Rust in the Linux ecosystem

Miguel Ojeda
ojeda@kernel.org
Credits & Acknowledgments

Rust
...for being a breath of fresh air

Kernel maintainers
...for being open-minded

Everyone that has helped Rust for Linux
(see credits in the RFC & patch series)
History

30 years of Linux

30 years of ISO C
Love story

30 years of Linux ❤ 30 years of ISO C

* Terms and Conditions Apply.
Why is C a good system programming language?
Why is C a good system programming language?

“Do you see any language except C which is suitable for development of operating systems?”
Why is C a good system programming language?

“Do you see any language except C which is suitable for development of operating systems?”

“I like interacting with hardware from a software perspective. And I have yet to see a language that comes even close to C.”

— Linus Torvalds 2012
Why is C a good system programming language?

“You can use C to generate good code for hardware.”

“When I read C, I know what the assembly language will look like.”

“The people that designed C ... designed it at a time when compilers had to be simple.”

“If you think like a computer, writing C actually makes sense.”
But...
But... UB
3.4.3

1 undefined behavior

behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this document imposes no requirements

2 Note 1 to entry: Possible undefined behavior ranges from ignoring the situation completely with unpredictable results, to behaving during translation or program execution in a documented manner characteristic of the environment (with or without the issuance of a diagnostic message), to terminating a translation or execution (with the issuance of a diagnostic message).

3 Note 2 to entry: J.2 gives an overview over properties of C programs that lead to undefined behavior.

4 EXAMPLE An example of undefined behavior is the behavior on dereferencing a null pointer.
Example of UB

— The value of the second operand of the / or % operator is zero (6.5.5).

```c
int f(int a, int b) {
    return a / b;
}
```
Example of UB

— The value of the second operand of the / or % operator is zero (6.5.5).

```c
int f(int a, int b) {
    return a / b;
}

UB \forall x f(x, 0);
```
Example of UB

Any other inputs that trigger UB?

```c
int f(int a, int b) {
    return a / b;
}
```
Example of UB

Any other inputs that trigger UB?

```c
int f(int a, int b) {
    return a / b;
}

UB f(INT_MIN, -1);
```
Instances of UB
Instances of UB

— The value of the second operand of the / or % operator is zero (6.5.5).
Instances of UB

The execution of a program contains a data race (5.1.2.4).

The second operand of the / or % operator is zero (6.5.5).
Instances of UB

- The execution of a program contains a data race (5.1.2.4).
- The second operand of the / or % operator is zero (6.5.5).
- An object is referred to outside of its lifetime (6.2.4).
The value of a pointer to an object whose lifetime has ended is used (6.2.4).

The second operand of the operator is zero (6.5.5).

An object is referred to outside of its lifetime (6.2.4).
An object is referred to outside of its lifetime (6.2.4).

The value of an object with automatic storage duration is used while it is indeterminate (6.2.4).

As a data race (5.1.2.4).

The value of a pointer whose lifetime has ended is used (6.7.9, 6.8).

Instances of UB
An object is referred to outside of its lifetime (6.2.4).

Instances of a program whose lifetime has ended is used (6.2.4).

The value of an object with a duration that does not have character type (6.2.6).

A trap representation is read by an I-value expression that is indeterminate (6.2.4).

A data race (5.1.2.4).
The value of an object with duration is used while it is indeterminate (6.2.4).

A trap representation is read by an lvalue expression that does not have character type (6.2.4).

The value of an array object is subtracted (6.5.6).

Instances of UB (6.7.9, 6.8).

An object is to outside of its lifetime (6.2.4).

Pointers that do not point into, or just beyond, the same array object are subtracted (6.2.4).
Pointers that do not point into, or just beyond, the same array object are subtracted (6.2.4).

The value of an object with a trap representation is read by an Ivalue expression that does not have character type (6.2.4).

A trap representation is read by an Ivalue expression that does not have character type (6.2.4).

The value of an object with a trap representation is read by an Ivalue expression that does not have character type (6.2.4).

A trap representation is read by an Ivalue expression that does not have character type (6.2.4).

As a data race (5.1.2.4).

