An introduction to making open source contributions

“Contributing to open source for fun ~~and profit~~”

Who am I?

- Integrations Engineer at Codeplay Software
  - Manage provisioning and configuration management of build infrastructure.
  - Build and maintain internal tooling.
- Rust Compiler Contributor
  - Member of non-lexical-lifetimes (NLL) working group.
  - Reviewer with merge privileges.

@davidtwco

@davidtwco

david@davidtw.co

https://davidtw.co

What is Codeplay?

codeplay®

Codeplay is internationally recognized for expertise in Heterogeneous Systems and has many years of experience in the development of compilers, runtimes, debuggers, test systems and other specialized tools.

What is open source?

“I feel like I should have heard about this…”

What is open source?

Open source software is software with source code that anyone can inspect, modify or enhance.

Today, “open source” designates a broader set of values:

- Open Exchange
- Collaborative Participation
- Transparency
- Meritocracy
- Community-oriented development

Why contribute to open source?

“I’m not sure about all this, I could be playing Red Dead Redemption 2 instead…”

Why contribute to open source?

It’s good for your career progression.

Contributions to open source projects are a great way to build up a portfolio of code that you can use to demonstrate your skills.

Becoming experienced and gaining expertise in a tool, language, framework or library through contributions makes you a desirable candidate for companies that use that project.

Why contribute to open source?

Gain experience contributing to large software projects in specialized fields.

Some fields are easiest to learn through practical experience on large open source projects.

For example, compilers are large and complex, being both substantial exercises in software engineering and a microcosm of computer science.

Compilers for well-known languages are ideal for newcomers to get experience in a production system and the problems that occur at that scale.

Why contribute to open source?

Learn to work and communicate in a distributed team.

Open source projects are often worked on by a distributed team of volunteers, each with different schedules and interests.

Working with people from diverse backgrounds and with differing levels of experience can be beneficial in broadening your horizons, and can introduce new ways of solving problems.

Being involved in open source also offers the opportunity to practice management and leadership skills, such as organizing teams of people, resolving conflicts and prioritizing work.

Why contribute to open source?

It’s good for everyone.

Contributing to open source software benefits everyone. We’ve all used open source projects – it wouldn’t be feasible for many small start-ups and projects to exist if they had to buy licenses for all the software they use or build all of their own stack.

Fix and improve projects you already use.

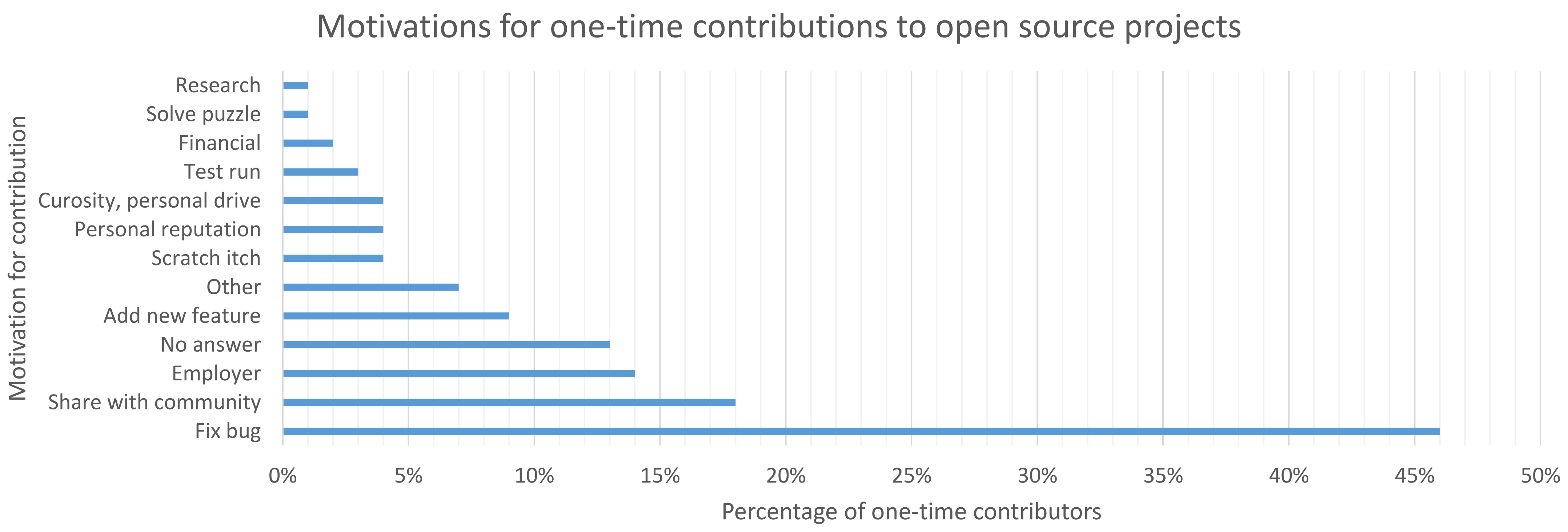
Open source software gives you the agency to improve, adapt and extend the tooling or libraries that you’re already using.

Research into open-source contributions

“I’m not going to have to read a paper, am I?”

Research into open source contributions

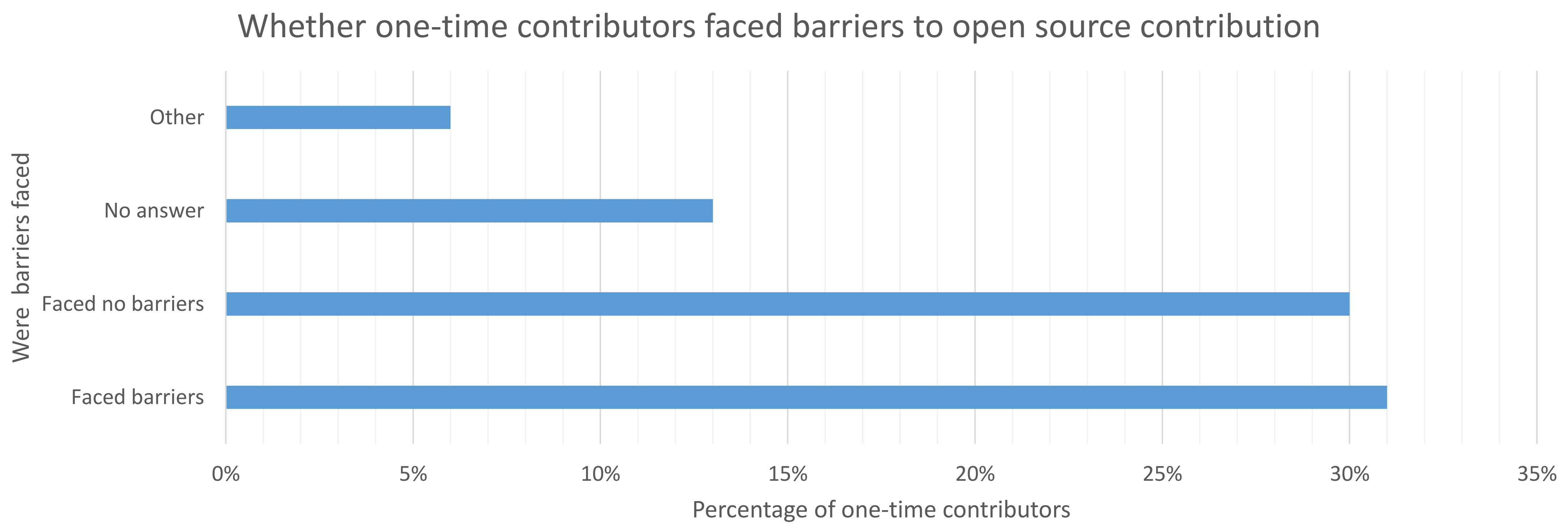
(graphs reproduced for purpose of readability in slides)



Amanda Lee, Jeffrey C. Carver, and Amiangshu Bosu. 2017. Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In Proceedings of the 39th International Conference on Software Engineering (ICSE '17). IEEE Press, Piscataway, NJ, USA, 187-197. DOI: https://doi.org/10.1109/ICSE.2017.25

