# RUST GRAZ – 01 GETTING STARTED

Lukas Prokop

July 25, 2019

# SETUP

Either play.rust-lang.org or [macOS/Linux/UNIX]

```
curl https://sh.rustup.rs -sSf | sh
```

*rustup:* Rust installer and version management tool

Use `rustup self uninstall` to uninstall

1. Determines host triple (arch&vendor&os): `x86_64-unknown-linux-gnu`
2. Installs rust tools to `/root/.cargo/bin`
3. Adds `/root/.cargo/bin` to PATH
4. Sets toolchain to `stable`

# RESOURCES

- doc.rust-lang.org
- doc.rust-lang.org/book/ or `rustup doc --book`
- doc.rust-lang.org/stable/rust-by-example/
- exercism.io/tracks/rust
- newrustacean.com (stopped in May 2019)
- rusty-spike.blubrry.net
- “Help Wanted: Research Questions in Rust - Aaron Turon - OPLSS 2018”

IMHO: huge bunch of community resources!

# TOOLING

In general: many features provided by the language (e.g. unlike C++ and Doxygen).

- **cargo** is rust's package (= crate) manager
- **rls** is rust's language server
- **rustfmt** formats/normalizes rust programs
- **rustdoc** generates documentation from comments
- **rust-gdb/rust-lldb** for debugging

# TOOLING

LLVM-based compiler and toolchain.
IMHO use an IDE for programming:

- IntelliJ IDEA with Rust plugin
- CLion with same plugin
- Visual Studio Code with `rust-lang.rust` extension
- …

# RELEASES

```
rustup install {stable,beta,nightly}
```

- **Stable** (1.36.0, default)
- **Beta** (1.37.0-beta.6, tests for stable release)
- **Nightly** (1.38.0-nightly, unstandardized features)

Rapid 6-week release schedule

set default release with:

```
rustup default {stable,beta,nightly}
```

# EDITIONS

| | | |
|---|---|---|
| **January 2012** | ⇒ | first numbered pre-alpha release |
| **May 2015** | ⇒ | 1.0.0 release (stable std API), “Rust 2015” |
| **Dec 2018** | ⇒ | 1.31 release, “Rust 2018” |

Editions (“long-term releases with a theme”):

- Rust 2015 ‘stability’
- Rust 2018 ‘productivity’

# SYNTACTIC NOTES FOR TODAY

# HELLO WORLD

```rust
fn main() {
    println!("Hello World!");
}
```

# HELLO WORLD

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

- `fn` for *function*
- C-like block structures, 4 spaces for indentation
- main function w/o return value
- `!` because `println` is a macro, not a function
- strings with "
- C-like semicolon after statements

# FORMATTING

```
fn main() {
    println!("Hello {:06b} {}!", 9, "rustaceans");
}
```

- featuring string formatting
- inspired by C's `printf` and Python's `str.format`
- `06` for left-padding with `0` character
- `b` for binary representation

```
fn main() {
    println!("Hello {no:06b} {who}!",
             no = 9, who = "rustaceans");
}
```

# ASSIGNMENT

```
fn main() {
    let attending_rustaceans = 9;
    println!("Hello {} rustaceans!", attending_rustaceans);
}
```

- `let` keyword for assignment
- underlines in variables by convention

# ASSIGNMENT

```
fn main() {
    let attending_rustaceans = 9;
    attending_rustaceans += 1;
    println!("Hello {} rustaceans!", attending_rustaceans);
}
```

# WAIT... WHAT?

```
error[E0384]: cannot assign twice to immutable variable `attending_
 --> test2.rs:3:5
  |
2 |     let attending_rustaceans = 9;
  |         -------------------- first assignment to `attending_rus
3 |     attending_rustaceans += 1;
  |     ^^^^^^^^^^^^^^^^^^^^^^^^^ cannot assign twice to immutable

error: aborting due to previous error

For more information about this error, try `rustc --explain E0384`.
```

# IMMUTABILITY BY DEFAULT!

# ASSIGNMENT

```
fn main() {
    let mut attending_rustaceans = 9;
    attending_rustaceans += 1;
    println!("Hello {} rustaceans!", attending_rustaceans);
}
```

- `mut` keyword for *mutability*

# HOW TO COMPILE?

```
user@sys / % cargo new hello_world --bin
     Created binary (application) `hello_world` package
user@sys / % cd hello_world/
user@sys /hello_world % ls -la
total 8
drwxrwxr-x 1 user user   54 Jul 25 12:03 .
drwxrwxrwt 1 root       root       1356 Jul 25 12:03 ..
drwxrwxr-x 1 user user   82 Jul 25 12:03 .git
-rw-rw-r-- 1 user user   19 Jul 25 12:03 .gitignore
-rw-rw-r-- 1 user user  131 Jul 25 12:03 Cargo.toml
drwxrwxr-x 1 user user   14 Jul 25 12:03 src
```

```
user@sys /hello_world % cat Cargo.toml
[package]
name = "hello_world"
version = "0.1.0"
authors = ["user <my@git-email.addr>"]
edition = "2018"

[dependencies]
user@sys /hello_world % cd src/
user@sys /hello_world/src % cat main.rs
fn main() {
    println!("Hello, world!");
}
user@sys /hello_world/src % cargo run
   Compiling hello_world v0.1.0 (/hello_world)
    Finished dev [unoptimized + debuginfo] target(s) in 0.42s
     Running `/hello_world/target/debug/hello_world`
Hello, world!
```