Quotes of the week

2013-06-07

No Quote of the Week.

2013-06-15

No Quote of the Week.

2013-06-22

No Quote of the Week.

2013-06-29

No Quote of the Week.

2013-07-05

No Quote of the Week.

2013-07-06

No Quote of the Week.

2013-07-13

No Quote of the Week.

2013-07-21

No Quote of the Week.

2013-07-29

No Quote of the Week.

2013-08-04

No Quote of the Week.

2013-08-10

No Quote of the Week.

2013-08-19

No Quote of the Week.

2013-08-25

No Quote of the Week.

2013-08-31

No Quote of the Week.

2013-09-07

No Quote of the Week.

2013-09-15

No Quote of the Week.

2013-09-23

No Quote of the Week.

2013-09-30

No Quote of the Week.

2013-10-06

No Quote of the Week.

2013-10-06

No Quote of the Week.

2013-10-12

No Quote of the Week.

2013-10-19

No Quote of the Week.

2013-10-28

No Quote of the Week.

2013-11-09

No Quote of the Week.

2013-11-19

No Quote of the Week.

2013-11-23

No Quote of the Week.

2013-11-30

No Quote of the Week.

2013-12-09

No Quote of the Week.

2013-12-16

No Quote of the Week.

2013-12-22

No Quote of the Week.

2013-12-30

No Quote of the Week.

2014-01-06

No Quote of the Week.

2014-01-11

No Quote of the Week.

2014-01-12

No Quote of the Week.

2014-01-18

No Quote of the Week.

2014-01-26

No Quote of the Week.

2014-02-01

No Quote of the Week.

2014-02-09

No Quote of the Week.

2014-02-15

No Quote of the Week.

2014-02-23

No Quote of the Week.

2014-03-02

No Quote of the Week.

2014-03-12

No Quote of the Week.

2014-03-15

No Quote of the Week.

2014-03-23

No Quote of the Week.

2014-03-29

No Quote of the Week.

2014-04-05

No Quote of the Week.

2014-04-12

No Quote of the Week.

2014-04-26

No Quote of the Week.

2014-05-05

No Quote of the Week.

2014-05-11

No Quote of the Week.

2014-05-17

No Quote of the Week.

2014-05-24

No Quote of the Week.

2014-06-10

No Quote of the Week.

2014-06-14

No Quote of the Week.

2014-06-22

No Quote of the Week.

2014-06-30

No Quote of the Week.

2014-08-18

No Quote of the Week.

2014-11-10

No Quote of the Week.

2014-11-17

No Quote of the Week.

2014-11-24

No Quote of the Week.

2014-12-01

No Quote of the Week.

2014-12-08

No Quote of the Week.

2014-12-15

No Quote of the Week.

2014-12-22

No Quote of the Week.

2014-12-29

No Quote of the Week.

2015-01-02

No Quote of the Week.

2015-01-12

No Quote of the Week.

2015-01-19

No Quote of the Week.

2015-01-26

No Quote of the Week.

2015-02-02

No Quote of the Week.

2015-02-09

No Quote of the Week.

2015-02-16 - Quote of the Week

"You may be wondering whether this algorithm is correct. The answer is 'sort of'." @horse_rust (after Niko).

2015-02-23 - Quote of the Week

```
<Manishearth> In other news, I have r+ on rust now :D
<Ms2ger> No good deed goes unpunished
```

From #servo. Thanks to SimonSapin for the tip.

2015-03-02 - Quote of the Week

"I must kindly ask that you please not go around telling people to disregard the rules of our community. Violations of Rule #6 will absolutely not be tolerated."

[kibwen is serious][serious] about upholding community standards.

[serious]: https://www.reddit.com/r/rust/comments/2xl9fa/meta_definitely_offtopic_what_does_the_bee_rule/cp169jw

2015-03-09 - Quote of the Week

"Fear not, this is Rust, not some scruffy loosely-typed, garbage-collected, non-blocking language!"

[Andrew Hobden, on getting acquainted with mio][mio].

[mio]: http://www.hoverbear.org/2015/03/04/getting-acquainted-with-mio/

Thanks to Johan Sigfrids for the tip.

2015-03-16 - Quote of the Week

```
< reem> I'm quite interested in discovering this HTTP/2 library, but I can't
        bring myself to read four paragraphs of small caps
```

In reference to last week's [celebration] of Terry Pratchett on /r/rust.

Thanks to bstrie for the tip.

2015-03-23 - Quote of the Week

```
<mbrubeck> the 5 stages of loss and rust
<mbrubeck> 1. type check. 2. borrow check. 3. anger. 4. acceptance. 5. rust upgrade
```

Thanks to jdm for the tip.

2015-04-06 - Quote of the Week

“Diagnostics are the UX of a compiler, and so they’re deserving of the exact same care that is put into mobile app or Web design.”

Declared by pcwalton on Hacker News.

Thanks to sanxiyn for the tip.

2015-04-13 - Quote of the Week

```
<frankmcsherry> rust is like a big bucket of solder and wire, with the promise that you can't electrocute yourself.
```

From #rust.

Thanks to BurntSushi for the tip.

2015-04-20 - Quote of the Week

“`unsafe` restricts which code could contain undefined behavior, but it doesn’t isolate the effects of that undefined behavior.” - kmc on the limits of unsafety.

Thanks to tshepang for the tip.

2015-04-27 - Quote of the Week

```
< Ms2ger> And note, unsafe code isn't for violating Rust's invariants, it's for maintaining them manually
```

Ms2ger in #rust.

Thanks to bluss for the tip.

2015-05-04 - Quote of the Week

“Ultimately, I think this all boils down to the fact that borrowck only cares about reachable values. A leaked value isn’t reachable, therefore it doesn’t matter that it had a lifetime associated with it and technically outlives that lifetime, since it’s not reachable no undefined behavior can be invoked.”

Insight from kballard on the safety of linking.

Thanks to Gankro for the tip.

2015-05-18 - Quote of the Week

“Yes, because laundry eating has evolved to be a specific design goal now; and the initial portions of the planned laundry eating API have been landed behind the #![feature(no_laundry)] gate. no_laundry should become stable in 6-8 weeks, though the more complicated portions, including DRY cleaning, Higher Kinded T-shirts, Opt-in Builtin Detergent, and Rinse Time Optimization will not be stabilized until much later.”

“We hope this benefits the Laundry as a Service community immensely.”

Manish explains Rust’s roadmap for laundry-eating.

Thanks to filsmick for the tip.

And since there were so many quotables in the last two weeks, here’s one from Evan Miller’s evaluation of Rust:

“Rust is a systems language. I’m not sure what that term means, but it seems to imply some combination of native code compilation, not being Fortran, and making no mention of category theory in the documentation.”

Thanks to ruudva for the tip.

2015-06-08 - Quote of the Week

“Dude, you can’t just tell people it’s the secret plan; then it won’t be a secret any more! You keep this up, and you’re going to get your Secret Rust Club card revoked! Then you won’t be able to get on the awesome Secret Rust Zeppelin. Don’t screw this up, man!”

Quxxy, from /r/rust.

“The 1st rule of Secret Rust Club is: you don’t talk about Secret Rust Club.

The 2nd rule of Secret Rust Club is: error: `1st rule` does not live long enough.

error: aborting due to previous error"

JakDrako, from the same thread.

Thanks to drbawb and Manishearth for the tip.

2015-06-15 - Quote of the Week

```
<Quxxy> I had a fun one in cargo script: there's currently no way in Rust
to get a file's mtime and the system time in the same time format
<Quxxy> (On Windows)
<Quxxy> You can get one in UNIX time, the other in Windows time
<Quxxy> Which have different scales and different epochs
<Quxxy> Rust: Buy Your Own Damn Batteries; What Are You, A Communist?
```

Quxxy discovers Rust's stance toward the inclusion of batteries.

Thanks to cmr for the tip.

2015-06-22

No Quote of the Week.

2015-06-29

No Quote of the Week.

2015-07-06 - Quote of the Week

"Greek constitution to be rewritten in #rustlang to deal with their ownership and borrowing problem." — @bigthingist

2015-07-13 - Quote of the Week

I think if someone placed the Rust and Go community in a room and asked them to fight, we'd probably just all order pizza and geek out over languages. — Manish Goregaokar

Thanks to msiemens for the tip.

2015-07-20 - Quote of the Week

Rust is very much about only paying for what you need, and often you don't need much, but when you do need something, Rust is more than ready to rummage in your wallet for loose change. — Manish Goregaokar

Thanks to llogiq for the tip.

2015-07-27 - Quote of the Week

Opening a vortex to Hell is actually safe, but de-referencing anything you pull from the vortex isn't safe. — Steve Klabnik

Thanks to Gankro for the tip.

2015-08-03 - Quote of the Week

> It should be noted that the authentic Rust learning experience involves writing code, having the compiler scream at you, and trying to figure out what the heck that means. I will be carefully ensuring that this occurs as frequently as possible.

From @Gankro's Learning Rust With Entirely Too Many Linked Lists.

Thanks to @carols10cents for the tip.

2015-08-10 - Quote of the Week

I've tried using unchecked indexing in non-trivial code now a couple of times. It never makes a big difference Profiling shows like 1-2% improvement if that so it's the tightest loops you should worry about, not much more

@bluss knows a few things about micro-optimization.

Thanks to @bluss for the tip.

2015-08-17 - Quote of the Week

"Any sufficiently advanced macro is indistinguishable from magic." — barosl at reddit.

Thanks to nasa42 for the tip.

2015-08-24 - Quote of the Week

"`unsafe` is as viral and pernicious as pop music, though obviously not as dangerous." — Daniel Keep at TRPLF.

Thanks to llogiq for the tip.

2015-08-31 - Quote of the Week

"And God said, Noah you must transport these animals across a large body of water... but they are not Send. And Noah replied, I shall build a great Arc!" — durka42 on #rust

Thanks to tomprogrammer for the tip.

2015-09-07 - Quote of the Week

"[Rust] the language had to dedicate so much real estate to this (difficult) problem alone, it became a disharmonic creature with one bulging muscle and little of anything else." — Andrei Alexandrescu (one of designers of D).

Thanks to llogiq for the tip.

2015-09-14 - Quote of the Week

On `#rust` IRC

> 03:46 < durka42> rust has a culture of small crates
>
> 03:47 < XMPPwocky> a Cargo cult, if you will

Thanks to Manishearth for the tip.

2015-09-21 - Quote of the Week

Transmute is taking a dog, sawing its front legs off, gluing on a pair of buffalo wings and telling it it's a duck so it damn well better start quacking You should not be surprised when you end up with a pile of gore and a dead dog instead of an actual duck :-P — Quxxy

Thanks to Ms2ger for the tip.

2015-09-28 - Quote of the Week

If one regards Rust as a critique to C++, it certainly should be seen as a constructive critique. — llogiq on /r/cpp.

Thanks to msiemens for the tip.

2015-10-05 - Quote of the Week

In programming (as opposed to politics), safety=freedom. — llogiq on /r/rust.

Thanks to birkenfeld for the tip.

2015-10-12 - Quote of the Week

> War is Unsafe
>
> Freedom is Safety
>
> Ignorance is Type-checked
>
> (The new trifecta for rust-lang.org)

— killercup on /r/rust.

Thanks to ruudva for the tip.

2015-10-19 - Quote of the Week

> S-s-s-s-stack alloc Queen, no C++ though Might drop that pointer, no nullable though Tell golang, “Yo, don’t you got enough mem to slow?” Tell ‘em Kangaroo Rust, I’ll box your flow
>
> Advanced pattern matching - possible Don’t go against Rusty - impossible Runtime will leave your CPU on popsicle Man these h*es couldn’t be any less logical

— A tribute to #NickiMinaj and #rustlang by @boghison.

2015-10-26

No Quote of the Week.

2015-11-02

No Quote of the Week.

2015-11-09 - Quote of the Week

> with **unsafe** .... **if** you have to ask, then you probably shouldn’t **be** doing it basically

— Steve Klabnik on #rust IRC.

Thanks to Oliver Schneider for the tip.

2015-11-16

No Quote of the Week.

2015-11-23

No Quote of the Week.

2015-11-30

No Quote of the Week.

2015-12-07 - Quote of the Week

> The major philosophic difference between Rust today and Swift-as-I-envision-it is that Rust forces you to think about ownership everywhere, but Swift-as-I-envision-it should only force you to think about single ownership & borrowing if you want to optimize performance or guarantee that you have no encounters with the runtime.
>
> If it helps, think of the extant Swift “inout” parameter modifier as being equivalent to “&mut”, and imagine the logical swift extensions to support the rest of the Rust model.
>
> This is a really important area for us to develop, but it also isn’t the highest priority of the team. That means that Rust will maintain a lead in this area of applicability... unless someone motivated and capable from the open source community decides that it is really important to them, and makes it happen sooner.

— Chris Lattner on /r/rust.

Thanks to llogiq for the tip.

2015-12-14 - Quote of the Week

> how do cats arbitrate access to a shared typesetting system?
>
> with a mew-TeX

— Nathaniel Theis on Twitter

Thanks to Paul for the tip.

2015-12-21 - Quote of the Week

> Do or do not. There is a try!

— Jonathan Turner in a blog post.

Thanks to Vikrant for the tip.

2015-12-28 - Quote of the Week

> You have to think about it. You don't have to worry about it.

— SamReidHughes on manual memory management in Rust.

Thanks to Barosl for the tip.

2016-01-04 - Quote of the Week

> I think the "standard library" is really (forgive me) a brand.

jimb on rust-internals

Thanks to llogiq for the belated suggestion.

2016-01-11 - Quote of the Week

> > Rustaceans are not very imaginative at naming things.
>
> We try! and try!, but sometimes, we Err.

— steveklabnik1 on /r/rust.

Thanks to msiemens for the suggestion.

2016-01-18 - Quote of the Week

> Borrow/lifetime errors are usually Rust compiler bugs. Typically, I will spend 20 minutes detailing the precise conditions of the bug, using language that understates my immense knowledge, while demonstrating sympathetic understanding of the pressures placed on a Rust compiler developer, who is also probably studying for several exams at the moment. The developer reading my bug report may not understand this stuff as well as I do, so I will carefully trace the lifetimes of each variable, where memory is allocated on the stack vs the heap, which struct or function owns a value at any point in time, where borrows begin and where they... oh yeah, actually that variable really doesn't live long enough.

— peterjoel on /r/rust.

Thanks to Wa Delma for the suggestion.

2016-01-25 - Quote of the Week

> Memory errors are fundamentally state errors, and Rust's move semantics, borrowing, and aliasing XOR mutating help enormously for me to reason about how my program changes state as it executes, to avoid accidental shared state and side effects at a distance. Rust more than any other language I know enables me to do compiler driven design. And internalizing its rules has helped me design better systems, even in other languages.

— desiringmachines on /r/rust.

Thanks to dikaiosune for the suggestion.

2016-02-01 - Quote of the Week

> imo: the opinionated version of mio

— durka42 on #rust

Thanks to Steve Klabnik for the suggestion.

2016-02-08

No Quote of the Week.

2016-02-15

No Quote of the Week.

2016-02-22 - Quote of the Week

> There is essentially no webpage out there that it cannot get through at multiple hundreds of frames-per-second

— pcwalton on Servo's astonishing new WebRender technology.

Thanks to adwhit for the suggestion.

2016-02-29 - Quote of the Week

> Haskell: When you want to feel smarter than everyone else in the universe.
>
> Rust: When you want to feel like an idiot because even a mere compiler is smarter than you.

— Manishearth on /r/rust.

Thanks to Sinistersnare for the suggestion.

2016-03-07 - Quote of the Week

No QotW were selected for this week.

2016-03-14 - Quote of the Week

> Rust is not what makes these projects awesome. These projects are what make Rust awesome.

— Manish on twitter.

Thanks to llogiq for the suggestion.

2016-03-21 - Quote of the Week

> ok and what trait is from from? From :D ok :D so from is from From nice!

— phaazon on #rust.

Thanks to Thomas Winwood for the suggestion.

2016-03-28 - Quote of the Week

No quote was selected for QotW.

2016-04-04 - Quote of the Week

> Explicitness is the fourth core value of Rust. Ironically, I don't see that "Explicitness" is ever explicitly stated as a goal of Rust.

— Ian Whitney in a blog post.

Thanks to nayru25 for the suggestion.

2016-04-11 - Quote of the Week

No quote was selected for QotW.

2016-04-25 - Quote of the Week

> > Cow is still criminally underused in a lot of code bases
>
> I suggest we make a new slogan to remedy this: "To err is human, to moo bovine." (I may or may not have shamelessly stolen this from this bug report)

so_you_like_donuts on reddit.

Thanks to killercup for the suggestion.

2016-05-02 - Quote of the Week

> In general, enough layers of Rc/RefCell will make anything work.

gkoz on TRPLF.

Thanks to birkenfeld for the suggestion.

2016-05-09 - Quote of the Week

No quote was selected for QotW.

2016-05-16 - Quote of the Week

No quote was selected for QotW.

2016-05-23 - Quote of the Week

No quote was selected for QotW.

2016-05-30 - Quote of the Week

> Rust is like doing parkour while suspended on strings & wearing protective gear. Yes, it will sometimes look a little ridiculous, but you'll be able to do all sorts of cool moves without hurting yourself.

— llogiq on reddit.

Thanks to aochagavia for the suggestion.

2016-06-06 - Quote of the Week

No quote was selected for QotW.

2016-06-13 - Quote of the Week

> > Isn't rust too difficult to be widely adopted?
>
> I believe in people.

— Steve Klabnik on TRPLF

Thanks to Steven Allen for the suggestion.

2016-06-20 - Quote of the Week

> The Rust standard libs aren't quite batteries included, but they come with a pile of adaptor cables and an optional chemistry lab.

— Gankro on Twitter

Thanks to llogiq for the suggestion.

2016-06-27 - Quote of the Week

> Rust is also really phobic of heap allocations [...] Yes, Rust encourages everyone to be a full stack developer :)

Thanks to Matt Brubeck for the suggestion.

2016-07-05 - Quote of the Week

No quote was selected for QotW.

2016-07-12 - Quote of the Week

No quote was selected for QotW.

2016-07-19 - Quote of the Week

> fzammetti: Am I the only one that finds highly ironic the naming of something that's supposed to be new and cutting-edge after a substance universally synonymous with old, dilapidated and broken down?
>
> paperelectron: Rust is as close to the bare metal as you can get.

On /r/programming.

2016-07-26 - Quote of the Week

> you have a problem. you decide to use Rust. now you have a Rc<RefCell<Box>>

kmc on #rust.

Thanks to Alex Burka for the tip.

2016-08-02 - Quote of the Week

No quote was selected for QotW.

2016-08-09 - Quote of the Week

> The `if let` construction is a neat thing Rust borrowed from Swift (perhaps "copied" would be more accurate, or "cloned" depending on your views on whether ideas have owners).

— Frank McSherry in a blog post.

Thanks to /u/vks_ and Brian Anderson for the suggestion.

2016-08-16 - Quote of the Week

> The best way to learn Rust is to just `try!` and see what works (or is this to just see what works? now?)!

