Rust for Linux

Miguel Ojeda
Wedson Almeida Filho
Agenda

Status update
   Community & team
   Industry support
   Kangrejos
   Tools
   Users

Upstreaming

Discussion topics
Status update
Growing Community

~460 subscribers in the **rust-for-linux** mailing list.

From ~340 last year.

Similar to the BPF and **linux-rt-users** lists.

Growing Community

The Zulip instance (i.e. chat) is growing too: ~530 users now!

---

https://rust-for-linux.zulipchat.com/stats
Growing Community

The Zulip instance (i.e. chat) is growing too: ~70 regulars

---

https://rust-for-linux.zulipchat.com/stats
Growing Community

8000+ messages in the last year, which represent 70% of the total.

---

https://rust-for-linux.zulipchat.com/stats
Growing Core Team

RUST
M:    Miguel Ojeda <ojeda@kernel.org>
M:    Alex Gaynor <alex.gaynor@gmail.com>
M:    Wedson Almeida Filho <wedsonaf@gmail.com>
R:    Boqun Feng <boqun.feng@gmail.com>
R:    Gary Guo <gary@garyguo.net>
R:    Björn Roy Baron <bjorn3_qh@protonmail.com>
R:    Benno Lossin <benno.lossin@proton.me>
R:    Andreas Hindborg <a.hindborg@samsung.com>
R:    Alice Ryhl <aliceryhl@google.com>
L:    rust-for-linux@vger.kernel.org
S:    Supported
W:    https://rust-for-linux.com
B:    https://github.com/Rust-for-Linux/linux/issues
C:    zulip://rust-for-linux.zulipchat.com
P:    https://rust-for-linux.com/contributing
T:    git https://github.com/Rust-for-Linux/linux.git rust-next
F:    Documentation/rust/
F:    rust/
F:    samples/rust/
F:    scripts/*rust*
K:    \b(?i:rust)\b
Growing Core Team

MAINTAINERS: add Benno Lossin as Rust reviewer

Benno has been involved with the Rust for Linux project for the better part of a year now. He has been working on solving the safe pinned initialization problem [1], which resulted in the pin-init API patch series [2] that allows to reduce the need for `unsafe` code in the kernel. He is also working on the field projection RFC for Rust [3] to bring pin-init as a language feature.

His expertise with the language will be very useful to have around in the future if Rust grows within the kernel, thus add him to the `RUST` entry as reviewer.

— Commit b0cf5d50210d (“MAINTAINERS: add Benno Lossin as Rust reviewer”)
Growing Core Team

MAINTAINERS: add **Andreas Hindborg** as Rust reviewer

Andreas has been involved with the Rust for Linux project for more than a year now. He has been primarily working on the **Rust NVMe driver** [1], presenting it in several places (such as LPC [2][3] and Kangrejos [4]).

In addition, he recently submitted the **Rust null block driver** [5] and has been reviewing patches in the mailing list for some months.

Thus add him to the `RUST` entry as reviewer.

— Commit 2a6f5df3cd94 (“MAINTAINERS: add Andreas Hindborg as Rust reviewer”)
Growing Core Team

MAINTAINERS: add Alice Ryhl as Rust reviewer

Alice has been involved with the Rust for Linux project for almost a year now. She has been primarily working on the Android Binder Driver [1].

In addition, she has been reviewing patches in the mailing list for some months and has submitted improvements to the core Rust support.

She is also part of the core maintainer team for the widely used library Tokio [2], an asynchronous Rust runtime.

Her expertise with the language will be very useful to have around in the future if Rust grows within the kernel, thus add her to the `RUST` entry as reviewer.