Research into open source contributions

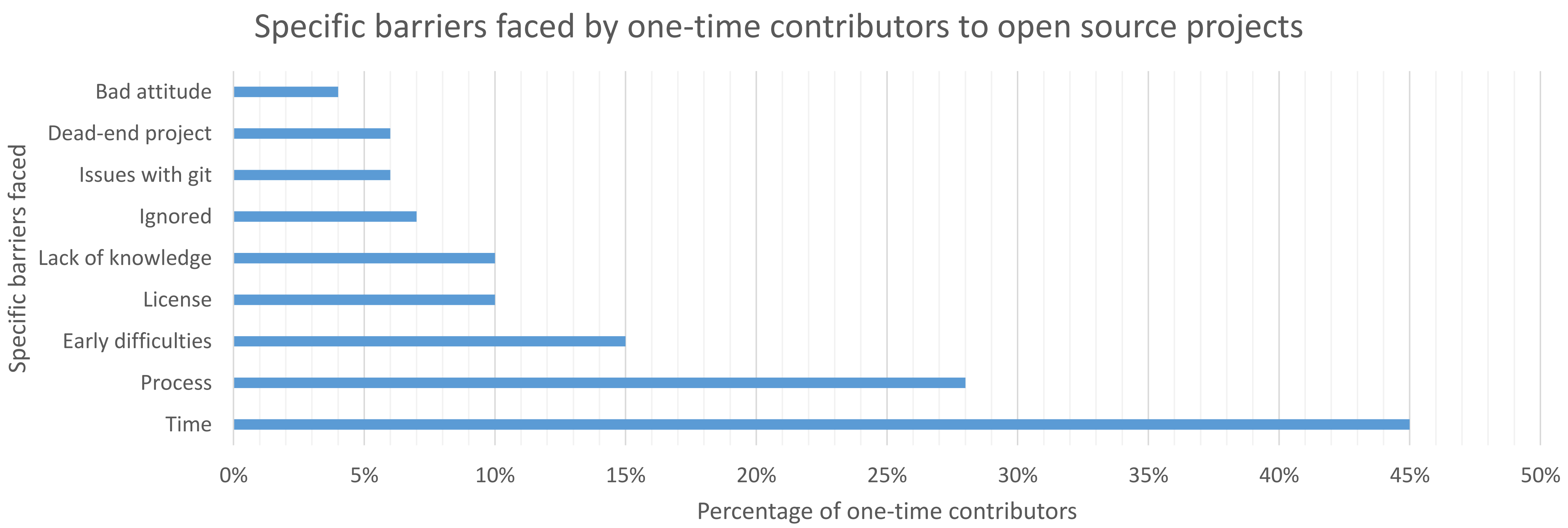
(graphs reproduced for purpose of readability in slides)



Amanda Lee, Jeffrey C. Carver, and Amiangshu Bosu. 2017. Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In Proceedings of the 39th International Conference on Software Engineering (ICSE '17). IEEE Press, Piscataway, NJ, USA, 187-197. DOI: https://doi.org/10.1109/ICSE.2017.25

Research into open source contributions

(graphs reproduced for purpose of readability in slides)



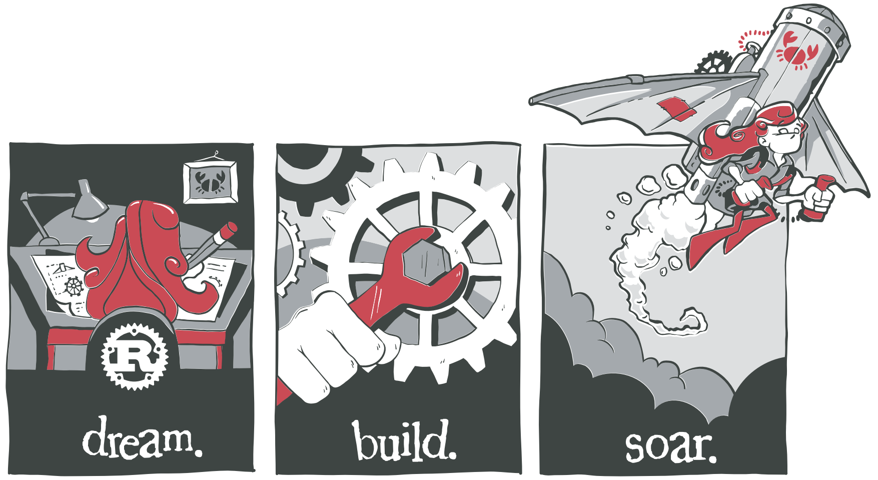
Amanda Lee, Jeffrey C. Carver, and Amiangshu Bosu. 2017. Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In Proceedings of the 39th International Conference on Software Engineering (ICSE '17). IEEE Press, Piscataway, NJ, USA, 187-197. DOI: https://doi.org/10.1109/ICSE.2017.25

What is Rust?

“You might have seen this slide coming…”

What is Rust?

Rust is a systems programming language that runs blazingly fast, prevents segfaults and guarantees thread safety.



- Zero cost abstractions
- Move semantics
- Guaranteed memory safety
- Threads without data races
- Trait-based generics
- Pattern matching
- Type inference
- Minimal runtime
- Efficient C bindings

Who uses Rust?

Deliveroo use Rust to quickly make assignment decisions in the food delivery network.

Canonical use Rust for everything from server monitoring to middleware.

Atlassian use Rust to analyse petabytes of source code.

Braintree is a payment processor using Rust to speed up batch processing and in command-line utilities.

Mozilla use Rust in building the Servo browser engine, integrating into Firefox, and other projects.

Samsung use Rust in memory-safe embedded applications in the SmartThings Hub and supporting cloud services.

Who uses Rust?

npm use Rust for replacing C and rewriting performance-critical bottlenecks in the registry service architecture.

Cloudflare use Rust as a replacement for memory-unsafe languages and are using it in their core edge logic.

Dropbox use Rust for synchronization logic in their desktop client and to optimize the storage of file data.

Discord use Rust to power the backend of their new game store, game SDK and multiplayer network layer.

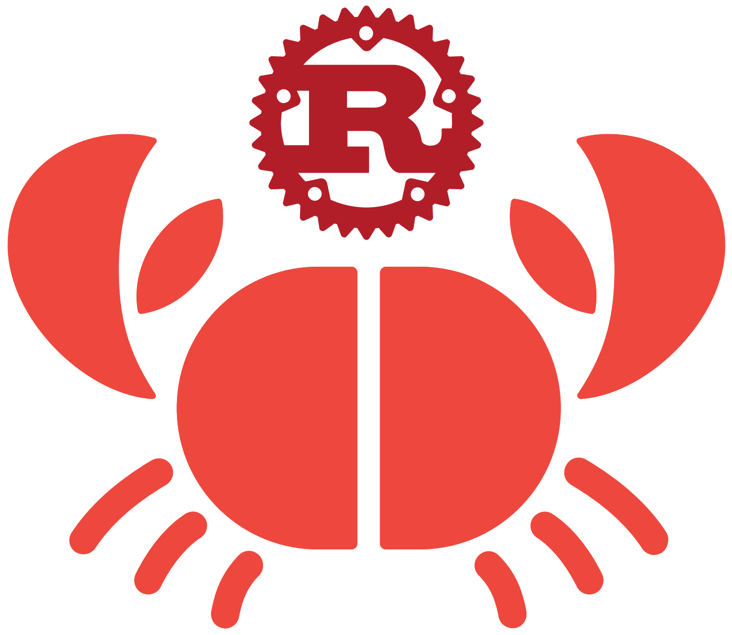
Google use Rust in "Garnet" layer of their new Fuchsia operating system and in other projects.

Chucklefish, developers of Starbound, use Rust in their next game, code-named Witchbrook.

What does this tutorial have to do with Rust?

I’ve been contributing to Rust for a while and will use my experiences with Rust to illustrate some lessons about making open source contributions.

