Experiences of a x86 maintainer

Feb 2009

Andi Kleen, Intel Open Source Technology Center
andi@firstfloor.org

Disclaimer

- This happened all before I joined Intel
- Not an Intel project
- Not speaking for Intel

What is a Linux maintainer?

- Patch collector
- Release manager for a subsystem
- Architect
- Default blamee
- Politician
- Sends patches for a subsystem to Linus
- But not absolute control over code
  - Linus and some other people can overrule
- Sometimes hard to not merge patches

More on maintainers

- On a larger project spending a lot of time on administrativa
- And code review
- Not that much time to code anymore
  - For large subsystems
- For complex projects also has to sub-delegate some areas
  - Become a middle manager

Code review

- ... is hard work
- required to keep linux coding standards up
- Normally would like to have reviewers on mailing lists do it
- Often the maintainer has to do the bulk of it in the end
  - Iterates until code is acceptable
  - Takes a lot of time
- Code reviewing on mailing list is an important contribution!

x86 maintenance

- Originally just worked on x86-64
- Project started with no clear kernel maintainer
  - Just a group of engineers
- Maintainer needed as interface to the outside world
- x86-64 kernel maintainer
  - but also x86-64 gcc/glibc/gdb/... maintainers
- Became defacto i386 maintainer too

Release trees

- Old
  - 2.4 tree main work
  - unstable 2.5 tree completely different
  - some distribution trees with lots of backports
- New
  - 2.6.x vs 2.6.x+1
  - Distribution trees
  - 3 month cycles

Phases of the project

- From novelty to commodity
- Complexity rising significantly
  - Not as much in the code
  - But in the platforms that need to work
- Interaction with other subsystems takes more and more time
- Farmed out some work
  - For example ACPI took over a lot of BIOS issues

From single platform to (nearly) everywhere

- First implementation on simulator
  - Then long time hiatus
- Then single hardware platform
- Then multiple platforms
- Then mass market with many more platforms
- Today (nearly) everywhere in PC space

How it started

- Initially mostly removing code from a copy of arch/i386
  - Goal was to get rid of old hardware workarounds
  - To get a cleaner and more manageable software
- Implemented 64bit support
  - done by a team
  - Various code areas redesigned
- Then was alone on the kernel side for over a year
  - Simulator only
  - Especially 2.5 was tough

Headaches

- New chipsets
  - Many new chipsets have one quirk or another
  - Especially those from smaller vendors
- BIOS
  - If Windows doesn’t use it ...
  - Servers are better than clients
  - The cheaper the system the worse the BIOS
    - But even expensive systems often have bad bugs
  - A lot of BIOS workarounds
  - Luckily significant part of it was handled by ACPI team
    - But still a lot of of non ACPI BIOS issues

32bit x86 maintenance

- i386 didn’t have a dedicated maintainer
  - resulted in some substandard code being merged
- Did i386 maintenance on the side
  - Primary focus was still on 64bit
- Plan to add 32bit support to 64bit
  - Get cleaner codebase
  - Never happened due to time constraints

Compat layer

- Allows to run 32bit x86 software under 64bit kernel
- In theory everything free can be recompiled...
  - but in practice it’s often not as easy
- Based on sparc compat layer
  - Not auto generated
- 98%+ compatible
- Wine was an interesting experience
  - First Solitaire
  - More compatible than the original
- Learned a lot of corner cases

Compat layer problems

- The kernel compat layer is quite good
- But relies on distributions shipping shared 32bit libraries
  - Didn’t spend enough effort educating
- Some popular distributions don’t ship 32bit compat libraries
- Large adoption hurdle for 64bit today

Bug management

- ... Still remember the day when I realized I couldn’t keep track of all bugs anymore
- Originally just bug list in a text file
- Then later handled by various people
  - Was difficult to track regressions
  - and determine release readiness
- Some bugs later handled in bugzilla
  - Most Linux subsystems still do it informally
  - ACPI is the main exception
- Also emergence of central bug masters

- Bug management is important

Last thoughts

- It’s very motivating when your code is widely used
- But it’s also a lot of work