— Commit d4d84eaa3f39 (“MAINTAINERS: add Alice Ryhl as Rust reviewer”)
Patch series submitters

Aakash Sen Sharma <aakashsensharma@gmail.com>
Alexander Pantyukhin <apantykhin@gmail.com>
Alice Ryhl <aliceryhl@google.com>
Andrea Righi <andrea.righi@canonical.com>
Andreas Hindborg <nmi@metaspace.dk>
Ariel Miculas <amiculas@cisco.com>
Asahi Lina <alina@asahilina.net>
Bagas Sanjaya <bagasdotme@gmail.com>
Ben Gooding <ben.gooding.dev@gmail.com>
Benno Lossin <benno.lossin@proton.me>
Björn Roy Baron <bjorn3_gh@protonmail.com>
Boqun Feng <boqun.feng@gmail.com>
Carlos Bilbao <carlos.bilbao@amd.com>
Conor Dooley <conor.dooley@microchip.com>
Costa Shulyupin <costa.shul@redhat.com>
Daniel Almeida <daniel.almeida@collabora.com>
David Gow <davidgow@gmail.com>
David Rheinsberg <david@readahead.eu>
Ethan D. Twardy <ethan.twardy@gmail.com>
Finn Behrens <fin@nyantec.com>
FUJITA Tomonori <tomo@exabit.dev>
Gary Guo <gary@garyguo.net>
Guillaume Plourde <gplourde@protonmail.com>
Jamie Cunliffe <Jamie.Cunliffe@arm.com>
Jiapeng Chong <jiapeng.chong@linux.alibaba.com>
Maíra Canal <mcanal@igalia.com>

Martin Rodriguez Reboredo <yakoyoku@gmail.com>
Masahiro Yamada <masahiroy@kernel.org>
Matteo Croce <teknoraver@meta.com>
Matthew Leach <dev@mattleach.net>
Matthew Maurer <mmaurer@google.com>
Michael Ellerman <mpe@ellerman.id.au>
Michele Dalle Rive <dallerivmichele@gmail.com>
Miguel Ojeda <ojeda@kernel.org>
Nick Desaulniers <nick.desaulniers@gmail.com>
Olof Johansson <olof@lixom.net>
Paran Lee <p4ranlee@gmail.com>
Patrick Blass <patrickblass@mailbox.org>
Qingsong Chen <changxian.cgs@antgroup.com>
Roy Matero <materoy@proton.me>
SeongJae Park <sj@kernel.org>
Thomas Bamelis <thomas@bamelis.dev>
Timo Grautstück <timogr@hotmail.de>
Trevor Gross <tgross@umich.edu>
TruongSinh Tran-Nguyen <i@truongsinh.pro>
Vinay Varma <varmavinaym@gmail.com>
Vincenzo Palazzo <vincenzo@palazzodesv@gmail.com>
WANG Rui <wangrui@loongson.cn>
Wedson Almeida Filho <wedsonaf@gmail.com>
Wei Liu <weiliu@kernel.org>
Wu XiangCheng <bobwx@gmail.cn>
Yang Yingliang <yangyingliang@huawei.com>
Yanteng Si <siyanteng@loongson.cn>
Maintainers getting involved

KUnit maintainers got Rust files in their MAINTAINERS entry.

MAINTAINERS: add Rust KUnit files to the KUnit entry

The KUnit maintainers would like to maintain these files on their side too (thanks!), so add them to their entry.

With this in place, `scripts/get_maintainer.pl` prints both sets of maintainers/reviewers (i.e. KUnit and Rust) for those files, which is the behavior we are looking for.

KERNEL UNIT TESTING FRAMEWORK (KUnit)
M: Brendan Higgins <brendanhiggins@google.com>
M: David Gow <davidgow@google.com>
...
F: lib/kunit/
+F: rust/kernel/kunit.rs
+F: scripts/rustdoc_test_*

— Commit 64bd4641310c ("MAINTAINERS: add Rust KUnit files to the KUnit entry")
Maintainers getting involved

Matthew Wilcox is willing to keep the Rust and C sides in sync:

I'm happy to commit to keeping the Rust implementation updated as I modify the C implementation of folios, but I appreciate that other maintainers may not be willing to make such a commitment.