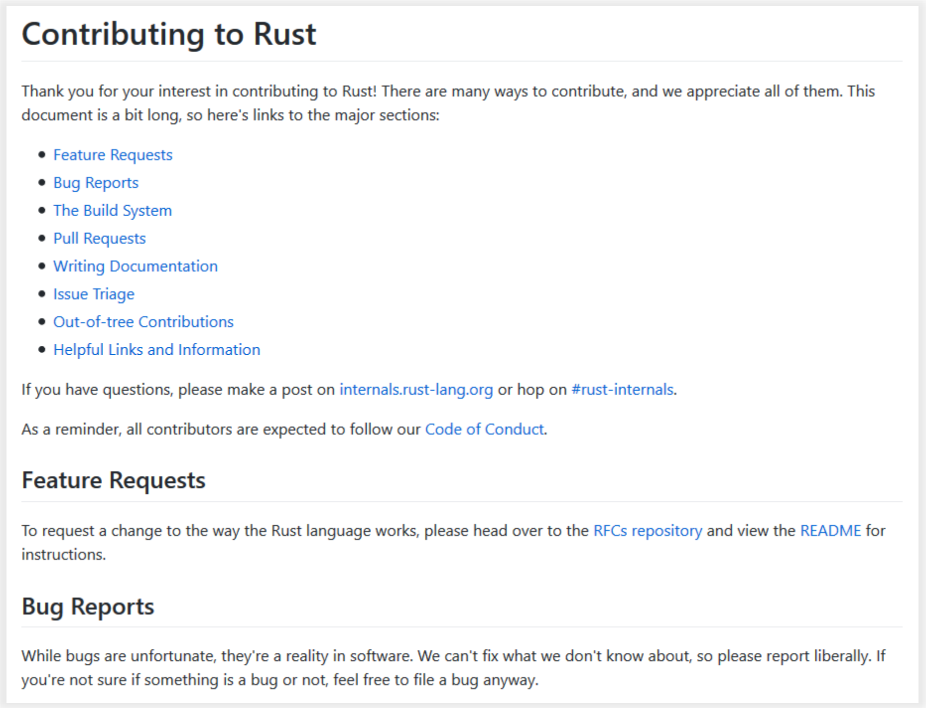
Rust is known for having a welcoming environment for new contributors and is ideal for learning about compilers and low-level systems languages.



Lessons learned from contributing to Rust

“I guess this is why he explained what Rust was..”

Lessons learned from contributing to Rust



Contributing to Rust

Thank you for your interest in contributing to Rust! There are many ways to contribute, and we appreciate all of them. This document is a bit long, so here's links to the major sections:

- Feature Requests
- Bug Reports
- The Build System
- Pull Requests
- Writing Documentation
- Issue Triage
- Out-of-tree Contributions
- Helpful Links and Information

If you have questions, please make a post on internals.rust-lang.org or hop on #rust-internals.

As a reminder, all contributors are expected to follow our Code of Conduct.

Feature Requests

To request a change to the way the Rust language works, please head over to the RFCs repository and view the README for instructions.

Bug Reports

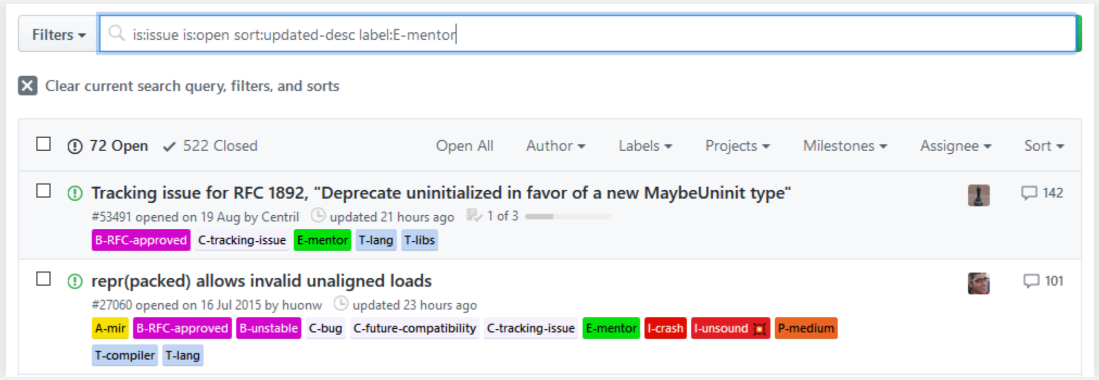
While bugs are unfortunate, they're a reality in software. We can't fix what we don't know about, so please report liberally. If you're not sure if something is a bug or not, feel free to file a bug anyway.

Process and workflow

Many large open source projects use similar workflows to software companies.

This process is often documented in a `CONTRIBUTING` file within the repository, but it varies for each project. Make sure to read the documentation for a project before contributing, this will help avoid some headaches.

Lessons learned from contributing to Rust



Filters ▾ is:issue is:open sort:updated-desc label:E-mentor

Clear current search query, filters, and sorts

72 Open ✓ 522 Closed Open All Author ▾ Labels ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

Tracking issue for RFC 1892, "Deprecate uninitialized in favor of a new MaybeUninit type" 142
#53491 opened on 19 Aug by Centril updated 21 hours ago 1 of 3
B-RFC-approved C-tracking-issue E-mentor T-lang T-libs

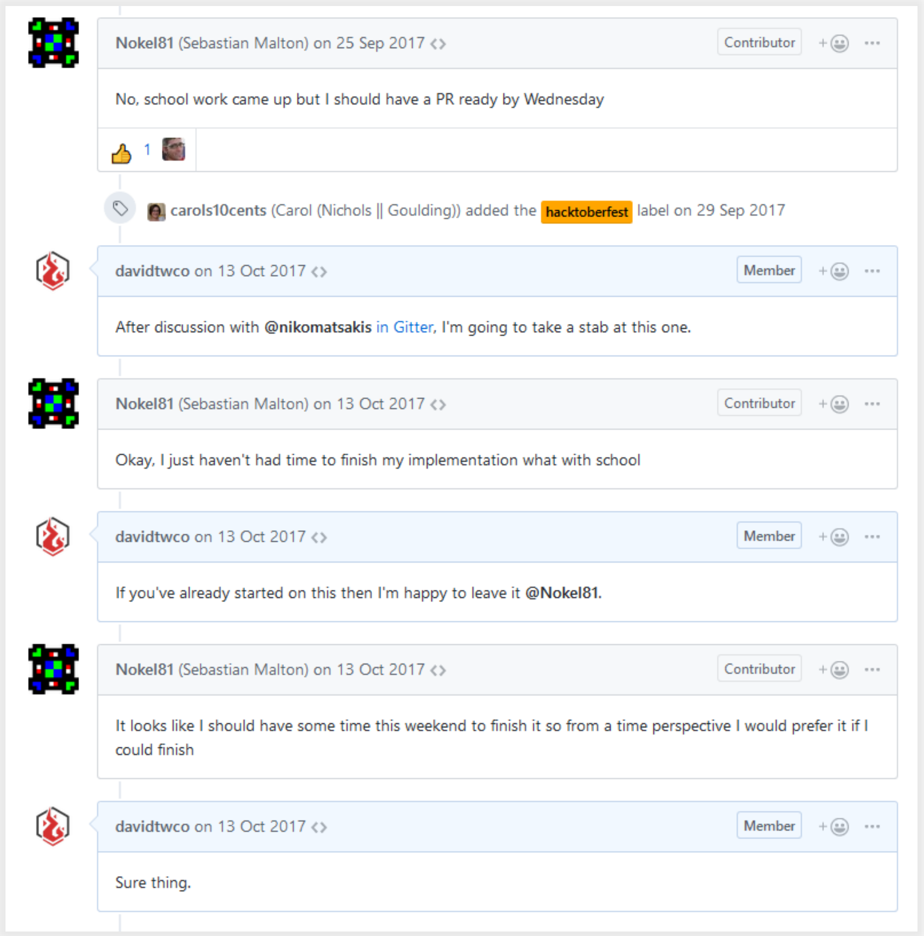
repr(packed) allows invalid unaligned loads 101
#27060 opened on 16 Jul 2015 by huonw updated 23 hours ago
A-mir B-RFC-approved B-unstable C-bug C-future-compatibility C-tracking-issue E-mentor I-crash I-unsound P-medium T-compiler T-lang

Finding a good issue to work on

Most large projects will have a subset of issues specifically labelled for new contributors.

If someone has commented suggesting they're going to work on the issue weeks ago, it's fine to reply asking if they're still working on it.

Lessons learned from contributing to Rust



Nokel81 (Sebastian Malton) on 25 Sep 2017 Contributor

No, school work came up but I should have a PR ready by Wednesday

👍 1

carols10cents (Carol (Nichols || Goulding)) added the hacktoberfest label on 29 Sep 2017

davidtwco on 13 Oct 2017 Member

After discussion with @nikomatsakis in Gitter, I'm going to take a stab at this one.