Instances of UB

An object is said to be outside of its lifetime (6.2.4).

ew; second operand of the operator is

whose lifetime, operator is used is (6.2.4).
So, what does Rust offer?
So, what does Rust offer?
Safety

Safety in Rust

≡

No undefined behavior
Safety

Safety in Rust ≠ Safety in “safety-critical” as in functional safety (DO-178B/C, ISO 26262, EN 50128…)

[Images of airplane, car, and train]
Safety examples

abort()s in C are Rust-safe
Safety examples

abort()s in C

⇒

are

Rust-safe

Even if your company goes bankrupt.
Safety examples

abort(())s in C

⇒

are Rust-safe

Even if your company goes bankrupt.
Even if somebody is injured.
Avoiding UB

```c
int f(int a, int b) {
    if (b == 0)
        abort();
    if (a == INT_MIN && b == -1)
        abort();
    return a / b;
}
```
Avoiding UB

```c
int f(int a, int b) {
    if (b == 0)
        abort();

    if (a == INT_MIN && b == -1)
        abort();

    return a / b;
}
```

f is a safe function
Safety examples

Rust panics are Rust-safe
Safety examples

Kernel panics are Rust-safe.
Safety examples

Uses after free, null derefs, double frees, OOB accesses, uninitialized memory reads, invalid inhabitants, data races...

⇒

are not Rust-safe
Safety examples

Uses after free, null derefs, double frees, OOB accesses, uninitialized memory reads, invalid inhabitants, data races...

⇒

are not

Rust-safe

Even if your system still works.
Safety examples

⇒

Race conditions are Rust-safe
Safety examples

⇒

Memory leaks are Rust-safe
Safety examples

⇒

Deadlocks are Rust-safe
Safety examples

⇒

Integer overflows are Rust-safe
Is avoiding UB that important?
Is avoiding UB that important?

\[ \sim 70\% \]

of vulnerabilities in C/C++ projects come from UB
Is avoiding UB that important?

~70% of the vulnerabilities Microsoft assigns a CVE each year continue to be memory safety issues

— https://msrc-blog.microsoft.com/2019/07/18/we-need-a-safer-systems-programming-language/
Is avoiding UB that important?

Mojave (aka macOS 10.14)

Apple released macOS 10.14 Mojave on September 24, 2018 and subsequently has issued 6 point releases.

<table>
<thead>
<tr>
<th>Total CVE Count</th>
<th>Memory Unsafety Bugs</th>
<th>Percentage</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>36</td>
<td>81.8%</td>
<td>10.14.6</td>
</tr>
<tr>
<td>45</td>
<td>40</td>
<td>88.9%</td>
<td>10.14.5</td>
</tr>
<tr>
<td>38</td>
<td>20</td>
<td>52.6%</td>
<td>10.14.4</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>95.7%</td>
<td>10.14.3</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>84.6%</td>
<td>10.14.2</td>
</tr>
<tr>
<td>71</td>
<td>40</td>
<td>56.3%</td>
<td>10.14.1</td>
</tr>
<tr>
<td>64</td>
<td>44</td>
<td>68.8%</td>
<td>10.14</td>
</tr>
</tbody>
</table>

Is avoiding UB that important?

The Chromium project finds that around 70% of our serious security bugs are memory safety problems. Our next major project is to prevent such bugs at source.

Is avoiding UB that important?

Most of Android’s vulnerabilities occur in the media and bluetooth components. Use-after-free (UAF), integer overflows, and out of bounds (OOB) reads/writes comprise 90% of vulnerabilities with OOB being the most common.

— https://security.googleblog.com/2019/05/queue-hardening-enhancements.html
Is avoiding UB that important?

Fish in a Barrel @LazyFishBarrel · Sep 9
5/8 vulnerabilities fixed in Firefox 92 are memory unsafety mozila.org/en-US/security... #memoryunsafety

Security Vulnerabilities fixed in Firefox 92
 mozila.org

Fish in a Barrel @LazyFishBarrel · Sep 1
13/19 (5/5 high) vulnerabilities fixed in Google Chrome 93.0.4577.63 are memory unsafety chromereleases.googleblog.com/2021/08/stable... #memoryunsafety

Stable Channel Update for Desktop
The Chrome team is delighted to announce the promotion of Chrome 93 to the stable channel for ... chromereleases.googleblog.com
Sure, UB is an issue and safe Rust does not have it…
Sure, UB is an issue and safe Rust does not have it…

...but does Rust really help, though?
Does Rust help?
Does Rust help?

I took a look at this spreadsheet published three weeks ago...
Does Rust help?

I took a look at this spreadsheet published three weeks ago...

Fuzzing 100+ open source projects with OSS-Fuzz - lessons learned.

31st August, 2021

David Korczynski & Adam Korczynski,
Security Research & Security Engineering

Does Rust help?

I filled the language column and plotted...
Does Rust help?

I filled the language column and plotted...
Does Rust help?

I filled the language column and plotted...
Does Rust help?

I filled the language column and plotted...
What else does Rust offer?

Language
What else does Rust offer?