— https://lore.kernel.org/rust-for-linux/ZTaDFe%2Fs2wvyl9u2@casper.infradead.org/
Sponsors & Industry support

- https://rust-for-linux.com/sponsors
- https://rust-for-linux.com/industry-and-academia-support
- https://www.memorysafety.org/initiative/linux-kernel/
- https://www.memorysafety.org/blog/rustls-and-rust-for-linux-funding-openssf/
Statements of support

“Being able to use Rust in the Linux kernel is an incredible milestone on the road to a more secure future for the Internet and everything else that depends heavily on Linux.”

— https://www.memorysafety.org/blog/rust-in-linux-just-the-beginning/
— https://rust-for-linux.com/industry-and-academia-support#ISRG
“Samsung is actively engaged in supporting the integration of Rust code into the Linux Kernel. Recognizing the significant benefits that Rust brings to kernel and system software development, particularly in terms of enhancing security and reducing critical bugs, Samsung is committed to enabling kernel developers to write block layer device drivers using the Rust programming language. By embracing modern programming languages like Rust, Samsung aims to attract new talent to systems development and promote memory safety within the Linux storage stack.”

---

[https://rust-for-linux.com/industry-and-academia-support#Samsung](https://rust-for-linux.com/industry-and-academia-support#Samsung)
“Cisco supports the inclusion and development of Rust in the Linux kernel as a way of eliminating memory safety bugs and vulnerabilities. We are developing a next-generation container filesystem in Rust and, to this end, we are contributing time, code, and the testing effort to the Rust for Linux project.”

— https://rust-for-linux.com/industry-and-academia-support#Cisco
Statements of support

“Collabora feels privileged to partner with customers who envision Rust as an integral part of the Linux kernel’s future. We are committed to supporting the integration of Rust into as many Linux subsystems as appropriate over the coming years. By doing so, this will enable our customers, and many more developers, to increase the reliability of their Linux kernel contributions. We extend our gratitude for the activities undertaken by the Rust for Linux Initiative.”

— https://rust-for-linux.com/industry-and-academia-support#Collabora
Rust support in the Ubuntu kernel

Using Rust, you can easily create your own kernel modules and share them with other Ubuntu users, without the need of any special toolchain or kernel requirements.

The generic kernel in Ubuntu already contains the Rust subsystem that is capable of running Rust modules.

From a user-space perspective developers just need to install the toolchain packages required to build kernel modules in Rust:

```
$ sudo apt install rustc-1.62 rust-1.62-src rustfmt-1.62  
bindgen-0.56 llvm clang gcc make  
\   
linux-lib-rust-$\{(uname -r)\} linux-headers-$\{(uname -r)\}
```

Distributing Rust kernel modules is also easy with Ubuntu, any Ubuntu user can recompile and load binary modules (.ko) directly into the generic kernel shipped with the distribution, like any other regular kernel module.

---

Kangrejos

The Rust for Linux Workshop

An event where people involved in the Rust for Linux discussions can meet in a single place before LPC.

https://kangrejos.com

https://lwn.net/Archives/ConferenceIndex/
#Kangrejos
The new website

rust-for-linux.com
The new website

Documentation and resources

---

https://rust-for-linux.com
The new website

Documentation and resources

Subprojects

— https://rust-for-linux.com
The new website

Documentation and resources

Subprojects

Tools

— https://rust-for-linux.com
The new website

Documentation and resources

Subprojects

Tools

Users

— https://rust-for-linux.com
The new website

Documentation and resources

Subprojects

Tools

Users

External links

— https://rust-for-linux.com
Coccinelle for Rust

Coccinelle is a tool for automatic program matching and transformation that was originally developed for making large scale changes to the Linux kernel source code (i.e., C code). Matches and transformations are driven by user-specific transformation rules having the form of abstracted patches, referred to as semantic patches. As the Linux kernel, and systems software more generally, is starting to adopt Rust, we are developing Coccinelle for Rust, to make the power of Coccinelle available to Rust codebases.