Nokel81 (Sebastian Malton) on 13 Oct 2017 Contributor

Okay, I just haven't had time to finish my implementation what with school

davidtwco on 13 Oct 2017 Member

If you've already started on this then I'm happy to leave it @Nokel81.

Nokel81 (Sebastian Malton) on 13 Oct 2017 Contributor

It looks like I should have some time this weekend to finish it so from a time perspective I would prefer it if I could finish

davidtwco on 13 Oct 2017 Member

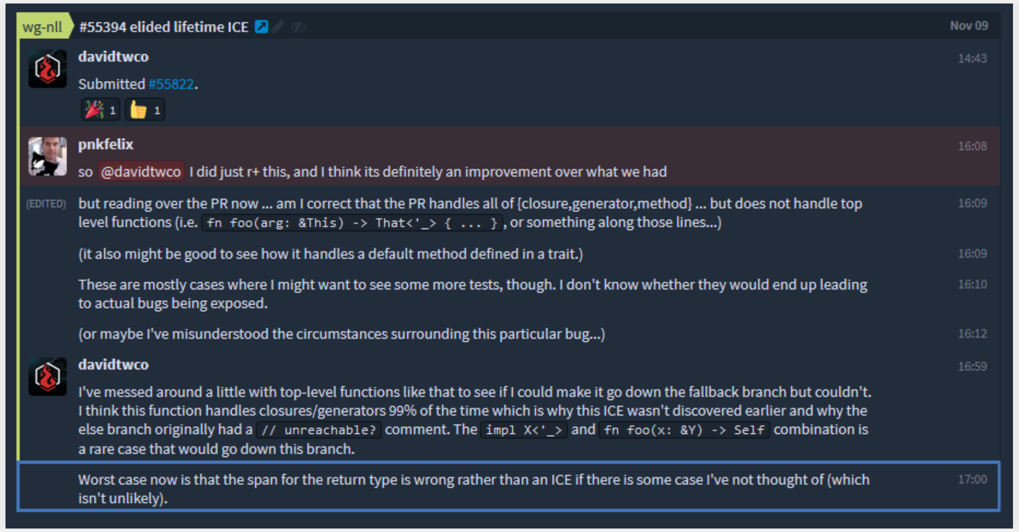
Sure thing.

rust-lang/rust#44495

Consider asking which issue would be good to start with in the project's IRC channel, Zulip or Discord.

Starting with a small project may seem less daunting, but it is often better to contribute to a larger project that is more likely to have resources available to dedicate to new contributors.

Lessons learned from contributing to Rust



wg-nll #55394 elided lifetime ICE Nov 09

davidtwco 14:43

Submitted #55822.

🎉 1 👍 1

pnkfelix 16:08

so @davidtwco I did just r+ this, and I think its definitely an improvement over what we had

(EDITED) but reading over the PR now ... am I correct that the PR handles all of {closure,generator,method} ... but does not handle top level functions (i.e. `fn foo(arg: &This) -> That<'_> { ... }`, or something along those lines...) 16:09

(it also might be good to see how it handles a default method defined in a trait.) 16:09

These are mostly cases where I might want to see some more tests, though. I don't know whether they would end up leading to actual bugs being exposed. 16:10

(or maybe I've misunderstood the circumstances surrounding this particular bug...) 16:12

davidtwco 16:59

I've messed around a little with top-level functions like that to see if I could make it go down the fallback branch but couldn't. I think this function handles closures/generators 99% of the time which is why this ICE wasn't discovered earlier and why the else branch originally had a `// unreachable?` comment. The `impl X<'_>` and `fn foo(x: &Y) -> Self` combination is a rare case that would go down this branch.

Worst case now is that the span for the return type is wrong rather than an ICE if there is some case I've not thought of (which isn't unlikely). 17:00

Communicating with other contributors

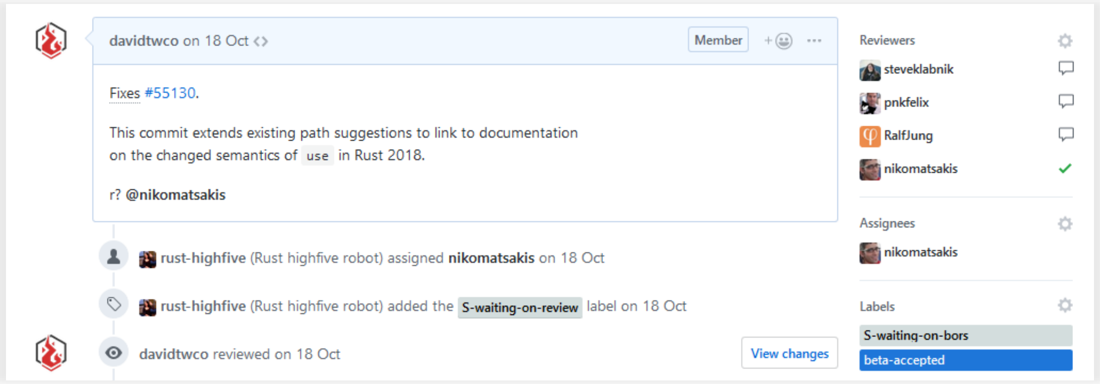
It’s important to remember that most contributors and maintainers of projects – even large ones – are volunteers, often juggling contributing with full-time work.

Often a maintainer will only have a limited amount of time in any given week to respond to issues or pull requests.

Lessons learned from contributing to Rust

- Keep realistic expectations about how quickly your messages will get replied to – it could be hours before anyone gets a chance to respond.
- Don’t repeatedly ping and prod a maintainer that you want a review or help from.
- It is better to ask someone a question rather than ask “are you around right now?” – if they aren’t available to respond right now then when they do see the message, they’ll not be able to answer your question and the communication will require another round trip to find out what you want to know.

Lessons learned from contributing to Rust



davidtwco on 18 Oct

Member

Fixes #55130.

This commit extends existing path suggestions to link to documentation on the changed semantics of `use` in Rust 2018.

r? @nikomatsakis

rust-highfive (Rust highfive robot) assigned nikomatsakis on 18 Oct

rust-highfive (Rust highfive robot) added the S-waiting-on-review label on 18 Oct

davidtwco reviewed on 18 Oct

View changes

Reviewers
- steveklabnik
- pnkfelix
- RalfJung
- nikomatsakis ✓

Assignees
- nikomatsakis

Labels
- S-waiting-on-bors
- beta-accepted

rust-lang/rust#55185

Requesting reviewers

Different projects have different mechanisms to request reviewers.

Smaller projects will often have no explicit mechanism for requesting a reviewer and the owner of the project will just comment when they have time.

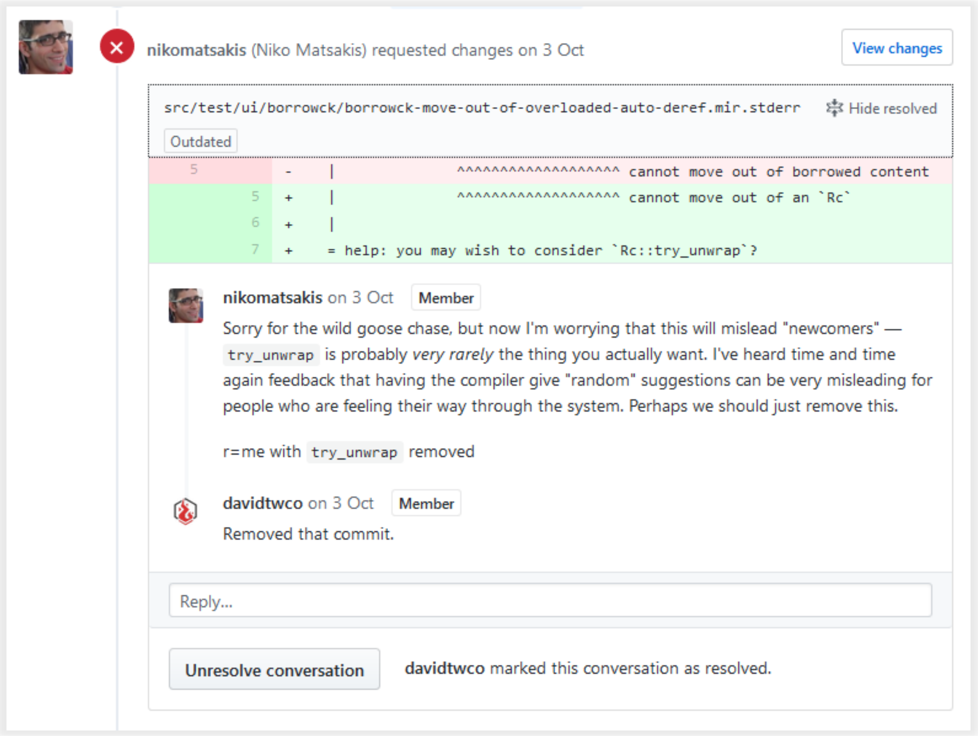
Lessons learned from contributing to Rust

Larger projects might require contributors request reviews from relevant maintainers for their change. There will often be documentation available about who is best for which parts of the codebase. GitHub will also suggest reviewers based on recent activity in related files.

