Lecture 0: Introduction to the course

Barinov Denis

February 7, 2024

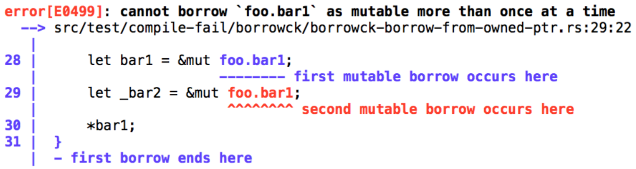
barinov.diu@gmail.com

Gonna play games?

Gonna play games?

EXPECTATION

REALITY



```
error[E0499]: cannot borrow `foo.bar1` as mutable more than once at a time
  --> src/test/compile-fail/borrowck/borrowck-borrow-from-owned-ptr.rs:29:22
   |
28 |     let bar1 = &mut foo.bar1;
   |                     -------- first mutable borrow occurs here
29 |     let _bar2 = &mut foo.bar1;
   |                      ^^^^^^^^ second mutable borrow occurs here
30 |     *bar1;
31 | }
   | - first borrow ends here
```

What is this course about?

The Rust programming language, its basic and advanced features.

- Basics: syntax, collections, traits...
- Some nightly features, such as trait specialization.
- Parallel and concurrent computing.
- Metaprogramming.
- Tooling around language.
- System safety.

Prerequisites

Nothing special:

- Knowledge of C++
- Basic understanding of parallel computing and concurrency
- Passion for coding

Useful links

- Lectures of the previous year
- Official Rustbook
- Rust reference
- Jon Gjengset
- Online IDE
- ...

How the course will change you

BEFORE

How the course will change you

BEFORE

AFTER

A brief history of Rust[1]

1. **(2006-2010)** Started at Mozilla by Graydon Hoare as a personal project.
2. **(2010-2012)** Rust is now a Mozilla project.
   - The team slowly grows allowing Rust to grow faster.
   - The aim is to make language that can catch critical mistakes before code even compiles.
3. **(2012-2014)** Rust improves type system.
   - To make the language safe, the team thought they need a garbage collector, but they figured out they don't need it: everything can be done in the level of type system!
   - Birth of Cargo - the Rust package manager. Influenced by `ruby` and `npm`.
4. **(2014-present)** Rust grows!

[1] The History of Rust talk.

Who uses Rust?

Google

- Pushing Rust to Linux Kernel
- Developing new OS Fuchsia in Rust[2]
- Enabled support of Rust in Android

Meta

- Mononoke - version control system
- Diem - blockchain
- Metaverse - virtual reality

[2] Count of lines of code in different languages.

Who uses Rust?

Amazon[3]

- Hired core developers of Tokio (the most popular framework for async)
- Firecracker - open source virtualization technology
- Bottlerocket OS - open-source Linux-based operating system meant for hosting containers
- Nitro - compute environments; underlying platform for Amazon EC2

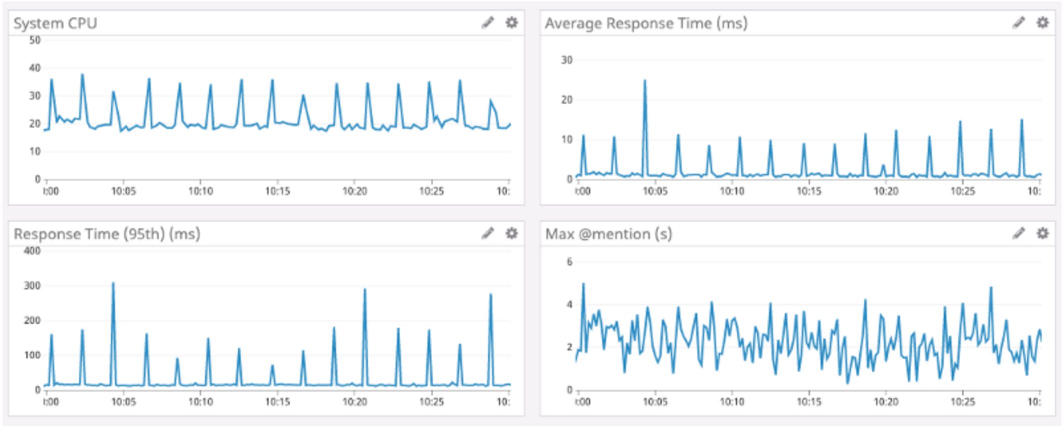
Microsoft

- Rewrited Windows component in Rust
- Official Rust WinAPI wrapper

[3] How our AWS Rust team will contribute to Rust's future successes

Who uses Rust?

