel community and you will have no influence over the direction of the kernel development.
- You have to keep up with kernel changes yourself: increasingly difficult, will become a maintenance nightmare.
- Usually not included with any linux distribution.

Lessons Learned

- Only use out-of-tree development for the initial phase and prototyping.
- Get into the kernel as soon as possible:
  Your driver will be updated for you whenever kernel APIs change.
- Targeting the Android kernel does not give you any benefits, that is still out-of-tree development.
- Backporting patches from the latest kernel to an older kernel is fairly easy, forward-porting your driver to the latest kernel will be increasingly difficult as time goes by.
- Backporting is a one-time job for each older kernel you need to do that. Forward porting is never ending. Note: drivers/media can be backported easily to 2.6.31 and up.
- Long-term support is important for certain applications: trivial if support is already in the mainline kernel.

Case Study 2

Getting ivtv into the kernel

Getting ivtv into the kernel

- Merging the branched i2c drivers (2.6.15, January 2006)
- Adding new i2c drivers (2.6.17/18)
- Adding required API functionality (2.6.18/2.6.22)
- Adding ivtv (2.6.22, July 2007)
- Adding the framebuffer device for the OSD (2.6.24, February 2008)
- Added the saa717x video encoder i2c driver (2.6.26, August 2008)

Adding APIs

- Post an RFC document on the linux-media mailinglist.
- Discuss via mailinglist and #v4l IRC channel.
- Update the RFC and repeat until everyone agrees.
- Implement the changes in the main API header and document them to keep the V4L2 API Specification up to date.
- There must be at least one driver using the new API before the API changes are accepted!

Lessons Learned

- Feed changes upstream in small chunks.
- Do not expect instant review of your code or RFCs: you often have to prod people after a week.
- Take the time to discuss API changes and be open to (constructive) criticism.
- Stay focused on the technical aspects.
- For very complex problems consider organizing a brainstorm session or visit conferences like the Embedded Linux Conference (Europe) and the Linux Plumbers Conference.

Lessons Learned

- Adding support for new features can take a long time.
- Start discussions on such features as early as possible.
- It is wise to submit drivers with a partial feature set and extend over time.

Case Study 3

The Xceive tuner

The Xceive tuner

- Xceive hybrid tuner: can do both digital (DVB) tuning and analog (V4L) tuning. Also requires complicated firmware setup.
- DVB and V4L did not have a common tuner framework and have radically different approaches to accessing i2c devices.
- Used heavily in em28xx-based USB devices.
- The em28xx maintainer developed an Xceive driver.
- His Xceive driver was part of a huge em28xx patch bomb: impossible to review.

The Xceive tuner

“I see the order is a problem there, I've redone the whole tree 3 times because of problems with some folks on the ML earlier since I wanted to keep the extra changes as small as possible I merged some patches finally.

As from my side I don't have the time to do a major rewrite or doing some reordering again, please try to make the best out of what's available now.

The patches include more than 9000 lines of code changes through the v4l and dvb framework, these changes have been done during the last 1 1/2 years of split development from the main v4l-dvb code, I know it's a lot but it's worth to get it done now. It's not only about adding support for one new device only.”

The Xceive tuner

- Initially refused to split it up, but later did. Tuner patch changed all DVB drivers and in fact introduced DVB bugs.
- Again refused to improve it: 'It works for me and the users of my em28xx driver.'
- Refused to compromise, antagonized other developers (personal attacks).
- Started userspace tuner to work around refusal to take his code.
- Linus refused to accept it. No need and invites binary drivers.

The Xceive tuner

- People started to take his GPL code and work on it themselves: 'Others are stealing my code!'

  “great you apply untested patches stolen from my repository! great.Take care that we don't meet again in reallife, I'm highly annoyed about how you proceed with my code.”

- Tried to force a split between the other V4L/DVB developers.
- Eventually took all his code offline.

Lessons Learned

- Do not make big patchbombs, make small (reviewable) changes at a time.
- Don't sit on your code for a long time. It's much better to upstream code on a regular basis.
- Even better, post early code to get feedback from experts.
- A mailinglist for your driver is an ivory tower: you only deal with people who need this driver, and you can loose sight of how it affects other devices.
- You do not own your code when creating open source SW.
- Do not reply to flames and personal attacks.
- Be aware of language and cultural barriers.
- It's not (just) about how smart you are, but it's about cooperation.

Organizational Guidelines

- Do not mix upstreaming activities with customer support activities. Instead dedicate some people to upstreaming work. Upstreaming activities all to often fail because this golden rule isn't followed.
- Upstreaming is a long-term commitment and involves building trust between the developers *(not* the company) doing the upstreaming work and the kernel community.
- The initial upstreaming effort will be substantial, but once done you can build on that for newer products and your maintenance will decrease substantially over time.
- Do not work for months on a new driver in silence before finally posting it: if you made a wrong decision early on then you may have to redo a lot of work. Post (or mail privately) your code early to ensure the right decisions are made.
- When in doubt: ask!

Organizational Guidelines

- Make budget available for visits to conferences like ELC(E) and LPC: it's the linux equivalent of a Microsoft Developer Conference.
- Open up your datasheets and supporting material. Datasheets rarely if ever contain really 'secret' information that would require an NDA. Opening them up makes it possible for others to contribute to your code.
- Consider making hardware available for testing to relevant subsystem maintainers/developers.
- If you develop ARM-based SoCs, then consider becoming a Linaro member.

Resources

- www.linuxtv.org
- Linux Device Drivers (O'Reilly, 3rd edition)
- http://lwn.net/Kernel/
- The kernel source
- Documentation/CodingStyle + scripts/checkpatch.pl
- http://www.kroah.com/log/linux/howto.html
- Conferences: Linux Plumbers Conference, Embedded Linux Conferences.

Questions?