Some projects will automatically assign reviewers using bots and some heuristic. There will often be a mechanism to request specific reviewers – this is useful if you want to request the author of mentoring instructions as a reviewer.

Different projects will require a different number of reviewers.

Lessons learned from contributing to Rust



nikomatsakis (Niko Matsakis) requested changes on 3 Oct

src/test/ui/borrowck/borrowck-move-out-of-overloaded-auto-deref.mir.stderr

```
5      -  |              ^^^^^^^^^^^^^^^^^^^ cannot move out of borrowed content
   5   +  |              ^^^^^^^^^^^^^^^^^^^ cannot move out of an `Rc`
   6   +  |
   7   +  = help: you may wish to consider `Rc::try_unwrap`?
```

nikomatsakis on 3 Oct Member

Sorry for the wild goose chase, but now I'm worrying that this will mislead "newcomers" — `try_unwrap` is probably *very rarely* the thing you actually want. I've heard time and time again feedback that having the compiler give "random" suggestions can be very misleading for people who are feeling their way through the system. Perhaps we should just remove this.

r=me with `try_unwrap` removed

davidtwco on 3 Oct Member

Removed that commit.

davidtwco marked this conversation as resolved.

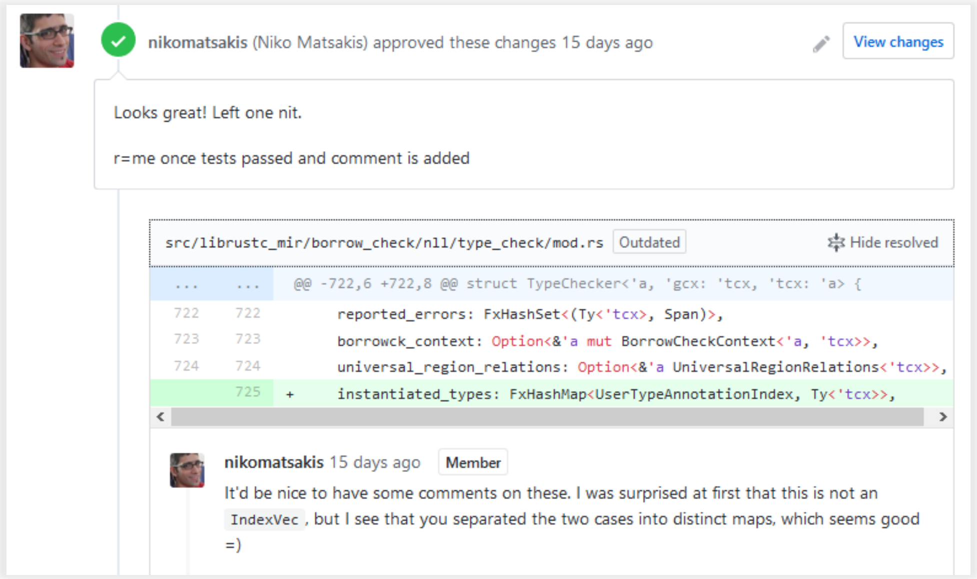
rust-lang/rust#54703

Responding to pull request feedback

Reviewers will often leave comments requesting changes or approving the pull request.

Often reviewers will be happy to explain or clarify their feedback.

Lessons learned from contributing to Rust



nikomatsakis (Niko Matsakis) approved these changes 15 days ago

View changes

Looks great! Left one nit.

r=me once tests passed and comment is added

src/librustc_mir/borrow_check/nll/type_check/mod.rs Outdated

Hide resolved

```
@@ -722,6 +722,8 @@ struct TypeChecker<'a, 'gcx: 'tcx, 'tcx: 'a> {
        reported_errors: FxHashSet<(Ty<'tcx>, Span)>,
        borrowck_context: Option<&'a mut BorrowCheckContext<'a, 'tcx>>,
        universal_region_relations: Option<&'a UniversalRegionRelations<'tcx>>,
+       instantiated_types: FxHashMap<UserTypeAnnotationIndex, Ty<'tcx>>,
```

nikomatsakis 15 days ago Member

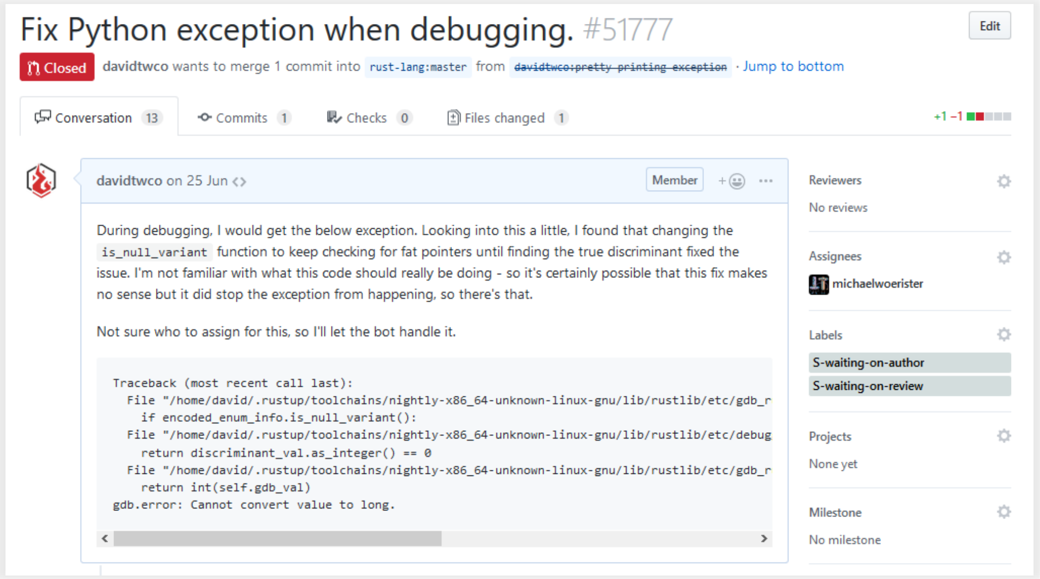
It'd be nice to have some comments on these. I was surprised at first that this is not an `IndexVec`, but I see that you separated the two cases into distinct maps, which seems good =)

rust-lang/rust#55937

Some projects will require that contributions conform to a specific style. This can vary from all formatting to just line length checks. There is often tooling available to automate this.

Reviewers often won't expect changes to be made immediately and it can often take a few days after the first review for your new changes to be reviewed.

Lessons learned from contributing to Rust



Fix Python exception when debugging. #51777

Closed davidtwco wants to merge 1 commit into `rust-lang:master` from ~~`davidtwco:pretty-printing-exception`~~ · Jump to bottom

davidtwco on 25 Jun

During debugging, I would get the below exception. Looking into this a little, I found that changing the `is_null_variant` function to keep checking for fat pointers until finding the true discriminant fixed the issue. I'm not familiar with what this code should really be doing - so it's certainly possible that this fix makes no sense but it did stop the exception from happening, so there's that.

