Why should you use Rust

Vlad Aleksashyn

November 29, 2018

Apriorit Dev Club 33

Intro

About me

- Vlad Aleksashyn
- Software Designed at Apriorit
- Mostly work with virtualization, Linux kernel and native
- Used to love C++ a lot, still love C
- Started using Rust at 2016 and regreat nothing
- Developed a real-time GPU ray tracer with Rust and Vulkan
- Now developing image compression library for RPD in Rust

Contact:

- GitHub: vaffeine
- Email: vlad.al.dp@gmail.com
- Skype: vlad.aleks

About Rust

- Started by Graydon Hoare in 2006
- Named after the rust family of fungi (?)
- First appeared July 7, 2010
- Successfully compiled itself in 2011
- First stable release 1.0 / May 15, 2015
- Was the 3rd most loved programming language in the 2015 Stack Overflow annual survey
- . . . and took first place in 2016, 2017, and 2018.
- Stable release 1.30.1 / November 8, 2018

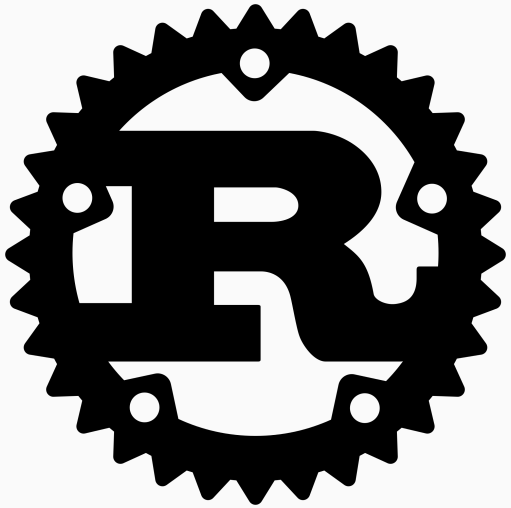


Figure 1: Rust logo

Code example #1

```rust
extern crate rand; // external library

use rand::Rng;

fn main() {
    let mut rng = rand::thread_rng();

    let numbers: Vec<i64> = (0..100)
        .map(|_| rng.gen_range(1, 42))
        .collect();

    println!("{:?}", numbers);
}
```

Code example #2

```
enum Event {
    Load,
    KeyPress(char),
    Click { x: i64, y: i64 }
}

fn process_event(event: Event) {
    match event {
        Event::Load => ...,
        Event::KeyPress(c) => ...,
        Event::Click {x, y} => ...,
    }
}
```

Companies that use Rust

And many many more. . .

