EMBEDDED
LINUX
CONFERENCE

The End of the Paved Road

Maintaining Linux Kernel 4.4 beyond LTS

Ulrich Hecht, Civil Infrastructure Platform Project

About this speaker

- Linux professional since 1998
- Kernel developer since 2013
- With the CIP Project since 2021

CIP? LTS? SLTS?

CIP: Civil Infrastructure Platform

- kernel and userspace for civil infrastructure applications

LTS: Long-Term Support kernel

- stable kernel branch maintained for a couple of years
- used by Android, Debian and others

SLTS: Super Long-Term Support kernel

- maintained for as long as it takes
- 10 years minimum

EMBEDDED
LINUX
CONFERENCE

Old Kernels in the Eyes of the Public

Maintaining a gerontic vendor kernel with a billion backported patches until the end of times? Sounds exciting (not really).

an embedded Linux developer

asked to join the CIP kernel maintenance team

It costs more money and time the older the kernel is to keep it "alive". It is cheaper and easier to use more modern kernels.

a kernel maintainer responding
to the announcement of the 4.4 SLTS kernel

EMBEDDED
LINUX
CONFERENCE

Timescales

The Linux kernel developer's timescale:

- Now: Where we live. Don't look back, the past is the worst.
- 12 years ago: antiquity (mainline kernel 2.6.39)
- 32 years ago: the beginning of time (kernel 0.01)

A good match for product developers thinking in similar timescales. (mobile phones and things like that)

EMBEDDED
LINUX
CONFERENCE

The Civil Infrastructure Timescale

The Civil Infrastructure timescale: The protagonists

Example: the CIP Project members

- There are currently nine members.
- All of them are corporations.
- Four have been around for more than a century.

These entities operate on super-human timescales.

The Stuff

By Hegor - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=11072148

The Civil Infrastructure timescale: The stuff

- Jaruga Hydroelectric Power Station:
  - Jaruga 2 built in 1903
  - still online in 2023 (**120 years** later)
  - no fundamental changes
  - refurbished about every 25 years
  - *not an outlier*

By Hegor - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=11072148

Civil infrastructure operates on super-human timescales.

EMBEDDED
LINUX
CONFERENCE

The Perfect Civil Infrastructure Kernel

The perfect civil infrastructure kernel:

- No regressions!
- Fix the bad stuff.
- Keep the same codebase as long as possible.
  - in major updates, regressions are *inevitable*.
  - recertification might be necessary **zOMG!!1! RUN!!**

But how?

EMBEDDED
LINUX
CONFERENCE

The SLTS Approach

Limiting the scope

SLTS is **not a general-purpose kernel**:

- limited to specific use cases
- focuses on CIP members' needs
- We know their platforms, we know their kernel configurations.

But SLTS is **not a proprietary kernel** either:

- We maintain the other stuff as well (best effort).
- We accept outside contributions.

Minimizing changes

- If it’s not trivial or important, it’s out.
  - Backporting may introduce bugs.
  - the more complicated the patch, the more likely that it will
- Good news: Most patches for stable kernels are small in scope.
  - unhandled corner cases (OOM, probe failure handling, ...)
  - thinkos (wrong return value, inverted conditions, ...)

The Process

The Process: Ingesting patches

- take in patches from nearest maintained LTS tree (4.14, ATM)
- A tool dredges new patches and tries to apply them to 4.4.
  - Whatever does not apply gets left out for now.
  - produces a list that we append to a log file (*v4.4.org*)

v4.4.org is a file that documents every patch that we do or do not include in the 4.4 SLTS kernel, including the "how" and "why".

(https://gitlab.com/cip-project/cip-kernel/lts-commit-list/-/raw/master/v4.4.org)

The Process: Reviewing ingested patches

- Review patches that fail to apply cleanly.
  - Most are not applicable at all.
  - Backport the easy ones that are not likely to break stuff.
  - Backport the important ones.
- Review patches that *don't* fail to apply.

  Sometimes a patch applies even though it shouldn't, or it is not necessary.

The Process: Testing and reviewing backports

- Run tests.
  - local compile tests
  - linux-cip CI
- Push a release candidate tree.
- Review backports via cip-dev mailing list.

The Process: Release

There are four flavors of the SLTS kernel.

- Once reviews are done and tests pass, release **4.4-st**, the "vanilla" SLTS kernel.
- Merge 4.4-st into the **4.4-cip** kernel tree, the SLTS kernel that includes CIP member patches, and release that.
- Update, merge and release the real-time branches **4.4-st-rt** and **4.4-cip-rt**.

The Point

The Point:

- "Cheap and easy" is not what is needed.
- Long-term stability "until the end of times" is.
- We are willing and able to provide it.

If you start your project with 6.1 SLTS today, we will keep it running.

For as long as it takes.

Thank you for listening!