Not sure who to assign for this, so I'll let the bot handle it.

```
Traceback (most recent call last):
  File "/home/david/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib/etc/gdb_r
    if encoded_enum_info.is_null_variant():
  File "/home/david/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib/etc/debug
    return discriminant_val.as_integer() == 0
  File "/home/david/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib/etc/gdb_r
    return int(self.gdb_val)
gdb.error: Cannot convert value to long.
```

Reviewers
No reviews

Assignees
michaelwoerister

Labels
S-waiting-on-author
S-waiting-on-review

Projects
None yet

Milestone
No milestone

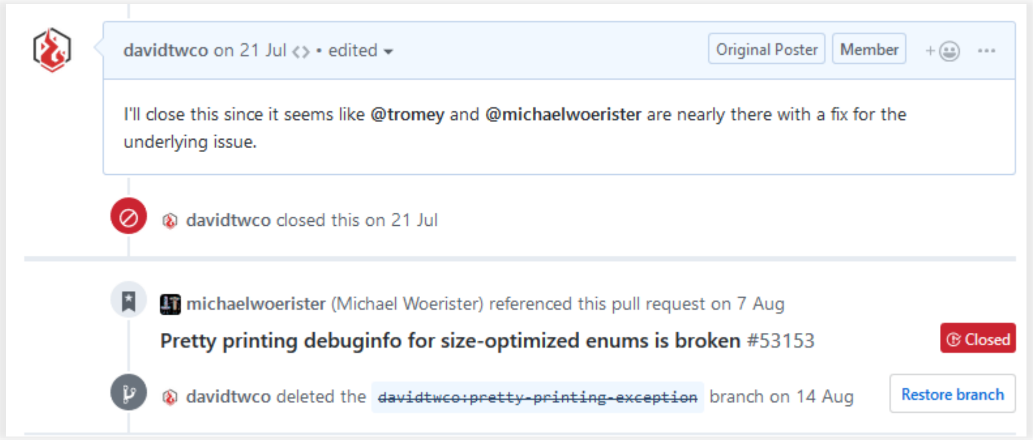
rust-lang/rust#51777

Handling rejected contributions

Sometimes pull requests get rejected, don't be discouraged or take it personally.

It's always a good idea to double check with a maintainer before starting on a issue!

Lessons learned from contributing to Rust



davidtwco on 21 Jul • edited · Original Poster · Member

I'll close this since it seems like @tromey and @michaelwoerister are nearly there with a fix for the underlying issue.

davidtwco closed this on 21 Jul

michaelwoerister (Michael Woerister) referenced this pull request on 7 Aug

Pretty printing debuginfo for size-optimized enums is broken #53153 · Closed

davidtwco deleted the ~~davidtwco:pretty-printing-exception~~ branch on 14 Aug · Restore branch

rust-lang/rust#51777

Rejections can happen for a variety of reasons:

- A change conflicts with planned work on the project.
- A fix doesn't solve the underlying issue.
- The project isn't interested in expanding scope to include the feature you've added.

If there's larger work or another fix planned, ask to be involved.

Rust’s techniques to encourage contributions

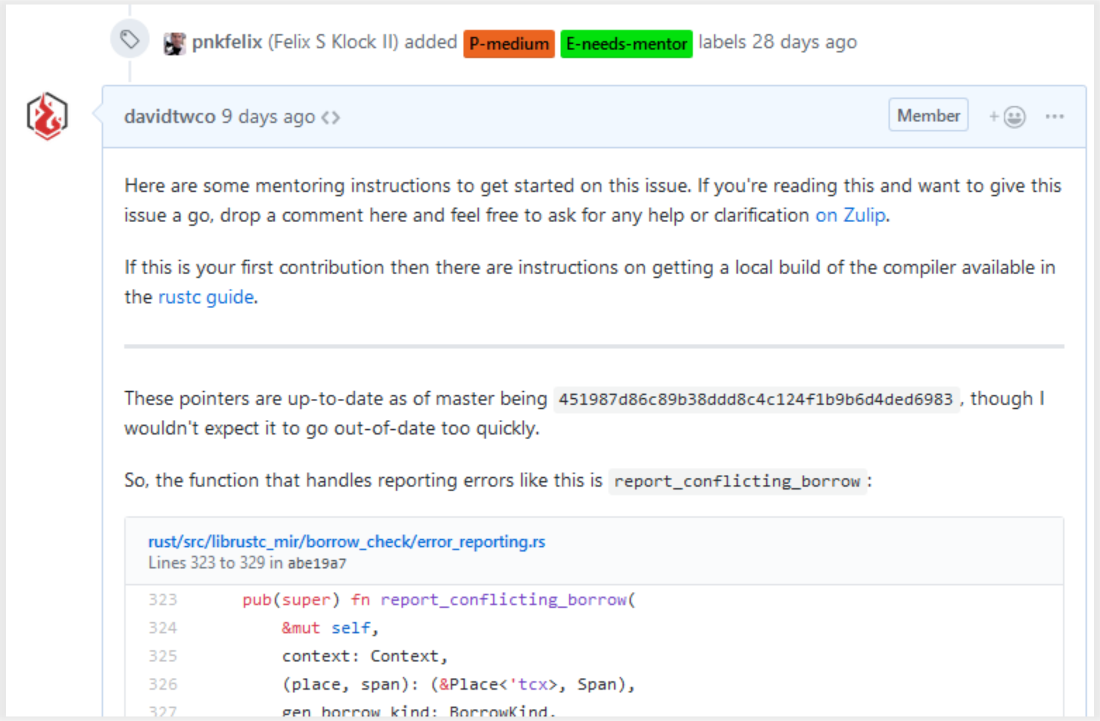
“I’m starting to notice a theme…”

Rust’s techniques to encourage contributions

Mentored issues

Issues tagged with *E-mentor* have mentoring instructions intended for new contributors, letting newcomers get started independently.

There are also the *E-easy* and the *E-help-wanted* labels!



pnkfelix (Felix S Klock II) added P-medium E-needs-mentor labels 28 days ago

davidtwco 9 days ago Member

Here are some mentoring instructions to get started on this issue. If you're reading this and want to give this issue a go, drop a comment here and feel free to ask for any help or clarification on Zulip.

If this is your first contribution then there are instructions on getting a local build of the compiler available in the rustc guide.

These pointers are up-to-date as of master being `451987d86c89b38ddd8c4c124f1b9b6d4ded6983`, though I wouldn't expect it to go out-of-date too quickly.

So, the function that handles reporting errors like this is `report_conflicting_borrow`:

rust/src/librustc_mir/borrow_check/error_reporting.rs
Lines 323 to 329 in abe19a7

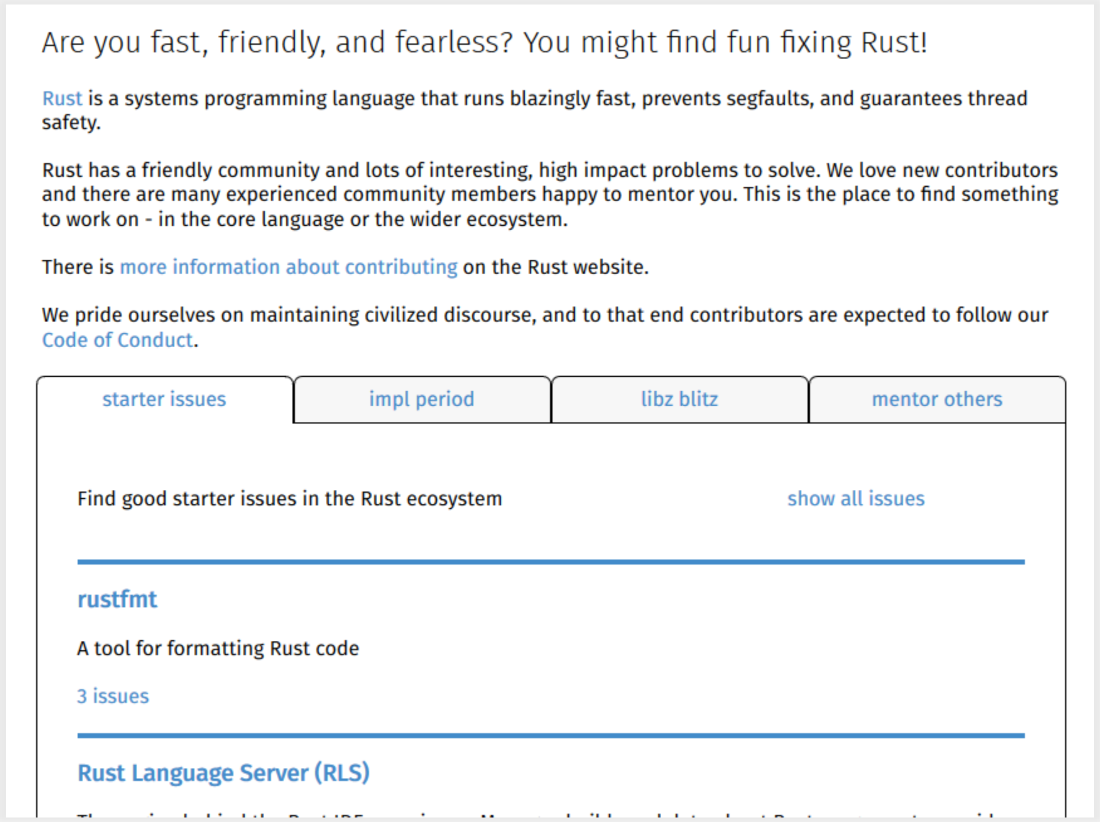
```
    pub(super) fn report_conflicting_borrow(
        &mut self,
        context: Context,
        (place, span): (&Place<'tcx>, Span),
        gen_borrow_kind: BorrowKind,
```

Rust’s techniques to encourage contributions

Impl period

At the end of 2017, Rust held its first “impl period” where the focus shifted to implementing accepted RFCs and attracting new contributors of all skill levels and experience.