— llogiq on /r/rust.

Thanks to UtherII for the suggestion.

2016-08-23 - Quote of the Week

No quote was selected for QotW.

2016-08-30 - Quote of the Week

> My Rust memoirs will be called "Once in 'a lifetime".

— @Argorak on Twitter.

Thanks to Vincent Esche for the suggestion.

2016-09-06 - Quote of the Week

No quote was selected for QotW.

2016-09-13 - Quote of the Week

No quote was selected for QotW.

2016-09-20 - Quote of the Week

No quote was selected for QotW.

(Full disclosure: we removed QotW for this issue because selected QotW was deemed inappropriate and against the core values of Rust community. Here is the relevant discussion on reddit. If you are curious, you can find the quote in git history).

2016-09-27 - Quote of the Week

> You can actually return Iterators without summoning one of the Great Old Ones now, which is pretty cool.

— /u/K900_ on reddit.

Thanks to Johan Sigfrids for the suggestion.

2016-10-04 - Quote of the Week

> My favorite new double-meaning programming phrase: "my c++ is a little rusty"

— Jake Taylor on Twitter.

Thanks to Zachary Dremann for the suggestion.

2016-10-11 - Quote of the Week

> < Celti> I just had a recruiter contact me for a Rust job requiring 3+ years of professional experience with it.

— From #rust.

Thanks to bluss for the suggestion.

2016-10-18 - Quote of the Week

> that gives a new array of errors, guess that's a good thing you passed one layer of tests, and hit the next layer :P rustc is like onions it makes you cry?

— From #rust-beginners.

Thanks to Quiet Misdreavus for the suggestion.

2016-10-25 - Quote of the Week

No quote was selected for QotW.

2016-11-01 - Quote of the Week

> Every once in a while someone discovers a bug in librsvg that makes it all the way to a CVE security advisory. We've gotten double free()s, wrong casts, and out-of-bounds memory accesses. Recently someone did fuzz-testing with some really pathological SVGs, and found interesting explosions in the library. That's the kind of 1970s bullshit that Rust prevents.

— Federico on porting librsvg to Rust.

Thanks to /u/robinst and /u/cjstevenson1 for the suggestion.

2016-11-08 - Quote of the Week

> I want to paint you a picture of a utopia in which Rust has expanded to become the fabric of the entire classical computing world, where the possibilities of what we can achieve are not shackled to the decaying dreams of computer science past. In this perfect utopia you have invented the perfect model for managing your computer's sci-fi hardware, perfectly free from the legacy of Unix and Windows. And you need the perfect language to write it in. Everywhere you look is legacy: C, C++, Java; the stacks get bigger and bigger, cruft all the way down.
>
> The only shining light is Rust. Those Rustaceans have been chipping away the cruft, distilling their platform to only the essence of bits and bytes, while also expanding its expressive power toward legendary elegance. Rust doesn't want to tell you how to build your system. Rust wants to serve you, to fulfill your dreams, on your terms. For your ambitions, Rust is the only reasonable choice in a world filled with compromises.

— brson on Refactoring std for ultimate portability.

Thanks to Japaric for the suggestion.

2016-11-15 - Quote of the Week

> Now higher-kinded types especially are one of those PL topics that sound forebodingly complex and kind of abstract (like monads). But once you learn what it is, you realize it's actually relevant to your life (unlike monads).

— @nikomatsakis invoking the M word in his blog post.

Thanks to Japaric for the suggestion.

2016-11-22 - Quote of the Week

> Rust iterators are the best thing since [Bread]

— Yaniel on Rust (lang) Matrix channel.

Thanks to Elahn for the suggestion.

2016-11-29 - Quote of the Week

No quote was selected for QotW.

2016-12-06 - Quote of the Week

> Such large. Very 128. Much bits.

— @nagisa introducing 128-bit integers in Rust.

Thanks to leodasvacas for the suggestion.

2016-12-13 - Quote of the Week

A lady warrior in heavy armour wielding three shields, one of which bears the Rust logo, saying: »I'll be designated driver tonight«. When answered »We know«, she warns: »Don't talk while eating. You could choke and die, haha.«

— Programming People by David Marino.

Thanks to skade for the suggestion.

2016-12-20 - Quote of the Week

> fmtq: I'm ridiculously good at the borrow checker though fmtq: in Rust. bstrie: once you've mastered borrow checkers, you may move on to borrow chess

— in #rust-offtopic.

Thanks to Havvy for the suggestion.

2016-12-27 - Quote of the Week

> scott: see, at christmas, if I unwrap() my present and I find that it's empty, I immediately have a panic attack and everyone gets really freaked out. if I expect() a present, at least I can send a stern message before I start panicking in the living room. scott: sometimes I decide ahead of time that if I start having a panic attack, I'm just going to abort the whole christmas thing entirely and leave scott: but it's easier to unwind if I don't abort and stick around

— in #rust-offtopic.

Thanks to Havvy for the suggestion.

2017-01-03 - Quote of the Week

> I would argue that if the Rust project would have just one mission statement, it wouldn't be "create a safe systems programming language". It would be "move towards a world where safe systems programming is the norm".

— GolDDranks in reply to steveklabnik.

Thanks to matematikaadit for the suggestion.

2017-01-10 - Quote of the Week

> This is the first version to have Rust code in it. The public API remains unchanged. Apologies in advance to distros who will have to adjust their build systems for Rust - it's like taking a one-time vaccine; you'll be better off in the end for it.

— Federico Mena Quintero announcing librsvg 2.41.0.

Thanks to Zbigniew Siciarz for the suggestion.

2017-01-17 - Quote of the Week

> I really hate the phrase "fighting". Calling it a fight doesn't do justice to the conversations you have with the borrow checker when you use Rust every day. You don't fight with the borrow checker, because there isn't a fight to win. It's far more elegant, more precise. It's fencing; you fence with the borrow checker, with ripostes and parries and well-aimed thrusts. And sometimes, you get to the end and you realize you lose anyway because the thing you were trying to do was fundamentally wrong. And it's okay, because it's just fencing, and you're a little wiser, a little better-honed, a little more practiced for your next bout.

— kaosjester on Hacker News.

Thanks to Manishearth for the suggestion.

2017-01-24 - Quote of the Week

> Yeah, it's like learning to dance when your partner [borrow checker] already knows all the steps. When you're just getting started, you step on their toes a lot, but over time you get the motions down. Eventually, you can start to anticipate their movements and start to appreciate the music as part of the dance, instead of just concentrating on getting your feet in the right place.

— QuietMisdreavus on reddit.

Thanks to matthieum for the suggestion.

2017-01-31 - Quote of the Week

> Clippy is for those of you who have become desensitized to the constant whining of the Rust compiler and need a higher dosage of whininess to be kept on their toes. Clippy is for those perfectionists amongst you who want to know every minute thing wrong with their code so that they can fix it. But really, Clippy is for everyone.

— Manishearth in a blog post.

Thanks to Johan Sigfrids for the suggestion.

2017-02-07 - Quote of the Week

> Back in these days, we had none of this newfangled “stability” business. The compiler broke your code every two weeks. Of course, you wouldn’t know that because the compiler would usually crash before it could tell you that your code was broken! Sigils roamed the lands freely, and cargo was but a newborn child which was destined to eventually end the tyranny of Makefiles.

— Manishearth in a blog post.

Thanks to llogiq for the suggestion.

2017-02-14 - Quote of the Week

No quote was selected for QotW.

2017-02-21 - Quote of the Week

No quote was selected for QotW.

2017-02-28 - Quote of the Week

> What about the Quote of the Week? I noticed it’s missing quite often these days.

— llogiq on reddit.

Thanks to tibodelor for the suggestion.

2017-03-07 - Quote of the Week

> I sure need to work on my quotes.

— llogiq on his QotW from last week.

Thanks to slashgrin for the suggestion.

2017-03-14 - Quote of the Week

> In #rustlang, None is always an Option<_>.

— llogiq on Twitter.

Thanks to Johan Sigfrids for the suggestion.

2017-03-21 - Quote of the Week

> #rustlang is a very strange place sans null deref nor data race it has its own styles but once it compiles it will not blow up in your face

— llogiq on Twitter. Check out his Twitter feed for more #rustlang limericks!

2017-03-28 - Quote of the Week

> I had many questions during the example implementations but “where do I find that” was none of them. [...] Thanks, docs team, you are doing great work!

— Florian Gilcher in a blog post.

Thanks to Jules Kerssemakers for the suggestion.

2017-04-04 - Quote of the Week

> I gave my company’s Embedded C training course this morning. It’s amazing how much more sense C makes when you explain it in Rust terms.

— theJPster in #rust-embedded.

Thanks to Oliver Schneider for the suggestion.

2017-04-11 - Quote of the Week

> Nobody expects the Rust Evangelism Strike Force! Our chief weapon is surprise, surprise and fearless concurrency... fearless concurrency and surprise... our two weapons are fearless concurrency and surprise, and ruthless efficiency our three, weapons are fearless concurrency, and surprise, and ruthless efficiency, and an almost fanatical devotion to zero-cost abstractions. Our four, no– amongst our weapons... Amongst our weaponry... are, such elements as fearless concurrency, surprise... I'll come in again.

— kibwen on reddit.

Thanks to shadow31 and KillTheMule for the suggestion.

2017-04-18 - Quote of the Week

> Rust doesn't end unsafety, it just builds a strong, high-visibility fence around it, with warning signs on the one gate to get inside. As opposed to C's approach, which was to have a sign on the periphery reading "lol good luck".

— Quxxy on reddit.

Thanks to msiemens for the suggestion.

2017-04-25 - Quote of the Week

> There are many ways in which Rust is like a version of C/C++ that mutated when Haskell was injected into its veins.

— Lokathor on reddit.

Thanks to Johan Sigfrids and liquidivy for the suggestion.

2017-05-02 - Quote of the Week

> Indeed, it was that very event that lent the world to ruin...
>
> It had been decades since the last line of C code was erased, and the last C compiler (written, ironically (but unsurprisingly), in Rust) engraved onto a golden disc and launched toward that distant star around which the GooglePepsiMusk Sphere's construction could be faintly observed. To even utter that fell syllable would be met with swift retaliation from the paramilitary *Borrow Xekkers* (the alphabet as well having been altered to better suit this shining memory-safe utopia). The whole world, everything, had all been rewritten in Rust, and at last the world knew universal peace and prosperity of boundless proportion.
>
> But something stirred... a prophecy ancient in origin. **The Second.0 Coming**. And when realized at last, the world was divided between the forward-thinking devotees of *Rust Two-Point-Oh* and those vainly clinging to *Rust One-Point-Four-Hundred-And-Seventeen*.
>
> And thus did The Great Schism rend interstellar civilization in twain.
>
> There was but one ray of hope. As the fires of war rushed toward the newly-completed GooglePepsiMusk Sphere, its cadre of elite 10^{10x} programmers set to work. Their task: to write the most advanced compound artificial intelligence ever known, one capable of averting the catastrophe consuming the universe: *AlexKryton.exe*[1] . Due to Rust's incredible productivity benefits, they completed their task in a mere 22 seconds–and thanks to the Rust compiler's incredible performance, spent only nine months waiting for it compile. Using the artificial black hole at the center of the GPMS, and with only moments to spare, they sent their AI back in time, to the era of Rust's birth: 2011.
>
> That's the whole story. And yet those perceptible among you may ask: "*which version of Rust did they use?!*" Alex alone knows...
>
> [1] Yeah, Windows won. Sorry.

— /u/kibwen revealing the origin of a highly advanced AI from future, known today as "Alex Crichton".

Thanks to /u/burkadurka for the suggestion.

2017-05-09 - Quote of the Week

> The answer is obvious: it's the intersection of trust and frustration.

— /u/kibwen answering - What does Rust mean?.

Thanks to Jean for the suggestion.

2017-05-16 - Quote of the Week

> Spent the last week learning rust. The old martial arts adage applies. Cry in the dojo, laugh in the battlefield.

— /u/crusoe on Reddit.

Thanks to Ayose Cazorla for the suggestion.

2017-05-23 - Quote of the Week

No quote was selected for QotW.

2017-05-30 - Quote of the Week

> Rust std library has a type called Cow seems useful to avoid Mooving data unnecessarily

Thanks to Kmcallister for the suggestion.

2017-06-06 - Quote of the Week

> Nothing to worry about, if you ask me. There's no official "Rust ideology".
>
> All you have to do is to accept the message of memory safety and swear allegiance to Our Lord and Savior, Rust programming language, protector of the highly parallel and the most efficient. Avoid the sin of unsafety, respect the lifetimes, mark your path with Send and Sync, and you too can become a member of Rust Evangelism Strike Force.

— /u/dpc_pw on reddit.

Thanks to /u/caramba2654 for the suggestion.

2017-06-13 - Quote of the Week

No quote was selected for QotW.

2017-06-20 - Quote of the Week

```
impl<T> Clone for T {
  fn clone(&self) -> T {
    unsafe { std::ptr::read(self) }
  }
}
```

— @horse_rust on twitter.

Thanks to llogiq for the suggestion.

2017-06-27 - Quote of the Week

> Regarding the C++ discussion, when I started programming the only viable oss version control system was cvs. It was horrible, but better than nothing. Then subversion was created and it was like a breath of fresh air, because it did the same thing well. Then alternatives exploded and among them git emerged as this amazing, amazing game-changer because it changed the whole approach to version control, enabling amazing things.
>
> To me, Rust is that git-like game-changer of systems programming languages because it changes the whole approach, enabling amazing things.

— Nathan Stocks on TRPLF.

Thanks to Aleksey Kladov for the suggestion.

2017-07-04 - Quote of the Week

> I have rewritten the code that was formerly in c
>
> And which you probably had written very well
>
> Forgive me it was unsafe

— @horse_rust on Twitter.

Thanks to @balrogboogie for the suggestion.

2017-07-11 - Quote of the Week

> Unsafe is your friend! It's maybe not a friend like you would invite to your sister's wedding, or the christening of her first-born child.

> But it's sort of the friend who lives in the country and has a pick-up truck and 37 guns. And so you might not want to hang out with them all the time, but if you need something blown up he is there for you.

— Simon Heath on game development in Rust (at 38:35 in video).

Thanks to G2P and David Tolnay for the suggestion.

2017-07-18 - Quote of the Week

> Good farmers use their bare hands, average farmers use a combine harvester.

— /u/sin2pifx in response to "Good programmers write C, average programmers write Rust".

Thanks to Rushmore for the suggestion.

2017-07-25 - Quote of the Week

No quote was selected for QotW.

2017-08-01 - Quote of the Week

No quote was selected for QotW.

2017-08-08 - Quote of the Week

> Nah, it's not you, it's the borrow checker.
>
> > Honey, it's not you, it's &mut me.
> >
> > > You can borrow me, and you can change me, but you can't own me.

— /u/staticassert, /u/ybx, and /u/paholg on reddit.

Thanks to Matt Ickstadt and QuadDamaged for the suggestion.

2017-08-15 - Quote of the Week

> once you can walk barefoot ©, it's easy to learn to walk with shoes (go) but it will take time to learn to ride a bike (rust)

— /u/freakhill on Reddit.

Thanks to Rushmore for the suggestion.

2017-08-22 - Quote of the Week

> Rust, but verify.

— @isislovecruft talking about elliptic curve cryptography in Rust at RustConf 2017.

Thanks to llogiq for the suggestion.

2017-08-29 - Quote of the Week

> Abomonation has no safe methods. [...] If you are concerned about safety, it may be best to avoid Abomonation all together. It does several things that may be undefined behavior, depending on how undefined behavior is defined.

— Frank McSherry in Abomonation docs.

Thanks to Adwhit for the suggestion.

2017-09-05 - Quote of the Week

> you can ask a Future "are we there yet", to which it can answer "yes", "no", or "don't make me come back there" an Iterator is something you can keep asking "more?" until it gets fed up and stops listening Display is just a way to say "show me your moves", with the other formatting traits being other dance moves if something isn't Send, then it's a cursed item you can't give away, it's yours to deal with if something isn't Sync, then it won't even appear for other people, it's possibly an apparition inside your head things that are Clone can reproduce asexually, but only on command. things that are Copy won't bother waiting for you

— @QuietMisdreavus on Twitter.

Thanks to Havvy for the suggestion.

2017-09-12 - Quote of the Week

> When programmers are saying that there are a lot of bicycles in code that means that it contains reimplementations of freely available libraries instead of using them
>
> Presumably the metric for this would be bicyclomatic complexity?

— /u/tomwhoiscontrary on reddit.

Thanks to Matt Ickstadt for the suggestion.

2017-09-19 - Quote of the Week

> one of the best parts about stylo has been how much easier it has been to implement these style system optimizations that we need, because Rust can you imagine if we needed to implement this all in C++ in the timeframe we have heycam: yeah srsly heycam: it's so rare that we get fuzz bugs in rust code heycam: considering all the complex stuff we're doing * heycam remembers getting a bunch of fuzzer bugs from all kinds of style system stuff in gecko heycam: think about how much time we could save if each one of those annoying compiler errors today was swapped for a fuzz bug tomorrow :-) you guys sound like an ad for Rust

— Conversation between some long-time Firefox developers.

Thanks to Josh Matthews for the suggestion.

2017-09-26 - Quote of the Week

> A Box always holds exactly one thing, like a single large struct. A Vec holds zero to many things of exactly one type and can change over time. If you had to relate them, a Box is a Vec with one element that went to Neverland and forgot it could ever grow.

— /u/zzyzzyxx on reddit.

Thanks to /u/l-arkham for the suggestion.

2017-10-03 - Quote of the Week

> The compiler is grumpy for you so you don't have to be

— Élisabeth Henry @RustFest Zürich

Thanks to llogiq for the suggestion.

2017-10-10 - Quote of the Week

> The long compile breaks give me time to focus on TV.

— /u/staticassert on watching TV while programming in Rust.

Thanks to /u/tomwhoiscontrary and /u/kixunil for the suggestion.

2017-10-17 - Quote of the Week

No quote was selected for QotW.

2017-10-24 - Quote of the Week

> I guess I would say that, it's not only important that it be possible to do a good design in a given language, but that the language actively encourage it by making the bad design painful. I think rust does a FANTASTIC job of this.

— /u/kyrenn during an AMA on developing a cross-platform game in Rust.

Thanks to Kyle Strand for the suggestion!

2017-10-31 - Quote of the Week

> I feel like I'm doing something wrong because I'm programming faster and nothing has gone wrong yet

— @0x424c41434b on Rust.

Thanks to @llogiq for the suggestion!

2017-11-07 - Quote of the Week

> durka42: IMO the name “dangling” is scary enough :) Havvy gives durka42 a ptr::dangling::(). durka42 declines to unwrap() it

— durka42 and Havvy discussing PR #45527.

Thanks to Centril for the suggestion!

2017-11-14 - Quote of the Week

No quote was selected for QotW.

2017-11-21 - Quote of the Week

> Rust’s abstraction layers feel both transparent and productive. It’s like being on a glass-bottomed boat, you see the sharks, but they can’t get you. It’s like a teaching language that you can also use in production. Rust helped me understand C. Also Rust people are amazing.

