The What, Who and How of the Linux Kernel

Vincent Sanders

What

What is the kernel?

- Provides Interface between applications and hardware.
- Services without policy.
- Requires additional “plumbing” userspace software to create complete system.
- Adds no intrinsic value to the system but without it there can be no applications.

Some current numbers

- LWN do regular articles on these numbers
- 10,246,692 Source lines of code in 3.4-rc6 (generated using David A. Wheeler's 'SLOCCount')
- 5,763,563 SLOC (56%) are drivers for hardware.
- 116,195 SLOC (1.2%) added between 3.2 and 3.3 release
- Over 1200 active contributors in 3.4 release
- Tens of thousands of contributors overall

Some context to those numbers

- The RedHat commercial Linux based OS has around 200 million SLOC.
- The Debian OS has circa 400 million SLOC.
- Kernel represents less than 5% of the OS
- One study mentions that most individual users actually run less than 10% of the kernel code.

Quick History

- Started by Linus Torvalds in 1991
- One of the Largest GPL v2 projects
- Hit human scaling issues with Torvalds showed up by the 2.4 to 2.6 development cycle.
- Initially Bitkeeper and then Git DVCS allowed scaling issues to be overcome.
- Moved to current rolling stable release process in 2005

Why would anyone change the kernel?

- Fix issues
- Add new drivers
- Extend existing interfaces
- Create new interfaces

Fixing bugs

- Most common operation
- Often submitted while using system for other purpose.
- Found while doing other kernel work
- Fixes submitted completely independently of other work.

Adding drivers

- The most common and least technically challenging operation is to add a device driver
- A device driver is simply a way to interface a physical peripheral into the kernel API.
- The kernel provides common API and infrastructure for many types of device (Networking, Audio, USB etc.)
- Using the kernel infrastructure is important for mainline acceptance (Android issues).

Extending and creating Interfaces

- Uncommon operation – the kernel already has interfaces for just about every driver need.
- Expensive in terms of resource and time
- Getting merged mainline may involve becoming maintainer for that new interface.
- Generally requires long term commitment to support the interface.

Who

The People

Linus Torvalds

Andrew Morton | Greg K H | David Miller

Lieutenants

Al Viro | Thomas Gleixner

Russell King | Ben Hutchings

David Brown | Dave Woodhouse | Matthew Garrett

Maintainers

Ben Dooks | Lennert Buytenhek

Vincent Sanders | Richard Purdie

Sjoerd Simons

Alban Crequy

Javier Martinez Canillas

Contributors

Interacting

- Universally with Email
- Some cliques communicate via private IRC
- Linux Kernel Mailing List (LKML) is a firehose (hundreds of messages per day)
- Some areas like networking have their own lists
- Adversarial to the point of offensive behaviour
- Some subsystems harder to work in than others
- Many maintainers concentrate only on their use cases.

Interacting Well

- Follow the best practice for interaction
  - **NEVER** use HTML mail
  - NEVER top post
  - NEVER private post
  - NEVER disparage the maintainer (even under provocation)
  - ALWAYS be concise while not omitting anything important.
  - ALWAYS ensure you have done as much research as possible and show it!

Truths about the community

- The social issues are often more problematic than the technical.
- Maintainers can often hold an opinion which will be contrary to the submitters.
- Submitting effectively invites a great deal of discussion, ultimately the maintainers opinion is the focus.
- The commercial politics and back biting especially in the directly competitive areas like ARM SoC and Networking drivers may be fierce.

How

The Release cycle

- Stable release every 10 to 12 weeks.
- Monotonically incrementing point version release numbers 3.4, 3.4, 3.5 etc.
- Once a release is made the “merge window” opens for around 10 days.
- All new work must have been submitted before the “merge window” to be considered.

The Release cycle

- Once the “merge window” closes the first release candidate (rc) for the release is made.
- Release candidates are produced every 7 - 10 days while stabilisation changes are merged.
- In parallel with stabilisation new changes are proposed to maintainers ready for acceptance during the next “merge window”

Ideal Submission process

- The feature is developed until it is minimally complete.
- The changes are generated as a series of patches.
- Patches submitted to appropriate public forum.
- Changes updated with feedback from forum.
- Patch series submitted to appropriate maintainer.
- Maintainer merges changes into kernel

Common issues with the ideal submission process.

- Changes are not of a high enough standard.
- Changes not implemented as the maintainer would like.
- The feature is controversial within the community.
- There is no maintainer for the area the feature applies to.
- Process failures by the maintainer.

Questions?