kernel.org development and the embedded world

Andrew Morton
<akpm@linux-foundation.org>
<akpm@google.com>
CELF Embedded Linux Conference
April 2008

Overview

- Assumed audience is developers of embedded products rather than developers of Linux itself
- Won't cover day-to-day interaction with the kernel.org team.
- Will cover economics of embedded, how this impacts their kernel.org participation
- Patch-hoarding, and why not to do it
- Why merge upstream?
- Which kernel version to use?
- Why do kernel.org developers give free support?
- The merge decision making processes

The economics of Linux-for-embedded

- The product's kernel version is usually not upgraded across product lifetime
  - In desktop and server kernel upgrades are expected
- In embedded the software reaches the end-user directly from the manufacturer
  - In desktop and server, it is usually via intermediaries
- Exceptional case: hardware manufacturers whose customers develop embedded products
- There seems to be generally less cash handy in the embedded segment
- Net effect: embedded development is underrepresented in kernel.org work, relative to its economic importance

Effects of embedded's under-representation

- There is a risk that it will tilt development too far toward server/desktop
- However the kernel.org team care about Linux-on-embedded and want it to work well
- So we do consider each change's effects on small systems
- However this is mainly a minimize-the-damage effort
  - New server/desktop features should not degrade embedded
  - But only a few developers are actively working to improve Linux for embedded
- This is unfortunate – just a single full-time “embedded maintainer” could have a large effect.

How you can help kernel.org development

- Not all contributions are code patches!
- We (and the kernel) can benefit from your experience
  - Tell us your problems
  - Which features are missing
  - Help us prioritize different proposals/features
  - Help us to direct those resources which we do have
- Read the websites, read the mailing lists and don't just lurk
- Review patches
  - Suggesting improvements is a good way to get work done for free
- The squeaky gate gets the oil

Patch hoarding

- Various embedded groups seem to like to maintain out-of-tree patchsets
  - These patchsets linger on for ever and never seem to get merged
- Due to lack of resources (I hope)
  - To retain a little competitive advantage (I hope not)
- It is inefficient and ineffective to have all these little patchpiles dotted around the place
  - Let's get them merged up

Why merge upstream? The pros

- The review, integration and release process will improve the code quality
- Once merged, your maintenance cost falls to near-zero
  - Others will improve it
  - Others will no longer break it (as much)
- Others will enhance it
- It gets more testing
- Others will fix bugs in it
- Eliminates possibility that a competing implementation will be merged
- You avoid the risk of having to migrate to a new implementation, or of maintaining yours for ever

Why merge upstream? The cons

- It requires some up-front work
  - But less long-term work (I think)
- You may be asked to change things
  - This hurts if you've already deployed it
    - This is a reason to merge before deploying!
- The feature becomes more easily available to your competitors
  - Tough luck, live with it.

Which kernel version to use?

- Many embedded developers use old kernels
  - And that's OK – they were fine kernels
    - Much of the development in recent years has been addressing large-system shortcomings
- However, freezing on an old kernel is a short-term strategy.
  - When the time comes to upgrade it can be painful
  - And you will want to upgrade, to pick up new features
- Using an old kernel rules out support from the kernel.org team
- A better strategy might be to develop product on current kernel.org release, freeze it at end of product development
  - Intermediate kernel providers should be able to help
  - kernel.org developers might be able to help

Which kernel version to use? (cont'd)

- A single-product company can feasibly freeze on one kernel version for ever
- This is probably less viable for multi-product companies

Free support!!

- Why do kernel.org developers give free support?
- Well, we don't.
- We will work with you to resolve kernel problems and shortcomings
- It is a mutually beneficial transaction
- If fixing your problem won't improve the mainline kernel we won't help you (much)
- If your problem is in 2.6.ancient then we won't be very interested
  - First question: “does it happen in current mainline”?
  - This is a strong reason for using recent kernels in product development

The merge decision

- The “dictator model”: maintainers make arbitrary decisions
  - We don't do this
- The “rule of law” model: maintainers decide whether a submission meets the written rules
  - Maintainer is a “judge” and not a “dictator”
    - We do do this
- Unfortunately there are no written rules ;)
  - Should meet coding guidelines (these *are* written)
  - Is useful to some kernel users
  - Long-term maintainable (traded off against usefulness)
  - Non-injurious to other parties <- most important?
  - If the feature is large: does it have a maintenance team?

The merge decision (cont'd)

- When in doubt: ask me!
  - I can help
    - That's what I'm here for
      - I'm really nice!
        - And I have great legs
        - But I'll only help you if there's something in it for me

Working with the kernel.org team

- Identify one or two specialists in your organization as the kernel.org interface
  - Don't expect every developer to work with the public developers
    - It just isn't efficient
      - Or effective
- When in doubt, ask me.