Contribution ID: **15** Type: **not specified**

Rust MC

CFP closes on July 14th.

Rust is a systems programming language that is making great strides in becoming the next big one in the domain. Rust for Linux is the project adding support for the Rust language to the Linux kernel.

Rust has a key property that makes it very interesting as the second language in the kernel: it guarantees no undefined behavior takes place (as long as unsafe code is sound). This includes no use-after-free mistakes, no double frees, no data races, etc. It also provides other important benefits, such as improved error handling, stricter typing, sum types, pattern matching, privacy, closures, generics, etc.

This microconference intends to cover talks and discussions on both Rust for Linux as well as other non-kernel Rust topics.

Possible Rust for Linux topics:

- Rust in the kernel (e.g. status update, next steps…).
- Use cases for Rust around the kernel (e.g. subsystems, drivers, other modules…).
- Discussions on how to abstract existing subsystems safely, on API design, on coding guidelines…
- Integration with kernel systems and other infrastructure (e.g. build system, documentation, testing and CIs, maintenance, unstable features, architecture support, stable/LTS releases, Rust versioning, third-party crates…).
- Updates on its subprojects (e.g. klint, pinned-init…).
- Rust versioning requirements and using Linux distributions' toolchains.

Possible Rust topics:

- Language and standard library (e.g. upcoming features, stabilization of the remaining features the kernel needs, memory model…).
- Compilers and codegen (e.g. rustc improvements, LLVM and Rust, rustc_codegen_gcc, gccrs…).
- Other tooling and new ideas (Coccinelle for Rust, bindgen, Compiler Explorer, Cargo, Clippy, Miri…).
- Educational material.
- Any other Rust topic within the Linux ecosystem.

Last year was the second edition of the Rust MC and the focus was on presenting and discussing the ongoing efforts by different parties that are using and upstreaming new Rust abstractions and drivers (Using Rust in the binder driver, Block Layer Rust API, Rust in V4L2: a status report and Converting a DRM driver to Rust) as well as those that are improving the ergonomics and tooling around it (Klint: Compile-time Detection of Atomic Context Violations for Kernel Rust Code, pin-init: Solving Address Stability in Rust and Coccinelle for Rust).

Since the MC last year, there has been continued progress from users (e.g. the Android Binder Driver getting closer to upstreaming all its dependencies) as well as new project announcements (e.g. Nova), the first Rust reference driver merged together with its abstractions (the Rust Asix PHY driver), Rust support for new architectures mainlined (LoongArch and arm64)…

Primary authors: OJEDA, Miguel; ALMEIDA FILHO, Wedson

Presenters: OJEDA, Miguel; ALMEIDA FILHO, Wedson

Track Classification: LPC Microconference Proposals