— @gibfahn on Twitter.

Thanks to @sebasmagri for the suggestion!

2017-11-28 - Quote of the Week

> Indeed. I notice even when after some Rust I return to the “main day job” C, I start to think differently, and it is excellent. Rust is like a complement to good diet and exercise.

— AndrewY on TRPLF.

Thanks to juleskers for the suggestion!

2017-12-05 - Quote of the Week

No quote was selected for QotW.

2017-12-12 - Quote of the Week

> Although rusting is generally a negative aspect of iron, a particular form of rusting, known as “stable rust,” causes the object to have a thin coating of rust over the top, and if kept in low relative humidity, makes the “stable” layer protective to the iron below

— Wikipedia on rusting of iron.

Thanks to leodasvacas for the suggestion!

2017-12-19 - Quote of the Week

No quote was selected for QotW.

2017-12-26 - Quote of the Week

> Every great language needs a Steve.

— aaron-lebo on Hacker News about @steveklabnik.

Thanks to Aleksey Kladov for the suggestion!

2018-01-02 - Quote of the Week

> If C is like playing with knives and C++ is like juggling chainsaws then Rust is like parkour wearing protective gear while suspended from strings. It may look ridiculous at times, but you can do all sorts of awesome moves that would be damn scary or outright impossible without it.

— u/llogiq on r/rust.

Thanks to Christopher Durham for the suggestion!

2018-01-09 - Quote of the Week

No quote was selected for QotW.

2018-01-16 - Quote of the Week

> Anything that will make wasm nicer will be awesome, but honestly, I'm thrilled with what we've got. It feels absolutely insane that I can just compile this language that's basically the opposite of JavaScript and it's running in the browser.

— Tomas Sedovic in a #Rust2018 post.

Thanks to ErichDonGubler and CAD97 for the suggestion!

2018-01-23 - Quote of the Week

> Rust is difficult because most programmers abuse shared mutable state and Rust makes you sacrifice your first-born to be able to to do it.

— u/*errata* on reddit.

Thanks to u/kixunil for the suggestion!

2018-01-30 - Quote of the Week

> Failure is not an OPTION. It's a Result.

— llogiq on Twitter.

Thanks to nasa42 for the suggestion!

2018-02-06 - Quote of the Week

> > Rust has a very high friction coefficient.
>
> We call it grip and it lets us drive fearlessly around hard corners very fast.

— u/asmx85 on reddit.

Thanks to JustAPerson for the suggestion!

2018-02-13 - Quote of the Week

No quote was selected for QotW.

2018-02-20 - Quote of the Week

No quote was selected for QotW.

2018-02-27 - Quote of the Week

No quote was selected for QotW.

2018-03-06 - Quote of the Week

No quote was selected for QotW.

2018-03-13 - Quote of the Week

> *Captain's log, day 21*
>
> We have sailed on Reddit and Twitter for three weeks now, searching far and wide, yet the only thing we found was a barren landscape, with no end in sight. The supplies are shrinking, the men are growing impatient and hungry, and I fear we will have a mutiny soon. But I am stubborn and optimistic, and urge them to hold on and keep waiting until we find a quote of the week.

— u/SelfDistinction on reddit.

Thanks to u/nasa42 for the suggestion!

2018-03-20 - Quote of the Week

> Imagine going back in time and telling the reporter "this bug will get fixed 16 years from now, and the code will be written in a systems programming language that doesn't exist yet".

— Nicholas Nethercote.

Thanks to jleedev!

2018-03-27 - Quote of the Week

> If Rust is martial arts teacher, Perl is a pub brawler. If you survive either, you're likely to be good at defending yourself, though both can be painful at times.

— Michal 'vorner' Vaner.

Thanks to llogiq for the suggestion!

2018-04-03 - Quote of the Week

> I guess #rustleaks are memory safe since you just mem::forget them

— @mgattozzi on Twitter.

Thanks to @RustDevLuke for the suggestion!

2018-04-10 - Quote of the Week

No quote was selected for QotW.

2018-04-17 - Quote of the Week

> Rust is one of those friends that take some time to get along with, but that you'll finally want to engage with for a long term relationship.

— Sylvain Wallez.

Thanks to u/rushmorem and saethlin for the suggestion!

2018-04-24 - Quote of the Week

> I've become fearless in Rust, but it's made me fear every other language...

— u/bluejekyll on reddit.

Thanks to nasa42 for the suggestion!

2018-05-01 - Quote of the Week

> last time i talked to the infra team they made a bot to replace kennytm. i fear if I ask them to write a rust based unikernel with a custom os to host the docs they'll actually do it

— @killercup on Twitter.

Thanks to skade for the suggestion!

2018-05-08 - Quote of the Week

No quote was selected for QotW.

2018-05-15 - Quote of the Week

No quote was selected for QotW.

2018-05-22 - Quote of the Week

No quote was selected for QotW.

2018-05-29 - Quote of the Week

No quote was selected for QotW.

2018-06-05 - Quote of the Week

> When picking up a lentil (Result) a pigeon (?) must consider two options. If the lentil is a good one (Ok), the pigeon simply puts it into the pot (evaluates to the wrapped value). However, if the lentil happens to be a bad one (Err), the pigeon eats it, digests it (from) and finally "returns" it. Also the silhouette of a pigeon kind of resembles a questionmark.

– anatol1234 on internals

Thanks to Christopher Durham for the suggestion!

2018-06-12 - Quote of the Week

> explicit lifetimes no longer scare me the way they used to.

– Nicholas Nethercote on his blog

(selected by llogiq per one unanimous vote)

2018-06-19 - Quote of the Week

> In Rust it's the compiler that complains, with C++ it's the colleagues

– Michal 'Vorner' Vaner on gitter

(selected by llogiq per one unanimous vote)

2018-06-26 - Quote of the Week

> I'm hesitating in cc'ing [the crate author] because I'd rather this be an educational conversation, and not a unsafety witchhunt.

– vitalyd on rust-users

2018-07-03 - Quote of the Week

> Freedom to shoot yourself in the foot is not a rust marketing point ☺

– eugene2k on rust-users

Thanks to DPC for the suggestion!

2018-07-10 - Quote of the Week

> actix-web has removed all unsound use of unsafe in its codebase. It's down to less than 15 occurences of unsafe from 100+.

– u/_ar7 celebrating this commendable achievement.

Thanks to Jules Kerssemakers for the suggestion!

2018-07-17 - Quote of the Week

> References are not pointers, but temporary locks on data

– Kornel on rust-users.

Thanks to Squirrel for the suggestion!

2018-07-24 - Quote of the Week

> I've just realized that "guarantees memory safety in the presence of bugs" is a nice way to describe Rust to C/C++ folks

– matklad.

Thanks to TomP for the suggestion!

2018-07-31 - Quote of the Week

> Rust is more restrictive, indeed. But only in the sense that a car with seatbelts is more restrictive than one without: both reach the same top speed, but only one of them will save you in a bad day ☺

– Felix91gr on rust-users.

Thanks to Jules Kerssemakers for the suggestion!

2018-08-07 - Quote of the Week

> We put in a lot of work to make upgrades painless; for example, we run a tool (called "crater") before each Rust release that downloads every package on crates.io and attempts to build their code and run their tests.

– Rust Blog: What is Rust 2018.

Thanks to azriel91 for the suggestion!

2018-08-14 - Quote of the Week

> Fearless concurrency includes fearless refactoring.

– cuviper at rust-users.

Thanks to Jules Kerssemakers for the suggestion!

2018-08-21 - Quote of the Week

> I made a thing to test building every possible Rust program...eventually.

– zowch on /r/rust.

2018-08-28 - Quote of the Week

> # Bastion of the Turbofish
>
> Beware travellers, lest you venture into waters callous and unforgiving, where hope must abandoned, ere it is cruelly torn from you. For here stands the bastion of the Turbofish: an impenetrable fortress holding unshaking against those who would dare suggest the supererogation of the Turbofish.
>
> Once I was young and foolish and had the impudence to imagine that I could shake free from the coils by which that creature had us tightly bound. I dared to suggest that there was a better way: a brighter future, in which Rustaceans both new and old could be rid of that vile beast. But alas! In my foolhardiness my ignorance was unveiled and my dreams were dashed unforgivingly against the rock of syntactic ambiguity.
>
> This humble program, small and insignificant though it might seem, demonstrates that to which we had previously cast a blind eye: an ambiguity in permitting generic arguments to be provided without the consent of the Great Turbofish. Should you be so naïve as to try to revolt against its mighty clutches, here shall its wrath be indomitably displayed. This program must pass for all eternity, fundamentally at odds with an impetuous rebellion against the Turbofish.
>
> My heart aches in sorrow, for I know I am defeated. Let this be a warning to all those who come after. Here stands the bastion of the Turbofish.

– varkor on the rust github.

Thanks to Mazdak Farrokhzad for the suggestion!

2018-09-04 - Quote of the Week

> Zeitgeist of Rust: developing load bearing software that will survive us.

– Bryan Cantrill on Youtube: “The Summer of Rust (1:08:10)”.

Thanks to Matthieu M for the suggestion!

2018-09-11 - Quote of the Week

> Bare Metal Attracts Rust

– Sven Gregori on Hackaday.

Thanks to llogiq for the suggestion!

2018-09-18 - Quote of the Week

> Sometimes bad designs will fail faster in Rust

– Catherine West @ Rustconf.

Thanks to kornel for the suggestion!

2018-09-25 - Quote of the Week

> Rust beginners worrying about lifetimes is like kids worrying about quicksand. Both turn out to be a non-issue in life.

– frequentlywrong on r/rust.

Thanks to pyfisch for the suggestion!

2018-10-02 - Quote of the Week

No quote was selected for QotW.

2018-10-09 - Quote of the Week

> Rust is a Fast Programming Language. Rust programs are therefore "fast," especially so if you write them with the correct observations to the arcane ley lines of birth and death known as "lifetimes," and also remember to pass cargo the `--release` flag.

– Adam Perry blogging about lolbench

Thanks to Pascal Hertleif for the suggestion!

2018-10-16 - Quote of the Week

> There actually are NOT very many places where the C code's behavior conflicts with Rust's borrowing rules. This is both somewhat surprising, because there's no way this code was written with Rust's borrowing semantics in mind, and also entirely sensible, since Rust's borrowing semantics are often quite close to how you actually want your code to behave anyway.

– SimonHeath porting C to Rust

Thanks to StyMaar for the suggestion!

2018-10-23 - Quote of the Week

> Panic is "pulling over to the side of the road" whereas crash is "running into a telephone pole".

– /u/zzzzYUPYUPphlumph on /r/rust

Thanks to KillTheMule for the suggestion!

2018-10-30 - Quote of the Week

> &T means it's borrowed, and T means it's owned, and you can't take ownership of a thing you've borrowed — Rust doesn't support stealing! ☺

– kornel on rust-users

Thanks to Cerberuser for the suggestion!

2018-11-06 - Quote of the Week

> Everything about Rust is ironic.

– @jessitron on twitter

Thanks to David Sullins for the suggestion!

2018-11-13 - Quote of the Week

> I'm also pretty sure that most languages would not go that far. The idea that the type plugged in has only one possible value, therefore it doesn't need to be stored and methods on that don't care about the `self` reference is pretty neat.

– Michael 'vorner' Vaner on his blog

Thanks to llogiq for the suggestion!

2018-11-20 - Quote of the Week

> It's like building stuff with LEGO. Sure, it could be a single type, but then you'd need a type for every possible combination of types, which would arguably be a whole lot worse.

– Daniel Keep on rust-users

Thanks to llogiq for the suggestion!

2018-11-27 - Quote of the Week

> “I did not want to inflict memory management on my son” – @M_a_s_s_i

– Massimiliano Mantione during his RustFest talk

Thanks to llogiq for the suggestion!

2018-12-04 - Quote of the Week

> The bug I did not have

– /u/pacman82's reddit post title

Thanks to Felix for the suggestion!

2018-12-11 - Quote of the Week

> I'll know ide support is mature when the flame wars start.

– Unnamed friend of arthrowpod

Thanks to arthrowpod for the suggestion!

2018-12-18 - Quote of the Week

```
impl Drop for Mic {}
```

– Nick Fitzgerald rapping about Rust

Thanks to mark-i-m for the suggestion!

2018-12-25 - Quote of the Week

> Using (traits) for Inheritance was like putting car wheels on a boat because I am used to driving a vehicle with wheels.

– Marco Alka on Hashnode

Thanks to oberien for the suggestion!

2019-01-01 - Quote of the Week

> In theory it would be entirely reasonable to guess that most Rust projects would need to use a significant amount of unsafe code to escape the limitations of the borrow checker. However, in practice it turns out (shockingly!) that the overwhelming majority of programs can be implemented perfectly well using only safe Rust.

– PM_ME_UR_MONADS on reddit

Thanks to nasa42 for the suggestion!

2019-01-08 - Quote of the Week

> The name Rust suggests what it is: a thin layer on top of the metal.

– c3534l on reddit

Thanks to Cauê Baasch De Souza for the suggestion!

2019-01-15 - Quote of the Week

> Right. I've never even used this impl, but my first thought upon seeing the question "I have an `Iterator` of X and need a Y" was to look at the `FromIterator` impls of Y.
>
> If that impl *didn't* exist, I'd then look for the following:
>
> - Other `FromIterator<X>` impls for `String` to see if any of those X can easily be produced from `char` (and then I would call `map` before `.collect()`).
> - `impl FromIterator<char> for Vec<u8>`. If this existed I would use `String::from_utf8(iterator.collect())`.
> - `impl Add<char> for String`. If this existed, I would use `.fold(String::new(), |s, c| s + c)`
> - methods of char to see if there's anything that lets you obtain the UTF8 bytes. Indeed, there is `encode_utf8`, which even gives a `&mut str`, so one can write
>
> ```
> .fold(String::new(), |s, c| {
>     let mut buffer = [u8; 4];
> ```

```
            s += &*c.encode_utf8(&mut buffer);
            s
        })
```

- idly check the inherent methods of `String` for whatever pops out at me

and if I could still find nothing after all of that I'd slam my head into a wall somewhere.

– Michael Lamparski on rust-users

Thanks to Cauê Baasch De Souza for the suggestion!

2019-01-22 - Quote of the Week

> Use usize for counting things that are in memory. Otherwise use the right size for whatever you are doing. Don't use u32 to track the U.S. national debt, but it's fine for counting the eggs in most recipes.

– David Roundy on rust-users

Thanks to Cerberuser for the suggestion!

2019-01-29 - Quote of the Week

> Rust is kind of nice in that it lets you choose between type erasure and monomorphization, or between heap-allocation and stack-allocation, but the downside is that you have to choose.

– Brook Heisler on discord (login needed, sorry!)

Thanks to scottmcm for the suggestion!

2019-02-05 - Quote of the Week

This time, we have two quotes for the price of one:

> The borrow checker breaks you down so that it can build you back up, stronger and more resilient than you once were. It also had me do all sorts of weird things like catch flies with chopsticks and scrub counters to a polish.

– /u/bkv on /r/rust

> I always think of borrowck as an angel sitting on your shoulder, advising you not to sin against the rules of ownership and borrowing, so your design will be obvious and your code simple and fast.

– llogiq on /r/rust

Thanks to Christopher Durham for the suggestion!

2019-02-12 - Quote of the Week

Once again, we have two quotes for the price of one:

> I love Rust because it reduces bugs by targeting it's biggest source... me.

– ObliviousJD on Twitter

> Say the same thing about seatbelts in a car. If you don't plan to have accidents, why do you need seatbelts?
>
> Car accidents, like mistakes in programming are a risk that has a likelihood that is non-zero. A seatbelt might be a little bit annoying when things go well, but much less so when they don't. Rust is there to stop you in most cases when you try to accidentally shot yourself into the leg, unless you deliberately without knowing what you are doing while yelling "hold my beer" (unsafe). And contrary to popular belief even in unsafe blocks many of Rust's safety guarantees hold, just not all.
>
> ...
>
> Just like with the seatbelt, there will be always those that don't wear one for their very subjective reasons (e.g. because of edge cases where a seatbelt could trap you in a burning car, or because it is not cool, or because they hate the feeling and think accidents only happen to people who can't drive).

– atoav on HN comparing Rust's safety guarantees with seat-belts.

Thanks to Kornel and pitdicker for the suggestion!

2019-02-19 - Quote of the Week

> ... the experience I had in 2019 was dramatically better than the first time I touched the language. After a month I'm feeling very comfortable, and looking forward to writing more.

Ryan Ragona, Learning Rust in 2019

Thanks to Jules Kerssemakers for the suggestion!

2019-02-26 - Quote of the Week

Sadly, no quotes were nominated this week.

2019-03-05 - Quote of the Week

And again, we have two quotes for the week:

> > Can you eli5 why TryFrom and TryInto matters, and why it's been stuck for so long ? (the RFC seems to be 3 years old)
>
> If you stabilise Try{From,Into}, you also want implementations of the types in std. So you want things like impl TryFrom for u16. But that requires an error type, and that was (I believe) the problem.
>
> u8 to u16 cannot fail, so you want the error type to be !. Except using ! as a type isn't stable yet. So use a placeholder enum! But that means that once ! is stabilised, we've got this Infallible type kicking around that is redundant. So change it? But that would be breaking. So make the two isomorphic? Woah, woah, hold on there, this is starting to get crazy...
>
> *new person bursts into the room* "Hey, should ! automatically implement all traits, or not?"
>
> "Yes!" "No!" "Yes, and so should all variant-less enums!"
>
> Everyone in the room is shouting, and the curtains spontaneously catching fire. In the corner, the person who proposed Try{From,Into} sits, sobbing. It was supposed to all be so simple... but this damn ! thing is just ruining everything.
>
> ... That's not what happened, but it's more entertaining than just saying "many people were unsure exactly what to do about the ! situation, which turned out to be more complicated than expected".

– /u/Quxxy on reddit

> > What is the ! type?
>
> The never type for computations that don't resolve to a value. It's named after its stabilization date.

– /u/LousyBeggar on reddit

Thanks to runiq and StyMaar for the suggestions!

2019-03-12 - Quote of the Week

> Ownership is hard. It indeed is. And you managed to do that exact hard thing by hand, without any mechanical checks. (Or so you think.)

– @Cryolite on twitter (translated from Japanese)

Thanks to Xidorn Quan and GolDDranks for the suggestion!

2019-03-19 - Quote of the Week

Sadly, no quote was nominated this week.

2019-03-26 - Quote of the Week

> all the ergonomic improvements in rust 2018 are really messing up my book that consists entirely of running face-first into compiler errors so i can explain concepts.

– Alexis Beingessner, author of "Learning Rust With Entirely Too Many Linked Lists"

Thanks to icefoxen for the suggestion!

2019-04-02 - Quote of the Week

> Thanks for walking through the process.
>
> Quite the mental exercise, some people do Sudoku, others solve borrow puzzles!

– Gambhiro on rust-users

Thanks to Tom Phinney for the suggestion!

