试用 Rust 编程

Rust 是这几年比较火的编程语言。2006 年时，Mozilla 的一个员工开始做一个个人项目，后来得到了公司的支持，于 2010 年发布这个项目。这个项目就叫 Rust。

接下来，运行起来 Rust 的第一个程序吧。打开官网，来看看如何把程序跑起来。

官网提供了一个脚本：

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

在 Mac 上也可以用 Mac 系统的包管理工具 Homebrew 来安装。可运行命令：

```
brew install rust
```

我这里用 Homebrew 来安装 rust。安装的时刻，让我们继续看官网。

接着我们看官网出现了 Cargo 这个东西，Rust 的构建工具和包管理工具。

官方网站上说：

- build your project with `cargo build`
- run your project with `cargo run`
- test your project with `cargo test`

告诉我们如何构建、运行和测试 Cargo 程序。

运行：

```
brew install rust
```

输出：

```
==> Downloading https://homebrew.bintray.com/bottles/rust-1.49.0_1.big_sur.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/5a238d58c3fa775fed4e12ad74109deff54a82a06cb6
######################################################################## 100.0%
==> Pouring rust-1.49.0_1.big_sur.bottle.tar.gz
==> Caveats
Bash completion has been installed to:
  /usr/local/etc/bash_completion.d
==> Summary
   /usr/local/Cellar/rust/1.49.0_1: 15,736 files, 606.2MB
```

这便安装成功了。

当在终端运行 `cargo` 时，输出如下：

```
Rust's package manager

USAGE:
    cargo [OPTIONS] [SUBCOMMAND]

OPTIONS:
    -V, --version           Print version info and exit
        --list              List installed commands
        --explain <CODE>    Run `rustc --explain CODE`
    -v, --verbose           Use verbose output (-vv very verbose/build.rs output)
    -q, --quiet             No output printed to stdout
        --color <WHEN>      Coloring: auto, always, never
        --frozen            Require Cargo.lock and cache are up to date
        --locked            Require Cargo.lock is up to date
        --offline           Run without accessing the network
    -Z <FLAG>...            Unstable (nightly-only) flags to Cargo, see 'cargo -Z help' for details
    -h, --help              Prints help information

Some common cargo commands are (see all commands with --list):
    build, b    Compile the current package
    check, c    Analyze the current package and report errors, but don't build object files
    clean       Remove the target directory
    doc         Build this package's and its dependencies' documentation
    new         Create a new cargo package
    init        Create a new cargo package in an existing directory
    run, r      Run a binary or example of the local package
    test, t     Run the tests
    bench       Run the benchmarks
    update      Update dependencies listed in Cargo.lock
    search      Search registry for crates
    publish     Package and upload this package to the registry
    install     Install a Rust binary. Default location is $HOME/.cargo/bin
    uninstall   Uninstall a Rust binary

See 'cargo help <command>' for more information on a specific command.
```

无需弄懂所有的命令。只需要知道常用的命令即可。build 和 run 命令很重要。

继续看官网文档:

```
Let's write a small application with our new Rust development environment. To start, we'll use Cargo
```

```
cargo new hello-rust
```

```
This will generate a new directory called hello-rust with the following files:
```

```
hello-rust
|- Cargo.toml
|- src
  |- main.rs
Cargo.toml is the manifest file for Rust. It's where you keep metadata for your project, as well as de
```

```
src/main.rs is where we'll write our application code.
```

这讲述了如何创建项目。接下来便创建。

```
$ cargo new hello-rust
```

```
Created binary (application) `hello-rust` package
```

我们用 VSCode 来打开项目。

main.rs:

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

接下来，很自然想到要 build 以及 run 程序。

```
$ cargo build
```

```
error: could not find `Cargo.toml` in `/Users/lzw/ideas/curious-courses/program/run/rust` or any parent
```

出错了。为什么。这表明 cargo 只能在项目下的目录运行。接下来进入子目录，运行了 `cd hello-rust`。

这时，想如果直接运行，会怎么样。

```
$ cargo run
```

```
   Compiling hello-rust v0.1.0 (/Users/lzw/ideas/curious-courses/program/run/rust/hello-rust)
    Finished dev [unoptimized + debuginfo] target(s) in 4.43s
     Running `target/debug/hello-rust`
Hello, world!
```

好了，成功了。输出了字符串，程序开始工作啦。

试着改动程序。

```
fn main() {
    println!(2+3);
}
```

cargo run 之后，出现了：

```
   Compiling hello-rust v0.1.0 (/Users/lzw/ideas/curious-courses/program/run/rust/hello-rust)
error: format argument must be a string literal
 --> src/main.rs:2:14
  |
2 |     println!(2+3);
  |              ^^^
  |
help: you might be missing a string literal to format with
  |
2 |     println!("{}", 2+3);
  |              ^^^^^


error: aborting due to previous error


error: could not compile `hello-rust`


To learn more, run the command again with --verbose.
```

还没学习任何 Rust 语法。凭着我们的直觉来改动代码出错了。这个出错提示很好，已经告诉我们怎么改了。

```
fn main() {
    println!("{}", 2+3);
}
```

这次改对了，果然输出了 5。

对了，build 会怎么样。

```
$ cargo build
    Finished dev [unoptimized + debuginfo] target(s) in 0.00s
```

为什么要有 build 呢。因为有可能我们只是想生成可执行程序，而并不想执行。也许对于一些庞大的程序，执行是费时的。也许我们想本地生成，然后传输到远程服务器去执行。

我们已经把 Rust 程序跑起来了。后面便是熟悉更多的 Rust 语言语法，来在 Rust 中找到我们在「解谜计算机科学」上讲述的变量、函数、函数调用和表达式等概念所对应的符号表示。

小练习

- 试着像上面的一样，学生在自己电脑上试用 Rust 编程。
- 练习完后，可提交一百字以内的总结或对本文的补充。