Why Discord is swithing from Go to Rust?



Why companies sometimes do not use Rust?

- Already wrote a lot of code in another language
- Company's internal tools do not support Rust, and maintenance could be costly
- In a big company, you should have your committee to help support language in the company.
- Hard to find developers in such a difficult and fresh language
- Rust developers are usually talented people with high salary expectations

But why are we learning Rust?

CVE-2008-0166

Bug in `glibc` that resulted in vulnerability in OpenSSL.

- `srandom()` - set seed for non-cryptographic pseudorandom number generator.
- If read from `/dev/random` failed, the following code is executed:

```c
struct timeval tv;
unsigned long junk;
gettimeofday(&tv, NULL);
srandom((getpid() << 16) ^ tv.tv_sec ^ tv.tv_usec ^ junk);
```

UB example

CVE-2008-0166

Bug in `glibc` that resulted in vulnerability in OpenSSL.

- `srandom()` - set seed for non-cryptographic pseudorandom number generator.
- If read from `/dev/random` failed, the following code is executed:

```c
struct timeval tv;
unsigned long junk;
gettimeofday(&tv, NULL);
srandom(junk);
```

One day, the compiler decided to remove everything except junk :D

But why are we learning Rust?

```
Vec* vec_new() {
    Vec vec;
    vec.data = NULL;
    vec.length = 0;
    vec.capacity = 0;
    return &vec;
}
```

Dangling pointer

```
Vec* vec_new() {
    Vec vec;
    vec.data = NULL;
    vec.length = 0;
    vec.capacity = 0;
    return &vec; // returning a reference to a local var
}
```

But why are we learning Rust?

```
void main() {
    Vec *vec = vec_new();

    /* ... */

    free(vec->data);
    vec_free(vec);
}
```

Double free

```
void main() {
    Vec *vec = vec_new();

    /* ... */

    free(vec->data);
    vec_free(vec); // double free
}
```

But why are we learning Rust?

```c
void main() {
    /* ... */

    int *n = &vec->data[0];
    vec_push(vec, 110);
    printf("%d\n", *n);

    /* ... */
}
```

Iterator Invalidation

```
void main() {
    /* ... */

    int *n = &vec->data[0];
    vec_push(vec, 110); // may be reallocation
    printf("%d\n", *n);

    /* ... */
}
```

C++ problems

What is common?

- UB
- Double free
- Dangling pointer
- Iterator invalidation

Is Rust safe?

Rust is theoretically proven to be safe.

Understanding and Evolving the Rust Programming Language, Ralf Jung, August 2020.

Awards:

- 2021 Otto Hahn Medal
- Honorable Mention for the 2020 ACM Doctoral Dissertation Award
- 2021 ETAPS Doctoral Dissertation Award

Is Rust safe?

RustBelt - formal model of Rust that includes core conceptions of language (borrowing, lifetimes, lifetime inclusion).

- Proof of safety of Safe[4] Rust
- Definition of sufficiency conditions for every type `T` to consider it safe abstraction
- Proof of soundness (no UB or Memory unsafety): `Cell`, `RefCell`, `thread::spawn`, `Mutex`, `RwLock`, `Arc`, ...

[4] There exists Unsafe Rust, but we will return to it later.

Rust pros summary

Rust language

Pros:

- Don't require any runtime
- Provides a thick layer of abstraction, allowing to write complex readable code: structures, generics, traits, closures, iterators...
- **No memory unsafety and undefined behavior**[5]
- Modern standard without any[6] incorrect decisions

According to Microsoft and Chromium, 70% of bugs involve memory unsafety.

[5] Unless you or your dependencies use `unsafe` incorrectly

[6] There exist not good decisions.

Organization

Organization

- The complexity of the course
- Lecture format
- Homework
- Projects