It launched the “findwork” site to make it easy to start contributing to the ecosystem.



Are you fast, friendly, and fearless? You might find fun fixing Rust!

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

Rust has a friendly community and lots of interesting, high impact problems to solve. We love new contributors and there are many experienced community members happy to mentor you. This is the place to find something to work on - in the core language or the wider ecosystem.

There is more information about contributing on the Rust website.

We pride ourselves on maintaining civilized discourse, and to that end contributors are expected to follow our Code of Conduct.

Find good starter issues in the Rust ecosystem

show all issues

rustfmt

A tool for formatting Rust code

3 issues

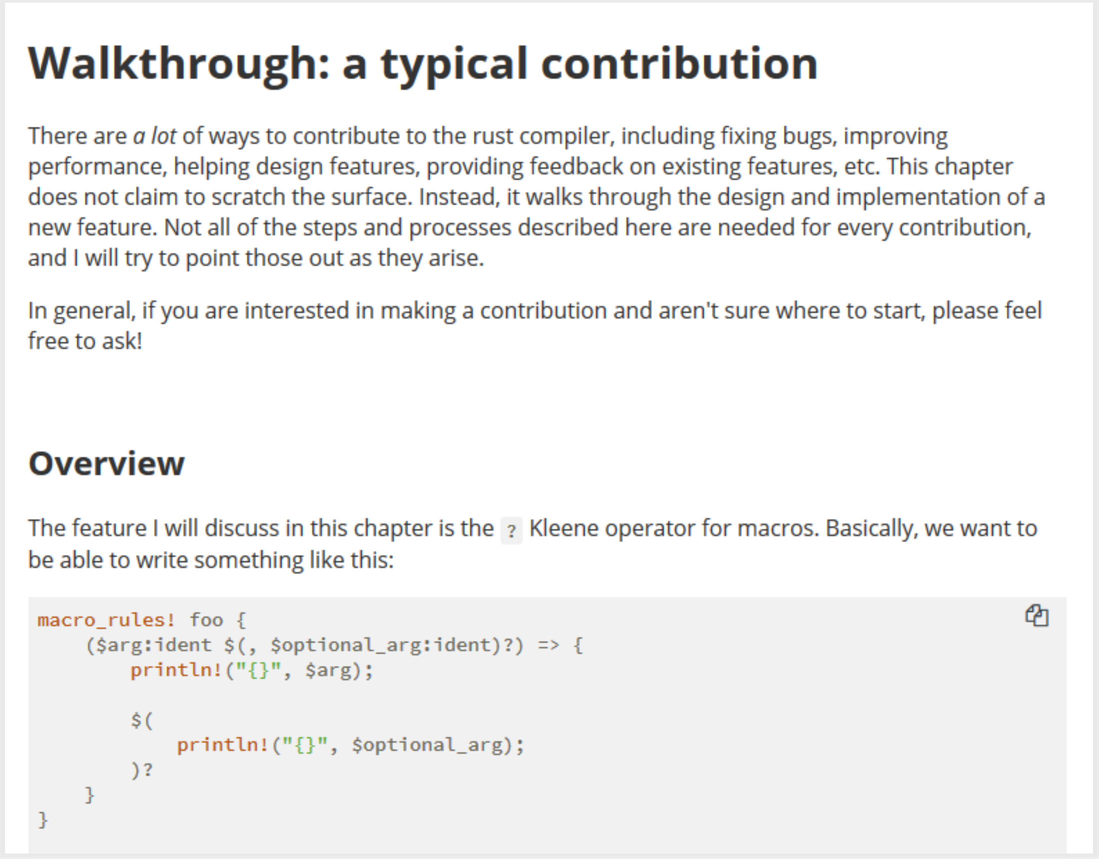
Rust Language Server (RLS)

Rust's techniques to encourage contributions

rustc guide

The rustc guide is a book explaining the internals of the compiler.

It covers simpler topics, such as how to add tests, to deep explanations of complicated compiler internals.



Walkthrough: a typical contribution

There are *a lot* of ways to contribute to the rust compiler, including fixing bugs, improving performance, helping design features, providing feedback on existing features, etc. This chapter does not claim to scratch the surface. Instead, it walks through the design and implementation of a new feature. Not all of the steps and processes described here are needed for every contribution, and I will try to point those out as they arise.

In general, if you are interested in making a contribution and aren't sure where to start, please feel free to ask!

Overview

The feature I will discuss in this chapter is the `?` Kleene operator for macros. Basically, we want to be able to write something like this:

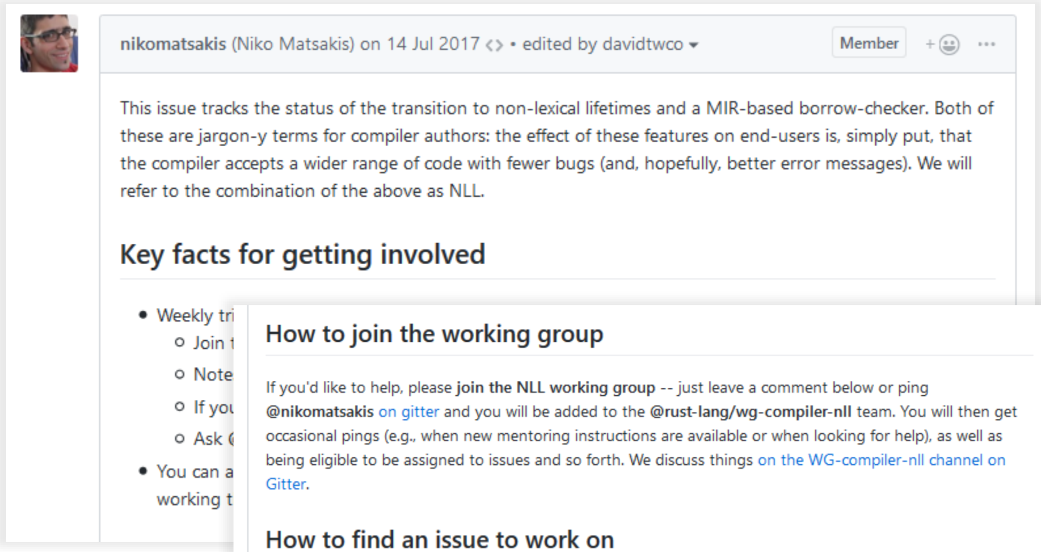
```
macro_rules! foo {
    ($arg:ident $(, $optional_arg:ident)?) => {
        println!("{}", $arg);

        $(
            println!("{}", $optional_arg);
        )?
    }
}
```

Rust’s techniques to encourage contributions

Working Groups

Rust has a handful of working groups that focus on specific parts of the project.

Each working group has a leader who can divide and coordinate work and be a point of contact for new contributors who have questions and need help.