Visit https://www.rust-lang.org/en-US/friends.html

TRust

Linux CVEs in 2018

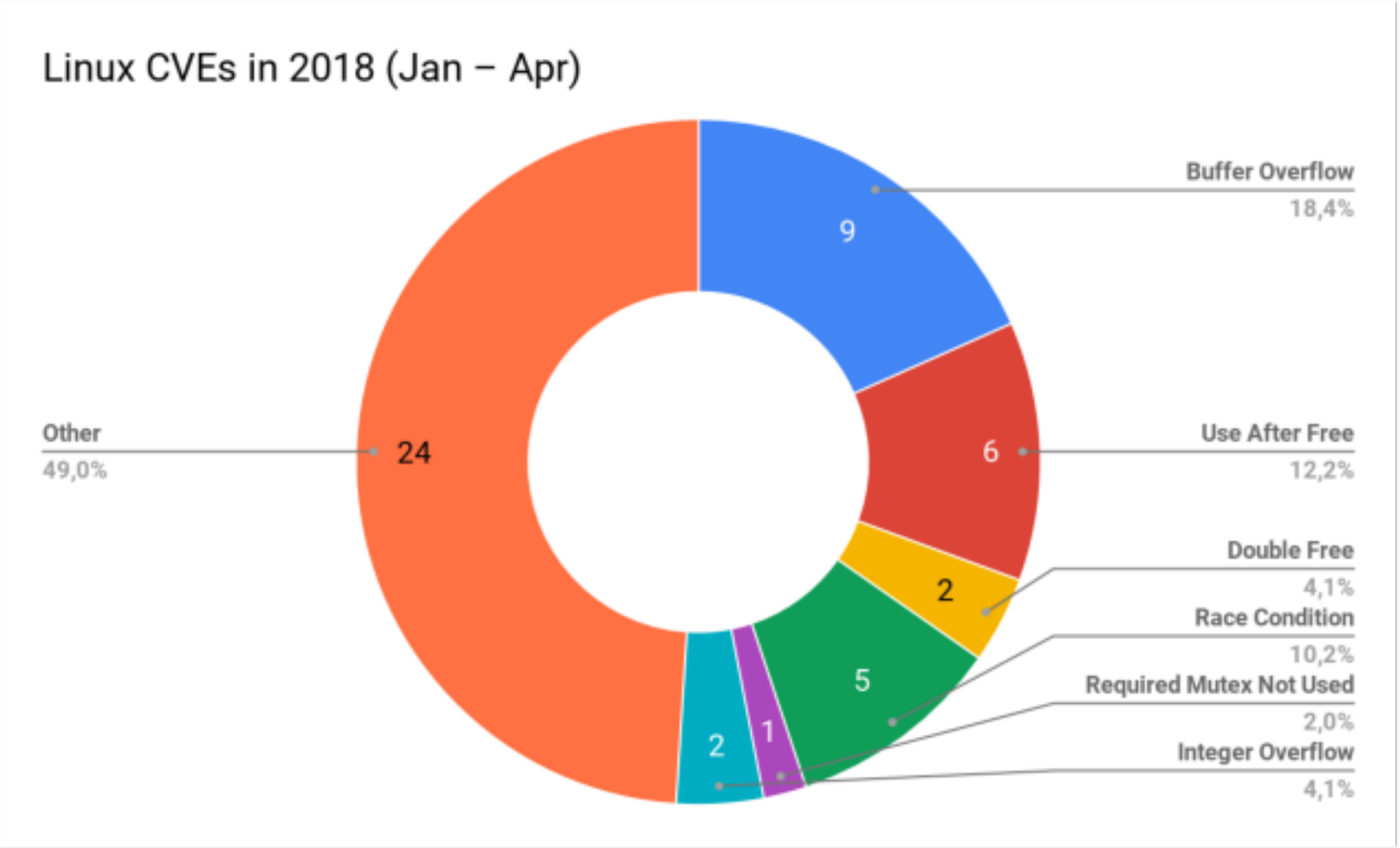


Figure 2: https://phil-opp.github.io/talk-konstanz-may-2018

Memory Safety

- No buffer overflows*
- No dangling pointers
- No data races

Guaranteed by Rust's ownership system
<u>at compile time!</u>

* unless you explicitly want them

Fearless concurency

C++

```cpp
std::vector data = {1, 2, 3};
// mutex is unrelated to data
std::mutex mutex;
...
// unsynchronized access
data.push_back(4);

// can forget to use lock guard
std::lock_guard lock(mutex);
data.push_back(5);
```

Rust

```rust
let data = vec![1, 2, 3];
// data is moved inside mutex
let mutex = Mutex::new(data);

// compilation error
data.push(4);

// can't access data w/o lock
let mut d = mutex.lock()?;
d.push(5);
```

Rust ensures that Mutex is locked before accessing data

Encapsulating Unsafety

Not everything can be verified at compile time.

For this cases Rust has unsafe blocks that allow to

- Dereference raw pointers
- Call unsafe functions
- Access mutable statics
- Implement unsafe traits

but only in this block.

Convenience

Clean generics

C++

```
class MyType {};
unordered_set<MyType>;
```

191 lines of compile errors with only 2 of them being usefull. But there will be contracts in ~~C++17~~ ~~C++20~~ C++24.

Rust

```
struct MyType;
HashSet::<MyType>::new();
```

the following trait bounds were not satisfied:
‘foo::MyType : std::cmp::Eq‘
‘foo::MyType : std::hash::Hash‘

Cool iterators and closures

Take the values produced by an instance of Counter, pair them with values produced by another Counter instance after skipping the first value, multiply each pair together, keep only those results that are divisible by 3, and add all the resulting values together:

```
let sum: u32 = Counter::new()
     .zip(Counter::new().skip(1))
     .map(|(a, b)| a * b)
     .filter(|x| x % 3 == 0)
     .sum();

assert_eq!(18, sum);
```

Effective error handling

```rust
use std::io::{self, Read};
use std::fs::File;

fn read_username_from_file()
                -> io::Result<String> {
    let mut s = String::new();

    File::open("hello.txt")?
        .read_to_string(&mut s)?;

    Ok(s)
}
```

No exceptions are needed!

Modules system

Project structure:

- src
  - client.rs
  - lib.rs
  - network
    - mod.rs
    - server.rs

```rust
// lib.rs
mod client;
mod network;

pub use network::Server;
pub use client::Client;
```