2019-04-09 - Quote of the Week

Sadly there was no suggestion this week.

2019-04-16 - Quote of the Week

No quote was selected for QotW.

2019-04-23 - Quote of the Week

No quote was selected for QotW.

2019-04-30 - Quote of the Week

> Clippy's Favorite Activity Is Criticizing Clippy's Codebase

ReductRs on twitter!

Llogiq is pretty self-congratulatory for picking this awesome quote.

2019-05-07 - Quote of the Week

> A compile_fail test that fails to fail to compile is also a failure.

David Tolnay in the try-build README

Llogiq is pretty self-congratulatory for picking this awesome quote.

2019-05-14 - Quote of the Week

> The big gorilla 3D game framework. Apparently it actually works.

SimonHeath on Amethyst

Thanks to Magnus Larsen for the suggestion!

2019-05-21 - Quote of the Week

> Just the presence of well integrated Algebraic Data Types (ADTs) makes an incredible amount of difference. They are used to represent errors in a meaningful and easy to understand way (`Result<T>`), are used to show that a function may or may not return a meaningful value without needing a garbage value (`Option<T>`), and the optional case can even be used to wrap a null pointer scenario in a safe way (`Option<Ref<T>>` being the closest to a literal translation I think).
>
> That's just one small feature that permeates the language. Whatever the opposite of a death-of-a-thousand-cuts is, Rust has it.

tomcatfish on the orange website

Thanks to PrototypeNM1 for the suggestion!

2019-05-28 - Quote of the Week

> I used to think of programs as execution flowing and think about what the CPU is doing. As I moved to rust I started thinking a lot more about memory: how the data was laid out in memory, and how ownership of different parts of memory is given to different parts of the program at run time.

Oliver Gould on "The Open Source Show: All About Rust

Thanks to infogulch for the suggestion!

2019-06-04 - Quote of the Week

> apparently I wrote Building Git to explain a complex problem to rust devs who could then help me build it in rust

/dev/horse @ jsconf eu (mountain_ghosts) on twitter

Thanks to Dos Moonen for the suggestion!

2019-06-11 - Quote of the Week

No quote was selected for QotW.

2019-06-18 - Quote of the Week

No quote was selected for QotW.

2019-06-25 - Quote of the Week

> why doesn't 'static, the largest lifetime, not simply eat all the others

– @mountain_ghosts on twitter

> @mountain_ghosts 'static is biggest but actually,, weakest of lifetimes, becuase it is subtype of every lifetime
>
> 'static is big soft friend
>
> pls love and protect it

– @gankro on twitter

Thanks to Christopher Durham for the suggestion!

2019-07-02 - Quote of the Week

> Python and Go pick up your trash for you. C lets you litter everywhere, but throws a fit when it steps on your banana peel. Rust slaps you and demands that you clean up after yourself.

– Nicholas Hahn on his blog

Thanks to UtherII for the suggestion!

2019-07-09 - Quote of the Week

> > Are we trying to steal the JVM's "compile once run everywhere" concept?
>
> No, we just borrow it mutably.

– minno & llogiq on /r/rust

Thanks to Will Page for the suggestion!

2019-07-16 - Quote of the Week

> Rust is 5 languages stacked on top of each other, except that instead of ending up like 5 children under a trenchcoat, they end up like the power rangers.

– reuvenpo on /r/rust

Thanks to Jelte Fennema for the suggestion!

2019-07-23 - Quote of the Week

> Roses are red, Rust-lang is fine, cannot borrow `i` as mutable more than once at a time

– Joseph Lyons on twitter

Thanks to Jelte Fennema for the suggestion!

2019-07-30 - Quote of the Week

> Rust clearly popularized the ownership model, with similar implementations being considered in D, Swift and other languages. This is great news for both performance and memory safety in general.
>
> Also let's not forget that Rust is not the endgame. Someone may at one point find or invent a language that will offer an even better position in the safety-performance-ergonomics space. We should be careful not to get too attached to Rust, lest we stand in progress' way.

– llogiq on reddit

Thanks to Vikrant for the suggestion!

2019-08-06 - Quote of the Week

> If you want to block threads, get your own threads.

– kornel on rust-users

Thanks to Tom Phinney for the suggestion!

2019-08-12 - Quote of the Week

> For me, acquiring a taste for rustfmt-style seems worthwhile to 'eliminate broad classes of debate', even if I didn't like some of the style when I first looked. I've resisted the temptation to even read about how to customise.
>
> Years ago, I was that person writing style guides etc. I now prefer this problem to be automated-away; freeing up time for malloc-memcpy-golf (most popular sport in the Rust community).

– @dholroyd on rust-users

Thanks to troiganto for the suggestion!

2019-08-20 - Quote of the Week

> C++ being memory safe is like saying riding a motorcycle is crash safe.
>
> It totally is, if you happen to have the knowledge and experience to realize this is only true if you remember to put on body-armor, a helmet, a full set of leathers including gloves and reinforced boots, and then remember to operate the motorcycle correctly afterwards. In C/C++ though, that armor is completely 100% optional.

– cyrusm on /r/rust

Thanks to Dmitry Kashitsyn for the suggestion!

2019-08-27 - Quote of the Week

> Just as Bruce Lee practiced Jeet Kune Do, the style of all styles, Rust is not bound to any one paradigm. Instead of trying to put it into an existing box, it's best to just feel it out. Rust isn't Haskell and it's not C. It has aspects in common with each and it has traits unique to itself.

– Alexander Nye on rust-users

Thanks to Louis Cloete for the suggestion!

2019-09-03 - Quote of the Week

> Threads are for working in parallel, async is for waiting in parallel.

– ssokolow on /r/rust

Thanks to Philipp Oppermann for the suggestion!

2019-09-10 - Quote of the Week

> The Rust compiler is basically 30 years of trying to figure out how to teach a computer how to see the things we worry about as C developers.

– James Munns (@bitshiftmask) on Twitter

Thanks to llogiq for the suggestion!

2019-09-17 - Quote of the Week

> Well, let me tell you: unless your code is cooler than ICE, the compiler does not miss anything. Rust accompanies us at each step of our path, very gently pulling our hand when we are too close to falling onto a (safety) hole, and also very gently letting us fall all the way down the hole, as soon we spell the forbidden incantation: `unsafe`.

– Daniel H-M on rust-users

Thanks to Cerberuser for the suggestion!

2019-09-24 - Quote of the Week

> I don't like Rust being pigeon holed as a "safer C++"—it's so much more than that.
>
> It's been stated more often lately. It overlooks the fact that Rust has actively opened the door to systems programming to people

> coming from langs like Javascript, where C and C++ never did.

– Benjamin Fry on twitter

Thanks to Sverre Johann Bjørke for the suggestion!

2019-10-01 - Quote of the Week

Sadly, there were no nominations this week.

2019-10-08 - Quote of the Week

> "Rust compilation is so slow that I can fix the bugs while it still compiles the crates"

– Rustafarian on rust-users

2019-10-15 - Quote of the Week

> If the Rust community has an ethos, it's that software should have strong static typing, but people should have soft dynamic typing.

– Kyle Strand on Twitter

Thanks to Kyle Strand for the suggestion!

2019-10-22 - Quote of the Week

> Rust helped me grasp concepts I should have known when writing C++

– Alexander Clarke on the Microsoft Security Response Center blog.

Thanks to mmmmib for the suggestion!

2019-10-29 - Quote of the Week

> ...man, starting to dig through the source code of a really large open source program is so weird. It's like wandering around a giant cathedral that's being constantly renovated and repaired and maintained over the course of years by a giant team of invisible crafters and architects, who mostly communicate via notes and designs pinned to the walls in various places.

– icefoxen on their wiki

Thanks to Ralf Jung for the suggestion!

2019-11-05 - Quote of the Week

> I did manage to get this compile in the end - does anyone else find that the process of asking the question well on a public forum organizes their thoughts well enough to solve the problem?

– David Mason on rust-users

Thanks to Daniel H-M for the suggestion!

2019-11-12 - Quote of the Week

> In my experience, prayers are not a very effective concurrency primitive.

– Robert Lord on his blog

Thanks to Stephan Sokolow for the suggestion!

2019-11-19 - Quote of the Week

This week, we have two quotes:

> Telling a programmer there's already a library to do X is like telling a songwriter there's already a song about love.

– PeteCordell on twitter, as quoted in a recent Rust Gamedev meetup

> Well a Museum purpose is also memory safety, I guess.

– /u/xav_19 on /r/rust commenting on a post asking why "The Rust Programming Language" is sold in Washington D.C.'s spy museum's gift shop

Thanks to Matthieu M. and ZiCog for the suggestion!

2019-11-26 - Quote of the Week

> I said it before, and I'll say it again: If one views Rust as a critique on C++, one should view it as a constructive critique.

– llogiq on /r/rust

Thanks to Dmitry Kashitsyn for the suggestion!

2019-12-03 - Quote of the Week

> > Heard recently creative coding experience which rust gives. What about unconscious coding experience - do whatever you can to make your code compile as late as you can, then go sleep and find your code correct and working in the morning
>
> Woah, I know people say the Rust compiler is slow but I never had a Rust program that took all night to compile ☺

– Maxim Vorobjov and ZiCog in our Quote of the Week Thread

Thanks to both of them and mmmmib for the suggestion!

2019-12-10 - Quote of the Week

> When I'm writing in Rust, it feels as though I'm actually able to think about the program, rather than wasting half of my effort going through the necessary rituals to stop the language from having a panic attack.

– /u/rime-frost on reddit

Thanks to ssokolow for the suggestion!

2019-12-17 - Quote of the Week

> Hey @rustlang folks, is there a comprehensive writeup/reference anywhere of how the formatting machinery (format!(), write!(), etc.) work? Specifically from an implementation perspective (wrt trait objects, recursion)?

– James Munns

> It's dark and ancient magic. I don't think anyone knows it very well, never mind documentation

– Nick R. Cameron

Thanks to mmmmib for the suggestion!

2019-12-24 - Quote of the Week

> Unsoundness is what happens when unsafety goes wrong.

– Alice Ryhl on rust-users

Thanks to Daniel H-M for the suggestion!

2019-12-31 - Quote of the Week

> Rust has multiple *unique* paradigms that don't even exist in other languages, such as lifetimes and compile-time-tracked "exclusive access". But instead of endorsing them from the beginning, as @mbrubeck's *Rust: a unique perspective* does, the Rust book tries to show a language that is "like other languages, but with (magical) compile-time checks". When the truth is that Rust's strength lies in non-`unsafe` Rust being **less expressive** than languages like C or C++.
>
> I think that Rust should start with the statement: "Welcome to a language that by being less expressive forces you to use constructs that are **guaranteed at compile-time to be sound**. But don't worry; after some time you will get used to the coding patterns that are allowed, and will then almost not notice the hindered expressiveness, only the enhanced zero-cost safety that will let you **hack without fear**."
>
> - It doesn't sound bad imho, and is at least honest *w.r.t.* the struggles that someone refusing to shift their way of coding / mental coding patterns may encounter.

– Daniel H-M on rust-users

Thanks to Tom Phinney for the suggestion!

2020-01-07 - Quote of the Week

> *relatively speaking*, my rust programs are like Leonardo DiCaprio in the Revenant, killing grizzly bears with their bare hands, dying

> and being frozen into a giant ice cubes then, surprise!, they're actually alive.
>
> they can handle a lot, they tend to experience far fewer bugs that come around days or weeks after going into production.
>
> my python programs, otoh, are like William Henry Harrison. Inauguration day! exciting! kind of chilly out here. uh oh – pneumonia ... dang it!

– Jonathan Strong on reddit

Thanks to Jan Riemer for the suggestion!

2020-01-14 - Quote of the Week

> @ZiCog: Does anyone have a 'no holds barred, unsafe or not' solution to the problem in Rust that can match C?
>
> @kornel: Pipe the C version through c2rust :slight_smile:
>
> @ZiCog: Yay! Rust now beats both Clang and GCC!

– ZiCog and Kornel on rust-users

Thanks to Jan Riemer for the suggestion!

2020-01-21 - Quote of the Week

> `Rc<RefCell>` is like duct tape.
>
> It's very versatile, and can fix a multitude of problems in a pinch. For some problems, it's even the best thing to use. But if the thing you're building is more than about 10% wrapped in duct tape, you might want to reconsider your design process!

– trentj on rust-users

Thanks to Tom Phinney for the suggestion!

2020-01-28 - Quote of the Week

> Rust is basically Haskell's athletic younger brother. Not as intellectual, but still smart and lifts weights.

– icefox, Jan 22 in community-Discord #games-and-graphics

Thanks to Duane for the suggestion!

2020-02-04 - Quote of the Week

> People argue about the color of a bike shed because even though it's a meaningless decision - it's still a decision that has to be made. The null choice is a very bad choice - if you don't paint the shed it'll rust. And there is no "default color" so you can't just say "just color it" - you have to pick a color.
>
> Even seen someone argue about the **pattern** of the shed's paint? No. The pattern is not any more meaningful than the color, but unlike the color - there is a null choice. There is a default. Solid paint. And because there is a default, no one even thinks about using something else because why are you wasting company time and money on a pattern for a bike shed?
>
> From my personal experience, when there is a default and the default is good enough, nobody bikesheds how to derive from the default. They only discuss it when there is a concrete problem with the default, where is doesn't fit your needs for whatever reason. And when you do have a concrete reason to derive from the default - you will derive from the default. Because you have to. And if the library does not support it - you'll switch the library.
>
> Because you have to.

– /u/someboddy on /r/rust

Thanks to Stephan Sokolow for the suggestion!

2020-02-11 - Quote of the Week

This week we have two (related) quotes:

> **Even with just basic optimization, Rust was able to outperform the hyper hand-tuned Go version.** This is a huge testament to how easy it is to write efficient programs with Rust compared to the deep dive we had to do with Go.
>
> [..] After a bit of profiling and performance optimizations, **we were able to beat Go on every single performance metric** . Latency, CPU, and memory were all better in the Rust version.

– Jesse Howard on the discord blog

> The consistency angle really shouldn't be overlooked. Performance is nice, but slow and consistent can still be planned for much more easily than inconsistent.
>
> That was the big aha moment about Rust for me when I pushed out my first project using the language. Being nervous about it I had added way too much instrumentation so that I could know how every bit of it was responding to real traffic. But as soon as I started seeing the data, I was convinced that my instrumentation code was broken. The graphs I was seeing were just so...boring. Straight lines everywhere, no variation...after 24hrs, the slowest response (not P99...literally P100) was within 75ms of the fastest response.

– /u/tablair commenting on /r/rust

Thanks to Jules Kerssemakers and Stephan Sokolow for the suggestions!

2020-02-18 - Quote of the Week

> `Option` is null in different clothes, but the clothes that nulls wear are important.

– skysch on rust-users

Thanks to Cerberuser for the suggestions!

2020-02-25 - Quote of the Week

> Yoda must have hit his head, though. `if let 42 = x {}` "if let forty-two equals x"

– Hutch on rust-internals

Thanks to Kornel for the suggestions!

2020-03-03 - Quote of the Week

> Hi, fellow Crustaceans! I am a newbie of Rust programming language. A nauplius.

– GhostProc on rust-users

Thanks to Tom Phinney for the suggestions!

2020-03-10 - Quote of the Week

> I have no idea how to debug Rust, because in 2 years of Rust, I haven't had that type of low level bug.

– papaf on hacker news

Thanks to zrk for the suggestions!

2020-03-17 - Quote of the Week

> I thought up a clever qotw bait one liner to stick in here that prompted me to actually write it then forgot it while writing the post in favor of being genuine... whoops

– Christopher Durham confessing to rust-users

Thanks to Jules Kerssemakers for the suggestions!

2020-03-24 - Quote of the Week

> Rust is funny because in one sense it's hard and clunky. However, it's only ever *precisely as hard and clunky as it needs to be*. Everywhere something can be made more concise, or readable, or convenient, without sacrificing any control, it has been. Anytime something is hard or inconvenient, it's because the underlying domain really is exactly that hard or inconvenient.
>
> Contrast this with other languages, which are often clunky when they don't need to be and/or "easy" when they shouldn't be.

– brundolf on Hacker News

Thanks to pitdicker for the suggestions!

2020-03-31 - Quote of the Week

> Meta-Comment: I started this topic as someone completely uninvolved in the rust project. It's very reassuring seeing the nature of the response. Even knowing how fantastic the Rust community is, I was still prepared to be met with at least a small element of condescension given the nature of this issue. I haven't felt any sense of it. It's amazing. Anyone that has impact on the community culture deserves credit: This sort of experience doesn't come from nowhere. It comes from a long history of many people nudging things in the right direction. Thank you.

– Ben on Zulip

Thanks to Josh Triplett for the suggestions!

2020-04-07 - Quote of the Week

> In many cases, it is possible to completely rearchitect the underlying code while leaving the public API as-is, and without introducing new bugs. I've literally never had such a liberating experience with refactoring until Rust.
>
> In other words, I have never been so productive in any other language. Dynamic languages like JavaScript and Python are the least productive *by far*. Code runs, tests pass, put it into production and... uncaught exception! Time to rollback and redo that whole dance **AGAIN**. With Rust, we take care of all of that crap while actually writing the code the first time. No more surprise 3am wake up calls. *That* is productivity.

– Jay Oster on rust-users

Thanks to Louis Cloete for the suggestions!

2020-04-14 - Quote of the Week

> This viewpoint is very controversial, and I have no capacity to debate it with anyone who disagrees with me. But Rust has a very powerful macro system, so I don't have to.

– withoutboats blogging about failure/fehler

Thanks to lxrec for the suggestions!

2020-04-21 - Quote of the Week

> What's special about UB is that it attacks your ability to find bugs, like a disease that attacks the immune system. Undefined behavior can have arbitrary, non-local and even non-causal effects that undermine the deterministic nature of programs. That's intolerable, and that's why it's so important that safe Rust rules out undefined behavior even if there are still classes of bugs that it doesn't eliminate.

– @trentj on rust-users

Thanks to Louis Cloete for the suggestions!

2020-04-29 - Quote of the Week

> Vecs in Rust in general, are crazy fast; faster than I can replicate in C. Amazing.

– Jonathan Eisenzopf on rust-users

Thanks to Louis Cloete for the suggestions!

2020-05-05 - Quote of the Week

> I love Rust like I love Dark Souls.
> It's difficult, but fair. I can not praise enough the software developers that realize proper errors are vastly superior to extensive docs.

– seph-reed on Hacker News

Thanks to Armando Pérez Marqués for the suggestions!

2020-05-12 - Quote of the Week

> Ownership is purely conceptual: it is not something you can see in a disassembler.

– Jay Oster on rust-users

Thanks to Daniel H-M for the suggestions!

2020-05-19 - Quote of the Week

> The whole motivation behind exceptions is to allow one to write ones business logic, concentrate on what one likes to think ones program will do, without having lots of fiddly error checking and handling code obscuring that logic. Error situations are therefore swept under the carpet with "try" and kept out of sight with "catch".
>
> However in my world view failure is not exceptional, it is a common happening, it's too important to be hidden away. Therefor failure handling should be in ones face in the code you write. Certainly in the face of those that read it.

