Learning Rust the Hands-On Way

Performant Software Systems with Rust — Lecture 2

Baochun Li, Professor

Department of Electrical and Computer Engineering

University of Toronto

- Rust is one of the most exciting languages in the recent years
- Yet, it is known to be a complex language with a steep learning curve

But I've got some good news for you.

It **can** be simple if you learn it the **hands-on** way.

Advanced, yet friendly, compiler

Unhelpful, cryptic errors abound

```
JSONDecodingError On Line 1
NullPointerException
IndexError In File <in>
```

Read the errors carefully

and the compiler will teach you itself

Let's Write Some ~~Rust~~ Javascript

```
1 function hello(name) {
2     return "hello " + name
3 }
```

```
(base) bli@sim ~ $ rustc main.rs
error: expected one of `!` or `::`, found `hello`
 --> main.rs:1:10
  |
1 | function hello(name) {
  | -------- ^^^^^ expected one of `!` or `::`
  | |
  | help: write `fn` instead of `function` to declare a function

error: aborting due to previous error
```

```
1 def hello():
2     return "Hello World"
3
4 print(hello())
```

```
(base) bli@sim ~ $ rustc main.rs
error: expected one of `!` or `::`, found `hello`
 --> main.rs:1:5
  |
1 | def hello():
  |     ^^^^^ expected one of `!` or `::`
  |     |
  |     help: write `fn` instead of `def` to declare a function

error: aborting due to previous error
```

```
1 fn hello(name) {
2     return "hello " + name
3 }
```

```
(base) bli@sim ~ $ rustc main.rs
error: expected one of `:`, `@`, or `|`, found `)`
 --> main.rs:1:14
  |
1 | fn hello(name) {
  |              ^ expected one of `:`, `@`, or `|`
  |
  = note: anonymous parameters are removed in the 2018 edition (see RFC 1685)
help: if this is a `self` type, give it a parameter name
  |
1 | fn hello(self: name) {
  |          +++++
help: if this is a parameter name, give it a type
  |
1 | fn hello(name: TypeName) {
  |              ++++++++++
help: if this is a type, explicitly ignore the parameter name
  |
1 | fn hello(_: name) {
  |          ++

error[E0601]: `main` function not found in crate `main`
 --> main.rs:3:2
  |
3 | }
  |  ^ consider adding a `main` function to `main.rs`

error[E0308]: mismatched types
 --> main.rs:2:12
  |
1 | fn hello(name) {
  |               - help: try adding a return type: `-> str`
2 |     return "hello " + name
  |            ^^^^^^^^^^^^^^^ expected `()`, found `str`

error: aborting due to 3 previous errors

Some errors have detailed explanations: E0308, E0601.
For more information about an error, try `rustc --explain E0308`.
```

```
error: expected one of `:`, `@`, or `|`, found `)`
  --> main.rs:1:14
1 | fn hello(name) {
  |              ^ expected one of `:`, `@`, or `|`
  |
help: if this is a `self` type, give it a parameter name
  |
1 | fn hello(self: name) {
  |          +++++
help: if this is a parameter name, give it a type
  |
1 | fn hello(name: TypeName) {
  |              ++++++++++
help: if this is a type, explicitly ignore the parameter
  |
1 | fn hello(_: name) {
  |          ++
```

```
1 help: if this is a parameter name, give it a type
2   |
3 1 | fn hello(name: TypeName) {
4   |              ++++++++++
```

```
1 fn hello(name: string) {
2     return "hello " + name
3 }
```

```
error[E0412]: cannot find type `string` in this scope
  --> main.rs:1:16
   |
1  | fn hello(name: string) {
   |                ^^^^^^ help: a struct with a similar
   |     name exists (notice the capitalization): `String`
```

```
1 fn hello(name: String) {
2     return "hello " + name
3 }
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
error[E0369]: cannot add `String` to `&str`
  --> main.rs:2:21
   |
2  |     return "hello " + name
   |            -------- ^ ---- String
   |            |        |
   |            |        `+` cannot be used to concatenate
   |            |        a `&str` with a `String`
   |            &str
   |
help: create an owned `String` on the left
      and add a borrow on the right
   |
2  |     return "hello ".to_owned() + &name
   |                    ++++++++++   +
```

```
1 fn hello(name: String) {
2     return "hello ".to_owned() + &name
3 }
```

```
error[E0308]: mismatched types
 --> main.rs:2:12
  |
1 | fn hello(name: String) {
  |              - help: try adding a return type: `-> String`
2 |     return "hello ".to_owned() + &name
  |            ^^^^^^^^^^^^^^^^^^^^^^^^^^ expected `()`,
  |             found `String`
For more information about an error, try `rustc
--explain E0308`.
```

