Production process, bug tracking and quality assurance of the Linux kernel

Andrew Morton
<akpm@linux-foundation.org>
<akpm@google.com>

Hannover Fair
April 2008

Overview

- Production process
  - Release cycle and kernel delivery
  - Contributors
  - Code flow
  - The -mm tree
  - The linux-next tree
- QA
  - Testing
  - Testing and the internet
- Bug reporting
  - Email
  - Kernel bugzilla
  - Shortcomings

Release cycle and kernel delivery

- Linus releases one 2.6.x kernel every 2-3 months
- This cycle is the central "heartbeat" of the kernel development and release process
- Linux distribution companies will pick a release from this ongoing output and will freeze on it, stabilise it, release it as a product and will provide post-release updates to it.
- This is the added value which you pay your distributor to provide
- The kernel.org team provide a technology. Distributors turn it into a product.
- Most people will get their kernels via a distributor

Release cycle and kernel delivery (cont'd)

- Embedded product developers will often get their kernel direct from kernel.org: they too are making a product and kernel.org provides that product's base technology.

Contributors

- Who are they and why do they work on the kernel.org kernel?
- Most contributors are engineers from companies which have Linux-related business
  - Distributors
  - IT hardware companies
  - Software companies
  - Embedded chip manufacturers (provide hardware input to...)
  - Embedded product developers
- Some contributors are unpaid enthusiasts
- Testing and reporting is a contribution as well
  - This is one area where the unpaid enthusiasts perhaps dominate.

Code flow

- The kernel is divided into "subsystems"
  - scsi, USB, locking, memory management, ext3, ia64 support, documentation, etc, etc.
  - There are probably 200-300 identifiable subsystems
  - Nearly 100 of the subystems have an identifiable maintainer
  - The maintainer writes code as well as taking contributions from others
  - The maintainer operates a "subsystem tree" for that subsystem.
    - Usually a git-based tree
  - That tree is available for others to test and to develop against all the time.

Code flow (cont'd)

- When Linus releases 2.6.X, we have a two-week "merge window" in which the subsytem maintainer is to merge the ready-to-release part of the subsystem tree into Linus's mainline kernel for 2.6.X+1.
- When the merge window ends, Linus declares 2.6.X+1-rc1.
- We then spend two months stabilising 2.6.X+1-rcY for the 2.6.X+1 release.
- At the same time, new code for 2.6.X+2 is accumulating in the subsystem trees.

The -mm tree

- While all this code is accumulating in the subsystem trees I will periodically merge all of it together and will release it as 2.6.X+1-rcY-mmZ.
- People will download and test the -mm tree and will report bugs.
- Hopefully they get fixed before the buggy code goes into Linus's mainline.
- The main objective of -mm is to avoid breaking Linus's tree too much
- It also effectively lengthens the stabilisation window from 2 months to as much as 4 months.

The -mm tree (cont'd)

- The -mm tree also gives us visibility into forthcoming conflicts between all the subsystem trees.
  - So we don't hit major problems during the very short merge window.
- The -mm tree also hosts about 100 subsytem trees. Most are small, a few are large.
- This is service I provide for subsystem maintainers who don't feel a need to run their own git tree
- Some of those subsystem trees don't even have a maintainer, so I am effectively taking that role.

The “linux-next” tree

- About 85% of kernel changes flow into mainline via the subsystem maintainers' trees. They are where most of the work resides.
- The patches which are hosted in -mm constitute the remaining 15%.
- It is a lot of work to merge together all the subsystem trees for -mm release, and this work could be performed by someone else.
- Stephen Rothwell is now doing that, and I shall soon switch the -mm tree to be based on his linux-next tree.
- So hopefully I will only have to perform integration of 15% of the Linux changes rather than 100%.

Quality Assurance: Testing

- An operating system is like no other software product
- The large variation in hardware, CPU types, BIOS implementations and workloads means that the kernel developer cannot fully test his code.
- It worked OK on the hardware which he has available
  - But there are millions of possible hardware combinations
  - Plus hardware and BIOSes are often buggy
  - Most kernel bugs are hardware dependent
  - Most hardware-dependent bugs cannot be reproduced on the hardware which the kernel developer has available.

Quality Assurance: Testing (con't)

- So to test a kernel change, that change needs to be executed upon many machines: thousands, tens of thousands, to find problems.
- A change needs to be tested by 1000's of humans as well – one person's usage pattern will trigger a bug which hundreds of others never encountered.
- Several groups have set up kernel testing systems: run lots of tests on a few machines.
  - These haven't been effective - they just don't find many bugs because a "few" machines do not have sufficient coverage.
  - It would be better to expend resources on fixing more of the bugs which are reported by the testers.

Quality Assurance: Testing and the internet

- All of this explains why the large and widespread community of testers is so important to kernel development. We cannot test our own code!
- So we have a very rapid and open "release early, release often" philosophy.
- Much of the kernel development system is built around this.
- We are utterly dependent upon our testers. If people were not to download and test new kernels and report problems, the whole project would fail.
- I am very aware of this and I do attempt to ensure that we cater to our testers as well as we can.

Bug reporting: email

- Most bugs are reported via email. They should be reported to the linux-kernel mailing list and/or to the subsystem-specific mailing list.
- Individual developers should be cc'ed, if they are known.
  - Reporters often get the routing wrong but that's OK - people will help to redirect the report appropriately (often I do this)
- Once a bug is reported, developers should (and sometimes do) start working on getting it fixed.

Bug reporting: email (cont'd)

- Often this will require additional help from the reporter, because only the reporter has the hardware upon which the bug can be reproduced.
  - We'll request extra information
  - We'll send patches and request testing
  - We may ask the reported to perform a bisection search through the kernel patch series to identify which patch caused the bug.

Bug reporting: bugzilla

- The kernel also has a bugzilla bug-tracking system.
- It is more appropriate for tracking bugs which have been present for a while.
- We prefer that bugs get fixed quickly via a short email interaction, but that doesn't always happen.
- Often people will report bugs via bugzilla which we would have preferred be reported via email.
  - But that's OK. I screen all bugzilla reports and will ensure that they are made known to the appropriate developers in the way which is appropriate to the particular bug and to the way in which that development team prefers to work.

Bug reporting: bugzilla (cont'd)

- I also will track emailed bug reports (from linux-kernel only) and will ask that they be recorded in bugzilla if they haven't been fixed after 3-6 months or so.

Bug reporting: shortcomings

- Many bug reports just get lost.
  - There is no identifiable maintainer
  - There is a maintainer, but he's asleep
  - We don't respond quickly enough and the reporter will disappear
    - He found a workaround
    - He went back to an old kernel
    - He obtained new hardware
    - He switched to Windows
- Every six months or so I will send out queries to 400-500 separate emailed bug reports
  - Ask the originator to raise a bugzilla report if the problem still exists
  - Many were fixed - Many many reporters simply don't reply at all. Probably they found a workaround.

Bug reporting: shortcomings

- I think we're losing a large number of valid bug reports this way.
- Each one is a lost opportunity to improve Linux