The Rust for Linux Kernel Report

Miguel Ojeda

ojeda@kernel.org

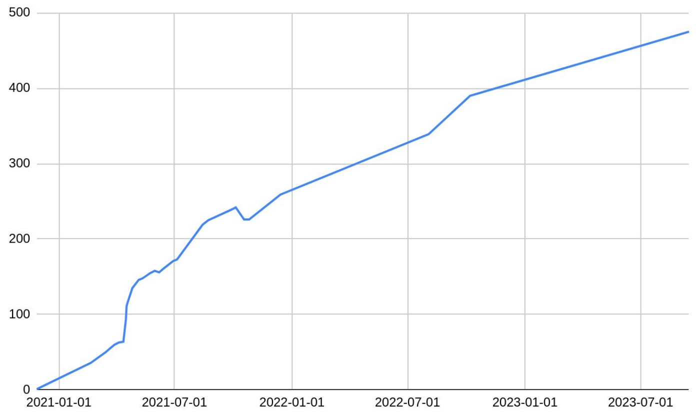
The last year

Growing Community

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

From ~340 last year.

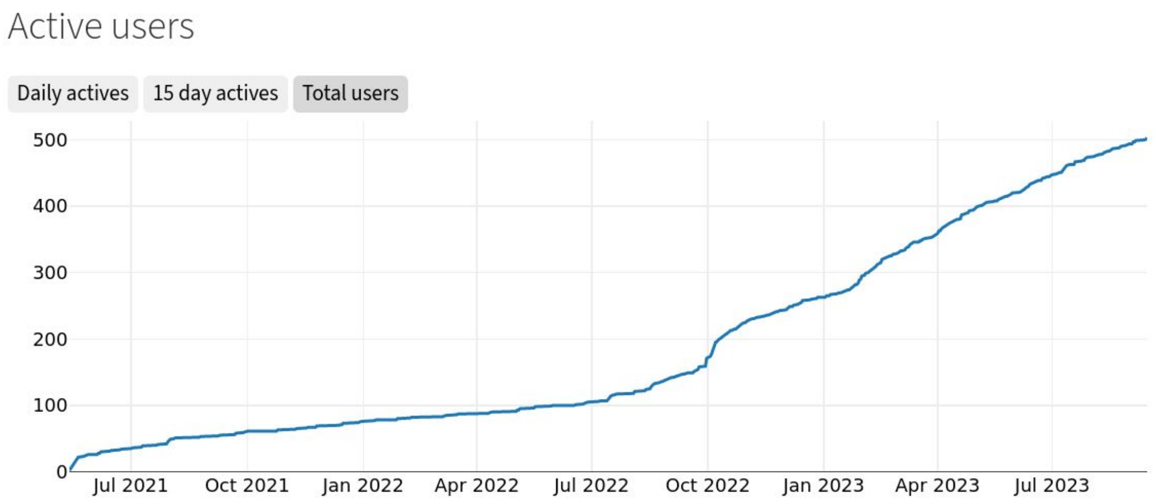
Similar to the BPF and `linux-rt-users` lists.



— https://subspace.kernel.org/vger.kernel.org.html

Growing Community

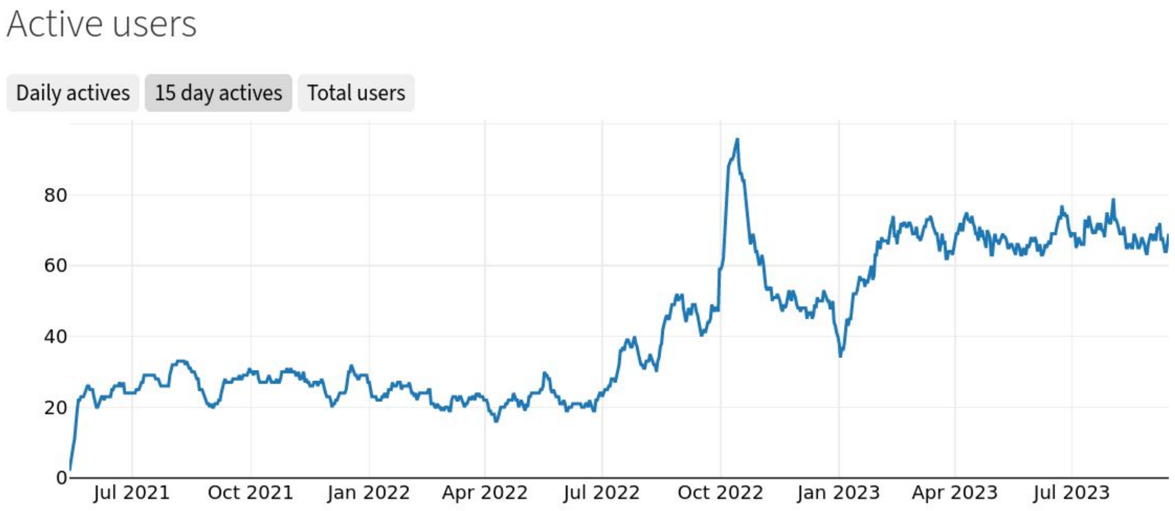
The Zulip instance (i.e. chat) is growing too: 500 users now!



— https://rust-for-linux.zulipchat.com/stats

Growing Community

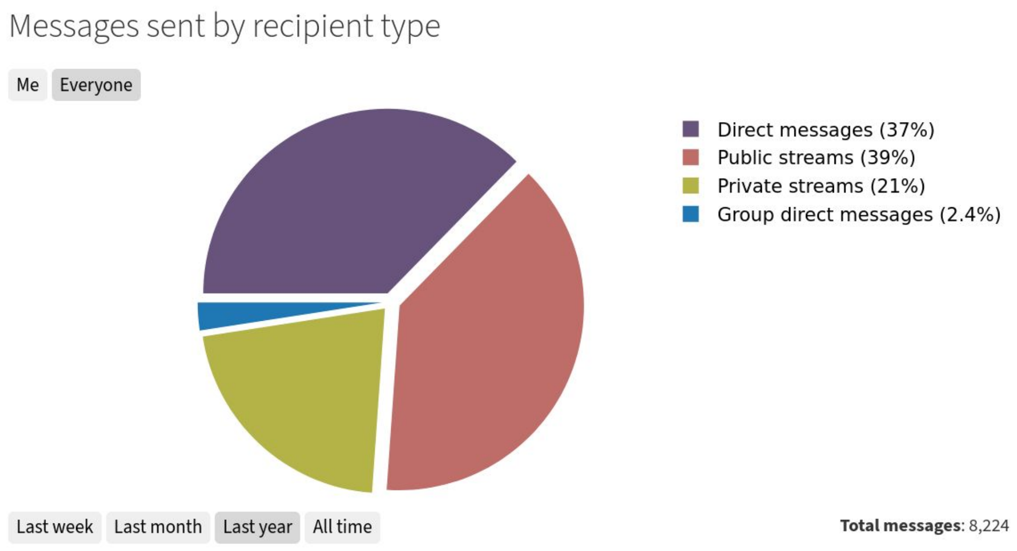
The Zulip instance (i.e. chat) is growing too: ~70 regulars



