Community and Embedded Linux

David Woodhouse

Embedded Linux Conference Europe

November 2008

Software and Solutions Group

What does “Embedded” mean anyway?

- Headless?
- Handheld?
- Low power?
- Physical size?
- Limited RAM?
- Limited persistent storage?
- Other...

So what about “Embedded maintainer”?

- Chasing patches?
- Looking out for stupidity elsewhere
  - Bloatwatch
- Contact point for Andrew Morton
- Encouraging people to work together
  - Companies
  - Community

Embedded "communities" and users

- OpenWRT
- OpenMoko
- OLPC
- Maemo
- Moblin
- Android
- handhelds.org

How well do they work with upstream?

- How many local patches not submitted?
- How visible and accessible is their work?
- How old is their kernel?
- How sane is their code?

How well do they work with upstream?

- OLPC
  - A dozen or so sets of changes, against 2.6.27.4:

    ```
    95 files changed, 7585 insertions(+), 1133 deletions(-)
    ```
- Moblin
  - 23 patches against 2.6.24:

    ```
    174 files changed, 120867 insertions(+), 208 deletions(-)
    ```
- OpenWRT
  - 160 patches against 2.6.27:

    ```
    410 files changed, 65387 insertions(+), 1027 deletions(-)
    ```
  - **AND** 425 extra files, with 125000 extra lines
- Maemo (Nokia Internet Tablets)
  - *FAIL*

Why is upstream so important?

- Ease of merging fixes and new features
- Avoids duplicated effort
- External code review
- Compile testing
- Automated bug checking (Coccinelle, etc.)
- Updates for kernel API changes
- “Janitors”

Why is upstream so important?

Divergence

Is Pain

Why do people “hoard” code?

- “Too hard” to ~~write decent code~~ get code accepted
- Not enough time
- Developed against ancient kernels
- Dubious legal issues
- Upstream resistance to changes

How can we make things better?

- Use git
- Keep separate git trees for “topics”
- Pull topic trees into “working” tree
  - Commit to working tree only as last resort
  - Regularly assess “outstanding” code in working tree
- Work with upstream maintainers regularly
- Call on dwmw2 and akpm where necessary

Adopt a driver!

- Take responsibility for shepherding code upstream
- Clean it up and make it look sane
- Talk to relevant maintainers
- Put it into a git tree based on a current kernel
- Where appropriate, get it into linux-next...
- ...or if it's still crap, GregKH's “staging” tree
- Make sure people know how to find you
- Respond to feedback

Conclusion

- We need to work better as a community before we can point fingers at companies who don't play nicely.

Questions?

Leap ahead™