Language

- Stricter type system
- Safe/unsafe split
- Modules & visibility
- Generics
- RAII
- Lifetimes
- Pattern matching
- Sum types
- Shared & exclusive references
- Modules & visibility
- Lifetimes
- Shared & exclusive references

Powerful hygienic and procedural macros
What else does Rust offer?

Standard library
What else does Rust offer?

- Standard library
  - Vocabulary types like `Result` and `Option`
  - Formatting
  - Collections
  - Processes & Threads
  - Paths & Filesystem
- Pinning
- Networking
- Iterators
- Checked, saturating & wrapping integer arithmetic primitives
What else does Rust offer?

Tooling
What else does Rust offer?

- Documentation generator
- Unit & integration tests
- Static analyzer
- Build system
- C ↔ Rust bindings generators
- Linter
- Macro debugging
- Formatter
- Great compiler error messages
- IDE tooling
- UBSAN-like interpreter
What else does Rust offer?

- Documentation generator
- Unit & integration tests
- Static analyzer
- Build system
- C ↔ Rust bindings generators
- Linter
- Macro debugging
- Formatter
- IDE tooling
- Great compiler error messages
- UBSAN-like interpreter

*plus the usual friends: gdb, lldb, perf, valgrind...*
The Rust community’s crate registry

Instantly publish your crates and install them. Use the API to interact and find out more information about available crates. Become a contributor and enhance the site with your work.

New Crates
- tracing-awc v0.1.0-beta.4
- rustfuck v0.1.1
- hashicorp-iru v0.1.5
- key-format v0.0.0

Most Downloaded
- rand
- syn
- rand core
- libc

Just Updated
- async-graphql-viz v0.1.0-alpha.13
- retroqwest v0.0.1-rc.4
- air- interpreter-wasm v0.14.0-alpha.22
- viz v0.1.0-alpha.13

9,427,285,669 Downloads
67,753 Crates in stock
What is the catch?
What is the catch?

Cannot model everything ⇒ Unsafe code required
What is the catch?

Cannot model everything ⇒ Unsafe code required

More information to provide ⇒ More complex language
What is the catch?

- Cannot model everything ⇒ Unsafe code required
- More information to provide ⇒ More complex language
- Extra runtime checks ⇒ Potentially expensive
What is the catch?

- Cannot model everything ⇒ Unsafe code required
- More information to provide ⇒ More complex language
- Extra runtime checks ⇒ Potentially expensive
- An extra language to learn ⇒ Logistics & maintenance burden
Why is C a good system programming language?

“You can use C to generate good code for hardware.”

“When I read C, I know what the assembly language will look like.”

“The people that designed C ... designed it at a time when compilers had to be simple.”

“If you think like a computer, writing C actually makes sense.”

Fast
Low-level
Simple
Fits the domain
Why is C a good system programming language?

“The people that designed C ... designed it at a time when compilers had to be simple.”

“If you think like a computer, writing C actually makes sense.”

“You can use C to generate good code for hardware.”

“When I read C, I know what the assembly language will look like.”

Fast: Yes
Low-level: Sometimes
Simple: Not really
Fits the domain: ...
Who is using Rust?
Production users

HealPay

We have been pushing Rust into every project we can. Currently we have several backend services built in rust.

Metaswitch

Rust is the primary programming language for all communications solutions architected using microservices methodologies.

360dialog

Most of our service consumers are written with Rust.
Mozilla

Building the **Servo** browser engine, integrating into Firefox, other projects.

Atlassian

We use Rust in a service for analyzing petabytes of source code.

System76

As a Linux-based computer-manufacturer, much of our infrastructure and desktop Linux projects are written in Rust. From hardware certification, flashing, and imaging; to system services and GTK3 desktop applications.
Fire and Emergency NZ

The New Zealand Fire Service is using a custom geolocation search engine, built in rust, that runs on embedded hardware within a fire truck to stream hazard information to a fire crew at an incident.

Dropbox

Optimizing cloud file storage.

Cloudflare

We are using Rust as a replacement for memory-unsafe languages (particularly C) and are using it in our core edge logic.
1Password

We use Rust to power the entire backend (encryption, networking, database, and business logic) of all our client apps.

Deliveroo

We are using Rust to quickly make assignment decisions in our food delivery network.

Canonical

Everything from server monitoring to middleware!
Projects written in Rust
Servo’s mission is to provide an independent, modular, embeddable web engine, which allows developers to deliver content and applications using web standards.

Servo is written in Rust, and shares code with Mozilla Firefox and the wider Rust ecosystem. Since its creation in 2012, Servo has contributed to W3C/WHATWG web standards by reporting specification issues and submitting new cross-browser automated tests, and core team members have co-edited new standards that have been adopted by other browsers. As a result, the Servo project helps drive the entire web platform forward while building on a platform of reusable, modular technologies that implement web standards.

