

Bases et exemples

2024-06-06

Guilhem Saurel

Available at

[https://homepages.laas.fr/gsaurel/
rust-intro.pdf](https://homepages.laas.fr/gsaurel/rust-intro.pdf)

Under License



<https://creativecommons.org/licenses/by-sa/4.0/>

Source

`https://gitlab.laas.fr/gsaurel/talks :`
`rust-intro.md`

Discussions & Companion project

`https://github.com/nim65s/RobotS`

Introduction

- 1 introduction
- 2 nice features in a language
- 3 nice features in the tooling
- 4 scope
- 5 limitations

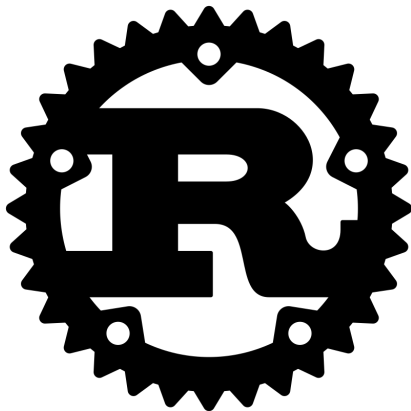


Figure 1: Rust

Base Tooling

Run the following in your terminal, then follow the onscreen instructions.

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



You appear to be running Unix. If not, [display all supported installers](#).

Figure 2: `https://rustup.rs/`

```
$ mkdir first
$ cd first
$ cargo init
  Creating binary (application) package
$ cargo run
  Compiling first v0.1.0 (/root/first)
  Finished `dev` profile [unoptimized + debuginfo]
  Running `target/debug/first`
Hello, world!
```

Language

```
$ cat src/main.rs
```

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

```
let x = 5;  
let mut y = 6;
```

```
let x = 5;  
let mut y = 6;  
  
x += 1; // Error  
y += 1; // Ok
```

Compiling first v0.1.0 (/tmp/first)

```
error[E0384]: cannot assign twice to immutable variable
```

```
--> src/main.rs:27:5
```

```
24 |         let x = 1;  
    |         -  
    |         |  
    |         first assignment to `x`  
    |         help: consider making this binding mutable
```

```
...  
27 |         x += 1;  
    |         ^^^^^^ cannot assign twice to immutable variable
```

For more information about this error, **try** `rustc --explain E0384`
error: could not compile `first` (bin "first") due to this error

- Signed integers: `i8`, `i16`, `i32`, `i64`, `i128` and `isize`
- Unsigned integers: `u8`, `u16`, `u32`, `u64`, `u128` and `usize`
- Floating point: `f32`, `f64`
- `char`: Unicode scalar values like `'a'`, `'α'` and `'∞'`
- `bool`: either `true` or `false`
- unit type: `()`

