Rust for Linux: an Overview

whoami

- Anisse Astier
- @Aissen@treehouse.systems
- https://anisse.astier.eu
- Kernel Recipes live blogger

Disclaimer

- Not a Rust for Linux contributor
- Watch Miguel Ojeda's presentation

Rust for Linux ?

- Meta-project, like the kernel
- Goal : Rust as a second language
- Upstream as a core goal

History (condensed)

- rust.ko by Taesoo Kim in 2013
- LSS talk by Thomas & Gaynor in 2019
- LPC discussion in 2020
- Project announcement by Ojeda et al in 2021
- Experiment merged into Linux 6.1 in 2022

Why Rust ?

- Usual tradeoff, pick two :
  - Dynamic Memory Allocations
  - Memory Safety
  - No GC and can be freerunning (native speed)

Why Rust ?

- ~~Usual tradeoff, pick two~~ Get all three:
  - Dynamic Memory Allocations
  - Memory Safety
  - No GC and can be freerunning (native speed)

Why Rust ?

- Quoting Lyude Paul, kernel developer
  - “[Memory safety] is the least convincing point.[…] We know C is unsafe, we’ve been writing it for decades”
- Ergonomics “Rust can be a kernel maintainer”
  - Enums and pattern matching
  - Traits and hygienic macros
  - Ownership and lifetimes
  - ...

Why not Rust?

- Learning takes time
- Architecture support in rustc
- Multiple toolchains: GCC vs LLVM
  - gccrs is progressing
  - so is rust_codegen_gcc

Rust for Linux strategy

- Docs everywere, including `///` SAFETY comments
- Use bindgen for FFI layer generation
- Safe abstractions on top of bindings, no direct call to bindings

Rust safe abstractions

- Need thorough analysis of C layer to understand constraints
- Not trivial to do a fully safe abstraction
- Depends on specific kernel domain

Rust abstractions in Linux

- Device / MiscDevice / platform::Device
- Alloc
- Workqueue
- PidNamespace
- ... (list is quite long, yet too short)

Rust drivers

- No-duplicate rule, except for reference drivers
- Merged upstream
  - Null block driver reimplementation
  - drm_panic qrcode generator (Linux BSOD)
  - Two phy drivers : ax88796b and qt2025

Upcoming and existing drivers

- Asahi Linux Apple GPU DRM driver
- Android binder and ashmem reimplementations
- NVME reimplementation
- Nova driver for NVIDIA GPUs
- And many others : tarfs, erofs, puzzlefs, i2c regulator driver, display interface…

Recent updates

- Rust minimum version (no longer fixed)
- Kernel now has its own alloc module and associated types KVec, KBox
- Rust project now builds the Linux kernel in CI to prevent breakage
- RFL is a Rust flagship goal in 2024H2 and 2025H1

Community outlook

- Positive outlook from Maintainers Summit
- Many supportive maintainers
- Linus said “steam right ahead”
- Supporting Rust is still optional… or is it?

Code examples

C vs Rust

You can't forget to clean up

In C

```
err_translate_failed:
err_bad_object_type:
err_bad_offset:
err_bad_parent:
err_copy_data_failed:
        binder_cleanup_deferred_txn_lists(&sgc_head, &pf_head);
        binder_free_txn_fixups(t);
        trace_binder_transaction_failed_buffer_release(t->buffer);
        binder_transaction_buffer_release(target_proc, NULL, t->buffer,
                                          buffer_offset, true);
        if (target_node)
                binder_dec_node_tmpref(target_node);
        target_node = NULL;
        t->buffer->transaction = NULL;
        binder_alloc_free_buf(&target_proc->alloc, t->buffer);
err_binder_alloc_buf_failed:
err_bad_extra_size:
        if (secctx)
                security_release_secctx(secctx, secctx_sz);
err_get_secctx_failed:
        kfree(tcomplete);
        binder_stats_deleted(BINDER_STAT_TRANSACTION_COMPLETE);
err_alloc_tcomplete_failed:
        if (trace_binder_txn_latency_free_enabled())
                binder_txn_latency_free(t);
        kfree(t);
        binder_stats_deleted(BINDER_STAT_TRANSACTION);
err_alloc_t_failed:
err_bad_todo_list:
err_bad_call_stack:
err_empty_call_stack:
err_dead_binder:
err_invalid_target_handle:
        /* it keeps going ... */
```