Support and donations 🍀

Interested in helping out the Servo Project? Please do! You could write code or documentation, test nightlies and file issues, or donate to help cover the project’s CI and hosting costs. If you know a company that would like to support the Servo Project, please have a look at our Participation Agreement or get in touch: info@servo.org.
The Rust Programming Language

This is the main source code repository for Rust. It contains the compiler, standard library, and documentation.

Note: this README is for users rather than contributors. If you wish to contribute to the compiler, you should read the Getting Started section of the rustc-dev-guide instead.

Quick Start

Read "Installation" from The Book.

Installing from Source

The Rust build system uses a Python script called x.py to build the compiler, which manages the bootstrapping process. It lives in the root of the project.

The x.py command can be run directly on most systems in the following format:

```
./x.py <subcommand> [flags]
```

This is how the documentation and examples assume you are running x.py.

Systems such as Ubuntu 20.04 LTS do not create the necessary python command by default when Python is installed that allows x.py to be run directly. In that case you can either create a symlink for python (Ubuntu provides the python-is-python3 package for this), or run x.py using Python itself:

```
# Python 3
python3 x.py <subcommand> [flags]
```

```
# Python 3.7
python3.7 x.py <subcommand> [flags]
```
Redox is a Unix-like Operating System written in Rust, aiming to bring the innovations of Rust to a modern microkernel and full set of applications.

- Implemented in Rust
- Microkernel Design
- Includes optional GUI - Orbital
- Supports Rust Standard Library

- MIT Licensed
- Drivers run in Userspace
- Includes common Unix commands
- Custom libc written in Rust (relibc)
Our mission is to enable secure, multi-tenant, minimal-overhead execution of container and function workloads.

Read more about the Firecracker Charter [here](#).

**What is Firecracker?**

Firecracker is an open source virtualization technology that is purpose-built for creating and managing secure, multi-tenant container and function-based services that provide serverless operational models. Firecracker runs workloads in lightweight virtual machines, called microVMs, which combine the security and isolation properties provided by hardware virtualization technology with the speed and flexibility of containers.

**Overview**

The main component of Firecracker is a virtual machine monitor (VMM) that uses the Linux Kernel Virtual Machine (KVM) to create and run microVMs. Firecracker has a minimalist design. It excludes unnecessary devices and guest-facing functionality to reduce the memory footprint and attack surface area of each microVM. This improves security, decreases the startup time, and increases hardware utilization. Firecracker has also been integrated in container runtimes, for example Kata Containers and Weaveworks Ignite.

Firecracker was developed at Amazon Web Services to accelerate the speed and efficiency of services like AWS Lambda.
ripgrep (rg)

ripgrep is a line-oriented search tool that recursively searches the current directory for a regex pattern. By default, ripgrep will respect .gitignore rules and automatically skip hidden files/directories and binary files. ripgrep has first class support on Windows, macOS and Linux, with binary downloads available for every release. ripgrep is similar to other popular search tools like The Silver Searcher, ack and grep.

Dual-licensed under MIT or the UNLICENSE.

CHANGELOG

Please see the CHANGELOG for a release history.

Documentation quick links

- Installation
- User Guide
- Frequently Asked Questions
- Regex syntax
- Configuration files
- Shell completions
- Building
- Translations
# hyperfine

A command-line benchmarking tool.

Demo: Benchmarking `fd` and `find`:

```
$ hyperfine --warmup 3 'fd -e jpg -uu' 'find -iname "*.jpg"'
Benchmark #1: fd -e jpg -uu
  Time (mean ± std): 329.5 ms ± 1.9 ms  [User: 1.019 s, System: 1.433 s]
  Range (min ... max): 326.6 ms - 333.6 ms

Benchmark #2: find -iname "*.jpg"
  Time (mean ± std): 1.253 s ± 0.016 s  [User: 461.2 s, System: 777.0 ms]
  Range (min ... max): 1.233 s - 1.278 s

Summary
'fd -e jpg -uu' ran
3.80 ± 0.05 times faster than 'Find -iname "*.jpg"'
```

## Features

- Statistical analysis across multiple runs.
- Support for arbitrary shell commands.
- Constant feedback about the benchmark progress and current estimates.
- Warmup runs can be executed before the actual benchmark.
- Warmup runs can be executed before each timing run.
- Cache-clearing commands can be set up before each timing run.
- Statistical outlier detection to detect interference from other programs and caching effects.
**wgpu**

wgpu is a cross-platform, safe, pure-rust graphics api. It runs natively on Vulkan, Metal, D3D12, D3D11, and OpenGL; and on top of WebGPU on wasm.