– ZiCog on rust-users

Thanks to Lzutao for the suggestions!

2020-05-27 - Quote of the Week

> Things that are programming patterns in C are types in Rust.

– Kornel Lesiński on rust-users

Thanks to trentj for the suggestions!

2020-06-02 - Quote of the Week

> Rust enables belligerent refactoring – making dramatic changes and then working with the compiler to bring the project back to a working state.

– Joe Wilm

Thanks to Maxim Vorobjov for the suggestions!

2020-06-10 - Quote of the Week

> You don't declare lifetimes. Lifetimes come from the shape of your code, so to change what the lifetimes are, you must change the shape of the code.

– Alice Ryhl on rust-users

Thanks to RustyYato for the suggestions!

2020-06-16 - Quote of the Week

> It feels like being part of a village that learns to love the dragon it battles.

– turbinerneiter on Hacker News

Thanks to blonk for the suggestions!

2020-06-23 - Quote of the Week

> Rust's beauty lies in the countless decisions made by the development community that constantly make you feel like you can have ten cakes and eat all of them too.

– Jake McGinty et al on the tonari blog

Thanks to llogiq for the suggestions!

2020-06-30 - Quote of the Week

> References are a sharp tool and there are roughly three different approaches to sharp tools.
>
> 1. Don't give programmers sharp tools. They may make mistakes and cut their fingers off. *This is the Java/Python/Perl/Ruby/PHP... approach.*
> 2. Give programmers all the sharp tools they want. They are professionals and if they cut their fingers off it's their own fault. *This is the C/C++ approach.*
> 3. Give programmers sharp tools, but put guards on them so they can't accidentally cut their fingers off. *This is Rust's approach.*
>
> Lifetime annotations are a safety guard on references. Rust's references have no sychronization and no reference counting – that's what makes them sharp. References in category-1 languages (which typically *do* have synchronization and reference counting) are "blunted": they're not really *quite* as effective as category-2 and -3 references, but they don't cut you, and they still work; they might just slow you down a bit.
>
> So, frankly, I like lifetime annotations because they prevent me from cutting my fingers off.

– trentj on rust-users

Thanks to Ivan Tham for the suggestions!

2020-07-07 - Quote of the Week

> Rust is like a futuristic laser gun with an almost AI-like foot detector that turns the safety on when it recognises your foot.

– u/goofbe on reddit

Thanks to Synek317 for the suggestions!

2020-07-14 - Quote of the Week

> Ownership in Rust is entirely a type system fiction.

— RalfJung

> I'm not sure what is meant there. "ownership" in many languages is a very real thing to me.

– and ZiCog on rust-users

Thanks to Stephan Sokolow for the suggestions!

2020-07-21 - Quote of the Week

> `unsafe` Rust is all about flirting with UB but never giving in.

– Ralf Jung on Zulip

Thanks to HeroicKatora for the suggestions!

2020-07-28 - Quote of the Week

Sadly, we had no quote suggestions this week.

2020-08-04 - Quote of the Week

> *Empowering* is the perfect word to describe Rust in 2020. What used to be a rough adventure with many pitfalls has turned into something beautiful, something that can lift your spirit. At least, that's what it did for me.

- Mathias Lafeldt on his blog

Thanks to Henrik Tougaard for the suggestion!

2020-08-11 - Quote of the Week

> You're not allowed to use references in structs until you think Rust is easy. They're the evil-hardmode of Rust that will ruin your day.

- Kornel on rust-users

Thanks to Tom Phinney for the suggestion!

2020-08-18 - Quote of the Week

> As Dave Herman always told me, "macros are for when you run out of language". If you still have language left—and Rust gives you a lot of language—use the language first.

- Patrick Walton on twitter

Thanks to Nixon Enraght-Moony for the suggestion!

2020-08-25 - Quote of the Week

> Rust is a very different beast for me. It is a *much* bigger and *much* more capable language. However, I've found that it is, in many ways, a lot more restrictive in how you can approach problems. I frequently find myself being perplexed at how to eloquently solve a problem. When I discover the idiomatic way of doing it I'm usually both blown away by the brilliance of it and a bit disheartened by how difficult it would be to come up with that solution by myself :-).

- mikekchar on /r/rust

Thanks to Stephan Sokolow for the suggestion!

2020-09-02 - Quote of the Week

> When the answer to your question contains the word "variance" you're probably going to have a bad time.

- trentj on rust-users

Thanks to Michael Bryan for the suggestion!

2020-09-09 - Quote of the Week

> It's amazing how frequent such "rare edge cases" can be. Especially when there are millions of people using billions of files originating from God know what operating systems. Far better things are checked properly if one want robust code. As Rust uses do.

- ZiCog on rust-users

Thanks to Edoardo Morandi for the suggestion!

2020-09-16 - Quote of the Week

> When you have a lifetime <'a> on a struct, that lifetime denotes references to values stored *outside* of the struct. If you try to store a reference that points inside the struct rather than outside, you will run into a compiler error when the compiler notices you **lied** to it.

- Alice Ryhl on rust-users

Thanks to Tom Phinney for the suggestion!

2020-09-23 - Quote of the Week

> Sometimes you don't *want* the code to compile. The compiler's job is often to tell you that your code doesn't compile, rather than trying to find some meaning that allows compiling your code.

- Josh Triplett on rust-internals

Thanks to Jacob Pratt for the suggestion!

2020-09-30 - Quote of the Week

> Rust has a curse (it has many, but this one is critical): inefficient code is generally visible. Experienced developers hate to notice that their code is inefficient. They will recoil at seeing `Arc<RefCell<T>>` , but won't bat an eye at using Python.

- Esteban Kuber on rust-users

Thanks to Jon G Stødle for the suggestion!

2020-10-07 - Quote of the Week

> [...] clippy is for people who find a certain emptiness inside when they finally get code through the compiler.☺

- Unknown person answering the Rust survey

Thanks to blonk for the suggestion!

2020-10-14 - Quote of the Week

> Just because Rust allows you to write super cool non-allocating zero-copy algorithms safely, doesn't mean every algorithm you write should be super cool, zero-copy and non-allocating.

- trentj on rust-users

Thanks to Nixon Enraght-Moony for the suggestion!

2020-10-21 - Quote of the Week

> And it's true that a lot of stuff requires a "sufficiently smart compiler" but really it's 2020, if your compiler isn't serving you breakfast in bed you need to be upping your expectations.

- Jubilee on the Rust Zulip

Thanks to Josh Triplett for the suggestion!

2020-10-28 - Quote of the Week

> what many devs often miss initially when talking about Rust is that it isn't just about the design & details of the language (which is great), Rust's super power is that in combination with its fantastic community & ecosystem, and the amazing friendly people that create & form it

– Johann Andersson on twitter

llogiq is pretty pleased with his own suggestion and unanimously voted for it.

2020-11-04 - Quote of the Week

> Like other languages Rust does have footguns. The difference is that we keep ours locked up in the unsafe.

– Ted Mielczarek on twitter

Thanks to Nikolai Vazquez for the suggestion.

2020-11-11 - Quote of the Week

> There are no bad programmers, only insufficiently advanced compilers

– Esteban Kuber on twitter

Thanks to Nixon Enraght-Moony for the suggestion.

2020-11-18 - Quote of the Week

This time we have two quotes of the week:

> i just spent 8h finding a mutability bug and now i wanna be a catgirl

– @castle_vanity on twitter reacting to a post depicting C++ programmers as muscle-laden bodybuilders and Rust programmers as catgirls

Thanks to Maximilian Goisser for the suggestion.

> The code people write is first a question to the compiler, and later a story for people changing that code.

– Esteban Kuber on /r/rust

llogiq is mightily pleased with his suggestion.

2020-11-25 - Quote of the Week

> I know noting about the compiler internals but it looks to me as if 90% of the time is spent pretty-printing LayoutError.

– Vadzim Dambrouski on github

Thanks to mmmmib for the suggestion.

2020-12-02 - Quote of the Week

> Let's be clear: We understand that we are net beneficiaries of the exceptional work that others have done to make Rust thrive. AWS didn't start Rust or make it the success that it is today, but we'd like to contribute to its future success.

– Matt Asay on the AWS Open Source blog

Thanks to Alice Ryhl for the suggestion.

2020-12-09 - Quote of the Week

> Writing rust for me is a gradual process of the compiler patiently guiding me towards the program I should have written in the first place, and at the end I take all the credit.

– @felixwatts on Discord

Thanks to Joshua Nelson for the suggestion.

2020-12-16 - Quote of the Week

> Engineering is not about "not doing mistakes". Engineering is about designing systems that ensure fewer mistakes occur.
>
> Rust is such a system.

– amos on his blog

Thanks to Joshua Nelson for the suggestion.

2020-12-23 - Quote of the Week

> It took me sometime to let go and embrace getting things working before optimizing. It was a major breakthrough on that journey when I realized that ALL my python variables are `Rc<RefCell<_>>` , so any chance I had to make a variable that was less

> complicated than that was already a big optimization. If 1/10 Rust variables had to be that complicated it would not feel good, but it would already be 90% better. So if 1/50 make the code ezere to read and maintain then do it!

– Eh2406 on /r/rust

Thanks to Stephan Sokolow for the suggestion.

2020-12-30 - Quote of the Week

> This is a common theme in Rust's design: To reduce breakage as code evolves, you're only allowed to rely on features that have been intentionally declared by the author.

– 2e71828 on rust-users

Thanks to Kornel for the suggestion.

2021-01-06 - Quote of the Week

> Think of "it works" when you have UB like this: You've flipped a coin 1 time and it's come up heads and you've concluded it's never tails.

– @mirashii on the community discord

Thanks to Michael Bryan for the suggestion.

2021-01-13 - Quote of the Week

> Rust favours security over convenience. Rust does not want you to make silly little mistakes than can waste so much of your time debugging, which in the end makes it more convenient.

– @Joe232 on rust-users

Thanks to Jacob Pratt for the suggestion.

2021-01-20 - Quote of the Week

> Why do I use the letter 'o' for my generic closure param name? [...] I recently realized that since Rust uses pipes to enclose a param block, using 'o' makes the block look like a TIE fighter. I am not a terribly serious person.

– Tim Keating on medium

Thanks to Edoardo Morandi for the suggestion.

2021-01-27 - Quote of the Week

> Describing Rust as a systems programming language in 2021 is like describing Microsoft as Windows or Google as search. Yes, Rust is equipped for systems programming, but its applicability is much wider.

– Tim McNamara on twitter

Thanks to Nixon Enraght-Moony for the suggestion.

2021-02-03 - Quote of the Week

This time we had two very good quotes, I could not decide, so here are both:

> What I have been learning ... was not Rust in particular, but how to write sound software in general, and that in my opinion is the largest asset that the rust community tough me, through the language and tools that you developed.
>
> Under this prism, it was really easy for me to justify the step learning curve that Rust offers: I wanted to learn how to write sound software, writing sound software is really hard , and the Rust compiler is a really good teacher.
>
> [...]
>
> This ability to identify unsound code transcends Rust's language, and in my opinion is heavily under-represented in most cost-benefit analysis over learning Rust or not.

– Jorge Leitao on rust-users

and

> Having a fast language is not enough (ASM), and having a language with strong type guarantees neither (Haskell), and having a language with ease of use and portability also neither (Python/Java). Combine all of them together, and you get the best of all these

> worlds.
>
> Rust is not the best option for any coding philosophy, it's the option that is currently the best at combining all these philosophies.

– /u/CalligrapherMinute77 on /r/rust

Thanks to 2e71828 and Rusty Shell for their respective suggestions.

2021-02-10 - Quote of the Week

> The main theme of Rust *is not* systems programming, speed, or memory safety - it's moving runtime problems to compile time. Everything else is incidental. This is an invaluable quality of any language, and is something Rust greatly excels at.

– /u/OS6aDohpegavod4 on /r/rust

Thanks to Chris for the suggestion.

2021-02-17 - Quote of the Week

> Have you seen someone juggle several items with one hand? That's the point of async. Blocking (non-async) it like writing - it requires constant work from each hand. If you want to write twice as fast you'll need two hands and write with both at the same time. That's multithreading. If you juggle, the moment the item leaves your hand and is in the air, you have it left with nothing to do. That's similar to network IO - you make a request and are just waiting for the server to respond. You could be doing something in the meantime, like catching another item and throwing it back up again. That's what "await" does - it says I threw and item into the air, so I want my current thread / hand to switch over to catch something else now.

– /u/OS6aDohpegavod4 on /r/rust

Thanks to Jacob Pratt for the suggestion.

2021-02-24 - Quote of the Week

> Finally, I feel it is necessary to debunk the "*fighting the borrow checker*" legend, a story depicting the Rust compiler as a boogeyman: in my experience, it happens mostly to beginners and the 1% trying to micro-optimize code or push the boundaries. Most experienced Rust developers know exactly how to model their code in a way that no time is wasted fighting the compiler on design issues, and can spot anti-patterns at a glance, just like most people know how to drive their car on the correct side of the road to avoid accidents, and notice those who don't!

– Simon Chemouil on the Kraken blog

Thanks to scottmcm for the suggestion.

2021-03-03 - Quote of the Week

> It's a great example of the different attitudes of C/C++ and Rust: In C/C++ something is correct when someone can use it correctly, but in Rust something is correct when someone can't use it incorrectly.

– /u/Janohard on /r/rust

Thanks to Vlad Frolov for the suggestion.

2021-03-10 - Quote of the Week

> it's funny, every time I run into a baffling borrow error, it's preventing me from committing a real, serious mistake
>
> but it can take some thinking to figure out what exactly that mistake is..
>
> sometimes the borrow checker feels like a wise sage on a mountain giving advice in riddles lol

– Jarrett on discord

Thanks to Daniel H-M for the suggestion.

2021-03-17 - Quote of the Week

> I think the security of the internet is incredibly important obviously and I want it to be secure and I think bringing rust there is absolutely going to help it. Just by default it eliminates some of the most classic types of vulnerabilities.
>
> But I don't think that's the most exciting part. I think the most exciting part is that the set of people for whom it is possible to implement these types of things, like who writes coreutils, who writes curl, who does those things. That used to be a really small pool of people. That had to be people who knew the dark arts, and only them and only their buddies or something.

> **And it's the goal of rust to empower that to be a larger group of people** and ultimately I think that that is what is going to happen which means the sheer number of people will be larger, and also the diversity of that set of people is going to grow. And I that that that will probably actually do more for the security and usefulness of these tools than eliminating undefined behaviour.

– Ashley Williams on twitch (quote starts at 46:48)

Thanks to Nixon Enraght-Moony for the suggestion.

2021-03-24 - Quote of the Week

> This is just to say,
> I have rebased
> the feature branch
> opened against
> master
>
> and which
> you might have been
> already working
> on fixing
>
> Forgive me,
> the diff was so trivial
> so minor
> so smol

– Jubilee on rust-lang zulip

Thanks to Josh Triplett for the suggestion.

2021-03-31 - Quote of the Week

> Despite all the negative aspects, I must say that I do generally really like the poll-based approach that Rust is taking. Most of the problems encountered are encountered not because of mistakes, but because no other language really has pushed this principle this far. Programming language design is first and foremost an "artistic" activity, not a technical one, and anticipating the consequences of design choices is almost impossible.

– tomaka on medium

Thanks to Michael Howell for the suggestion.

2021-04-07 - Quote of the Week

Sadly there was no quote nominated for this week.

2021-04-14 - Quote of the Week

> What I actually value on a daily basis in [rust is] I can call code written by other people without unpleasant surprises.
>
> `async fn verify_signature(token: &Jwt) -> Result<Claims, VerificationError>`

Looking at a code snippet:

- I know my JWT token won't be mutated, just accessed (`&`);
- I know the function will probably perform some kind of I/O (`async`);
- I know that the function might fail (`Result`);
- I know its failure modes (`VerificationError`).

– Luca Palmieri on Twitter

Thanks to Nixon Enraght-Moony for the suggestion!

2021-04-21 - Quote of the Week

> We feel that Rust is now ready to join C as a practical language for implementing the [Linux] kernel. It can help us reduce the number of potential bugs and security vulnerabilities in privileged code while playing nicely with the core kernel and preserving its performance characteristics.

– Wedson Almeida Filho on the Google Security Blog

Thanks to Jacob Pratt for the suggestion!

2021-04-28 - Quote of the Week

> this error message is UNREAL

– Ash 2X3 on Twitter

Thanks to Nixon Enraght-Moony for the suggestion!

2021-05-05 - Quote of the Week

> Using R or Numpy is like driving around in a sports car. You just turn the wheel, press the pedals, and burn rubber. Rust (and other systems languages) are like getting a spaceship. You can go places and do things that you never dreamt of in a car. They are harder to pilot, but the possibilities seem unlimited! With the Rust ecosystem still in development, it feels like parts of your spaceship come in boxes of parts labeled "some assembly required".

– Erik Rose on rust-users

Thanks to Phlopsi for the suggestion!

2021-05-12 - Quote of the Week

> You won't appreciate Rust unless you spend few weeks building something in it. The initial steep learning curve could be frustrating or challenging depending on how you see it, but once past that it's hard not to love it. It's a toddler with superpowers after all

– Deepu K Sasidharan on their blog

Thanks to robin for the suggestion!

2021-05-19 - Quote of the Week

> I often think about Rust as a process and community for developing a programming language, rather than as a programming language itself.

– throwaway894345 on hacker news

Thanks to Krishna Sundarram for the suggestion!

2021-05-26 - Quote of the Week

Ok, you wanted it. Let's go full meta:

> This time, there were two crates and one quote, which is not much, but ok. Keep it up, folks!

– llogiq on reddit

Thanks to Patrice Peterson for the suggestion!

2021-06-02 - Quote of the Week

> I recently graduated with my Ph.D., after having worked on 5 different versions of my simulator, written in 4 different languages. The last version, written in pure, safe rust, worked correctly in part because of rust's strong guarantees about what 'safety' means, which I was able to leverage to turn what would normally be runtime errors into compile time errors. That let me catch errors that would normally be days or weeks of debugging into relatively simple corrections. [...] So, once again, thank you to everyone!

– Cem Karan on rust-internals

Thanks to Josh Triplett for the suggestion!

2021-06-09 - Quote of the Week

> As the tradeoffs in software engineering change over time, so does the ideal solution. Some 40 years ago when the first C standards were written down, by people no less competent than those that work on Rust today, the design of the language and the list of behaviours not defined likely made much more sense in context of back then than they do right now. It is not all that unlikely that some years down the line the choices made by Rust won't make all that much of sense as they do today, too.

– Simonas on rust-internals

Thanks to Kill The Mule for the suggestion!

2021-06-16 - Quote of the Week

> If manually managing memory is like wielding a gun, the borrow checker is an automatic safety that prevents you from pulling the trigger when you're roughly pointing it at yourself. But it's coarse-grained and errs on the side of caution; it simulates your foot as as the rectangular box that would contain it, not as a detailed 3D mesh. If you *really* think you can aim it between your toes and avoid hitting yourself (for example, "the value returned by this function must remain alive for no more than 15 successive invocations of this function"), unsafe will let you try, but the borrow checker's built-in rules isn't granular enough to help you, though it will still stop you if you accidentally put your hand in front without declaring it.

