試用 Rust 編程

Rust 是這幾年比較熱門的程式語言。2006 年時，Mozilla 的一名員工開始做一個個人項目，後來得到了公司的支持，於 2010 年發佈了這個項目。這個項目就叫 Rust。

接下來，讓我們來運行第一個 Rust 程式吧。打開官網，來看看如何把程式跑起來。

官網提供了一個腳本：

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

在 Mac 上也可以用 Mac 系統的套件管理工具 Homebrew 來安裝。可運行命令：

```
brew install rust
```

我這裡用 Homebrew 來安裝 rust。安裝的時刻，讓我們繼續看官網。

接著我們看官網出現了 Cargo 這個東西，Rust 的構建工具和套件管理工具。

官方網站上說：

- 用 `cargo build` 來構建你的項目
- 用 `cargo run` 來運行你的項目
- 用 `cargo test` 來測試你的項目

告訴我們如何構建、運行和測試 Cargo 程式。

運行：

```
brew install rust
```

輸出：

```
==> 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
```

這便安裝成功了。

當在終端運行 cargo 時，輸出如下：

```
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.
```

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

繼續看官網文檔：

```
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.
```

這講述了如何創建項目。接下來便創建。

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

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

我們用 VSCode 來打開項目。

main.rs:

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

接下來，很自然想到要 build 以及 run 程式。

```
$ cargo build
```

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

出錯了。為什麼。這表明 cargo 只能在項目下的目錄運行。接下來進入子目錄，運行了 `cd hello-rust`。

這時，想如果直接運行，會怎麼樣。

```
$ 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。輸出了字符串，程式開始工作啦。

試著改動程式。

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

`cargo run` 之後，出現了：

```
   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.
```

還沒學習任何 Rust 語法。憑著我們的直覺來改動程式出錯了。這個出錯提示很好，已經告訴我們怎麼改了。

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

這次改對了，果然輸出了 5。

對了，build 會怎麼樣。

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

為什麼要有 build 呢。因為有可能我們只是想生成可執行程式，而並不想執行。也許對於一些龐大的程式，執行是費時的。也許我們想本地生成，然後傳輸到遠端伺服器去執行。

我們已經把 Rust 程式跑起來了。後面便是熟悉更多的 Rust 語言語法，來在 Rust 中找到我們在「解謎計算機科學」上講述的變量、函數、函數調用和表達式等概念所對應的符號表示。

小練習

- 試著像上面的一樣，學生在自己電腦上試用 Rust 編程。
- 練習完後，可提交一百字以內的總結或對本文的補充。