Examples

Changing a method call sequence in the Rust implementation:

```rust
eexpression tcx, arg;
- tcx.type_of(arg)
+ tcx.bound_type_of(arg).subst_identity()
```

Merging some lifetimes in tokio:

```rust
identifier f, P, p;
type T1, T2;
- f<P: T1>(p: P) -> T2
+ f(p: Impl T1) -> T2
    [...]```

Current status

Coccinelle for Rust is still in development. For more information, see the project website.
Results

![Graph showing read throughput comparison between C and Rust for different queue depths.](image)

- 12th Gen Intel(R) Core(TM) i7-1260P
- 32 GB DRAM
- 1x INTEL MEMPEK1W016GA (PCIe 3.0 x2)
- Debian Bullseye userspace
Tools

**rustc_codegen_gcc** — Antoni Boucher

Compiles & QEMU-boots mainline without source changes.

[https://github.com/rust-lang/rustc_codegen_gcc](https://github.com/rust-lang/rustc_codegen_gcc)

**GCC Rust (gccrs)** — Arthur Cohen, Philip Herron


[https://github.com/Rust-GCC/gccrs](https://github.com/Rust-GCC/gccrs)

**Coccinelle for Rust** — Julia Lawall, Tathagata Roy

Recently published.

[https://gitlab.inria.fr/coccinelle/coccinelleforrust](https://gitlab.inria.fr/coccinelle/coccinelleforrust)
Tools

rustc_codegen_gcc — Antoni Boucher
  Compiles & QEMU-boots mainline without source changes.
  https://github.com/rust-lang/rustc_codegen_gcc

GCC Rust (gccrs) — Arthur Cohen, Philip Herron
  https://github.com/Rust-GCC/gccrs
Tools

**rustc_codegen_gcc** — Antoni Boucher
Compiles & QEMU-boots mainline without source changes.
https://github.com/rust-lang/rustc_codegen_gcc

**GCC Rust (gccrs)** — Arthur Cohen, Philip Herron
https://github.com/Rust-GCC/gccrs

**Coccinelle for Rust** — Julia Lawall, Tathagata Roy
Recently published.
https://gitlab.inria.fr/coccinelle/coccinelleforrust
Tools

**rustc_codegen_gcc** — Antoni Boucher
Compiles & QEMU-boots mainline without source changes.
[https://github.com/rust-lang/rustc_codegen_gcc](https://github.com/rust-lang/rustc_codegen_gcc)

**GCC Rust (gccrs)** — Arthur Cohen, Philip Herron
[https://github.com/Rust-GCC/gccrs](https://github.com/Rust-GCC/gccrs)

**Coccinelle for Rust** — Julia Lawall, Tathagata Roy
Recently published.
[https://gitlab.inria.fr/coccinelle/coccinelleforrust](https://gitlab.inria.fr/coccinelle/coccinelleforrust)

See Julia’s talk at the Rust MC on Wednesday!
rustc_codegen_gcc

```
/tmp $ git clone https://github.com/Rust-for-Linux/linux/ --single-branch --branch=rust-next
Cloning into 'linux'...
remote: Enumerating objects: 9716732, done.
remote: Counting objects: 100% (629/629), done.
remote: Compressing objects: 100% (629/629), done.
remote: Total 9716732 (delta 0), reused 629 (delta 0), pack-reused 9716183
Receiving objects: 100% (9716732/9716732), 4.49 GB | 3.86 MB/s, done.
Resolving deltas: 100% (7944120/7944120), done.
Updating files: 100% (81757/81757), done.
```

```
/arch $(/usr/bin/detect_arch)  
/extra  
/block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt  
```