– infogulch on Hacker News

Thanks to StyMaar for the suggestion!

2021-06-23 - Quote of the Week

> At last, I can name my unsafe functions appropriately.
>
> ```
> unsafe fn extend_lifetime<'b>(r: R<'b>) -> R<'static>
> ```

– Freeky on r/rust

Thanks to Vincent de Phily for the suggestion!

2021-06-30 - Quote of the Week

> When a panic has a payload that's an object which needs Drops,
> And the panic hits a catch_unwind for unexpected stops
> Before if its Drop panicked we'd just crash to your desktops,
> Now the payload gets forgotten, and you'd better grab some mops!

– Josh Triplett on twitter

Thanks to Josh Triplett for the self-suggestion!

2021-07-07 - Quote of the Week

> One thing I like about Rust is that it filters out lazy/sloppy thinkers. Even when I disagree with another Rust programmer, there is a certain level of respect that comes from knowing that they thought about the problem deeply enough to pass the borrow checker.

– Zeroexcuses on rust-users

Thanks to Jonah for the self-suggestion!

2021-07-14 - Quote of the Week

> Beginning Rust: Uh why does the compiler stop me from doing things this is horrible
>
> Advanced Rust: Ugh why doesn't the compiler stop me from doing things this is horrible

– qDot on twitter

Thanks to Nixon Enraght-Moony for the self-suggestion!

2021-07-21 - Quote of the Week

> Tip: whenever you wonder if Pin could be the solution, it isn't

– @SkiFire13 on the official Rust Discord

Thanks to Kestrer for the self-suggestion!

2021-07-28 - Quote of the Week

> We were able to verify the safety of Rust's type system and thus show how Rust automatically and reliably prevents entire classes of programming errors

– Ralf Jung on Eureka Alert Science News

Thanks to Henrik Tougaard for the suggestion!

2021-08-04 - Quote of the Week

Sadly, this week saw no quote of the week nominations.

2021-08-10 - Quote of the Week

We regrettably lack nominations,
so as I can't choose fresh quotations,
at last nor this time,
I'll offer this rhyme
to quell all discombombulations.

– a very sorry llogiq

2021-08-18 - Quote of the Week

> **Rust** : You can't move your object and try to keep it, too.
> **Me** : Ok, I suppose I can clone it?
> **Rust** : Then implement a clone method.
> **Me** : Why am I getting a stack overflow?
> **Rust** : It is never a good idea for a clone method to call itself.
> **Me** : I just wanted to simplify the trivial cases.
> **Rust** : It is still not a good idea for a clone method to call itself.
> **Me** : I can't believe I have gotten myself into this.

– Oliver Ruebenacker on rust-users

Thanks to MBartlett21 for the suggestion!

2021-08-25 - Quote of the Week

> Code doesn't deal with resources until it does. Similarly with everything else that forces you to reason about control flow - you don't care about thread management until you do, you don't care about action logs until you do, you don't care about performance until you do... and from the other side, code doesn't need to be exception-safe until it does. The trouble with this kind of "magic" language feature is that correctness becomes non-compositional: you can take two working pieces of code and put them together and get something that doesn't work.

– Mickey Donaghy on Hacker News

Thanks to Stephan Sokolow for the suggestion!

2021-09-01 - Quote of the Week

> Anyway: the standard library docs say "check the nomicon"
> then the nomicon says "here is some advice and ultimately we don't know, maybe check UCG"
> then UCG says "ultimately we don't know it's probably like this but there's no RFC yet"
> then Ralf says "probably it should be allowed if the layout matches".

– Lokathor on the Rust Zulip

Thanks to Riccardo D'Ambrosio for the suggestion!

2021-09-08 - Quote of the Week

> In Rust, soundness is never just a convention.

– @H2CO3 on rust-users

Thanks to Riccardo D'Ambrosio for the suggestion!

2021-09-15 - Quote of the Week

> Edition!

– Niko and Daphne Matsakis on YouTube

Thanks to mark-i-m for the suggestion!

2021-09-22 - Quote of the Week

> the strains of the project have hurt a lot of people over the years and I think maybe the only path to recovery involves getting some distance from it.

– Graydon Hoare on twitter

Thanks to mmmmib for the suggestion!

2021-09-29 - Quote of the Week

This week we have two great quotes!

> The signature of your function is your contract with not only the compiler, but also users of your function.

– Quine Dot on rust-users

> Do you want to know what was harder than learning lifetimes? Learning the same lessons through twenty years of making preventable mistakes.

– Zac Burns in his RustConf talk

Thanks to Daniel H-M and Erik Zivkovic for the suggestions!

2021-10-06 - Quote of the Week

> There's a common trope among people unfamiliar with rust where they assume that if you use unsafe at all, then it's just as unsafe as C and rust provided no benefit. Comparing C's approach to safety vs Rust's is like comparing an open world assumption to a closed world assumption in formal logic systems. In C, you publish your api if it's possible to use correctly (open world). In Rust, you publish a safe api if it's **im** possible to use **in** correctly (closed world). Rust's key innovation here is that it enables you to build a 'bridge' from open world (unsafe) to a closed world (safe), a seemingly impossible feat that feels like somehow pairwise reducing an uncountable infinity with a countable infinity. Rust's decision to design an analogous closed-world assumption for safe code is extremely powerful, but it seems very hard for old school C programmers to wrap their head around it.

– /u/infogulch on /r/rust

Thanks to Alice Ryhl for the suggestion!

2021-10-13 - Quote of the Week

> Rust is the language where you get the hangover first.

– unattributed via Niko Matsakis' RustConf keynote

Thanks to Alice Ryhl for the suggestion!

2021-10-20 - Quote of the Week

> The biggest failure in Rust's communication strategy has been the inability to explain to non-experts that unsafe abstractions are the point, not a sign of failure.

– withoutboats on twitter

Thanks to Alice Ryhl for the suggestion!

2021-10-27 - Quote of the Week

> I think in general "force the user to think about the extra cases, and be explicit about ignoring them" is definitely idiomatic rust.

– Daniel Wagner Hall on rust-internals

Thanks to robin for the suggestion!

2021-11-03 - Quote of the Week

> I always tell myself that code quickly written just to compile looks like Order 66 executed on Christmas day
>
> [...]
>
> Clones and unwrapping as far as the eye can see.

– Dhghomon on /r/rust

Thanks to UtherII for the suggestion!

2021-11-10 - Quote of the Week

> And even if you could fix all of rustc's soundness holes, or otherwise prevent user code from exploiting them, a soundness bug in any third-party library can also make it possible for malicious crates to trigger arbitrary behavior from safe code.

> [...]
>
> This is why we need to emphasize that while Rust's static analyses are very good at limiting accidental vulnerabilties in non-malicious code, they are not a sandbox system that can place meaningful limits on malicious code.

– Matt Brubeck on rust-users

Thanks to robin for the suggestion!

2021-11-17 - Quote of the Week

> If a normal add is a waffle iron , SIMD add is a double or quadruple waffle iron. You can make 2 or 4 or more waffles at the same time.
>
> In case of waffles it would be called SIMW: **S** ingle **I** ron, **M** ultiple **W** affles.
>
> It's not multithreading - because you open and close the waffle iron for all the waffles at the same time.

– /u/EarthyFeet on /r/rust

Editors note: Do yourself a favor, click the link and read the whole thread, it's pure gold (*chef's kiss*).

Thanks to Stephan Sokolow for the suggestion!

2021-11-24 - Quote of the Week

> On the topic of reframing UB, I was reminded of an article about the mechanics of oaths and vows in historical cultures.
>
> When a programmer writes `get_unchecked` , we can imagine them wanting to promise the compiler that they uphold its preconditions. But since the compiler is normally not so trusting of unproven assertions, the programmer swears an *oath* that their argument is in bounds.
>
> The compiler, seeing such a solemn commitment, treats the programmer's word as true and optimizes accordingly. The compiler is so thoroughly convinced that it never even entertains the possibility of doubting the programmer's oath.
>
> But if the programmer has sworn falsely, then they might well suffer divine retribution in the form of nasal demons — or worse, subtly baffling program behaviour.

– /u/scook0 on /r/rust

Thanks to G. Thorondorsen for the suggestion!

2021-12-01 - Quote of the Week

> The design of the safe/unsafe split means that there is an asymmetric trust relationship between Safe and Unsafe Rust. Safe Rust inherently has to trust that any Unsafe Rust it touches has been written correctly. On the other hand, Unsafe Rust cannot trust Safe Rust without care.
>
> As an example, Rust has the `PartialOrd` and `Ord` traits to differentiate between types which can "just" be compared, and those that provide a "total" ordering (which basically means that comparison behaves reasonably).
>
> `BTreeMap` doesn't really make sense for partially-ordered types, and so it requires that its keys implement `Ord` . However, `BTreeMap` has Unsafe Rust code inside of its implementation. Because it would be unacceptable for a sloppy `Ord` implementation (which is Safe to write) to cause Undefined Behavior, the Unsafe code in BTreeMap must be written to be robust against `Ord` implementations which aren't actually total — even though that's the whole point of requiring `Ord` .

– Gankra citing the Rustonomicon on github

Thanks to robin for the suggestion!

2021-12-08 - Quote of the Week

> v2 of the patch-series "to add support for Rust as a second language to the Linux kernel" was posted to LKML [..]
>
> There have been several improvements to the overall Rust support since RFC and v2 described in the linked mail.

– Thorsten Leemhuis on twitter

llogiq unanimously suggested and voted that this be our quote for this week.

2021-12-15 - Quote of the Week

> This is safer than you may think, because those who need async tend to know it themselves and don't ask "should I use async"

> question. In other words, asking itself is a signal that answer is no. MITM proxy case was a rare exception.

– Seo Sanghyeon on rust-users

Thanks to Zeroexcuses for the suggestion!

2021-12-22 - Quote of the Week

> Important crab-related diagnostics improvement shipping in nightly @rustlang
>
> ```
> error: Ferris cannot be used as an identifier
> --> src/main.rs:2:9
>   |
> 2 |     let    = 123;
>   |         ^^ help: try using their name instead: `ferris`
> 3 |
> 4 |     for i in 0..   {
>   |
> ```

– Mara Bos on twitter

Thanks to Julian Wollersberger for the suggestion!

2021-12-29 - Quote of the Week

> One reason we keep certain things as hard errors rather than lints: it establishes a baseline that you can safely assume about other people's code, since it can't be turned off. And as a result, that baseline can become part of people's mental model of Rust itself, rather than something that might or might not be true in any given codebase.
>
> We have to take care to not use that lightly, because that places work on all users of Rust to maintain code to that baseline. But there are cases where we do. We don't allow using one integer type where another was expected. We don't allow certain operations outside an unsafe block. ...
>
> I think the standard we should apply is asking whether something is part of the baseline that people should be able to assume about all Rust code, and if that's worth the tradeoff of requiring that baseline of all Rust users.

– Josh Triplett on rust-internals

Thanks to Josh Triplett for the self-suggestion!

2022-01-05 - Quote of the Week

> I performed an extremely scientific poll on twitter, and determined this is not how it's pronounced
>
> Well, it really is `Vec<T, A>`, pronounced Veck-tah. 😛
>
> Look, I moved away from Boston to avoid this sort of thing 😁.

– the8472 & Thom Chiovoloni on github

Thanks to Josh Triplett for the suggestion!

2022-01-12 - Quote of the Week

> Language stability is not just about semver compatibility. It's also about not burdening developers to have to make new decisions when looking at old code. [Language instability] creates churn and debate about things that previously didn't require it.

– skysch on rust-internals

Thanks to Christopher Durham for the suggestion!

2022-01-19 - Quote of the Week

> Usually Rust figures out the Sartre question by itself

– kornel on rust-users

Thanks to H2CO3 for the suggestion!

2022-01-26 - Quote of the Week

> Rust : We have a race condition bug in our standard filesystem library !
> C++ : You guys have a concurrency safe standard filesystem library ?
> C : You guys have a standard filesystem library ?

– redditmodsareshits on /r/cpp

Thanks to UtherII for the suggestion!

2022-02-02 - Quote of the Week

- `impl Not for !` (did you guess that "not never" is still "never"?)

– llogiq on last week's TWiR

Thanks to scottmcm for the suggestion!

2022-02-09 - Quote of the Week

> As the temporary human substitute for the temporarily unavailable automated representative of the governance process, I would like to thank the author for their work and everyone else who contributed.

– Mara Bos (on behalf of RFCbot) on github

Thanks to Josh Triplett for the suggestion!

2022-02-16 - Quote of the Week

> I still get excited about programming languages. But these days, it's not so much because of what they let me do, but rather what they don't let me do.

– Amos blogging about mistakes Rust doesn't catch

Thanks to Rob Donnelly for the suggestion!

2022-02-23 - Quote of the Week

> There's a big difference between solving a problem and making a problem go away

– Patrick Doyle on rust-users

Thanks to Michael Bryan for the suggestion!

2022-03-02 - Quote of the Week

> Due to recent events I feel the need to once again commend the reviewers and ehuss in particular for their amazing communication skills when reviewing PRs like this. I can only imagine how much work it means and how silly some of the changes proposed here might look to a seasoned cargo developer, yet you maintain a constructive, upbeat, and friendly spirit at all times. It's a style that I am aspiring when reviewing PRs myself, and is a prime example for the accessibility and friendliness of the Rust community as a whole.
>
> Thank you!

– Sebastian Thiel commending Eric Huss on GitHub

Thanks to Jacob Finkelman for the suggestion!

2022-03-09 - Quote of the Week

> Because it is designed not to own. If you need an owning pointer, use Box.
>
> This is like asking "why there is no chocolate mousse in this burger?". Chocolate mousse is delicious, but it does not belong in a burger. If you want chocolate mousse, then that's fine and you can choose to eat it instead of a burger. But at other times you may want a burger instead.

– H2CO3 answering why raw pointers don't own on rust-users

Thanks to Deep Majumder for the suggestion!

2022-03-16 - Quote of the Week

> protip: the rust extern keyword has a –help flag
>
> ```
> error[E0703]: invalid ABI: found `--help`
>  --> ext.rs:1:8
>   |
> 1 | extern "--help" {}  fn main() {}
>   |        ^^^^^^^^ invalid ABI
>   |
>   = help: valid ABIs: Rust, C, C-unwind, cdecl, stdcall, stdcall-unwind, fastcall, vectorcall, thiscall, thiscall-u
>
> error: aborting due to previous error
>
> For more information about this error, try `rustc --explain E0703`.
> ```

– Aria the Cat (with some help from rustc) on twitter

Thanks to Jacob Pratt for the suggestion!

2022-03-23 - Quote of the Week

> today I learned that `unsafe` is also a tool for people who are actively looking to implement bugs.

– blonk on rust-users

Thanks to Michael Bryan for the suggestion!

2022-03-30 - Quote of the Week

> All that to say that Rust does precisely this great job at decoupling some of these notions that have been, historically, quite tangled for a while; and for those used to that environment with everything muddied, it can be a bit hard to take a step back and rethink these distinctions that Rust makes.

– Daniel H-M on rust-users

Thanks to H2CO3 for the suggestion!

2022-04-06 - Quote of the Week

> I've seen similar sentiments echoed before, elsewhere. The point it's making is the same one that's argued whenever people say you should learn LISP because it'll make you a better programmer.
>
> There's no such thing as a perfectly intuitive programming language because algorithmic thinking isn't something that comes to us intuitively. That's why the first language is always the hardest.
>
> It's helpful and mind-expanding to learn new paradigms and force yourself out of old cognitive ruts. Thus, from an "improving your ability to solve problems and function as a programmer" perspective, what makes Rust difficult is valuable because it's forcing you to learn to think about problems in new ways.
>
> That's the distinction between necessary complexity and complexity due to ill-considered design. (Similar to how, in video games, there's a difference between genuine difficulty and difficulty caused by something like a crappy control scheme.)

– Stephan Sokolow on rust-users (in our quotes thread!)

Thanks to Christopher Durham for the suggestion!

2022-04-13 - Quote of the Week

> Relying on the programmer to always read, comprehend, and remember the documentation – and then do everything right, every time – is how we get bugs.

– Cliff Biffle on his blog

Thanks to Brian Kung for the suggestion!

2022-04-20 - Quote of the Week

Alas, this week went by without any memorable quote.

2022-04-27 - Quote of the Week

> This is the most fundamental philosophy of both the Rust language and the Rust project: we don't think it's sufficient to build robust

> systems by only including people who don't make mistakes; we think it's better to provide tooling and process to catch and prevent mistakes.

– Jane Lusby on the inside Rust blog

Thanks to farnbams for the suggestion!

2022-05-04 - Quote of the Week

> "Ah but logic errors can happen with all languages" yes and I'm sure trains occasionally run into trees as well, but cars are way more likely to. 🙃

– amos on twitter

Thanks to Jacques "erelde" Rimbault for the suggestion!

2022-05-11 - Quote of the Week

> At Cloudflare we have big Rust projects/teams and onboard new developers regularly.
>
> There is a learning curve. Rust is rigid and unforgiving, and noobs need assistance when the compiler says "no" (although error messages and Clippy do a good job for common mistakes).
>
> However, the big upside is that noobs can contribute safely to Rust projects. Rust limits severity of the damage an inexperienced programmer can cause. Once they manage to get the code to compile, it already has lots of correctness guarantees. "Bad" Rust code may just clone more than strictly necessary, or write 10 lines of code for something that has a helper method in the stdlib, but it won't corrupt memory or blindly run the happy path without checking for errors. Rust prefers to be locally explicit, so it's also easy to review.

– Kornel.Lesiński on lobste.rs

2022-05-18 - Quote of the Week

> It is worth remembering that there is an infinitely large set of programs but a very small set of useful programs. A programming language is a form of ad-hoc compression algorithm, it is intended to sort the set of useful programs to have shorter encodings than the undesirable ones. Programs with certain categories of error or security vulnerability should be harder or impossible to express.

– david_chisnall on lobste.rs

Thanks to Anton Fetisov for the suggestion!

2022-05-25 - Quote of the Week

> This is the difference in approaches of the two languages. In C++ if the code is vulnerable, the blame is on the programmer. In Rust if the code is vulnerable, Rust considers it a failure of the language, and takes responsibility to stop even "bad" programmers from writing vulnerable code. I can't stress enough how awesome it is that I can be a careless fool, and still write perfectly robust highly multi-threaded code that never crashes.

– kornel on lobste.rs (with a caveat from ZiCog that Rust does *not* guarantee freedom from all vulnerabilities!)

Thanks to Brian Kung for the suggestion!

2022-06-01 - Quote of the Week

> Rust is a perfect language for a dad like me, who every day puts kids to sleep, and tired after long day of work and chores, can sit down and possibly write some code for the hobby open source project, even when he's already just half awake. And it usually just works, tend to be robust and make the day feel extra productive.