— https://rust-for-linux.zulipchat.com/stats

Growing Community

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


Messages sent by recipient type

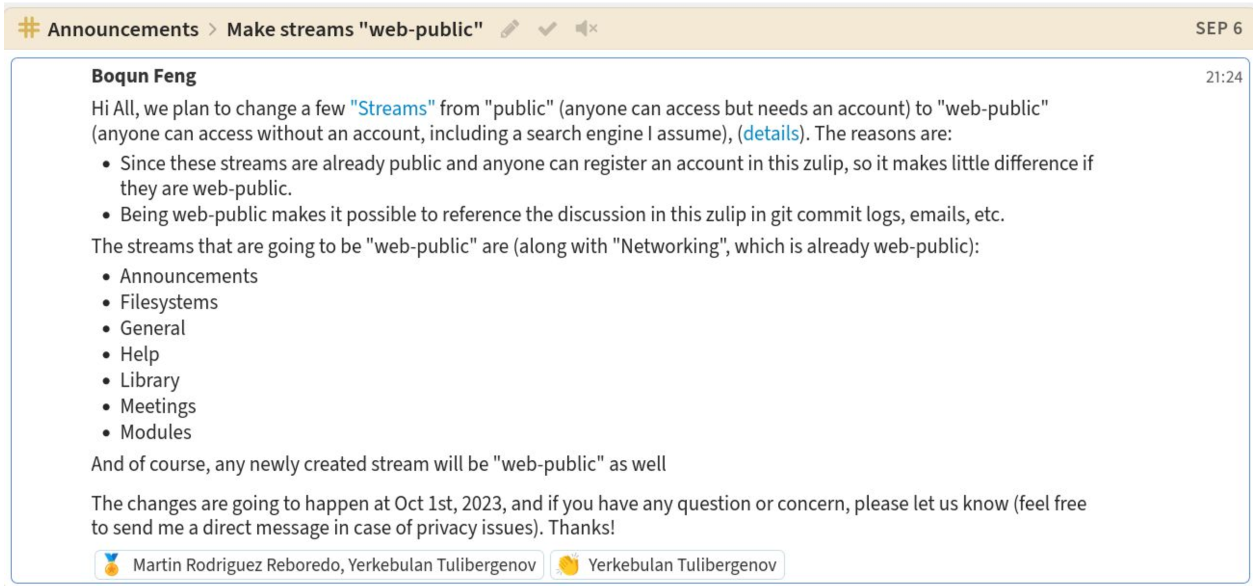
🟪 Direct messages (37%)
🟥 Public streams (39%)
🟨 Private streams (21%)
🟦 Group direct messages (2.4%)

Total messages: 8,224

— https://rust-for-linux.zulipchat.com/stats

Growing Community

Speaking of Zulip, we are making it anonymously readable.



Announcements > Make streams "web-public" SEP 6

Boqun Feng 21:24

Hi All, we plan to change a few "Streams" from "public" (anyone can access but needs an account) to "web-public" (anyone can access without an account, including a search engine I assume), (details). The reasons are:

- Since these streams are already public and anyone can register an account in this zulip, so it makes little difference if they are web-public.
- Being web-public makes it possible to reference the discussion in this zulip in git commit logs, emails, etc.

The streams that are going to be "web-public" are (along with "Networking", which is already web-public):

- Announcements
- Filesystems
- General
- Help
- Library
- Meetings
- Modules

And of course, any newly created stream will be "web-public" as well

The changes are going to happen at Oct 1st, 2023, and if you have any question or concern, please let us know (feel free to send me a direct message in case of privacy issues). Thanks!

🏅 Martin Rodriguez Reboredo, Yerkebulan Tulibergenov 👏 Yerkebulan Tulibergenov

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>
Arnaldo Carvalho de Melo <acme@kernel.org>
Asahi Lina <lina@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@google.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>
Michael Ellerman <mpe@ellerman.id.au>
Michele Dalle Rive <dallerivemichele@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.cqs@antgroup.com>
Roy Matero <materoy@proton.me>
SeongJae Park <sj@kernel.org>
Thomas Bamelis <thomas@bamelis.dev>
Timo Grautstück <timo.gr@hotmail.de>
Trevor Gross <tmgross@umich.edu>
TruongSinh Tran-Nguyen <i@truongsinh.pro>
Vinay Varma <varmavinaym@gmail.com>
Vincenzo Palazzo <vincenzopalazzodev@gmail.com>
WANG Rui <wangrui@loongson.cn>
Wedson Almeida Filho <wedsonaf@gmail.com>
Wei Liu <wei.liu@kernel.org>
Wu XiangCheng <bobwxc@email.cn>
Yang Yingliang <yangyingliang@huawei.com>
Yanteng Si <siyanteng@loongson.cn>

...as well as other contributors

Even more people involved in other tags, e.g. `Signed-off-by`, `Reviewed-by`...

Kernel maintainers are also getting involved!

- e.g. 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.
```

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

...as well as other contributors

Even more people involved in other tags, e.g. `Signed-off-by`, `Reviewed-by`...

Kernel maintainers are also getting involved!

- e.g. KUnit maintainers got Rust files in their `MAINTAINERS` entry.