```
# arch $(/usr/bin/detect_arch)  
#
```

```
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

```
# arch $(/usr/bin/detect_arch)  
#
```

```
/config scripts/basic/fixed.o  
/config scripts/kconfig/conf.o  
/config scripts/kconfig/confdata.o  
/config scripts/kconfig/exp.o  
/config scripts/kconfig/lexer.lex.c  
/config scripts/kconfig/parser.tab.[ch]  
/config scripts/kconfig/lexer.lex.o  
/config scripts/kconfig/menu.o  
/config scripts/kconfig/parser.tab.o  
/config scripts/kconfig/preprocess.o  
/config scripts/kconfig/symbol.o  
/config scripts/kconfig/util.o  
/config scripts/kconfig/conf
```

```
** Default configuration is based on `x86_64_defconfig'
```

```
# configuration written to .config
```

```
# make menuconfig
```

```
# arch $(/usr/bin/detect_arch)  
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

```
# arch $(/usr/bin/detect_arch)  
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

```
# arch $(/usr/bin/detect_arch)  
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

```
# arch $(/usr/bin/detect_arch)  
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

```
# arch $(/usr/bin/detect_arch)  
# extra  
# block COPYING crypto drivers include io_uring Kbuild kernel LICENSES Makefile mm README samples security tools virt
```

---

Antoni Boucher
rustc_codegen_gcc

— Antoni Boucher
— Antoni Boucher
File: samples/rust/rust_minimal.o

Hex dump of section '.comment':
0x00000000 00727573 74632076 65727369 6f6e2031 .rustc version 1
0x00000010 2e373352 e302d6e69 6768746c 79202832 .75.0.-nightly (2
0x00000020 34393632 34623530 20323032 332d3130 49624b50 2023-10
0x00000030 2d323029 20776974 68206c69 62676363 -20) with libgcc

— Antoni Boucher
GCC Rust (gccrs)

Kangrejos 2023 status report

Upstreaming in GCC 13.1

- The compiler was not released since it was still missing some basic features
- We spent a lot of time sending in patches and getting them approved, and gccrs is now a full part of GCC
- This makes our work much easier for the planned first release of gccrs within GCC, which should happen with GCC 14.1

— Arthur Cohen
GCC Rust (gccrs)

Technical side of things

- Macro name resolution
- Fixes to macro expansion to properly handle eager expansion of builtin macros
- Fixed point expansion and name resolution algorithm
- Implementation of derive macros framework, support for Clone and Copy
- Closures support
- Iterators support
- Binding associated types (`core::ops::Add<Output = i32>>`)`
- Procedural macros are almost completely implemented
- Unicode support
- Support for `Fn` traits
- Integration of rustc error codes within the compiler
  - This will help us pass the rustc testsuite when the time comes
- Compiler intrinsics
- Complete rework of our name resolution pass
- Jakub Dupak's master thesis is about integrating polonius to gccrs, in order to have access to a borrow checker

— Arthur Cohen
GCC Rust (gccrs)

Upcoming work

- Mostly upstreaming work... which takes a very long time
- Support for `format_args!()` builtin macro
  - Kernel print macros, `println!()` in core ...

Upstreaming in GCC 14.1

- Patch upstreaming has started again
- Sending in commits which affect common GCC parts (such as the build system)
- Once these are accepted, we will begin upstreaming all of the work we did since ~April 2023, which is around 900 commits
- We are hoping to be released as part of GCC 14.1

Talks

- FOSDEM
- GNU Cauldron (22/09/2023)
- EuroRust (13/10/2023)

— Arthur Cohen
Rust for Linux for Compiler Explorer

Already prototyped and discussed with Matt Godbolt.
   The basic setup is quite straightforward.
   A reasonable set of versions and kernel configs should be OK resource-wise.

Useful for development as well as training.

Makes it trivial to check how code is actually generated in the kernel.
   e.g. no need to remember what flags to pass.