Let's try running `rustc --explain E0308`

Expected Type Did Not Match the Received Type

```
 1 fn plus_one(x: i32) -> i32 {
 2     x + 1
 3 }
 4
 5 plus_one("Not a number");
 6 //         ^^^^^^^^^^^^^^ expected `i32`, found `&str`
 7
 8 if "Not a bool" {
 9 // ^^^^^^^^^^^^ expected `bool`, found `&str`
10 }
11
12 let x: f32 = "Not a float";
13 //     ---    ^^^^^^^^^^^^ expected `f32`, found `&str`
14 //     |
15 //     expected due to this
```

```
1 fn hello(name: String) → String {
2     return "hello ".to_owned() + &name
3 }
```

```
1 $ cargo build
2 Finished dev [unoptimized + debuginfo] target(s) in 0.01s
```

The Compiler Teaches You

https://rustup.rs

```
curl --proto '=https' --tlsv1.2 -sSf
https://sh.rustup.rs | sh
```

Beautiful one-liner install

- except if you are running Windows

Updating Rust

```
rustup update
```

Local Offline Docs

```
rustup doc
```

Or directly open the Rust Book

```
rustup doc --book
```

Hello World

```
1 fn main() {
2     println!("Hello, world!");
3 }
```

Compiling Your Code

```
rustc main.rs
```

Creating a Project with Cargo

```
1 $ cargo new hello
2 $ cd hello
```

Three items have been created for you

- `Cargo.toml`
- `src/main.rs`
- local git repo

Cargo.toml

```toml
[package]
name = "hello"
version = "0.1.0"
edition = "2021"

[dependencies]
```

Building Your Project

```
1 $ cargo clean           # remove all previous builds
2 $ cargo build           # build the project incrementally
3 $ cargo build --release # build release binaries
```

Running Your Project

```
$ cargo run # run your code in debug mode
```

Checking Your Project

```
1 $ cargo check # check for errors
```

Cargo: Swiss Army Knife

```
cargo doc                   # local package documentation
cargo bench                 # built-in benchmarking
cargo test                  # built-in parallel testing
cargo add aws-sdk           # easily add dependencies
cargo install               # install exes into .cargo/bin
cargo clippy                # run the code linter
cargo publish               # publish packages to crates.io
```

Modern Tooling

- **cargo** - packaging, building
- **cargo fmt** - standard formatting
- **cargo test** - doc and unit tests
- **cargo bench** - benchmarking
- **cargo clippy** - code linting
- **rustup** - rust version switching

https://github.com/rust-lang/rustlings

Additional Resources

- Everything related to Rust on fasterthanli.me
- Rust By Example https://doc.rust-lang.org/rust-by-example/
  - follows the same chapter ordering to the Rust Book

Rust

Your code can be **perfect**

Our First Rust Project

The Guessing Game

```rust
use std::io;

fn main() {
    println!("Guess the number!");
    println!("Please input your guess.");

    let mut guess = String::new();

    io::stdin()
        .read_line(&mut guess)
        .expect("Failed to read line");

    println!("You guessed: {}", guess);
}
```

```rust
use std::io;

fn main() {
    println!("Guess the number!");
    println!("Please input your guess.");

    let mut guess = String::new();

    io::stdin()
        .read_line(&mut guess)
        .expect("Failed to read line");

    println!("You guessed: {}", guess);
}
```

Demo

Generating a Secret Number

Cargo.toml

```
1 [dependencies]
2 rand = "0.8"
```

Updating Crates

```
$ cargo update
    Updating crates.io index
    Updating rand v0.8.5 → v0.9.2
```

Generating a Random Number

```
 1 use std::io;
 2 use rand::Rng;
 3
 4 fn main() {
 5     println!("Guess the number!");
 6
 7     let secret_number =
 8         rand::thread_rng().gen_range(1..=100);
 9
10     println!("The secret number is: {secret_number}");
```

Consulting local docs

```
cargo doc --open
```

Comparing the Guess to the Secret Number

```
 1 use std::cmp::Ordering;
 2
 3 fn main() {
 4     // --snip--
 5
 6     match guess.cmp(&secret_number) {
 7         Ordering::Less => println!("Too small!"),
 8         Ordering::Greater => println!("Too big!"),
 9         Ordering::Equal => println!("You win!"),
10     }
11 }
```

Demo

Resolving the Compile-Time Error

```
let mut guess = String::new();

io::stdin()
    .read_line(&mut guess)
    .expect("Failed to read line");

let guess: u32 = guess.trim().parse().expect("Please type
a number!");
```

Adding a Loop

```
1       // --snip--
2       println!("The secret number is: {secret_number}");
3
4       loop {
5           println!("Please input your guess.");
6
7           // --snip--
8       }
```

Quitting after a Correct Guess

```
        // --snip--
        match guess.cmp(&secret_number) {
            Ordering::Less => println!("Too small!"),
            Ordering::Greater => println!("Too big!"),
            Ordering::Equal => {
                println!("You win!");
                break;
            }
        }
```

Handling Invalid Input

```
        // --snip--
        let guess: u32 = match guess.trim().parse() {
            Ok(num) => num,
            Err(_) => continue,
        };

        println!("You guessed: {guess}");
```

You have just built your first Rust project!

Required Additional Reading

The Rust Programming Language, Chapter 1-2