Tools

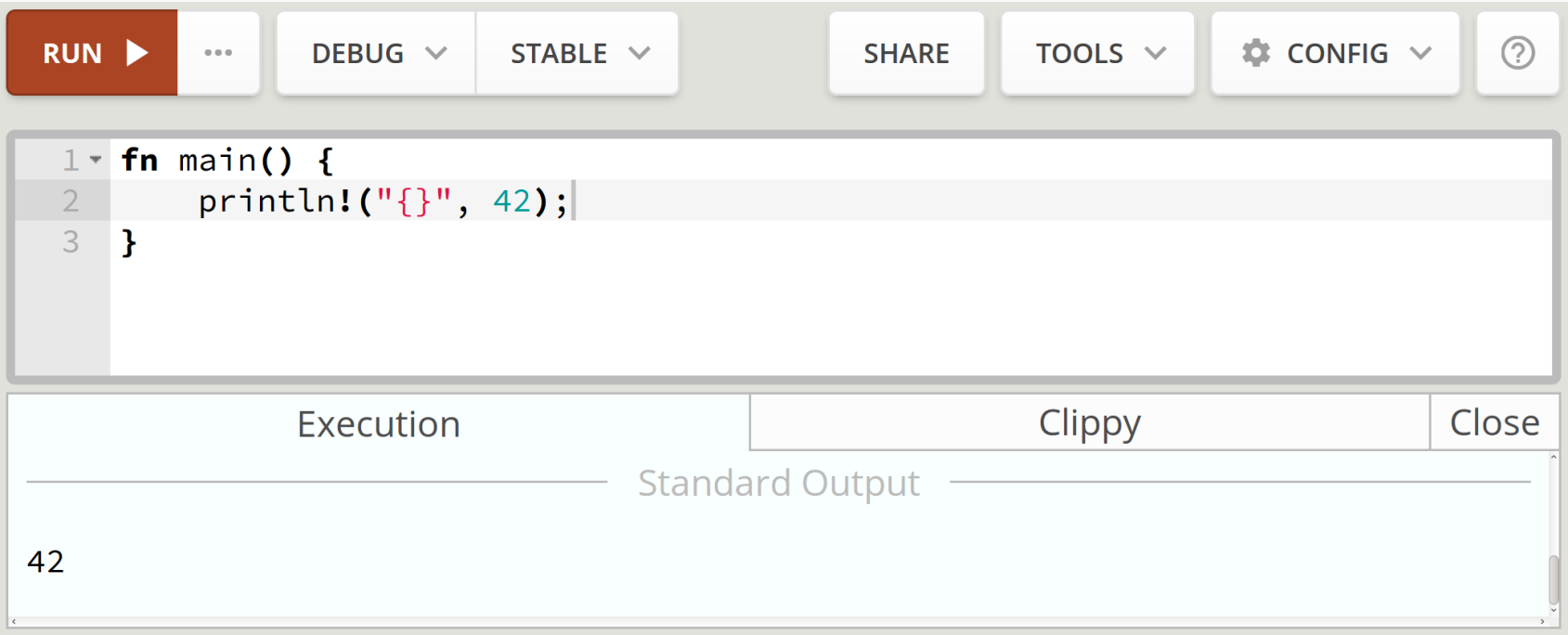
Cargo

- Over 15000 crates on crates.io
- Simply specify the desired version
  - Add single line to Cargo.toml
- Cargo takes care of the rest
  - Downloading, building, linking

Figure 3: Cargo logo

Tools

- rustup: Use multiple Rust versions for different directories
- rustfmt: Format Rust code according to style guidelines
- clippy: Additional warnings for dangerous or unidiomatic code
- Rust Playground: Run and share code snippets in your browser



Testing

Built-in testing framework:

```
fn add_two(a: i32) -> i32 {
        a + 2
}

#[test]
fn test_add_two() {
        assert_eq!(4, add_two(2));
}

#[bench]
fn bench_add_two(b: &mut Bencher) {
        b.iter(|| add_two(2));
}
```

What can I use Rust for?

Native applications

Tier 1:

- x86 Linux 2.6.18+
- x86 OSX 10.7+
- x86 Windows 7+

Tier 2:

- ARM64
- ARMv5-ARM7
- MIPS and MIPS64
- PowerPC
- on Linux, FreeBSD, NetBSD, Android and iOS

Kernel, bootloader, firmware, etc.

- You can compile Rust without standard library
- You will still have plenty of features from Core library
- Take a look at tutorials by Philipp Oppermann and Redox OS

Networking

Cool low level async networking with tokio.rs:

```
let listener = TcpListener::bind(&addr)?;

let server = listener.incoming()
        .map_err(|e| eprintln!("error"))
        .for_each(|sock| {
                let (reader, writer) = sock.split();
                let bytes_copied = copy(reader, writer);
                let handle_conn = bytes_copied.map(|amt| {
                        println!("wrote {:?} bytes", amt)
                }).map_err(|err| {
                        eprintln!("IO error {:?}", err)
                });
                tokio::spawn(handle_conn)
        });
tokio::run(server);
```

Web applications

Safety aimed web framework Rocket:

```
#[get("/<name>/<age>")]
fn hello(name: String, age: u8) -> String {
        format!("Hello, {} year old named {}!",
                                age, name)
}

fn main() {
        rocket::ignite()
                .mount("/hello", routes![hello])
                .launch();
}
```

Web client code

There are only 3 languages, that could be compiled to WebAssembly and run in client's web browser:

- C
- C++
- Rust

QA