Ideally, also providing an Executor:
   QEMU booting up a kernel.
   Having a window to write an init script.
   Useful for trainings etc.
More Compiler Explorer

Ideally, we would like to get:

- **bindgen** as a compiler (versioned)
- **rustfmt** as a compiler (versioned)
- Clippy as a compiler (versioned)
- Augmenting compiler diagnostics with hyperlinks and custom actions.
- Pre-filling flags (e.g. `--edition` for Rust) instead of the Overrides' implicit approach.
The original discussion on this started early 2021.

It has been a long time coming, but we got the OK to go ahead.

So expect the Rust generated docs to appear in that domain soon.

Per-tag access will be possible (e.g. v6.4, v6.6-rc1 and so on).

Some details are still open.

   e.g. should we have a top bar for “tag” selection?
Mitigations

-Zfunction-return support submitted to rustc.
https://github.com/rust-lang/rust/pull/116892

Patch series submitted to the kernel.
RETPOLINE, SLS, RETHUNK.
https://lore.kernel.org/rust-for-linux/20231023174449.251550-1-ojeda@kernel.org/

With both pieces, we are able to compile a kernel with objtool enabled for Rust that does not generate the corresponding warnings.
CFI and KCFI

“Working on fixing the known issues [1], but these are corner cases and hopefully shouldn't affect the Linux kernel/Rust-for-Linux.

Fixed building the standard library and its dependencies with CFI enabled.

Working on fixing CFI violations in the standard library [2][3] -- so far there are only 2 total, and by disabling CFI in these locations, all core and std tests pass.

The third violation mentioned in the GitHub issue is actually a bug in the CFI implementation I'm finishing a fix for.”

[1] https://github.com/rust-lang/rust/issues?q=is%3Aopen+label%3APG-exploit-mitigations%3ACFI

— Ramon de C Valle
Reviewers’ Recommendations

“This is a list of topics about which developers may want rules of thumb or checklists to start with. This also helps reviewers to understand the code quickly and provide useful feedbacks. Note that among all the reviewers, there is one we care most: the future yourself.

These recommendations may be incomplete, since both Rust and Linux are moving targets. In case where this document doesn’t cover, please consider the following:

- Be Rust idiomatic as hard as possible.
- Being explicit first and then improving ergonomic usually work.
- If you find a good and reasonable way for a certain problem, please do add it in this document!”

— Boqun Feng
Deprecating the `rust` branch

The `rust` branch was the original branch where development happened for two years.

We kept it synchronized with mainline (by merging Linus’ tree into it), but otherwise it did not get new features.

Recently, the latest major user (that we are aware of), the NVMe driver, got rebased on top of `rust-next`.

Thus the branch is now frozen/archived.

---

[https://rust-for-linux.com/branches](https://rust-for-linux.com/branches)
Introducing the `rust-dev` branch

A branch intended for:

- Early testing by taking patches without too much concern.
  Can also be done during the merge window.

- Easier development.

- Knowing what is in the queue.

Typically rebased on top of `rust-next` often.

Patches (that are not RFCs) should not be based on it.

Managed by Boqun Feng.

— [https://rust-for-linux.com/branches](https://rust-for-linux.com/branches)
Upstreaming
Upstreamed code

6.1: Initial merge (minimal support, Rust 1.62.0).

6.2: Opaque, Either, CString, CStr, BStr, #[vtable], concat_idsents!, {static, build}_assert!, the rest of pr_*! and more error codes, dbg!...


6.4: pinned-init API, AlwaysRefCounted, ARef, Lock, Guard, Mutex, SpinLock, CondVar, Task, uapi crate...

6.5: Rust 1.68.2 (first upgrade), pinned-init improvements, Error’s name() support, AsRef for Arc...

6.6: Rust documentation tests as KUnit tests, pinned-init features, paste!, Rust 1.71.1, bindgen 0.65.1, rust_is_available series...