```
 KERNEL UNIT TESTING FRAMEWORK (KUnit)
 M:   Brendan Higgins <brendanhiggins@google.com>
 M:   David Gow <davidgow@google.com>
 L:   linux-kselftest@vger.kernel.org
 ...
 F:   include/kunit/
 F:   lib/kunit/
+F:   rust/kernel/kunit.rs
+F:   scripts/rustdoc_test_*
 F:   tools/testing/kunit/
```

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

External interest

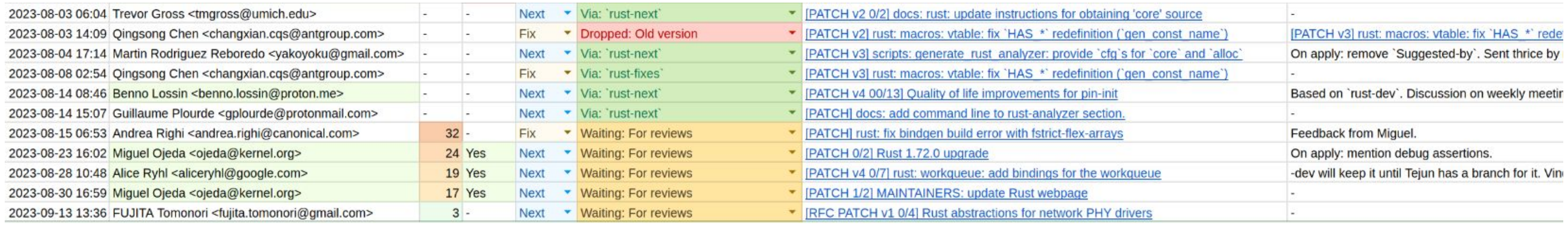
There has been a lot of interest from other parties:

- Companies wanting to write drivers
- Backporting requests (to 5.15 and 6.1 in particular)
- Students wanting to play with the Rust support

The Core Team spreadsheet

We are using Google Sheets to manage the patch series and PRs.

It looks like this:

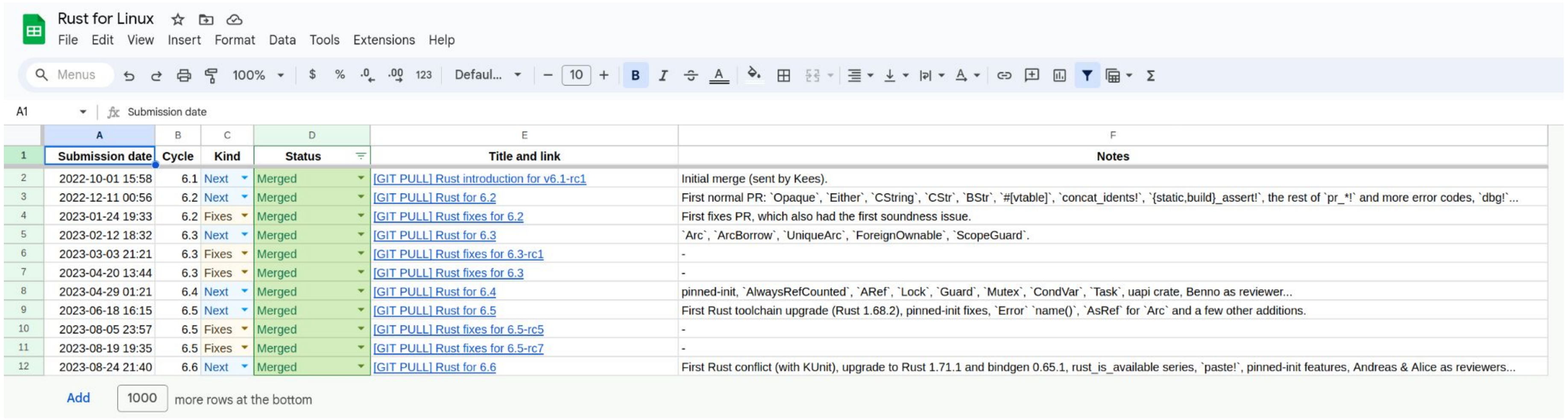


2023-08-03 06:04	Trevor Gross <tmgross@umich.edu>	-	-	Next	Via: `rust-next`	[PATCH v2 0/2] docs: rust: update instructions for obtaining 'core' source	-
2023-08-03 14:09	Qingsong Chen <changxian.cqs@antgroup.com>	-	-	Fix	Dropped: Old version	[PATCH v2] rust: macros: vtable: fix `HAS_*` redefinition (`gen_const_name`)	[PATCH v3] rust: macros: vtable: fix `HAS_*` rede
2023-08-04 17:14	Martin Rodriguez Reboredo <yakoyoku@gmail.com>	-	-	Next	Via: `rust-next`	[PATCH v3] scripts: generate_rust_analyzer: provide `cfg`s for `core` and `alloc`	On apply: remove `Suggested-by`. Sent thrice by
2023-08-08 02:54	Qingsong Chen <changxian.cqs@antgroup.com>	-	-	Fix	Via: `rust-fixes`	[PATCH v3] rust: macros: vtable: fix `HAS_*` redefinition (`gen_const_name`)	-
2023-08-14 08:46	Benno Lossin <benno.lossin@proton.me>	-	-	Next	Via: `rust-next`	[PATCH v4 00/13] Quality of life improvements for pin-init	Based on `rust-dev`. Discussion on weekly meetir
2023-08-14 15:07	Guillaume Plourde <gplourde@protonmail.com>	-	-	Next	Via: `rust-next`	[PATCH] docs: add command line to rust-analyzer section.	-
2023-08-15 06:53	Andrea Righi <andrea.righi@canonical.com>	32	-	Fix	Waiting: For reviews	[PATCH] rust: fix bindgen build error with fstrict-flex-arrays	Feedback from Miguel.
2023-08-23 16:02	Miguel Ojeda <ojeda@kernel.org>	24	Yes	Next	Waiting: For reviews	[PATCH 0/2] Rust 1.72.0 upgrade	On apply: mention debug assertions.
2023-08-28 10:48	Alice Ryhl <aliceryhl@google.com>	19	Yes	Next	Waiting: For reviews	[PATCH v4 0/7] rust: workqueue: add bindings for the workqueue	-dev will keep it until Tejun has a branch for it. Vin
2023-08-30 16:59	Miguel Ojeda <ojeda@kernel.org>	17	Yes	Next	Waiting: For reviews	[PATCH 1/2] MAINTAINERS: update Rust webpage	-
2023-09-13 13:36	FUJITA Tomonori <fujita.tomonori@gmail.com>	3	-	Next	Waiting: For reviews	[RFC PATCH v1 0/4] Rust abstractions for network PHY drivers	-

The Core Team spreadsheet

We are using Google Sheets to manage the patch series and PRs.

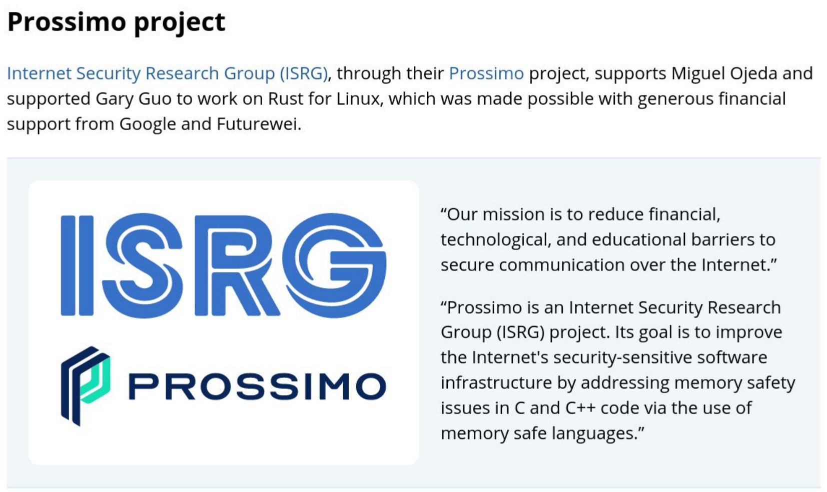
It looks like this:



Submission date	Cycle	Kind	Status	Title and link	Notes
2022-10-01 15:58	6.1	Next	Merged	[GIT PULL] Rust introduction for v6.1-rc1	Initial merge (sent by Kees).
2022-12-11 00:56	6.2	Next	Merged	[GIT PULL] Rust for 6.2	First normal PR: `Opaque`, `Either`, `CString`, `CStr`, `BStr`, `#[vtable]`, `concat_idents!`, `{static,build}_assert!`, the rest of `pr_*!` and more error codes, `dbg!`...
2023-01-24 19:33	6.2	Fixes	Merged	[GIT PULL] Rust fixes for 6.2	First fixes PR, which also had the first soundness issue.
2023-02-12 18:32	6.3	Next	Merged	[GIT PULL] Rust for 6.3	`Arc`, `ArcBorrow`, `UniqueArc`, `ForeignOwnable`, `ScopeGuard`.
2023-03-03 21:21	6.3	Fixes	Merged	[GIT PULL] Rust fixes for 6.3-rc1	-
2023-04-20 13:44	6.3	Fixes	Merged	[GIT PULL] Rust fixes for 6.3	-
2023-04-29 01:21	6.4	Next	Merged	[GIT PULL] Rust for 6.4	pinned-init, `AlwaysRefCounted`, `ARef`, `Lock`, `Guard`, `Mutex`, `CondVar`, `Task`, uapi crate, Benno as reviewer...
2023-06-18 16:15	6.5	Next	Merged	[GIT PULL] Rust for 6.5	First Rust toolchain upgrade (Rust 1.68.2), pinned-init fixes, `Error` `name()`, `AsRef` for `Arc` and a few other additions.
2023-08-05 23:57	6.5	Fixes	Merged	[GIT PULL] Rust fixes for 6.5-rc5	-
2023-08-19 19:35	6.5	Fixes	Merged	[GIT PULL] Rust fixes for 6.5-rc7	-
2023-08-24 21:40	6.6	Next	Merged	[GIT PULL] Rust for 6.6	First Rust conflict (with KUnit), upgrade to Rust 1.71.1 and bindgen 0.65.1, rust_is_available series, `paste!`, pinned-init features, Andreas & Alice as reviewers...

Add 1000 more rows at the bottom

Rust for Linux Sponsors



Prossimo project

Internet Security Research Group (ISRG), through their Prossimo project, supports Miguel Ojeda and supported Gary Guo to work on Rust for Linux, which was made possible with generous financial support from Google and Futurewei.

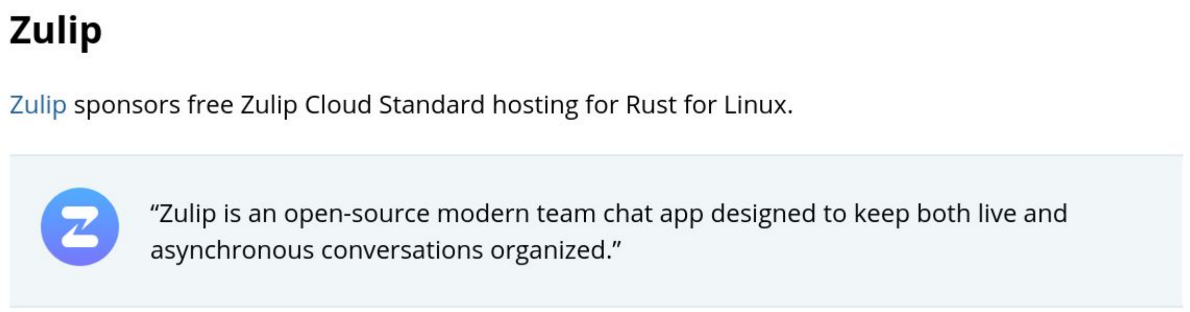
“Our mission is to reduce financial, technological, and educational barriers to secure communication over the Internet.”

“Prossimo is an Internet Security Research Group (ISRG) project. Its goal is to improve the Internet's security-sensitive software infrastructure by addressing memory safety issues in C and C++ code via the use of memory safe languages.”

— https://rust-for-linux.com/sponsors

— https://www.memorysafety.org/initiative/linux-kernel/

Rust for Linux Sponsors



Zulip

Zulip sponsors free Zulip Cloud Standard hosting for Rust for Linux.

"Zulip is an open-source modern team chat app designed to keep both live and asynchronous conversations organized."

— https://rust-for-linux.com/sponsors

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

Statements of support

“**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.”

SAMSUNG

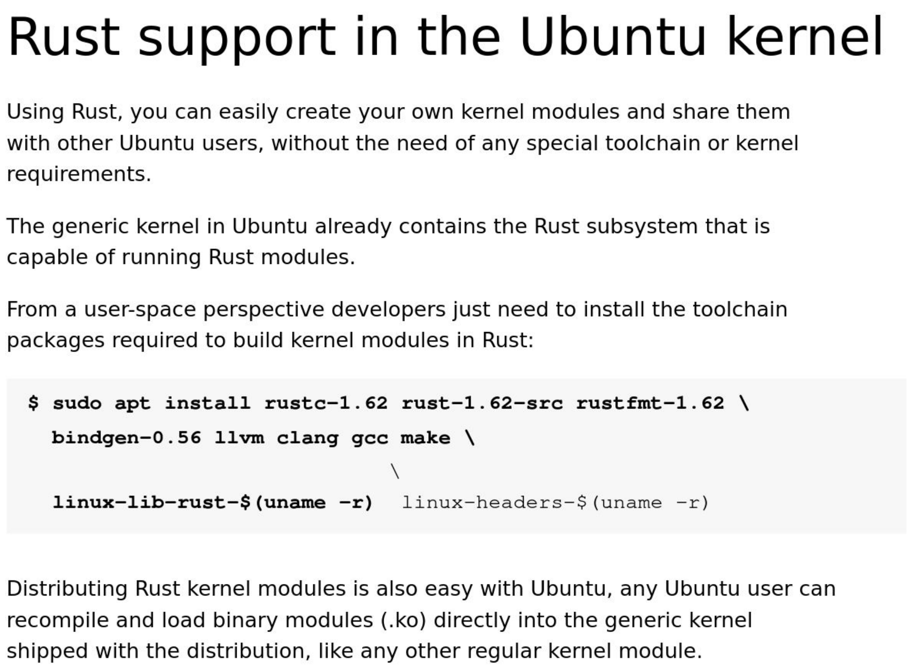
— https://rust-for-linux.com/industry-and-academia-support#Samsung

Statements of support

“**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

Distributions



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)
```

