rust 开发环境部署

1. 运行安装脚本

```
#打开终端，输入如下命令，进行脚本的安装，这个命令将下载一个脚本并开始安装 `rustup` 工具，此工具将安装 Rust 的最新稳定版本。可能会提示你输入管理员密码。
curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh
#预期如下输出
   default host triple: x86_64-apple-darwin
     default toolchain: stable (default)
               profile: default
  modify PATH variable: yes
1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
# 输入1或回车即可
# 安装成功，将会出现如下
Rust is installed now. Great!
```

2. 安装结果验证

```
cargo -V
# 如果出现版本号和日期，恭喜你，Rust安装成功了
cargo 1.64.0 (387270bc7 2022-09-16)
```

3. Rust安装项检查

```
# 可以确认安装了哪些toolchains、target,当前是版本是测试版本，还是正式版本
➜  ~ rustup show
Default host: x86_64-apple-darwin
rustup home:  /Users/fcihpy/.rustup

installed toolchains
--------------------
stable-x86_64-apple-darwin (default)
nightly-2022-02-23-x86_64-apple-darwin
nightly-2022-06-30-x86_64-apple-darwin
nightly-x86_64-apple-darwin
1.61-x86_64-apple-darwin
1.63-x86_64-apple-darwin
1.63.0-x86_64-apple-darwin
installed targets for active toolchain
--------------------------------------
wasm32-unknown-unknown
x86_64-apple-darwin
```

```
active toolchain
----------------
stable-x86_64-apple-darwin (default)
rustc 1.64.0 (a55dd71d5 2022-09-19)
```

4. 运行环境切换使用

```
# 将默认编译环境切换到stable
rustup default stable
# 将默认编译环境切换到nightly
rustup default nightly
# 使用rustup show确认切换是否成功
```

5. 安装必要的系统组件

```
# macOS下
$ xcode-select --install
# 其它相关组件，可以使用brew install安装
# linux下
sudo apt install -y cmake pkg-config libssl-dev git build-essential clang libclang-dev
curl libz-dev
```

6. 根据需要添加rust开发组件

```
rustup component add rust-src clippy
```

7. 安装打包为wasm的目标库

```
rustup target add wasm32-unknown-unknown
```

8. 安装rust上关工具

```
cargo install xcargo
cargo install nightly
cargo install cargo-contract
```

9. 项目目录

```
➜  project cargo new helloworld
     Created binary (application) `helloworld` package
➜  helloworld git:(master) ✗ tree
.
├── Cargo.toml    //项目配置文件,依赖库的添加
└── src
    └── main.rs   //项目入口文件
1 directory, 2 files
```

10. 常用Rust 命令

```
# 创建一个带有可执行文件的项目
cargo new <project_name>
# 创建一个带无可执行文件、只包含功能的项目
cargo new <project_name> --lib
# 运行全部测试用例
cargo test
# 运行带有handle的测试用例
cargo test handle
# 编译并运行当前项目
cargo run
# 编译当前项目为debug模式
cargo build
# 编译当前项目为release模式，会比较慢
cargo build --release
# check 依赖
cargo check
# 格式化代码
cargo format
```

11. Rust 卸载

```
# 在终端输入如下命令
$ rustup self uninstall
```