6.7: Workqueue abstractions, Rust 1.73.0, toybox support (Android), x86 IBT, webpage and Maintainer Entry Profile document.

RFCs/WIP: Binder, NVMe, DRM (Apple GPU, VGEM), file systems (tarfs, PuzzleFS), PHY, V4L2 codecs...
Better ergonomics for pinned initialisation

The Linux kernel has many data structures that require stable addresses
For example, struct `list_head`, described previously [here](#)

We had no ergonomic way of initialising them in Rust
In Rust, safe initialisation happens before we know the destination address
We needed unsafe blocks for this originally

We introduced `pin-init`
Allows us to initialise pinned objects without unsafe blocks – see Benno's talk at the Rust MC on Wednesday!
Unexpected safety issue

Deadlocks are **safe** in Rust
   A deadlock doesn't result in undefined behaviour

This isn't true in the Linux kernel
   In certain configurations, a situation that should have lead to a deadlock, **leads to** user-after-free

We need to avoid sleeping in atomic context **for safety**
   Previously, we believed we only needed it for correctness

We introduced **klint**
   Static analysis to detect context violations – see Gary's talk at the Rust MC on Wednesday!
Block layer abstractions

The community suggested that we implement an NVMe driver in Rust

Andreas presented performance numbers for that in LPC last year

We wrote block layer abstractions as part of that effort

The NVMe and Null blk drivers use these abstractions

We are improving and working on upstreaming the abstractions

So that block layer drivers can be written in Rust – see Andreas' talk at the Rust MC on Wednesday!
Android Binder

We had Binder as a WIP patch in the original Rust RFC in 2021
    It is Android's driver for IPC

It was the first non-trivial Rust driver
    But the community considered it too atypical and wanted to see other drivers
    So we temporarily shifted our focus away

It is now feature complete
    It's intended to replace the C implementation – see Alice's talk at the Rust MC on Wednesday!
Virtual file system

Needed by two Rust file systems: tarfs and puzzlefs
Per recommendation, only providing abstractions for needed features

An RFC patch series was posted
Some feedback provided, working on v2

Some unsoundness still present
When unregistering file systems
Discussion topics
Soundness issues for stable

A soundness issue in Rust is a mistake that could cause otherwise safe Rust code to introduce UB.

- They may not materialize in current kernel/compiler versions.
- However, we could have concrete instances where they are a real issue, especially considering distributions and out-of-tree modules.

We want to evaluate how feasible it would be to backport these long-term.

- So far, we have backported several.
- Probably worth a mention in the stable kernel rules.
Rust version policy

We cannot guarantee newer Rust versions will work due to the unstable features in use.

The Rust language is stable, i.e. it promises backwards compatibility.

In other words, our “minimum version” is in the future.

Thus, for now, we are tracking the latest version of the Rust compiler.

Quite unusual for the kernel.

Stable backports have not been an issue so far.

Should get easier as features get stabilized and we can establish the minimum version.

— https://rust-for-linux.com/rust-version-policy
Duplicate drivers exception

A few maintainers are open to the idea of experimenting with Rust, but they may want to start simple with a driver they are familiar with.

However, such a driver would violate the "no duplicate drivers" rule.

Others have expressed an interest in writing Rust drivers, but the required abstractions are not there, and merging those would break the "no code without an expected in-tree user" rule.

Some maintainers may want to avoid a flag day, or may prefer to iterate in-tree.

For these and other reasons, we have requested an exception for Rust drivers.

— [MAINTAINERS SUMMIT] The Rust Experiment
Rust for Linux

Miguel Ojeda
Wedson Almeida Filho
Backup slides
Drivers:
- `my_foo` driver

Include:
- `bindgen` bindings crate

Linux tree:
- `kernel` crate
- `foo` subsystem
- `bar` subsystem

Safe Abstractions:
- `Safe` subsystem

Forbidden!