In Rust

```
}
```

Alice Rhyl and Carlos Llamas on binder rewrite

Minimal Rust driver

```
use kernel::prelude::*;

module! {
    type: RustMinimal,
    name: "rust_minimal",
    author: "Rust for Linux Contributors",
    description: "Rust minimal sample",
    license: "GPL",
}
```

Minimal Rust driver

```
module! {
    type: RustMinimal,
    name: "rust_minimal",
    author: "Rust for Linux Contributors",
    description: "Rust minimal sample",
    license: "GPL",
}

struct RustMinimal {
    numbers: KVec<i32>,
}
```

Minimal Rust driver

```
impl kernel::Module for RustMinimal {
    fn init(_module: &'static ThisModule) -> Result<Self> {
        pr_info!("Rust minimal sample (init)\n");

        let mut numbers = KVec::new();
        numbers.push(72, GFP_KERNEL)?;
        numbers.push(200, GFP_KERNEL)?;

        Ok(RustMinimal { numbers })
    }
}
```

Minimal Rust driver

```
impl Drop for RustMinimal {
    fn drop(&mut self) {
        pr_info!("My numbers are {:?}\n", self.numbers);
        pr_info!("Rust minimal sample (exit)\n");
    }
}
```

Macros for firmware abstraction

```
#[versions(AGX)]
#[repr(C)]
pub(crate) struct RunFragment {
    pub(crate) tag: u32,

    #[ver(V >= V13_0B4)]
    pub(crate) counter: U64,
```

Macros for firmware abstraction

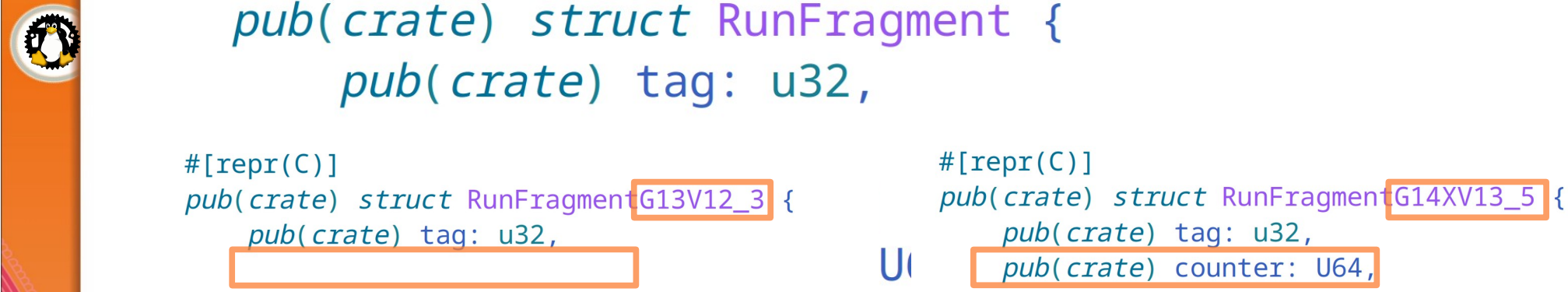
```
static AGX_VERSIONS: VersionConfig = VersionConfig {
    fields: &["G", "V"],
    enums: &[
        &["G13", "G14", "G14X"],
        &["V12_3", "V12_4", "V13_0B4", "V13_2", "V13_3", "V13_5"],
    ],
    versions: &[
        &["G13", "V12_3"],
        &["G14", "V12_4"],
        &["G13", "V13_5"],
        &["G14", "V13_5"],
        &["G14X", "V13_5"],
    ],
};
```

Macros for firmware abstraction

```
#[versions(AGX)]
#[repr(C)]
pub(crate) struct RunFragment {
    pub(crate) tag: u32,

    #[ver(V >= V13_0B4)]
    pub(crate) counter: U64,
```

Macros for firmware abstraction



```
#[versions(AGX)]
#[repr(C)]
pub(crate) struct RunFragment {
    pub(crate) tag: u32,
```

```
#[repr(C)]
pub(crate) struct RunFragmentG13V12_3 {
    pub(crate) tag: u32,
```

```
#[repr(C)]
pub(crate) struct RunFragmentG14XV13_5 {
    pub(crate) tag: u32,
    pub(crate) counter: U64,
```

Future ? No deadlocks

- Wait, Rust does not protect against deadlocks !
- …
- Or does it ?
- “Safety in an Unsafe World” talk by Joshua Liebow-Feeser
- Mutex Direct Acyclic Graph
  - Mutex acquire order verified at compile-time
  - Works for Fuschia Netstack3’s 77 mutexes

Thank you!

Questions?