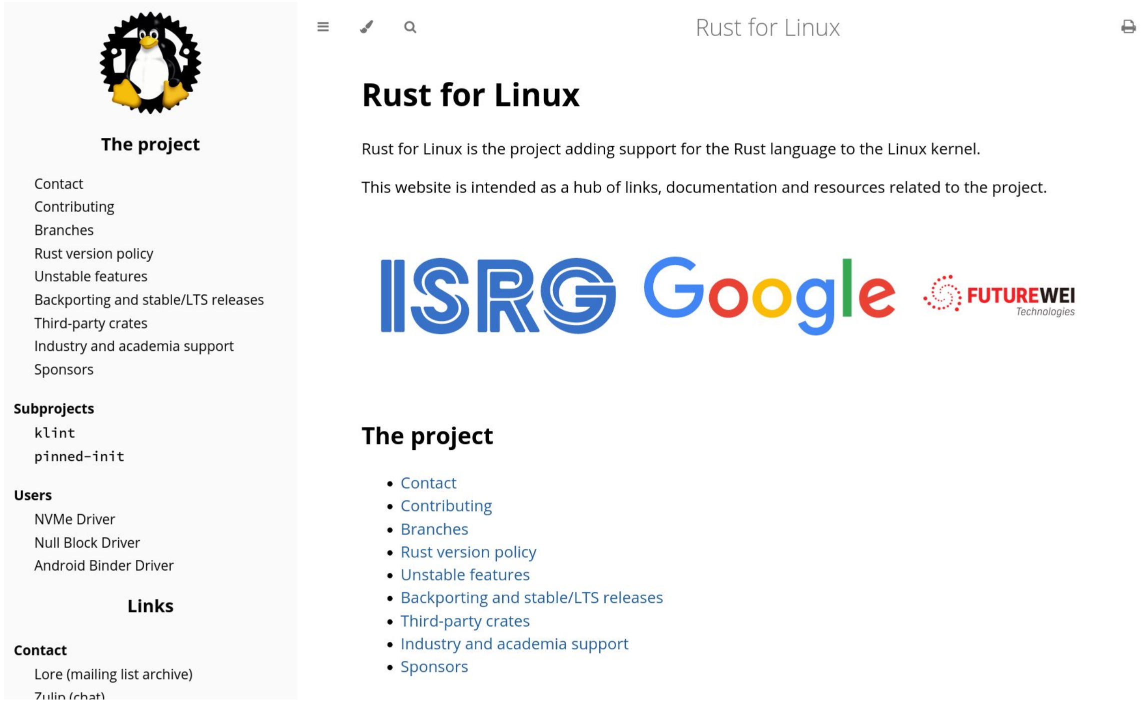
Distributing Rust kernel modules is also easy with Ubuntu, any Ubuntu user can recompile and load binary modules (.ko) directly into the generic kernel shipped with the distribution, like any other regular kernel module.

Ubuntu

Canonical

— https://ubuntu.com/blog/get-familiar-with-rusty-kernel-programming-in-ubuntu-lunar-lobster

rust-for-linux.com



Rust for Linux

The project

Contact
Contributing
Branches
Rust version policy
Unstable features
Backporting and stable/LTS releases
Third-party crates
Industry and academia support
Sponsors

Subprojects

`klint`
`pinned-init`

Users

NVMe Driver
Null Block Driver
Android Binder Driver

Links

Contact

Lore (mailing list archive)

Rust for Linux

Rust for Linux is the project adding support for the Rust language to the Linux kernel.

This website is intended as a hub of links, documentation and resources related to the project.

The project

- Contact
- Contributing
- Branches
- Rust version policy
- Unstable features
- Backporting and stable/LTS releases
- Third-party crates
- Industry and academia support
- Sponsors

— https://rust-for-linux.com

rust-for-linux.com

6 months old.

A hub of links, documentation and resources related to the project.

Essentially, everything that cannot got into the kernel tree.

- Contributing Guidelines
- Subproject pages
- Users

It’s going to be our current webpage (`W:` entry) in `MAINTAINERS`.

As well as our Maintainer Entry Profile field (`P:` entry).

- https://docs.kernel.org/maintainer/maintainer-entry-profile.html

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 can now finally be frozen/archived.

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.

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!”

The merges - v6.1

> Rust introduction for v6.1-rc1
>
> The initial support of Rust-for-Linux comes in roughly 4 areas:
>
> - Kernel internals (kallsyms expansion for Rust symbols, %pA format)
>
> - Kbuild infrastructure (Rust build rules and support scripts)
>
> - Rust crates and bindings for initial minimum viable build
>
> - Rust kernel documentation and samples
>
> Rust support has been in linux-next for a year and a half now, and the
> short log doesn't do justice to the number of people who have contributed
> both to the Linux kernel side but also to the upstream Rust side to
> support the kernel's needs. Thanks to these 173 people, and many more,
> who have been involved in all kinds of ways:

— https://lore.kernel.org/lkml/202210010816.1317F2C@keescook/

The merges - v6.1

```
Miguel Ojeda, Wedson Almeida Filho, Alex Gaynor, Boqun Feng, Gary Guo,
Björn Roy Baron, Andreas Hindborg, Adam Bratschi-Kaye, Benno Lossin,
Maciej Falkowski, Finn Behrens, Sven Van Asbroeck, Asahi Lina, FUJITA
Tomonori, John Baublitz, Wei Liu, Geoffrey Thomas, Philip Herron,
Arthur Cohen, David Faust, Antoni Boucher, Philip Li, Yujie Liu,
Jonathan Corbet, Greg Kroah-Hartman, Paul E. McKenney, Josh Triplett,
Kent Overstreet, David Gow, Alice Ryhl, Robin Randhawa, Kees Cook,
Nick Desaulniers, Matthew Wilcox, Linus Walleij, Joe Perches, Michael
Ellerman, Petr Mladek, Masahiro Yamada, Arnaldo Carvalho de Melo,
Andrii Nakryiko, Konstantin Shelekhin, Rasmus Villemoes, Konstantin
Ryabitsev, Stephen Rothwell, Andy Shevchenko, Sergey Senozhatsky, John
Paul Adrian Glaubitz, David Laight, Nathan Chancellor, Jonathan
Cameron, Daniel Latypov, Shuah Khan, Brendan Higgins, Julia Lawall,
Laurent Pinchart, Geert Uytterhoeven, Akira Yokosawa, Pavel Machek,
David S. Miller, John Hawley, James Bottomley, Arnd Bergmann,
Christian Brauner, Dan Robertson, Nicholas Piggin, Zhouyi Zhou, Elena
Zannoni, Jose E. Marchesi, Leon Romanovsky, Will Deacon, Richard
Weinberger, Randy Dunlap, Paolo Bonzini, Roland Dreier, Mark Brown,
Sasha Levin, Ted Ts'o, Steven Rostedt, Jarkko Sakkinen, Michal
Kubecek, Marco Elver, Al Viro, Keith Busch, Johannes Berg, Jan Kara,
David Sterba, Connor Kuehl, Andy Lutomirski, Andrew Lunn, Alexandre
```

