

01 Rust 的发展历程

每一门编程语言都有其特定的历史时期、以及其要解决的问题。这就决定了每门语言都有自己独特的设计哲学。在学习一门新的编程语言之前，了解其发展进程，对于学习过程中更加深刻的理解语言的特性有很大的帮助。

Rust 的发展历史

最早是在 2006 年，Rust 是当时在 Mozilla 任职的 Graydon Hoare 的个人项目。在 2009 年的时候，Mozilla 有一个研究型的浏览器项目 Servo，正在寻找一种更加安全高效的语言，所以 Mozilla 就官方赞助了 Rust 项目。

Rust 在 2012 年的时候，推出第一个内部可用版本；2015 年 5 月 15 日，第一个稳定版本 Rust 1.0 正式发布。在 1.0 发布不久之后，Rust 团队改变了版本命名策略，采用年度的方式来命名 Rust 版本。所以，Rust 2015 引入了一些重要的语言和库的改进，例如：模块系统、Error trait 以及标准库的改进。Rust 2018 是下一个重要的版本，故名思义，这个版本是在 2018 年发布的，它带来了一些重要的语言和工具链的改进，包括全新的宏系统、`async/await` 异步编程支持、更好的错误处理等。

在 2020 年，由于众所周知的原因，Mozilla 在全球范围内裁员。随着 Servo 团队的解散，不免让大家对 Rust 这门生命力旺盛的语言的未来有所担忧。2021 年 8 月份，由 AWS、华为、Google、Microsoft 以及 Mozilla 五家公司联合成立了 Rust 基金会，全面接手了 Rust 的治理工作，以及提供资金支持。大家可以看到，Rust 基金会中只有华为这一家中国公司，所以华为在前几天宣布[鸿蒙将支持使用 Rust 开发](#)，这个消息无疑也给 Rust 的竞争力提升添加了浓墨重彩的一笔。

截至目前，Rust 最新的发布版本是 2021，对工具链 Cargo、闭包匿名生命周期等进行了改进，解决了一些痛点。

目前，全球范围内有很多公司都已经或者开始使用 Rust 来开发他们的系统或者产品。例如：

- Mozilla 使用 Rust 开发 Firefox 浏览器核心组件
- Dropbox 在其后端存储系统中广泛使用 Rust

- AWS 在其云计算基础设施中采用 Rust，特别是用于开发工具和系统级组件
- Microsoft 在一些关键项目中开始采用 Rust，例如 Azure IoT Edge 和 Windows Subsystem for Linux 2 (WSL 2)
- Cloudflare 使用 Rust 开发了一些核心网络和安全工具

而最为重磅的是，从 Linux 6.1 开始，接受由 Rust 编写的内核。Linux 分为内核和发行版两个概念，其中 Linux 内核是由其创始人 Linus 领导的小组来维护，而 Linux 的发行版就非常多了，例如：CentOS、Ubuntu、Fedora、Debian、Red Hat Linux 等等。

The screenshot shows the InfoQ website's homepage with a navigation bar at the top. Below the header, there are several event cards for "InfoQ Live Roundtable", "QCon London", "InfoQ Dev Summit Boston", and "QCon San Francisco". The main content area features a news article titled "Linux 6.1 Officially Adds Support for Rust in the Kernel" by Sergio De Simone. The article discusses the history of Rust support in the kernel and its current status. To the right of the article, there is a sidebar titled "RELATED CONTENT" with links to other articles like "Visual Studio 17.9 Preview 2: .NET MAUI, C++" and "Swift 5.9 Backtracer is Improves Readability". There is also a newsletter sign-up form.

另外，如果不考虑玩游戏的因素，强烈建议大家使用 Linux 系统来进行开发，我个人比较推荐 Ubuntu 桌面版，外观很漂亮，操作丝滑，使用起来不需要太大的学习成本。或者使用苹果电脑（必须是自带的 macOS 系统哦）来做开发。虽然说，Windows 10 以来，增加了很多开发者友好的能力，但是在命令行方面还是不如 Linux 以及 macOS（基于 Unix 的）。现在常见的各种语言以及开发框架都是在 Linux 上用起来更好。包括很多 IDE 也都是跨平台的，即使 Windows 自己家的 Visual Studio Code 也是可以在 macOS 和 Linux 上使用的。所以，使用 Linux 桌面系统进行开发是完全没有问题的。

Rust 的特点

Rust 语言具有以下特点：

- **高性能。** Rust 速度惊人且内存利用率极高。由于没有运行时和垃圾回收，它能够胜任对性能要求特别高的服务，可以在嵌入式设备上运行，还能轻松和其他语言集成。
- **可靠性。** Rust 丰富的类型系统和所有权模型保证了内存安全和线程安全，让你在编译期就能够消除各种各样的错误。
- **生产力。** Rust 拥有出色的文档、友好的编译器和清晰的错误提示信息，还集成了一流的工具——包管理器和构建工具，智能地自动补全和类型检验的多编辑器支持，以及自动格式化代码等等。

鉴于 Rust 有如此显著的优点，被越来越多的系统或者软件采用，也有很多其他语言的库被使用 Rust 重写以获得更好的性能以及安全性。

Rust 可以做什么？

Rust 是一种通用编程语言，也即它不是针对某一特定的领域设计的，而是针对通用任务设计的。我们可以用 Rust 来开发：

- 命令行应用
- WebAssembly
- 网络服务，例如 HTTP 服务器、业务系统后端 API 等
- 嵌入式
- 图形界面（GUI）应用
- Web 页面