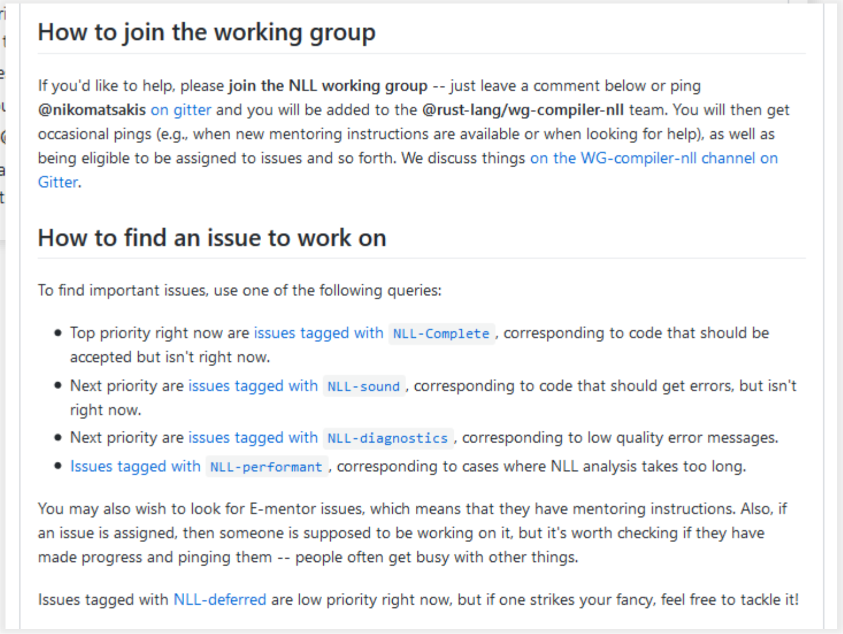


nikomatsakis (Niko Matsakis) on 14 Jul 2017 <> • edited by davidtwco ▾ Member

This issue tracks the status of the transition to non-lexical lifetimes and a MIR-based borrow-checker. Both of these are jargon-y terms for compiler authors: the effect of these features on end-users is, simply put, that the compiler accepts a wider range of code with fewer bugs (and, hopefully, better error messages). We will refer to the combination of the above as NLL.

Key facts for getting involved

- Weekly tri
  - Join t
  - Note
  - If you
  - Ask @
- You can a working t



How to join the working group

If you'd like to help, please **join the NLL working group** -- just leave a comment below or ping **@nikomatsakis** on gitter and you will be added to the **@rust-lang/wg-compiler-nll** team. You will then get occasional pings (e.g., when new mentoring instructions are available or when looking for help), as well as being eligible to be assigned to issues and so forth. We discuss things on the WG-compiler-nll channel on Gitter.

How to find an issue to work on

To find important issues, use one of the following queries:

- Top priority right now are issues tagged with `NLL-Complete`, corresponding to code that should be accepted but isn't right now.
- Next priority are issues tagged with `NLL-sound`, corresponding to code that should get errors, but isn't right now.
- Next priority are issues tagged with `NLL-diagnostics`, corresponding to low quality error messages.
- Issues tagged with `NLL-performant`, corresponding to cases where NLL analysis takes too long.

You may also wish to look for E-mentor issues, which means that they have mentoring instructions. Also, if an issue is assigned, then someone is supposed to be working on it, but it's worth checking if they have made progress and pinging them -- people often get busy with other things.

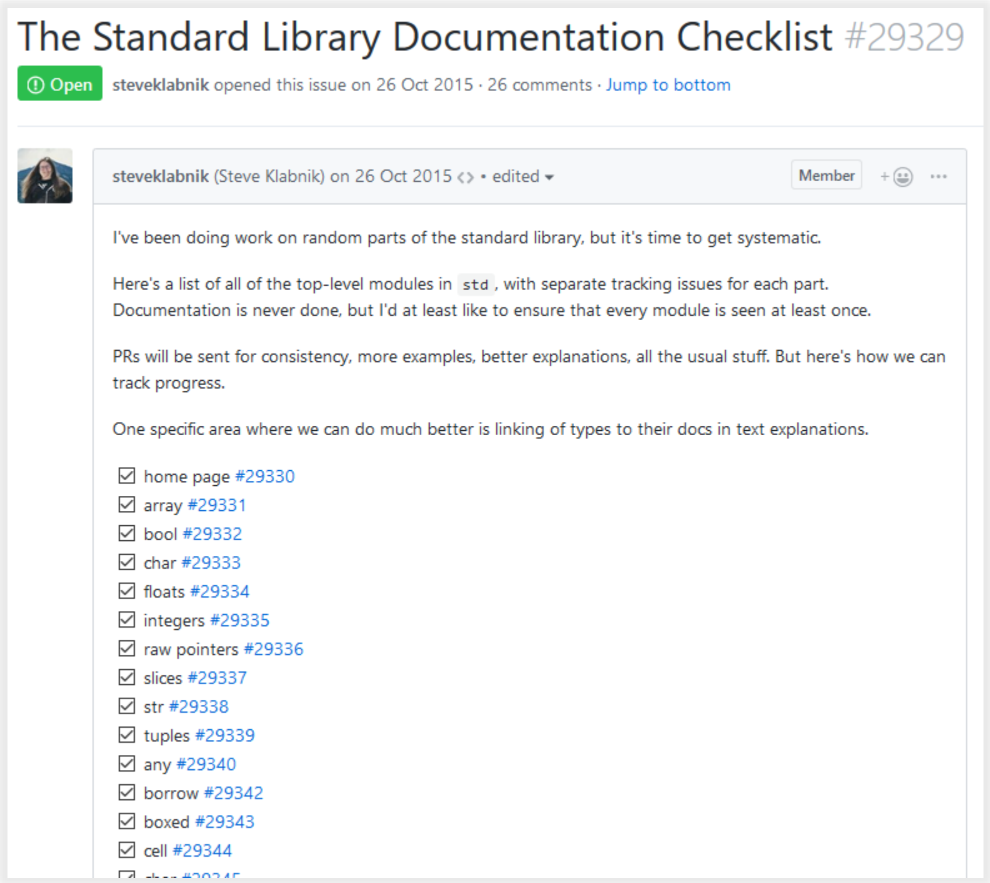
Issues tagged with NLL-deferred are low priority right now, but if one strikes your fancy, feel free to tackle it!

Rust’s techniques to encourage contributions

Quest issues

Quest issues break large tasks into small repetitive chunks that are easy for new contributors to jump in and tick off some items.

Instructions are often provided that apply to all the checklist items. This is a great way to crowdsource large tasks and are often wrapped up really quickly!



The Standard Library Documentation Checklist #29329

Open steveklabnik opened this issue on 26 Oct 2015 · 26 comments · Jump to bottom

steveklabnik (Steve Klabnik) on 26 Oct 2015 • edited Member

I've been doing work on random parts of the standard library, but it's time to get systematic.

Here's a list of all of the top-level modules in `std`, with separate tracking issues for each part. Documentation is never done, but I'd at least like to ensure that every module is seen at least once.

PRs will be sent for consistency, more examples, better explanations, all the usual stuff. But here's how we can track progress.

One specific area where we can do much better is linking of types to their docs in text explanations.

- ☑ home page #29330
- ☑ array #29331
- ☑ bool #29332
- ☑ char #29333
- ☑ floats #29334
- ☑ integers #29335
- ☑ raw pointers #29336
- ☑ slices #29337
- ☑ str #29338
- ☑ tuples #29339
- ☑ any #29340
- ☑ borrow #29342
- ☑ boxed #29343
- ☑ cell #29344

How you can get started in open source?

“I’m sold, how do I get started?”

How you can get started in open source?

24 Pull Requests

Contribute to a open source project every day before Christmas!

“You’ve been benefiting from the use of open source projects all year. Now is the time to say thanks to the maintainers of those projects, and a little birdy tells me that they love receiving contributions!”

https://24pullrequests.com

How you can get started in open source?

First timers only

An initiative to encourage projects to add `first-timers-only` labels to issues that are ideal for new contributors.

Some contributions will be small and better for getting used to the contribution process than the project itself.

FIRST TIMERS ONLY
Friendly Open Source projects should reserve specific issues for newbies.

https://www.firsttimersonly.com/

How you can get started in open source?

Contribute to Rust!

There are lots of opportunities to get started within the Rust ecosystem. I’ve written some mentoring instructions for issues you can get started with:

- #56654 (missing a note describing the type of the non-Copy moved variable)
- #56657 (missing indication that a "partial move" has occurred)

How you can get started in open source?

Contribute to something else!

Projects have varying levels of support for new contributors, you should find a project that matches your interests and has suitable provisions for new contributors.

Conclusion

You should contribute to open source!

Through contributing to Rust, I’ve become more knowledgeable software engineer, met great people, learned more about the inner workings of compilers and gained opportunities I wouldn’t have otherwise.

Whether you want to scratch an itch, fix a bug or gain more experience in your field, consider contributing to open source!

Thanks for listening!

Any questions?

@davidtwco

@davidtwco

david@davidtw.co

https://davidtw.co