— https://lore.kernel.org/lkml/202210010816.1317F2C@keescook/

The merges - v6.1

```
Belloni, Peter Zijlstra, Russell King, Eric W. Biederman, Willy
Tarreau, Christoph Hellwig, Emilio Cobos Álvarez, Christian Poveda,
Mark Rousskov, John Ericson, TennyZhuang, Xuanwo, Daniel Paoliello,
Manish Goregaokar, comex, Josh Stone, Stephan Sokolow, Philipp Krones,
Guillaume Gomez, Joshua Nelson, Mats Larsen, Marc Poulhiès, Samantha
Miller, Esteban Blanc, Martin Schmidt, Martin Rodriguez Reboredo,
Daniel Xu, Viresh Kumar, Bartosz Golaszewski, Vegard Nossum, Milan
Landaverde, Dariusz Sosnowski, Yuki Okushi, Matthew Bakhtiari, Wu
XiangCheng, Tiago Lam, Boris-Chengbiao Zhou, Sumera Priyadarsini,
Viktor Garske, Niklas Mohrin, Nándor István Krácser, Morgan Bartlett,
Miguel Cano, Léo Lanteri Thauvin, Julian Merkle, Andreas Reindl,
Jiapeng Chong, Fox Chen, Douglas Su, Antonio Terceiro, SeongJae Park,
Sergio González Collado, Ngo Iok Ui (Wu Yu Wei), Joshua Abraham,
Milan, Daniel Kolsoi, ahomescu, Manas, Luis Gerhorst, Li Hongyu,
Philipp Gesang, Russell Currey, Jalil David Salamé Messina, Jon Olson,
Raghvender, Angelos, Kaviraj Kanagaraj, Paul Römer, Sladyn Nunes,
Mauro Baladés, Hsiang-Cheng Yang, Abhik Jain, Hongyu Li, Sean Nash,
Yuheng Su, Peng Hao, Anhad Singh, Roel Kluin, Sara Saa, Geert
Stappers, Garrett LeSage, IFo Hancroft, and Linus Torvalds.
```

— https://lore.kernel.org/lkml/202210010816.1317F2C@keescook/

The merges - v6.2

```
Rust changes for v6.2

The first set of changes after the merge, the major ones being:

- String and formatting: new types `CString`, `CStr`, `BStr` and
  `Formatter`; new macros `c_str!`, `b_str!` and `fmt!`.

- Errors: the rest of the error codes from `errno-base.h`, as well as
  some `From` trait implementations for the `Error` type.

- Printing: the rest of the `pr_*!` levels and the continuation one
  `pr_cont!`, as well as a new sample.

- `alloc` crate: new constructors `try_with_capacity()` and
  `try_with_capacity_in()` for `RawVec` and `Vec`.
```

— https://lore.kernel.org/rust-for-linux/20221211005609.270457-1-ojeda@kernel.org/

The merges - v6.2

```
- Procedural macros: new macros `#[vtable]` and `concat_idents!`, as
  well as better ergonomics for `module!` users.

- Asserting: new macros `static_assert!`, `build_error!` and
  `build_assert!`, as well as a new crate `build_error` to support them.

- Vocabulary types: new types `Opaque` and `Either`.

- Debugging: new macro `dbg!`.
```

— https://lore.kernel.org/rust-for-linux/20221211005609.270457-1-ojeda@kernel.org/

The merges - v6.3

```
Rust changes for v6.3

More core additions, getting closer to a point where the first Rust
modules can be upstreamed. The major ones being:

- Sync: new types 'Arc', 'ArcBorrow' and 'UniqueArc'.

- Types: new trait 'ForeignOwnable' and new type 'ScopeGuard'.

There is also a substantial removal in terms of lines:

- 'alloc' crate: remove the 'borrow' module (type 'Cow' and trait
  'ToOwned').
```

— https://lore.kernel.org/rust-for-linux/20230212183249.162376-1-ojeda@kernel.org/

The merges - v6.4

```
Rust changes for v6.4

More additions to the Rust core. Importantly, this adds the pin-init
API, which will be used by other abstractions, such as the
synchronization ones added here too:

  - pin-init API: a solution for the safe pinned initialization problem.
    This allows to reduce the need for 'unsafe' code in the kernel when
    dealing with data structures that require a stable address. Commit
    90e53c5e70a6 ("rust: add pin-init API core") contains a nice
    introduction -- here is an example of how it looks like:
```

— https://lore.kernel.org/lkml/20230429012119.421536-1-ojeda@kernel.org/

The merges - v6.4

```
#[pin_data]
struct Example {
    #[pin]
    value: Mutex<u32>,
    #[pin]
    value_changed: CondVar,
}

impl Example {
    fn new() -> impl PinInit<Self> {
        pin_init!(Self {
            value <- new_mutex!(0),
            value_changed <- new_condvar!(),
        })
    }
}

// In a `Box`.
let b = Box::pin_init(Example::new())?;

// In the stack.
stack_pin_init!(let s = Example::new());
```

— https://lore.kernel.org/lkml/20230429012119.421536-1-ojeda@kernel.org/

The merges - v6.4

```
- 'sync' module: new types 'LockClassKey' ('struct lock_class_key'),
  'Lock', 'Guard', 'Mutex' ('struct mutex'), 'SpinLock'
  ('spinlock_t'), 'LockedBy' and 'CondVar' (uses 'wait_queue_head_t'),
  plus macros such as 'static_lock_class!' and 'new_spinlock!'.

  In particular, 'Lock' and 'Guard' are generic implementations that
  contain code that is common to all locks. Then, different backends
  (the new 'Backend' trait) are implemented and used to define types
  like 'Mutex':

      type Mutex<T> = Lock<T, MutexBackend>;

  In addition, new methods 'assume_init()', 'init_with()' and
  'pin_init_with()' for 'UniqueArc<MaybeUninit<T>>' and 'downcast()'
  for 'Arc<dyn Any + Send + Sync>'; as well as 'Debug' and 'Display'
  implementations for 'Arc' and 'UniqueArc'. Reduced stack usage of
  'UniqueArc::try_new_uninit()', too.
```

— https://lore.kernel.org/lkml/20230429012119.421536-1-ojeda@kernel.org/

The merges - v6.4

```
- 'types' module: new trait 'AlwaysRefCounted' and new type 'ARef'
  (an owned reference to an always-reference-counted object, meant to
  be used in wrappers for C types that have their own ref counting
  functions).

  Moreover, new associated functions 'raw_get()' and 'ffi_init()'
  for 'Opaque'.

- New 'task' module with a new type 'Task' ('struct task_struct'), and
  a new macro 'current!' to safely get a reference to the current one.

- New 'ioctl' module with new '_IOC*' const functions (equivalent to
  the C macros).

- New 'uapi' crate, intended to be accessible by drivers directly.

- 'macros' crate: new 'quote!' macro (similar to the one provided in
  userspace by the 'quote' crate); and the 'module!' macro now allows
  specifying multiple module aliases.

- 'error' module: new associated functions for the 'Error' type,
  such as 'from_errno()' and new functions such as 'to_result()'.

- 'alloc' crate: more fallible 'Vec' methods: 'try_resize` and
  'try_extend_from_slice' and the infrastructure (imported from
  the Rust standard library) they need.
```

— https://lore.kernel.org/lkml/20230429012119.421536-1-ojeda@kernel.org/

The merges - v6.5

```
Rust changes for v6.5

A fairly small one in terms of feature additions. Most of the changes in
terms of lines come from the upgrade to the new version of the toolchain
(which in turn is big due to the vendored 'alloc' crate).

 - Upgrade to Rust 1.68.2:

   This is the first such upgrade, and we will try to update it often
   from now on, in order to remain close to the latest release, until
   a minimum version (which is "in the future") can be established.

   The upgrade brings the stabilization of 4 features we used (and 2
   more that we used in our old 'rust' branch).

   Commit 3ed03f4da06e ("rust: upgrade to Rust 1.68.2") contains the
   details and rationale.
```