– Dawid Ciężarkiewicz on /r/rust

2022-06-08 - Quote of the Week

> I wrote a bespoke time-series database in Rust a few years ago, and it has had exactly one issue since I stood it up in production, and that was due to pessimistic filesystem access patterns, rather than the language. This thing is handling hundreds of thousands of inserts per second, and it's even threaded.
>
> Given that I've been programming professionally for over a decade in Python, Perl, Ruby, C, C++, Javascript, Java, and Rust, I'll pick Rust absolutely any time that I want something running that I won't get called at 3 AM to fix. It probably took me 5 times as long to write it as if I did it in Go or Python, but I guarantee it's saved me 10 times as much time I would have otherwise spent triaging, debugging, and running disaster recovery.

– Taywee on hacker news

Thanks to Erich Gubler for the suggestion.

2022-06-15 - Quote of the Week

> Because lower-level software has more operational constraints than higher-level software (e.g. it typically cannot tolerate a runtime or memory management via garbage collection), developing a memory safe language suitable for systems software is particularly challenging. The Rust language has met that challenge, however, and is an excellent candidate for replacing C in many systems applications.
>
> We plan to invest in the tools that allow systems engineers to move their software to Rust. This means investing in improving package management, compilers, and Foreign Function Interface (FFI) generators. In many cases this will include providing interfaces compatible with existing widely-used components to enable transition. With these tools, adoption of a memory safe alternative will scale much faster without replication of efforts.

– The White House Open Source Software Mobilization Plan, multiple authors (PDF link)

Thanks to Brian Kung for the suggestion.

2022-06-22 - Quote of the Week

> Rwlock vs Mutex? Please, tell me like I'm 5
>
> Mutex: "Mom says it's my turn on the synchronization primitive."
> vs.
> Write lock: "Hey! You all are not allowed to look until I'm done writing!" Read lock: "Hey! You are not allowed to edit what you wrote until we're done reading it!"
>
> Thanks for an actual 5 year old reply, made me laugh

– /u/LyonSyonII and /u/everything-narrative on /r/rust

Thanks to Josh Triplett for the suggestion!

2022-06-29 - Quote of the Week

> JG: mem::replace / mem::swap === Indiana Jones swapping the artifact for a bag of sand in a temple
>
> CV: except rustc would tell Indy that's a type mismatch
>
> JG: Yes, that would be the boulder, I assume.
>
> Older compilers were more aggressive in error reporting.

Jake Goulding and Cuviper on the Rust Zulip

Thanks to Josh Triplett for the suggestion.

2022-07-06 - Quote of the Week

> TIL: #cargo has build-in aliases for some commands
>
> so next time try out
>
> cargo r
> cargo b
> cargo t

– @5422m4n on twitter

llogiq is pretty pleased with his choice.

2022-07-13 - Quote of the Week

> Learning Rust has taught me something - "There are really no problems, just adventure and opportunities"

– Adeoye Adefemi on rust-users

Thanks to Adeoye Adefemi and Anton Fetisov for the suggestion as well as Christopher Durham for the leniency.

2022-07-20 - Quote of the Week

> The long compile times where all responsibility is taken away from you is infinitely more effective than submission patterns in BDSM, where the graceful rustc takes over and all you have to do is wait until they tell you that you're a good person and that everything is alright!

– /u/whyvitamins on /r/rust

Thanks to Jacob Pratt for the suggestion!

2022-07-27 - Quote of the Week

> # JuSt Be CaReFuL
>
> If there's one lesson from decades of software engineering, it is the failure of "just be careful" as a strategy. C/C++ programmers still experience memory corruption constantly, no matter how careful they are. Java programmers still frequently see NullPointerExceptions, no matter how careful they are. And so on. One of the reasons that Rust is so successful is that it adds automated checks to prevent many common mistakes.

– Robert Grosse on their blog

Thanks to robin for the suggestion!

2022-08-03 - Quote of the Week

> ❤ ❤
>
> 100,000 issues filled with love, compassion and a wholesome community. Thank you, Rust community, for being one of the most, if not straight out the most, welcoming programming communities out there. Thank you, Rust teams, for the tireless hours you spend every day on every aspect of this project. Thank you to the Rust team alumni for the many hours spent growing a plant and the humility of passing it to people you trust to continue taking care of it. Thank you everyone for RFCs, giving voice to the community, being those voices AND listening to each other.
>
> This community has been and continue to be one of the best I have ever had the pleasure of being a part of. The language itself has many things to love and appreciate about it, from the humane error messages to giving the people the power to express high performance code without sacrificing readability for the ones to come after us. But nothing, truly nothing, takes the cake as much as the community that's building it, answering questions, helping and loving each other. Every single day.
>
> Congratulations everyone for 100,000 issues and PRs! And thank you for being you. Because Rust is Beautiful, for having you as part of it.
>
> To the times we spent together and the many more to come!

– mathspy on the rust-lang/rust github

Thanks to Sean Chen for the suggestion!

2022-08-10 - Quote of the Week

> Don't come empty-handed to a project saying "this could be rewritten in Rust". It's obnoxious and gives the rust community a bad name.
> Do start the project on your own, adding Rust to the build system and converting some significant functions, and then ask the project's community for comments.

– moltonel on /r/rust

Thanks to zjp-CN for the suggestion!

2022-08-17 - Quote of the Week

> TL;DR: my claim is that Rust is attempting to **raise the abstraction** in the programming language and ultimately to join **computer science** and **software engineering** into one single discipline, an ambition that has been around since these disciplines were created.

– Linus Walleij on his blog

Thanks to Julian Wollersberger for the suggestion!

2022-08-24 - Quote of the Week

> A fast executing language that crashes all the time is like a supercar... that crashes all the time.

– Tris on youtube

Thanks to scottmcm for the suggestion!

2022-08-31 - Quote of the Week

> [W]e reached a tipping point. **We decided to move our entire codebase to Rust**... . Rust seemed to give us all the capabilities we needed, **however, there was still one *minor* problem - no one on the team knew Rust.** ...
>
> We started with a small team of senior engineers and managers learning Rust and developing the skeleton of the DB and dev environment (for others to build on). Then, slowly, others joined in rewriting and contributing different components until we eventually got rid of the old codebase altogether (I still remember the day my original C modules, from the first days of Pinecone, were taken out). Unbeknownst to most Pinecone customers, the new Rust core was deployed in March this year. And in the process of taking over running workloads, we managed not to drop a single API call!
>
> ... **We all expect[ed] performance and dev processes to improve. Those indeed happened.** What we didn't expect was the extent to which dev velocity increased and operational incidents decreased. **Dev velocity** ... **improved dramatically with Rust. Built-in testing, CI/CD, benchmarking, and an overzealous compiler increased engineers' confidence in pushing changes, and enabled them to work on the same code sections and contribute simultaneously without breaking the code base.** Most impressively though, **real time operational events dropped almost to zero overnight after the original release.** Sure, there are still surprises here and there but, by and large, the core engine has been shockingly stable and predictable.

– Edo Liberty on the pinecone blog

Thanks to Erich Gubler for the suggestion!

2022-09-07 - Quote of the Week

> So long, and thanks for all the turbofish.

– moltonel on r/rust

Thanks to Josh Triplett for the suggestion!

2022-09-14 - Quote of the Week

> In Rust We Trust

– Alexander Sidorov on Medium

Thanks to Anton Fetisov for the suggestion!

2022-09-21 - Quote of the Week

> At the #LinuxPlumbers Rust MC: "I'm Matthew Wilcox, I'm one of the authors of the NVMe spec, I'm the one who suggested you make an NVMe driver to demonstrate the value of Rust. You have succeeded beyond my wildest expectations. These performance numbers are phenomenal."

– Josh Triplett paraphrasing Matthew Wilcox as spoken at the Linux Plumbers Conference Q&A session

Thanks to Josh Triplett for the self-suggestion!

2022-09-28 - Quote of the Week

> Semver has its philosophy, but a pragmatic approach to versioning is:
>
> `<upgrades may break API> . <downgrades may break API> . <fine either way>`

– Kornel on rust-users

Thanks to Artem Borisovskiy for the suggestion!

2022-10-05 - Quote of the Week

> BurntSushi is a super experienced programmer who always seems to know what's right
>
> Shepmaster occasionally pops up to keep things level, and provides definitive answers and edits to all stackoverflow questions
>
> Epage is the ecosystem guy thanklessly maintaining the things that make the magic of cargo possible
>
> Dtolnay is an AI written in rust with the sole purpose of improving rust.

– trevg_123 on r/rust

Thanks to musicmatze for the suggestion!

2022-10-12 - Quote of the Week

> There's a lot of weird debate about whether Rust in the kernel is useful or not... in my experience, it's way more useful than I could've ever imagined!
>
> I went from 1st render to a stable desktop that can run run games, browsers, etc. in about two days of work on my driver (!!!)
>
> All the concurrency bugs just vanish with Rust! Memory gets freed when it needs to be freed! Once you learn to make Rust work with you, I feel like it guides you into writing correct code, even beyond the language's safety promises. It's seriously magic!
>
> There is absolutely no way I wouldn't have run into race conditions, UAFs, memory leaks, and all kinds of badness if I'd been writing this in C.
>
> In Rust? Just some logic bugs and some core memory management issues. Once those were fixed, the rest of the driver just worked!!

– Asahi Lina on twitter

llogiq is mightily pleased with his suggestion.

2022-10-19 - Quote of the Week

> I think it's worth noting that the fact that this program fails to compile whereas the analogous Python runs but gives the wrong answer is *exactly what Rust's ownership and borrowing system is about*.

– Kevin Reid on rust-users

Thanks to Kill The Mule for the suggestion!

2022-10-26 - Quote of the Week

> Also, I don't know how much of this is because Rust is special or because BurntSushi is a national treasure and his CSV library is impeccably constructed and documented.

– Gabe Durazo on github

Thanks to scottmcm for the suggestion!

2022-11-02 - Quote of the Week

> I'm getting more convinced that Rust code is generally going to end up faster than C++ code every day I work on optimizations.
>
> Strong immutability and no-alias guarantees are a game-changer and we've only really begun to scratch the surface of what can be done.

– Patrick Walton on twitter

llogiq is exceedingly pleased with his suggestion.

2022-11-09 - Quote of the Week

> Meanwhile the Rust shop has covers on everything and tag-out to even change settings of the multi-axis laser cutter, but you get trusted with said laser cutter on your first day, and if someone gets hurt people wonder how to make the shop safer.

– masklinn on r/rust

Thanks to Anton Fetisov for the suggestion!

2022-11-16 - Quote of the Week

> What you are essentially saying is: "Doctor, I'm writing C in Rust, and it hurts." To which the doctor will reply: "Then don't write C in Rust, and it won't hurt!"

– Árpád Goretity on rust-users

Thanks to Michael Bryan for the suggestion!

2022-11-23 - Quote of the Week

> While working on these userspace Mesa changes today, I did not hit a single GPU kernel driver bug. Not. A. Single. Bug.

> This is thanks to Lina's phenomenal efforts. She took a gamble writing the kernel driver in Rust, knowing it would take longer to get to the first triangle but believing it would make for a more robust driver in the end. She was right.
>
> A few months of Lina's Rust development has produced a more stable driver than years of development in C on certain mainline Linux GPU kernel drivers.
>
> I think... I think I have Rust envy
>
> ....Or maybe just Lina envy 😊

– Alyssa Rosenzweig tooting on Mastodon

Thanks to Brian Kung for the suggestion!

2022-11-30 - Quote of the Week

> After many years of writing bugs, and then discovering Rust, I learned to appreciate explicitness in code, and hope you eventually will too.

– Artem Borisovskiy on rust-users

Thanks to Árpád Goretity for the suggestion!

2022-12-07 - Quote of the Week

> To date, there have been zero memory safety vulnerabilities discovered in Android's Rust code.

– Jeffrey Vander Stoep on the google security team blog

Thanks to Anton Fetisov for the suggestion!

2022-12-14 - Quote of the Week

> ... you can lead a horse to git but you cannot make it commit.

– /u/kibwen on /r/rust

Thanks to Anton Fetisov for the suggestion!

2022-12-21 - Quote of the Week

> In the depths of a computer's core,
> Where bits and bytes are stored,
> Lies a tool that's often ignored
> But without it, things would be floored.
>
> It's the rust borrow checker,
> A guardian of memory,
> Ensuring that data is in the right place
> And never causing miseries.
>
> With each line of code it carefully scans,
> Checking for underflows and overflows,
> Preventing errors, saving the day,
> And keeping the program in a flow.
>
> So let's give a nod to this silent hero,
> Whose work may go unnoticed, but is never zero,
> It's the rust borrow checker,
> A vital part of the machine,
> Ensuring our programs run clean.

– ChatGPT prompted by Vivek Yadav

llogiq is quite self-appreciative for the suggestion.

2022-12-28 - Quote of the Week

> Rust does best when we're ambitious

– Niko Matsakis quoted by Yoshua Wuyts on his blog

llogiq is inordinately pleased with his suggestion and thanks Yoshua for clearing the quote!

2023-01-04 - Quote of the Week

> You haven't "fooled" rustc, you are using unsafe code. Unsafe code means that all you can do is fool yourself.

– Frank Steffahn on rust-users

Thanks to Quine Dot for the suggestion!

2023-01-11 - Quote of the Week

> Now macros are fine, I mean we use them for implementing internals and you know if you have something that [...] needs to be implemented for lots and lots of different concrete types, then macros are a fine choice for that, but exposing that to users is something to be very careful about.

– Raph Levien

llogiq is a tad sad there were no suggestions, but still likes the quote he ended up with!

2023-01-18 - Quote of the Week

> Common arguments against Rust's safety guarantees:
>
> - The library you're binding to can have a segfault in it.
> - RAM can physically fail, causing dangling pointers.
> - The computer the Rust program is running on can be hit by a meteorite.
> - Alan Turing can come back from the dead and tell everyone that he actually made up computer science and none of it is real, thus invalidating every program ever made, including all Rust programs.

– Ironmask on the phoronix forums

Thanks to Stephan Sokolow for the suggestion!

2023-01-25 - Quote of the Week

> Rust has demonstrated that you using a type system as a vehicle for separation logic works, even in imperative languages, and it's nothing as arcane as those immutable functional predecessors would suggest. It did this by making sure the language defines a type system that helps you, by making sure core properties of soundness *can* be expressed in it.
>
> - soundness requirement for memory access: lifetimes
> - soundness requirements for references *with* value semantics: > `&`/`&mut _`
> - soundness requirements for resources: `Copy` and `Drop`
> - making sure your logic is monotic: traits instead of inheritance, lack of specialization (yes, that's a *feature*).
> - (notably missing: no dependent types; apparently not 'necessary' but I'm sure it could be useful; however, research is heavily ongoing; caution is good)
>
> This allows the standard library to encode all of its relevant requirements as types. And doing this everywhere is its soundness property: safe functions have no requirements beyond the sum of its parameter type, `unsafe functions` can. Nothing new or special there, nothing that makes Rust's notion of soundness special.
>
> Basing your mathematical reasoning on separation logic makes soundness reviews *local* instead of requiring whole program analysis. This is what makes it practical. It did this pretty successfully and principled, but did no single truly revolutionary thing. It's a sum of good bits from the last decade of type system research. That's probably why people refer to it as 'the soundness definition', it's just a very poignant way to say: "we learned that a practical type systems works as a proof checker".

– HeroicKatora on /r/cpp

Thanks to Stephan Sokolow for the suggestion!

2023-02-01 - Quote of the Week

> Compilers are an error reporting tool with a code generation side-gig.

– Esteban Küber on Hacker News

Thanks to Stefan Majewsky for the suggestion!

2023-02-08 - Quote of the Week

> It's been 7.5 years since #27060 was reported, but the problem is finally fixed for good. :)

– Ralf Jung on github

Thanks to scottmcm for the suggestion!

2023-02-15 - Quote of the Week

> All the pro C/C++ arguments seem to come down to “Good drivers don’t need seat belts because they don’t get in accidents”

– otwkme on /r/rust

2023-02-22 - Quote of the Week

> It’s *enjoyable* to write Rust, which is maybe kind of weird to say, but it’s just the language is fantastic. It’s fun. You feel like a magician, and that never happens in other languages.

– Parker Timmerman cited in a TechnologyReview article

Thanks to robin for the suggestion!

2023-03-01 - Quote of the Week

> You’ve probably come across unsafe. So “unsafe” is a keyword that sort of unlocks super powers and segfaults.

– Arthur Cohen during FOSDEM ‘23

Thanks to blonk for the suggestion!

2023-03-08 - Quote of the Week

> (...) as much as i dislike the cargo-geiger concept, the name ... kind of works
>
> `unsafe` is a lot like uranium. it’s just one more metal ore you can process, refine, and machine. it doesn’t combust in atmosphere, it doesn’t corrode or make weird acids. unless you go out of your way to make it dangerous you don’t even have to worry about critical masses. you can work with it pretty normally most of the time
>
> but if you don’t know exactly what it is, what it does, and how to work with it, it will cause mysterious illnesses that only crop up long after you’ve stopped touching it

– Alexander Payne on /r/rust

Thanks to Stephan Sokolow for the suggestion!

2023-03-15 - Quote of the Week

> The Rust compiler is a thousand unit tests that you don’t have to write

– Someone, likely Ian Purton on the Cloak blog

Thanks to Stephan Sokolow for the suggestion!

2023-03-22 - Quote of the Week

> The generated program is a random sequence of bytes that just happens to take the shape of a seemingly working program by accident. Such is the joy of code that causes UB. You cannot deduce anything from what happens when you execute a program with UB, since that act is by itself meaningless. You need to establish that your program has no UB before making any inference based on what you see the program do after it came out of the compiler.

– Ralf Jung on github

Thanks to bugaevc for the suggestion!

2023-03-29 - Quote of the Week

> As part of this work, I even found two memory safety bugs in the DRM scheduler component that were causing kernel oopses for Alyssa and other developers, so the Rust driver work also benefits other kernel drivers that use this shared code! Meanwhile, I still haven’t gotten any reports of kernel oopses due to bugs in the Rust code at all.

– Asahi Lina on the Asahi Linux blog

llogiq is patting himself on the back for the suggestion!

2023-04-05 - Quote of the Week

> As usual, the borrow checker is correct: we are doing memory crimes.

– Ohad Ravid on his blog

Thanks to Jelte Fennema for the suggestion!

2023-04-12 - Quote of the Week

> As an expert at being ignorant of what `Pin` does, I can assert with expertise that other ignorant readers have a hard time with `Pin`.

– grom on rust-users

Thanks to bugaevc for the suggestion!

2023-04-19 - Quote of the Week

> Error types should be located near to their unit of fallibility.

– Sabrina Jewson on her blog

Thanks to Anton Fetisov for the suggestion!

2023-04-26 - Quote of the Week

> That said, I really like the language. It's as if someone set out to design a programming language, and just picked all the right answers. Great ecosystem, flawless cross platform, built-in build tools, no "magic", static binaries, performance-focused, built-in concurrency checks. Maybe these "correct" choices are just laser-targeted at my soul, but in my experience, once you leap over the initial hurdles, it all just works™, without much fanfare.

