Moving forward with Rust in V4L2

Daniel Almeida
Consultant Software Engineer, Collabora
daniel.almeida@collabora.com

What has been done so far?

What we have so far

- Abstractions for some V4L2 data types
- A *very* thin videobuf2 abstraction (you can create a queue)
- Abstractions for some VIDIOC_* ioctls
- The necessary code to get the driver to probe
- A module that prints to the terminal when processing some of the VIDIOC_* ioctls

Why we should experiment with Rust in V4L2?

Why Rust?

- V4L2 takes in *a lot* of untrusted data from userland
- Rust can help mitigate this problem at compile time
- There are low-risk components to experiment with
- This gives maintainers the time to evaluate whether Rust works for the subsystem

I discussed this topic during the Media Summit 2023

Roadblocks and feedback

Roadblocks and feedback

- V4L2 can’t keep up with the workload as is

Roadblocks and feedback

- V4L2 can’t keep up with the workload as is
- Not enough reviews and maintainers

Roadblocks and feedback

- V4L2 can’t keep up with the workload as is
- Not enough reviews and maintainers
- There are long-standing issues with some C frameworks (e.g. media controller lifetime issues)

Roadblocks and feedback

- V4L2 can’t keep up with the workload as is
- Not enough reviews and maintainers
- There are long-standing issues with some C frameworks (e.g. media controller lifetime issues)
- Huge fear of breaking existing C code

Roadblocks and feedback

- V4L2 can’t keep up with the workload as is
- Not enough reviews and maintainers
- There are long-standing issues with some C frameworks (e.g. media controller lifetime issues)
- Huge fear of breaking existing C code
- **There must be more contributors working on this**

We want to unblock this effort

Which is why we are proposing a virtual stateless codec driver in Rust

We already have a similar driver in C

Virtual stateless codec driver

- Requires adding Rust abstractions for important components in V4L2
- Not much more effort to write a driver for real hardware!
- Shows the community how a V4L2 Rust driver will look like
- This is not a critical component, just another driver
- **Maintainers will have actual code to judge**

Eventually, the Rust driver can replace visl

Open questions

- What happens if the C API is changed and it breaks the Rust bindings? Can we detect automatically?
    - Maybe CONFIG_MEDIA_EXPERIMENTAL to buy time?
- Collabora-maintained branch? Is this helpful?
- Anything else we can do to drive this?

Feedback?

Thank you!