MKProjects 2021

THE GIST OF RUST

They said perfection is impossible and yet here it is...

MUSTAFIF KHAN

PUBLISHED BY MKPROJECTS

MKPROJ.COM

Contents

Getting Started

THE GIST OF RUST

1. What is Rust?

Rust was born in the research labs of Mozilla, and was aimed to be a memory safe, zero abstractions language. Not only is it statically typed (knows the data type at compile time), but is a high-level programming langauge able to perform low-level tasks. It is being considered the replacement of C/C++ due to it's safety with memory and threads (in concurrent tasks), and this is all thanks to Rust's strict ownership rules.

Now Rust does have a steep learning curve, however if you come from languages such as C++, OCAML, you may have it easier than others that come from langauges like Python. However even though it has a steep learning curve, it is definitely worth the investment, and as a fellow Rustacean, it definitely changes your perspective of handling situations.

Here are a few things that Rust does that I just love:

- Rust has a nice package manager called **Cargo**
- Rust has documentation syntax that is used to well document your *Crate (A Rust Package)*
- Rust's compiler is very helpful and can help solve your problem at times

To really know if Rust is for you, it's good to know what it can be used for, so here's some common use cases:

- Systems Programming (Kernel Modules, CLI Applications, etc.)
- Web Servers (Rcoket, Actix, Warp)
- Desktop Applications (GTK)

In the next few sections, we will discuss core ideas to learn about Rust, but we still recommend doing some more reading on your own. The Rust Foundation provides their own documentation on various topics, and can be found at https://www.rust-lang.org/learn.

To install, visit https://rustup.rs and follow the instructions it gives to your particular system.

THE GIST OF RUST

2. What is Cargo?

We talked a little bit about Cargo already, but to review, Cargo is Rust's package manager and is your goto tool to create Rust programs. In Rust we have two different type of projects, **bin** & **lib**, which stand for binary & library respectively.

- **Binary**: A program built to be executed, the program is compiled and executed by it's binary file.
- **Library**: A program built to store various modules, functions, etc. The program is meant to store various components a binary project may use.

So why don't we create a cargo project? By default it's binary, and for our purposes that's completely fine.

```
# To create a new project use cargo new <project name>
$ cargo new hello_world
     Created binary (application) ‘hello_world‘ package
$ cd hello_world # Change directory to hello_world
$ ls
Cargo.toml src
```

A Cargo package contains the following:

- Cargo.toml (Used to specify metadata of the project)
- src (Contains all of the source code of the project)