— https://lore.kernel.org/rust-for-linux/20230618161558.1051269-1-ojeda@kernel.org/

The merges - v6.5

```
- pin-init API:

  Several internal improvements and fixes to the pin-init API, e.g.
  allowing to use 'Self' in a struct definition with '#[pin_data]'.

- 'error'  module:

  New 'name()' method for the 'Error' type (with 'errname()'
  integration), used to implement the 'Debug' trait for 'Error'.

  Add error codes from 'include/linux/errno.h' to the list of Rust
  'Error' constants.

  Allow specifying error type on the 'Result' type (with the default
  still being our usual 'Error' type).

- 'str' module:

  'TryFrom' implementation for 'CStr', and new 'to_cstring()' method
  based on it.
```

— https://lore.kernel.org/rust-for-linux/20230618161558.1051269-1-ojeda@kernel.org/

The merges - v6.5

```
- 'sync' module:

  Implement 'AsRef' trait for 'Arc', allowing to use 'Arc' in code that
  is generic over smart pointer types.

  Add 'ptr_eq' method to 'Arc' for easier, less error prone comparison
  between two 'Arc' pointers.

  Reword the 'Send' safety comment for 'Arc', and avoid referencing it
  from the 'Sync' one.

- 'task' module:

  Implement 'Send' marker for 'Task'.

- 'types' module:

  Implement 'Send' and 'Sync' markers for 'ARef<T>' when 'T' is
  'AlwaysRefCounted', 'Send' and 'Sync'.

- Other changes:

  Documentation improvements and '.gitattributes' change to start
  using the Rust diff driver.
```

— https://lore.kernel.org/rust-for-linux/20230618161558.1051269-1-ojeda@kernel.org/

The merges - v6.6

```
Rust changes for v6.6

In terms of lines, most changes this time are on the pinned-init API
and infrastructure. While we have a Rust version upgrade, and thus a
bunch of changes from the vendored 'alloc' crate as usual, this time
those do not account for many lines.

Toolchain and infrastructure:

 - Upgrade to Rust 1.71.1. This is the second such upgrade, which is a
   smaller jump compared to the last time.

   This version allows us to remove the '__rust_*' allocator functions
   -- the compiler now generates them as expected, thus now our
   'KernelAllocator' is used.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
  It also introduces the 'offset_of!' macro in the standard library
  (as an unstable feature) which we will need soon. So far, we were
  using a declarative macro as a prerequisite in some not-yet-landed
  patch series, which did not support sub-fields (i.e. nested structs):

      #[repr(C)]
      struct S {
          a: u16,
          b: (u8, u8),
      }

      assert_eq!(offset_of!(S, b.1), 3);

- Upgrade to bindgen 0.65.1. This is the first time we upgrade its
  version.

  Given it is a fairly big jump, it comes with a fair number of
  improvements/changes that affect us, such as a fix needed to support
  LLVM 16 as well as proper support for '__noreturn' C functions, which
  are now mapped to return the '!' type in Rust:

      void __noreturn f(void); // C
      pub fn f() -> !;         // Rust
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
- 'scripts/rust_is_available.sh' improvements and fixes.

  This series takes care of all the issues known so far and adds a few
  new checks to cover for even more cases, plus adds some more help
  texts. All this together will hopefully make problematic setups
  easier to identify and to be solved by users building the kernel.

  In addition, it adds a test suite which covers all branches of the
  shell script, as well as tests for the issues found so far.

- Support rust-analyzer for out-of-tree modules too.

- Give 'cfg's to rust-analyzer for the 'core' and 'alloc' crates.

- Drop 'scripts/is_rust_module.sh' since it is not needed anymore.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
Macros crate:

 - New 'paste!' proc macro.

   This macro is a more flexible version of 'concat_idents!': it allows
   the resulting identifier to be used to declare new items and it
   allows to transform the identifiers before concatenating them, e.g.

       let x_1 = 42;
       paste!(let [<x _2>] = [<x _1>];);
       assert!(x_1 == x_2);

   The macro is then used for several of the pinned-init API changes in
   this pull.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
Pinned-init API:

 - Make '#[pin_data]' compatible with conditional compilation of fields,
   allowing to write code like:

       #[pin_data]
       pub struct Foo {
           #[cfg(CONFIG_BAR)]
           a: Bar,
           #[cfg(not(CONFIG_BAR))]
           a: Baz,
       }

 - New '#[derive(Zeroable)]' proc macro for the 'Zeroable' trait, which
   allows 'unsafe' implementations for structs where every field
   implements the 'Zeroable' trait, e.g.:

       #[derive(Zeroable)]
       pub struct DriverData {
           id: i64,
           buf_ptr: *mut u8,
           len: usize,
       }
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
- Add '..Zeroable::zeroed()' syntax to the 'pin_init!'  macro for
  zeroing all other fields, e.g.:

      pin_init!(Buf {
          buf: [1; 64],
          ..Zeroable::zeroed()
      });

- New '{,pin_}init_array_from_fn()' functions to create array
  initializers given a generator function, e.g.:

      let b: Box<[usize; 1_000]> = Box::init::<Error>(
          init_array_from_fn(|i| i)
      ).unwrap();

      assert_eq!(b.len(), 1_000);
      assert_eq!(b[123], 123);
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
- New '{,pin_}chain' methods for '{,Pin}Init<T, E>' that allow to
  execute a closure on the value directly after initialization, e.g.:

      let foo = init!(Foo {
          buf <- init::zeroed()
      }).chain(|foo| {
          foo.setup();
          Ok(())
      });

- Support arbitrary paths in init macros, instead of just identifiers
  and generic types.

- Implement the 'Zeroable' trait for the 'UnsafeCell<T>' and
  'Opaque<T>' types.

- Make initializer values inaccessible after initialization.

- Make guards in the init macros hygienic.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The merges - v6.6

```
'allocator' module:

 - Use 'krealloc_aligned()' in 'KernelAllocator::alloc' preventing
   misaligned allocations when the Rust 1.71.1 upgrade is applied later
   in this pull.

   The equivalent fix for the previous compiler version (where
   'KernelAllocator' is not yet used) was merged into 6.5 already,
   which added the 'krealloc_aligned()' function used here.

 - Implement 'KernelAllocator::{realloc, alloc_zeroed}' for performance,
   using 'krealloc_aligned()' too, which forwards the call to the C API.

'types' module:

 - Make 'Opaque' be '!Unpin', removing the need to add a 'PhantomPinned'
   field to Rust structs that contain C structs which must not be moved.

 - Make 'Opaque' use 'UnsafeCell' as the outer type, rather than inner.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

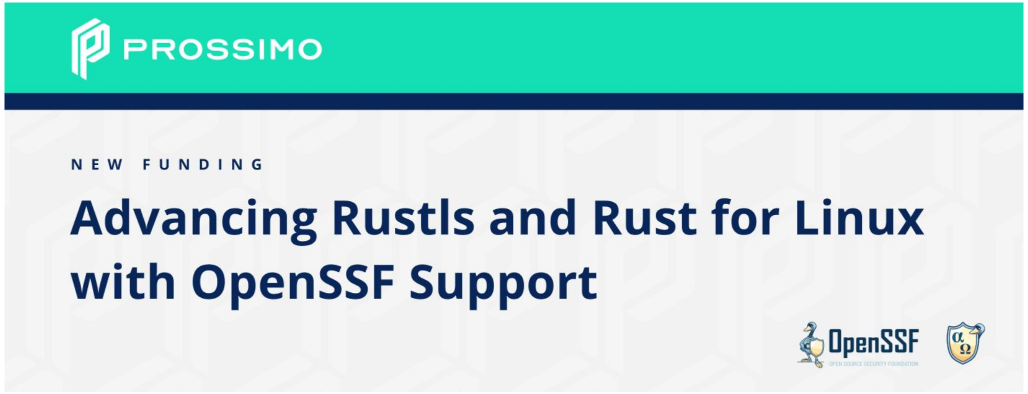
The merges - v6.6

```
Documentation:

 - Suggest obtaining the source code of the Rust's 'core' library using
   the tarball instead of the repository.

MAINTAINERS:

 - Andreas and Alice, from Samsung and Google respectively, are joining
   as reviewers of the "RUST" entry.

As well as a few other minor changes and cleanups.
```

— https://lore.kernel.org/rust-for-linux/20230824214024.608618-1-ojeda@kernel.org/

The future

Rust for Linux Sponsors



PROSSIMO

NEW FUNDING

Advancing Rustls and Rust for Linux with OpenSSF Support

OpenSSF

OpenSSF

OPEN SOURCE SECURITY FOUNDATION

Alpha-Omega

— https://www.memorysafety.org/blog/rustls-and-rust-for-linux-funding-openssf/

rust.docs.kernel.org

The original discussion on this started early 2021.