The api is based on the WebGPU standard. It serves as the core of the WebGPU integration in Firefox, Servo, and Deno.

**Repo Overview**

The repository hosts the following libraries:

- wgpu v0.16.1 | docs | passing - User facing Rust API.
- wgpu-core v0.10.2 | docs | passing - Internal WebGPU implementation.
- wgpu-hal v0.10.7 | docs | passing - Internal unsafe GPU API abstraction layer.
- wgpu-types v0.10.0 | docs | passing - Rust types shared between all crates.
- deno_webgpu v0.16.0 - WebGPU implementation for the Deno JavaScript/TypeScript runtime

The following binaries:

- player - standalone application for replaying the API traces.
Welcome to Veloren!

Veloren is a multiplayer voxel RPG written in Rust. It is inspired by games such as Cube World, Legend of Zelda: Breath of the Wild, Dwarf Fortress and Minecraft.

Veloren is fully open-source, licensed under GPL 3. It uses original graphics, musics and other assets created by its community. Being contributor-driven, its development community and user community is one and the same: developers, players, artists and musicians come together to develop the game.
Links

https://servo.org/
https://github.com/rust-lang/rust
https://www.redox-os.org/
https://github.com/firecracker-microvm/firecracker
https://github.com/BurntSushi/ripgrep
https://github.com/sharkdp/hyperfine
https://github.com/gfx-rs/wgpu
https://veloren.net/
https://rg3d.rs
Projects looking to take advantage of Rust
Rust for Linux
Memory safety for the Internet's most critical infrastructure

Initiatives

NTP
Let's create a memory safe NTP implementation.

View initiative  Project Status: Pending funding
curl
Let's make TLS and HTTP networking code in curl memory-safe.

View initiative  Project Status: In progress

Rustls
Let's get the Rustls TLS library ready to replace OpenSSL in as many projects as possible.

View initiative  Project Status: In progress

mod_tls
Let's make it possible to use memory safe TLS networking in Apache httpd.

View initiative  Project Status: In progress
Entities supporting Rust
Members

Founding Platinum

aws  Google  HUAWEI

Microsoft  mozilla
In the kernel...

“**Google** supports and contributes directly to the Rust for Linux project. Our Android team is evaluating a new Binder implementation and considering other drivers where Rust could be adopted.”

[Source](https://lore.kernel.org/lkml/20210704202756.29107-1-ojeda@kernel.org/)
In the kernel...

“Arm recognises the Rust value proposition and is actively working with the Rust community to improve Rust for Arm based systems. A good example is Arm’s RFC contribution to the Rust language which made Linux on 64-bit Arm systems a Tier-1 Rust supported platform. Rustaceans at Arm are excited about the Rust for Linux initiative and look forward to assisting in this effort.”

— https://lore.kernel.org/lkml/20210704202756.29107-1-ojeda@kernel.org/
In the kernel...

“Microsoft’s Linux Systems Group is interested in contributing to getting Rust into Linux kernel. Hopefully we will be able to submit select Hyper-V drivers written in Rust in the coming months.”

— https://lore.kernel.org/lkml/20210704202756.29107-1-ojeda@kernel.org/
Rust in the Linux ecosystem

Miguel Ojeda
ojeda@kernel.org
Backup slides
THE RUST PROGRAMMING LANGUAGE

STEVE KLARBK and CAROL NICHOLS,
with CONTRIBUTIONS from THE RUST COMMUNITY

COVERS RUST 2018

Programming Rust
Fast, Safe Systems Development

Jim Blandy,
Jason Orendorff &
Leonora F. S. Tindall
C Charter

6. **Keep the spirit of C.** The Committee kept as a major goal to preserve the traditional spirit of C. There are many facets of the spirit of C, but the essence is a community sentiment of the underlying principles upon which the C language is based. The C11 revision added a new facet to the original list of facets. The new spirit of C can be summarized in phrases like:

(a) *Trust the programmer.*
(b) *Don't prevent the programmer from doing what needs to be done.*
(c) *Keep the language small and simple.*
(d) *Provide only one way to do an operation.*
(e) *Make it fast, even if it is not guaranteed to be portable.*
(f) *Make support for safety and security demonstrable.*

— N2086 C2x Charter - Original Principles

12. **Trust the programmer, as a goal, is outdated in respect to the security and safety programming communities.** While it should not be totally disregarded as a facet of the spirit of C, the C11 version of the C Standard should take into account that programmers need the ability to check their work.

— N2086 C2x Charter - Additional Principles for C11