- Arrays: [1, 2, 3]
- Tuples: (1, true)

```
std::string::String  
std::vec::Vec<T>  
std::boxed::Box<T>  
std::rc::Rc<T>  
std::sync::Arc<T>
```

```
std::string::String  
std::vec::Vec<T, A = Global>  
std::boxed::Box<T, A = Global>  
std::rc::Rc<T, A = Global>  
std::sync::Arc<T, A = Global>
```

```
struct Pose {  
    x: f64,  
    y: f64,  
}
```

```
enum Mode {  
    Idle,  
    Move(Pose, f64),  
}
```

```
struct Robot {  
    name: String,  
    pose: Pose,  
    mode: Mode,  
}
```

```
struct Transform<Scalar, Angle> {  
    center: (Scalar, Scalar),  
    angle: Angle,  
    scale: Scalar,  
}
```

```
enum Result<T, E> {  
    Ok(T),  
    Err(E),  
}
```

```
enum Result<T, E> {  
    Ok(T),  
    Err(E),  
}  
  
type MyRet = Result<f64, f64>;  
let x = MyRet::Ok(4.25);  
let y = MyRet::Err(f64::NAN);  
  
x.is_ok();           // true  
x.is_err();          // false  
x.map(|n| n * 2.0);  // Ok(8.5)  
y.is_ok();           // false  
y.is_err();          // true  
y.map(|n| n * 2.0);  // Err(NAN)
```

```
let file = File::create("data.txt");  
match file {  
  Err(e) => eprintln!("cant create file: {e:?}"),  
  Ok(mut f) => {  
    let ret = f.write_all(b"hello");  
    match ret {  
      Err(e) => eprintln!("cant create file: {e:?}"),  
      Ok(()) => println!("file written."),  
    }  
  }  
}
```

```
fn main() -> std::io::Result<()> {  
    let mut file = File::create("data.txt");  
    file.write_all(b"hello");  
    println!("file written.");  
    Ok(())  
}
```

```
fn main() -> std::io::Result<()> {  
    let mut file = File::create("data.txt");  
    file.write_all(b"hello");  
    println!("file written.");  
    Ok(())  
}  
  
type std::io::Result<T> = Result<T, std::io::Error>;
```

```
enum Option<T> {  
    None,  
    Some(T),  
}
```

```
enum Option<T> {  
    None,  
    Some(T),  
}
```

```
type MyOpt = Option<f64>;  
let x = MyOpt::Some(4.25);  
let y = MyOpt::None;
```

```
x.is_some();           // true  
x.is_none();           // false  
x.map(|n| n * 2.0);    // Some(8.5)  
y.is_some();           // false  
y.is_none();           // true  
y.map(|n| n * 2.0);    // None
```

```
trait SetSpeed {  
    fn set_speed(&mut self, tgt: f64) -> Result;  
}
```

```
trait SetSpeed {  
    fn set_speed(&mut self, tgt: f64) -> Result;  
}  
  
impl SetSpeed for Robot {  
    fn set_speed(&mut self, tgt: f64) -> Result {  
        match self.mode {  
            Mode::Idle => Err(Error::IsIdle),  
            Mode::Move(pose, _) => {  
                self.mode = Mode::Move(pose, tgt);  
                Ok(())  
            }  
        }  
    }  
}
```

```
#[tokio::main]
async fn main() -> Result {
    let port = 1234;
    tokio::spawn(async move {
        serve(port).await;
    });

    let mut client = Client::connect(port).await?;
    client.set("hello", "world").await?;

    let result = client.get("hello").await?;
    println!("got {result:?}");

    Ok(())
}
```

```
#[tokio::main]
async fn main() {
    let (tx, mut rx) = mpsc::channel(32);

    tokio::spawn(async move {
        tx.send("sending from first handle").await;
    });

    while let Some(message) = rx.recv().await {
        println!("GOT = {}", message);
    }
}
```

Rust Embedded

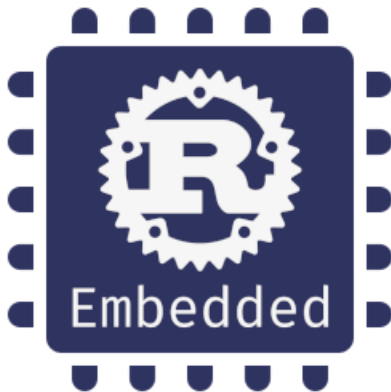


Figure 3: Rust Embedded

- PAC: Peripheral Access Crates
- HAL: Hardware Abstraction Layers
- embedded-hal

- PAC: Peripheral Access Crates
- HAL: Hardware Abstraction Layers
- embedded-hal

ref. https://www.youtube.com/watch?v=vLYit_HHPaY

- RTIC
- Embassy

Web



Leptos

Figure 4: Fast full-stack web apps in Rust

Tooling

```
$ rustup component add clippy  
$ cargo clippy  
$ cargo clippy --fix
```

```
$ cargo install probe-rs-tools  
$ cargo embed
```

```
$ cargo install cargo-leptos  
$ cargo leptos watch
```

Démo !

<https://github.com/nim65s/RobotS>

Applications

- grep → ripgrep
- find → fd
- ls → LSDeluxe / eza
- du → dust
- cat → bat
- ctrl-r → zoxide
- terminal → alacritty
- prompt → starship
- history → atuin
- make → just
- watch → watchexec / bacon
- screen / tmux → zellij
- diff → delta
- bash → nushell

Python

- ruff
- uv

JavaScript

- deno
- parcel

Alternatives

- Go
- Swift

- Zig

Limitations

Not that easy

Recompile all the things

You might need nix ;)

Thanks for your attention :)

References

- <https://doc.rust-lang.org/rust-by-example/>
- <https://doc.rust-lang.org/book/>