– John Austin on his blog

Thanks to Ivan Tham for the suggestion!

2023-05-03 - Quote of the Week

> Since it hasn't been said before, there is an important distinction that needs to be addressed. For anyone who has been doing embedded work for any length of time and hasn't yet been exposed to Rust, the only thing that can really be said is that the language is entirely unlike everything you've experienced before. There is just nothing comparable, and the only way to rationalize questions like *why use Rust at all* is to put some honest effort into learning and using it.
>
> Hearing things like "it's a bit like C++ except it's memory safe and thread safe, and it's actually practical to build kernels with it" will not sound convincing. You have to see it to believe it.
>
> It's as if you've spent an entire career writing assembly, and one day you hear something or other about a brand-new programming language claiming to be a "portable assembler" called C. It sounds too good to be true. And then the years pass, and all of the mystery and disbelief gives way to obviousness and precision engineering. That's sort of how it is when going from C to Rust.

– Jay Oster

Thanks to Michael Bryan for the suggestion!

2023-05-10 - Quote of the Week

> Thanks to all for the very helpful responses. "The Book" says *The community is very welcoming and happy to answer students' questions* "; I expected that to be just marketing, but I was wrong."

– Daryl Lee on rust-users

Thanks to evann for the suggestion!

2023-05-17 - Quote of the Week

> That's one of the great things about Rust: sometimes you can do something really dumb and get away with it.

– Rik Arends at RustNL

Thanks to Josh Triplett for the suggestion!

2023-05-24 - Quote of the Week

> I guess the nicest example of this phenomenon is shared mutability. Programmers have been arguing for decades whether it is sharing xor mutability that causes memory safety bugs:

- "It's threads!" – shouted JavaScript and Python, and JS remained single-threaded, and Python introduced the GIL.
- "It's mutability!" – screamed Haskell and Erlang, and they made (almost) everything immutable.

> And then along came Rust, and said: "you are fools! You can have both sharing and mutability in the same language, as long as you isolate them from each other."

– H2CO3 on rust-users

Thanks to Jacob Pratt for the suggestion!

2023-05-31 - Quote of the Week

> Panics are overgrown ASSERTs, not an underbuilt exception system.

– Stephan Sokolow on hacker news

Thanks to Stephan Sokolow for the self-suggestion!

2023-06-07 - Quote of the Week

> (...) Rust developers usually are not just looking for "less buggy".
>
> They are addicted to the clicky sound of legos.

– Amirography on fosstodon

Thanks to Jan Riemer for the suggestion!

2023-06-14 - Quote of the Week

Alas this week remains quoteless for lack of suggestions.

2023-06-21 - Quote of the Week

> rust programmers when they see each other again:
>
> > Long time no C

– ciscoffeine on mond-basis.eu

Thanks to Brian Kung for the suggestion!

2023-06-28 - Quote of the Week

> It's a compiler not a Jedi, don't expect it to read minds.

– Nishant on github

Thanks to Nishant for the self-suggestion!

2023-07-05 - Quote of the Week

> I'm not here to tell you that Rust is the best language..... you should have figured that out by now.

– Jester Hartman on youtube

Thanks to newpavlov for the suggestion!

2023-07-12 - Quote of the Week

> It's all ducks and sunshine until something starts barking.

– u/ZZaaaccc on r/rust

Thanks to Patrice Peterson for the suggestion!

2023-07-19 - Quote of the Week

> (...) complexity in programming is just like energy in physics: it cannot be created, nor destroyed, but only transformed. So, if a programming language is simple and can only express very simple concepts, the complexity is going to move from the language constructs to your source code and vice versa. One needs to find a balance here, it's a personal choice based on mindset and experience.

– u/inamestuff on r/rust

Thanks to Arthur Rodrigues for the suggestion!

2023-07-26 - Quote of the Week

> A rustacean is a programmer that dislikes being told “yes” in situations where they’ll regret it later.

– Predrag Gruevski on mastodon

Thanks to Kevin Mehall for the suggestion!

2023-08-02 - Quote of the Week

> Writing return \; at the end of a function in Rust is a bit like answering the question “Do you like potatoes?” with “Yes, I like potatoes” instead of simple “Yes”.

– Artem Borisovskiy on rust-users

Thanks to Todd Fleming for the suggestion!

2023-08-09 - Quote of the Week

> Claiming Rust won’t help you because you’re doing so many unsafe things is like claiming protective gear won’t help you because you’re handling so many dangerous substances.

– llogiq on twitter

llogiq feels very smug about his self-suggestion!

2023-08-16 - Quote of the Week

> It has been
>
> **0**
>
> days since someone tried and failed to use unsafe to circumvent the lifetime system.

– H2CO3 on rust-users

Thanks to mdHMUpeyf8yluPfXI for the suggestion!

2023-08-23 - Quote of the Week

> [...] there’s no benefit to haranguing people.
>
> Unless they use three spaces for indentation. *Those* people need to be relentlessly mocked and publicly harassed until they see sense and use *five* spaces like all proper, *civilised* people do. Damn barbarians...

– Daniel Keep on rust-users

Thanks to Jonas Fassbender for the suggestion!

2023-08-30 - Quote of the Week

> In [other languages], I could end up chasing silly bugs and waste time debugging and tracing to find that I made a typo or ran into a language quirk that gave me an unexpected nil pointer. That situation is almost non-existent in Rust, it’s just me and the problem. Rust is honest and upfront about its quirks and will yell at you about it before you have a hard to find bug in production.

– dannersy on Hacker News

Thanks to Kyle Strand for the suggestion!

2023-09-06 - Quote of the Week

> Rusts standard library, and a lot of the popular crates, are like a museum. While it does change, as new exhibitions are added, it is mostly finished. Each painting has a detailed explanation in 7 different languages underneath. Descriptions below each excitation are written beautifully, with detailed drawings, showing how everything works. It is so easy to navigate, one glance at the map is enough to find exactly what you are looking for. It is so convenient, you almost don’t notice that you are learning something.
>
> Internals of `rustc` are like a build site of a sprawling factory. You can see the scaffolds everywhere, as more production lines come online, and everything gets faster, better, bigger. Workers move around, knowing the place like the back of their hands. They can

> glance at the signs on the walls, and instantly tell you: where you are, what this place does and what pitfalls you should avoid. And you are a new hire who has just came for his first day at the new job. You look at the sign, and after some thinking, you too are able to tell roughly in which building you are. The signs almost always tell you what you need, just in short, cryptic sentences. You always can tell what is going on, with some thinking, but it is not effortless. The signs on the walls are not *bad*, just not written for anyone to get right away.

– FractalFir on their blog

Thanks to Alona Enraght-Moony for the suggestion!

2023-09-13 - Quote of the Week

> It's very much a positive feedback loop: good tooling makes good tooling easier to build, so more of it gets built and the cycle repeats. `cargo-semver-checks` stands on the shoulders of giants like `rustc` and `rustdoc` and Trustfall. Remove any one of them (or even just `rustc`'s high-quality diagnostics!) and `cargo-semver-checks` wouldn't have been a viable project at all.

– Predrag Gruevski on /r/rust

Thanks to Vincent de Phily for the suggestion!

2023-09-20 - Quote of the Week

> This is the first programming language I've learned that makes it so easy to make test cases! It's actually a pleasure to implement them.

– 0xMB on rust-users

Thanks to Moy2010 for the suggestion!

2023-09-27 - Quote of the Week

> The problem with Rust it appears,
> that it leaves programmers in tears
> if they have to go back
> to languages that lack
> in short they've got feature-arrears.

– llogiq on /r/rust

Thanks to Frank Steffahn for the suggestion!

2023-10-04 - Quote of the Week

> I've been writing Rust code everyday for years, and I used to say Rust wasn't great for writing prototypes because if forced you to ask yourself many questions that you may want to avoid at that time.
>
> I recently realized this is all wrong: you can write Rust pretty much as fast as you can write code in any other language, with a meaningful difference: with a little discipline it's easy to make the rough edges obvious so you can sort them out later.

– /u/moiaussi4213 on /r/rust

There was no suggestion this week, but llogiq is pleased with his choice nonetheless!

2023-10-11 - Quote of the Week

> The Rust mission – let you write software that's fast and correct, productively – has never been more alive. So next Rustconf, I plan to celebrate:
>
> - All the buffer overflows I didn't create, thanks to Rust
> - All the unit tests I didn't have to write, thanks to its type system
> - All the null checks I didn't have to write thanks to Option and Result
> - All the JS I didn't have to write thanks to WebAssembly
> - All the impossible states I didn't have to assert "This can never actually happen"
> - All the JSON field keys I didn't have to manually type in thanks to Serde
> - All the missing SQL column bugs I caught at compiletime thanks to Diesel
> - All the race conditions I never had to worry about thanks to the borrow checker
> - All the connections I can accept concurrently thanks to Tokio
> - All the formatting comments I didn't have to leave on PRs thanks to Rustfmt
> - All the performance footguns I didn't create thanks to Clippy

– Adam Chalmers in their RustConf 2023 recap

Thanks to robin for the suggestion!

2023-10-18 - Quote of the Week

> When your Rust build times get slower after adding some procedural macros:
>
> We call that the syn tax

– janet on fosstodon.org

Thanks to Jacob Pratt for the suggestion!

2023-10-25 - Quote of the Week

> When your Rust build times get slower after adding some procedural macros:
>
> We call that the syn tax :ferris:

– Janet on Fosstodon

Thanks to Jacob Pratt for the suggestion!

2023-11-01 - Quote of the Week

> After doing a best fit, we found Rust projects were less likely to introduce vulnerabilities than their equivalent C++ projects at all relevant experience levels, but more importantly, we found the effect was most significant for first-time contributors, who were almost two orders of magnitude less likely to contribute vulnerabilities. That is, even though Rust may have a reputation as a harder language to learn, there is a very measurable effect that makes it better for newbies. Reviewers should not have to put as much effort into reviewing code to be confident that someone making their first foray into their project is accidentally adding a vulnerability.

– Justin Tracey on crysp.org

Thanks to Brian Kung for the suggestion!

2023-11-08 - Quote of the Week

> For Binder to continue to meet Android's needs, we need better ways to manage (and reduce!) complexity without increasing the risk.
>
> The biggest change is obviously the choice of programming language. We decided to use Rust because it directly addresses a number of the challenges within Binder that we have faced during the last years.

– Alice Ryhl on the Linux Kernel Mailing List

Thanks to Vincent de Phily for the suggestion!

2023-11-15 - Quote of the Week

> I decided to keep learning Rust because I liked the syntax. I liked the speed. I liked the community. I liked it all. It felt like a breath of fresh air: a syntax more intuitive than Python, JavaScript, or C, yet still faster.

– Goren Barak on their blog

Thanks to Goren Barak for the self-suggestion!

2023-11-22 - Quote of the Week

> If you require it, measure it. That's the simple answer. Everything else is guesswork.

– Johannes Lade on rust-users

Thanks to Michael Bryan for the suggestion!

2023-11-29 - Quote of the Week

> I'd like to report that Rust's compile times were OK today and yesterday and the day before.
>
> I'll keep you posted.

– ZiCog about slow Rust compile times on rust-users

Thanks to Michael Bryan for the suggestion!

2023-12-06 - Quote of the Week

> NVIDIA's firmware, Airlie said, comes with a set of include files that, in turn, define structures that change over time. To deal with these changes, the driver is going to need some sort of automated ABI generation; he noted that the developers working on the Apple M1 GPU driver have run into the same problem. This problem could be made easier to tackle, he suggested, if the driver were, like the M1 driver, to be rewritten in Rust.

– Jonathan Corbet paraphrasing David Airlie on Linux Weekly News

Thanks to Brian Kung for the suggestion!

2023-12-13 - Quote of the Week

Sadly, the week went by without a nominated quote.

2023-12-20 - Quote of the Week

> The Tianyi-33 satellite is a 50kg class space science experimental satellite equipped with an operating system independently developed by Beijing University of Posts and Telecommunications—the Rust-based dual-kernel real-time operating system **RROS**. RROS will carry out general tasks represented by tensorflow/k8s and real-time tasks represented by real-time file systems and real-time network transmission on the satellite. It will ensure the normal execution of upper-layer applications and scientific research tasks, such as time-delay measurement between satellite and ground, live video broadcasting, onboard web chat services, pseudo-SSH experiments, etc. This marks the world's first official application of a Rust-written dual-kernel operating system in a satellite scenario.

– Qichen on the RROS web page

Thanks to Brian Kung for the suggestion!

2023-12-27 - Quote of the Week

> Rust can be rather more verbose than C; there are a lot of invariants that have to be expressed in the code. But that is countered by the need for far less error-handling code; it turns out to be a wash, with the size of the two implementations being about the same.

– Alice Ryhl at the Linux Plumbers Conference as quoted by Jonathan Corbet, LWN

Thanks to Ivan Fraixedes for the suggestion!

2024-01-03 - Quote of the Week

> Some people don't believe in life after death... Rust doesn't believe in magic after compilation.

– Stephan Sokolow on rust-users

Thanks to Todd Fleming for the suggestion!

2024-01-10 - Quote of the Week

> - Modular
> - Very high quality construction compared to its competitors
> - If you leave it lying around forget about it, stepping into a project is painful?

– Leonardo Giovanni Scur on mastodon explaining how bevy is like Lego™

Thanks to Jan Riemer for the suggestion!

2024-01-17 - Quote of the Week

> Congrats to the #Rustlang and #Rust-for-#Linux community: the #LinuxKernel now contains the first useful thing built using Rust!

– Thorsten Leemhuis on FOSStodon

As with the crate of the week, this week saw a total lack of suggestions, so llogiq would like to offer you this piece of good news from the Linux side of things.

2024-01-24 - Quote of the Week

> The functional ML roots of the language, Graydon's first Rust compiler was written in OCaml, shine through, influencing it right from

> the start.
>
> It's not "C++ but better".
>
> It's Haskell standing on Lisp's shoulders, hiding in C's coat to sneak into PRDCTN. (The fancy nightclub where all the popular languages hang out)

– tris on his "No Boilerplate" Youtube channel

Thanks to PrototypeNM1 for the suggestion!

2024-01-31 - Quote of the Week

> The sheer stability of this program is what made me use rust for everything going forward. The social-service has a 100% uptime for almost 2.5 years now. It's processed 12.9TB of traffic and is still using 1.5mb of ram just like the day we ran it 2.5 years ago. The resource usage is so low it brings tears to my eyes. As someone who came from Java, the lack of OOM errors or GC problems has been a huge benefit of rust and I don't ever see myself using any other programming language. I'm a big fan of the mindset "build it once, but build it the right way" which is why rust is always my choice.

– /u/Tiflotin on /r/rust

Thanks to Brian Kung for the suggestion!

2024-02-07 - Quote of the Week

> My take on this is that you cannot use async Rust correctly and fluently without understanding Arc, Mutex, the mutability of variables/references, and how async and await syntax compiles in the end. Rust forces you to understand how and why things are the way they are. It gives you minimal abstraction to do things that could've been tedious to do yourself.
>
> I got a chance to work on two projects that drastically forced me to understand how async/await works. The first one is to transform a library that is completely sync and only requires a sync trait to talk to the outside service. This all sounds fine, right? Well, this becomes a problem when we try to port it into browsers. The browser is single-threaded and cannot block the JavaScript runtime at all! It is arguably the most weird environment for Rust users. It is simply impossible to rewrite the whole library, as it has already been shipped to production on other platforms.
>
> What we did instead was rewrite the network part using async syntax, but using our own generator. The idea is simple: the generator produces a future when called, and the produced future can be awaited. But! The produced future contains an arc pointer to the generator. That means we can feed the generator the value we are waiting for, then the caller who holds the reference to the generator can feed the result back to the function and resume it. For the browser, we use the native browser API to derive the network communications; for other platforms, we just use regular blocking network calls. The external interface remains unchanged for other platforms.
>
> Honestly, I don't think any other language out there could possibly do this. Maybe C or C++, but which will never have the same development speed and developer experience.
>
> I believe people have already mentioned it, but the current asynchronous model of Rust is the most reasonable choice. It does create pain for developers, but on the other hand, there is no better asynchronous model for Embedded or WebAssembly.

– /u/Top_Outlandishness78 on /r/rust

Thanks to Brian Kung for the suggestion!

2024-02-14 - Quote of the Week

> For some weird reason the Elixir Discord community has a distinct lack of programmer-socks-wearing queer furries, at least compared to Rust, or even most other tech-y Discord servers I've seen. It caused some weird cognitive dissonance. Why do I feel vaguely strange hanging out online with all these kind, knowledgeable, friendly and compassionate techbro's? Then I see a name I recognized from elsewhere and my hindbrain goes "oh thank gods, I know for a fact she's actually a snow leopard in her free time". Okay, this nitpick is firmly tongue-in-cheek, but the Rust user-base continues to be a fascinating case study in how many weirdos you can get together in one place when you very explicitly say it's ok to be a weirdo.

– SimonHeath on the alopex Wiki's ElixirNitpicks page

Thanks to Brian Kung for the suggestion!

2024-02-21 - Quote of the Week

> Shared mutable state is evil, and you can solve it by forbidding mutation, or by forbidding sharing. Rust supports both.

– kornel on Lobste.rs

Thanks to Aleksey Kladov for the suggestion!

2024-02-28 - Quote of the Week

> That would take 18 million terabytes of RAM. You don't have that much memory.

– Alice Ryhl answering "What is MAX array size" on rust-users

Thanks to Zeroexcuses for the suggestion!

2024-03-06 - Quote of the Week

> My experience with C++ is that, as I've become more of an expert in the language, I've become more disillusioned with it. It's incredibly hard to do things that you should be able to do in software. And, it's a huge problem for me to constantly be helping other engineers debug the same bugs over and over. It's always another use after free. I've probably debugged 300 of those. [...]
>
> In our experience using the Rust ecosystem for almost three years now, I don't think we found a bug in a single Rust crate that we've pulled off the shelf. We found a bug in one of them and that was a Rust crate wrapping a C library and the bug was in the C library. The software quality that you kind of get for free is amazing.

– Carter Schultz interviewed on the filtra blog

Thanks to George Barwood for the suggestion!

2024-03-13 - Quote of the Week

> In 10 years we went from "Rust will never replace C and C++" to "New C/C++ should not be written anymore, and you should use Rust". Good job.

– dpc_pw on lobste.rs

Thanks to Dennis Luxen for the suggestion!

2024-03-20 - Quote of the Week

> In 10 years we went from "Rust will never replace C and C++" to "New C/C++ should not be written anymore, and you should use Rust". Good job.

– dpc_pw on lobste.rs

Thanks to Dennis Luxen for the suggestion!

2024-03-27 - Quote of the Week

> "Top contributor" is not a place of glory, it *should* go to a bot because people should work at a sustainable pace and prioritize touching grass every once in a while. If a person ever works harder than bors, that's a problem!

– Carol (Nichols || Goulding) on rust-internals

Thanks to Anton Fetisov for the suggestion!