It has been a long time coming, but this summer 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).

- Should we have a “tag” selection top bar?

`rfl` CLI

- A CLI tool for “all” your Rust for Linux needs.
  - Running all the pre-submission tests we want, commit-by-commit
    - `rustfmtcheck, rustdoc, rusttest...`
    - Perhaps including the old Ksquirrel checks.
  - Running the CI.
  - Preparing the kernel for Rust usage (e.g. setting up tooling, configuring the kernel...).
    - Useful for testing across upgrades (since we fix the version).
    - Support stable kernels too.

CI

We had some in the old `rust` branch.

We are going to set it up again for `rust-next` and the other branches.

This is in addition to KernelCI and Intel LKP.

Merging our own CI files into the kernel?

- Linus just merged DRM’s GitLab CI integration!

  https://lore.kernel.org/lkml/CAPM=9tz=gx2_zrQ2XD8JNwW1dg6b+Byr5FgYAAq+2f29rydcgg@mail.gmail.com/

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 is 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.
- Better integration with GCC Rust and `rustc_codegen_gcc`.

The `rust-{net,*}` branches

Branches intended for synchronization between developers.

Similar to the old `rust` branch, but for particular subsystems.

To be discussed.

CFI

“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:open+label:PG-exploit-mitigations+CFI
[2] https://github.com/rust-lang/rust/issues/115199
[3] https://github.com/rust-lang/rust/pull/115200

— Ramon de C Valle

`rustc_codegen_gcc`

Implemented:

- LTO
- CPU features detection
- All SIMD tests pass
- More function attributes (including `inline`)
- `noalias/restrict`

Implemented, but needs more work:

- Unwinding

Many bug fixes

— Antoni Boucher

`rustc_codegen_gcc`

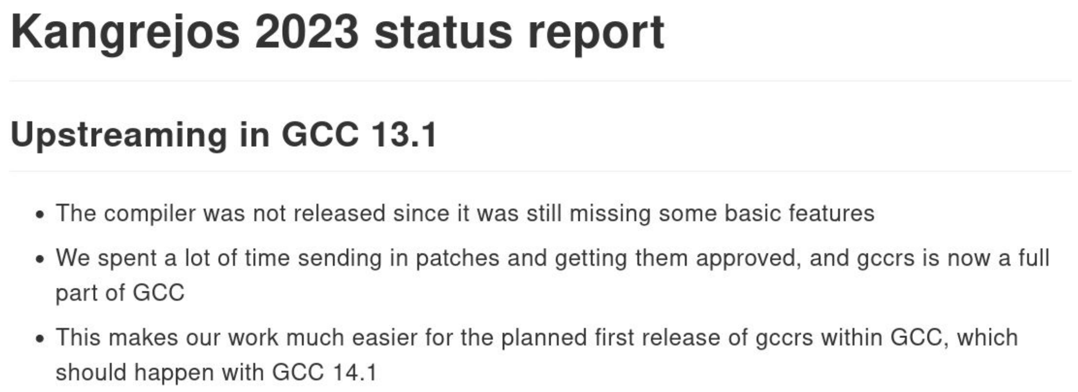
“I was able to finish my PR to fix the handling of CPU features and made another one to support `relocation-model=static` and now `rustc_codegen_gcc` can compile Rust for Linux (`rust-next` branch) without any patches.

> (well, it still requires patches to GCC like the rest of `rustc_codegen_gcc`, but I will merge them upstream eventually).

This requires compiling the sysroot with `CG_RUSTFLAGS=-Cpanic=abort` because we currently cannot only compile panic-abort with this flag.”

— Antoni Boucher

GCC Rust (`gccrs`)



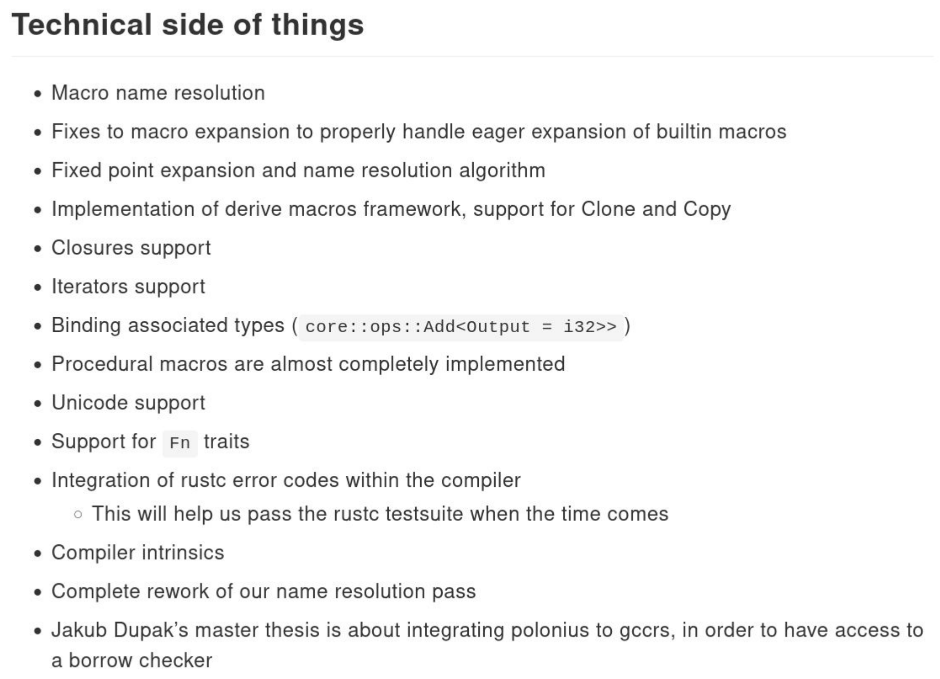
Kangrejos 2023 status report

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`)

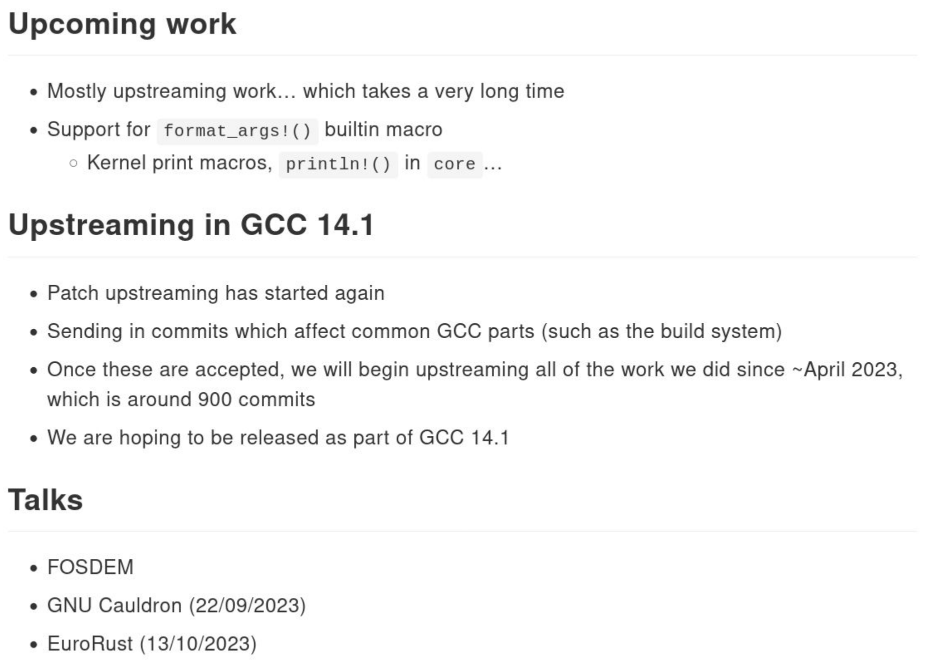


Technical side of things

- 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

Some extra discussion topics

Mailing List Bot & Ksquirrel

The “no dead code” rule

The “no duplicate drivers” rule 🍻

Should a soundness issue be backported?

Support in distributions

Build system

Thank you!

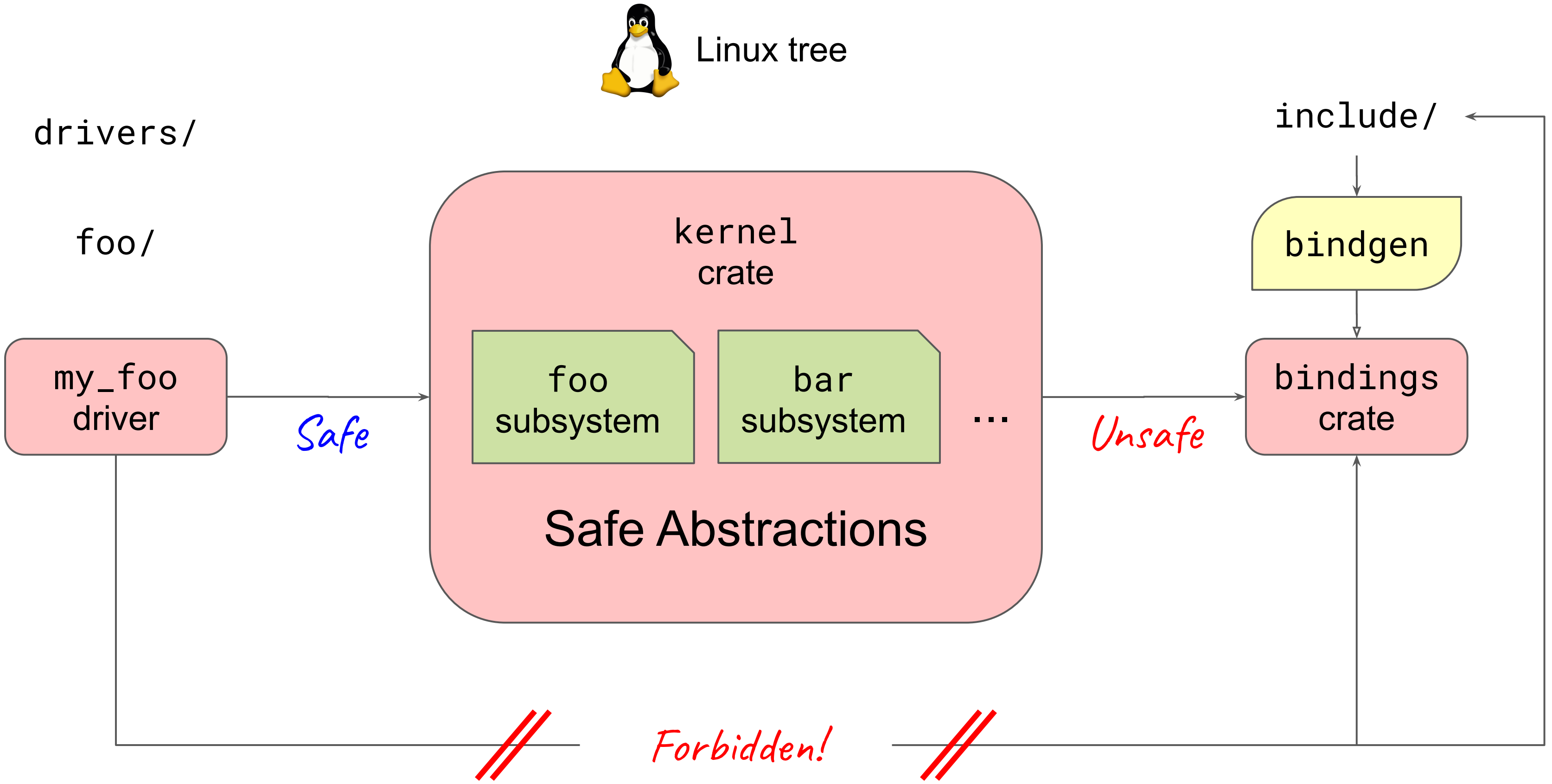
Questions?

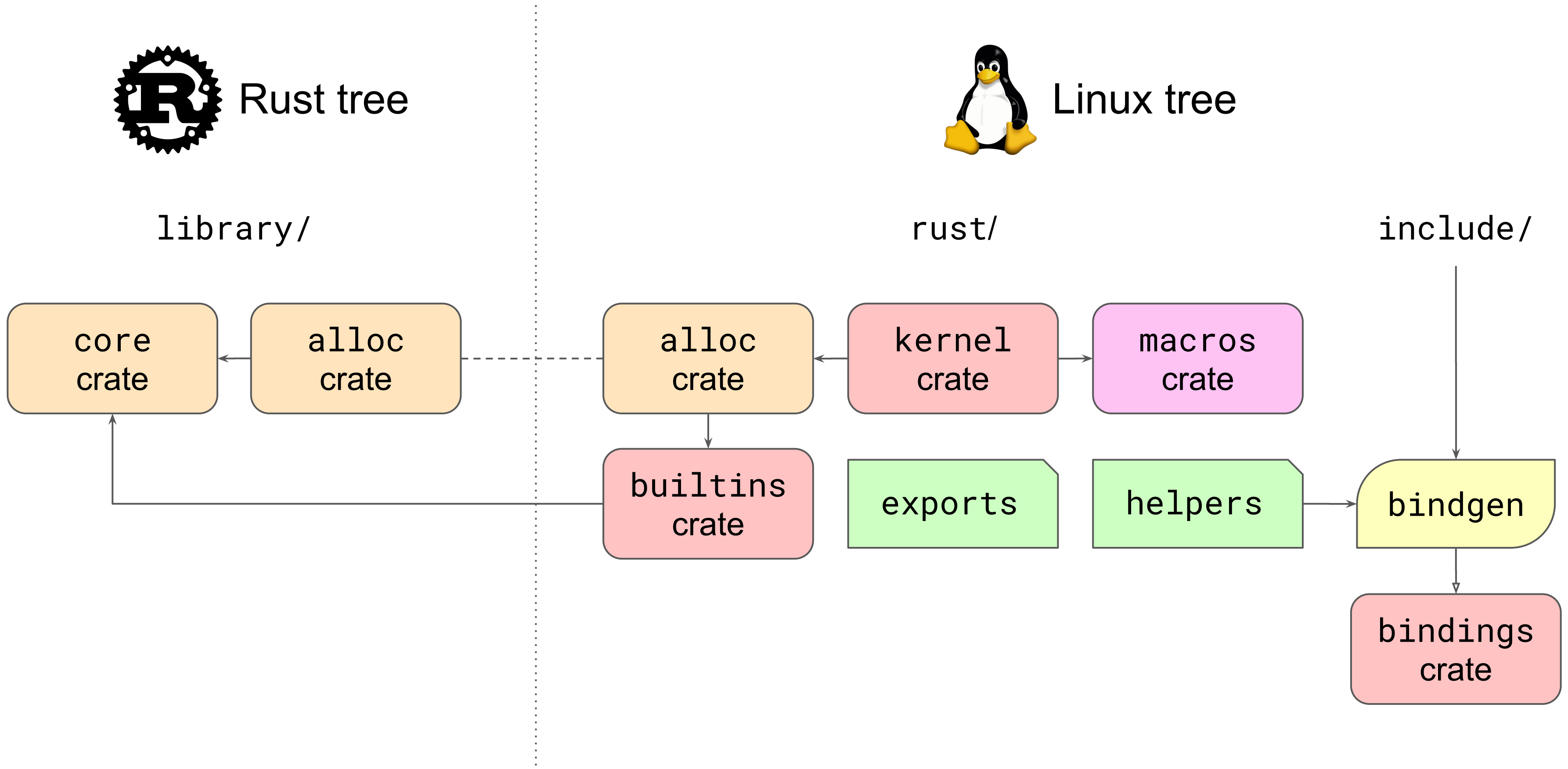
The Rust for Linux Kernel Report

Miguel Ojeda

ojeda@kernel.org

Backup slides


Linux tree
drivers/
foo/
my_foo driver
Safe
kernel crate
foo subsystem
bar subsystem
...
Safe Abstractions
Unsafe
include/
bindgen
bindings crate
Forbidden!


Rust tree
Linux tree
library/
rust/
include/
core crate
alloc crate
alloc crate
kernel crate
macros crate
builtins crate
exports
helpers
bindgen
bindings crate