安装 Rust 环境

Linux、macOS 系统

在 Linux 或者 macOS 系统上，可以直接使用命令行来安装 Rust 环境。打开一个终端窗口，执行下面的命令即可完成 Rust 环境的安装：

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

如果遇到网络原因导致安装失败的，可以多尝试几次即可成功。

安装成功之后，在`$HOME/.cargo`目录下即是Rust的相关文件。其中`$HOME/.cargo/bin`也被安装脚本自动加入到`$PATH`环境变量中了，重新启动终端窗口，执行`cargo --version`命令，如果看到版本信息的输出，表示Rust安装成功。

Windows系统

在Windows系统中安装Rust，需要首先安装Microsoft C++ Build Tools。打开浏览器访问：<https://visualstudio.microsoft.com/visual-cpp-build-tools>，下载并按照引导安装。



Microsoft C++ Build Tools安装完毕之后，打开浏览器访问<https://www.rust-lang.org/learn/get-started>下载Rust的安装包。

同样，如果遇到网络问题，多试几次即可。

也可以在前述页面中找到离线安装包，下载之后进行安装。

也可以从我的百度网盘下载：链接: <https://pan.baidu.com/s/1C8sHPsq129iNhn2XUFbYU> 提取码: unxc。

The screenshot shows a file download interface with the following details:

| 文件名 | 修改时间 | 大小 |
|--|------------------------------|------|
| vs_BuildTools.exe | Microsoft C++ Build Tools | 3.8M |
| rustup-init.exe | 2024-05-16 00:37 | 12M |
| rust-1.78.0-x86_64-pc-windows-msvc.msi | Rust 离线安装包 for 64bit Windows | |

安装完成之后，Rust 的相关文件也是安装到 `C:\Users\your_user_name\.cargo` 路径下的，默认情况下系统路径也已经设置完毕。

打开 Windows PowerShell 或者命令行检测安装是否成功，执行命令：`cargo --version`，如果可以出现正确的版本号则表示安装及设置成功。如果系统环境变量未能成功设置，需要进行手动设置，将 `C:\Users\your_user_name\.cargo\bin` 路径加入到系统的 Path 变量中。关于 Windows 上设置环境变量的操作方式，可以参考：https://blog.csdn.net/weixin_39934453/article/details/136782285。

关键工具

安装完 Rust 之后，有几个关键的命令行工具，你可以进入到安装目录下的 `bin` 文件夹看到这些文件。这里列出几个关键的、常用的命令。

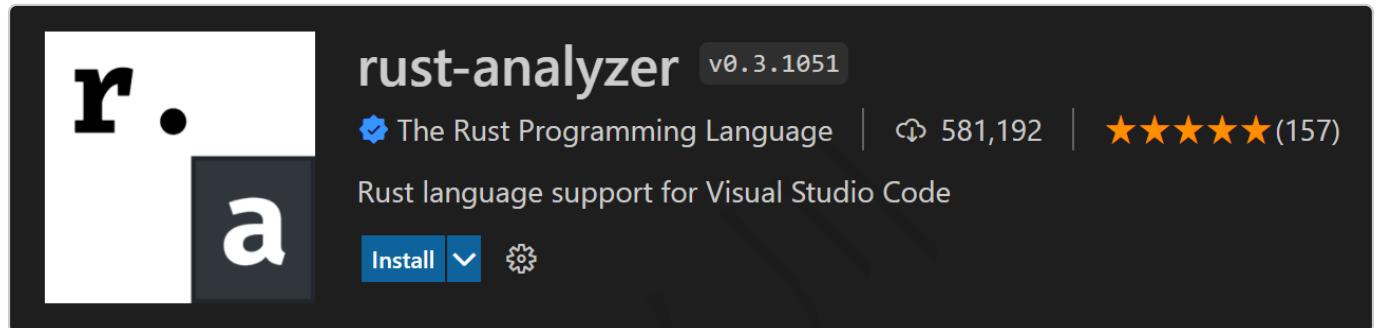
- `rustup` - Rust 工具链管理工具，这个命令主要是用来管理 Rust 的版本及相关工具
 - 执行 `rustup update` 命令可以升级当前电脑上 Rust 的版本到正式发布的最新版本
- `cargo` - Rust 项目管理工具，包管理工具
- `rustc` - 将 Rust 源代码编译成可执行程序。这个命令很少真正使用，大部分时候我们都是使用 `cargo` 命令来管理和构建项目

设置 IDE 或编辑器

支持 Rust 的编辑器有很多，例如 VS Code、Sublime Text 等；大名鼎鼎的 JetBrains 公司也在近期为 Rust 开发了一个 IDE 叫做 RustRover，~~目前这个产品还处于抢先体验版本，不确定未来是否收费。~~ 2024-05-22 RustRover 正式发布，个人免费，商用收费。

设置 VS Code

在 VS Code 中开发 Rust 项目需要安装 rust-analyzer 插件：



安装完毕之后，就可以使用了。

Hello World

依照传统，我们的第一个 Rust 程序还是 Hello World。

Step 1: 在自己的电脑上创建一个文件夹 `rust_hello_world`。

Step 2: 在 `rust_hello_world` 文件夹下创建文件 `main.rs`。

Step 3: 使用代码编辑工具编辑 `main.rs` 文件，其内容如下：

```
1 fn main() {  
2     println!("Hello world!");  
3 }
```

Step 4: 打开一个命令行窗口，进入到项目目录 `rust_hello_world`，然后执行命令：`rustc main.rs`。命令成功执行之后，会在当前目录生成一个可执行文件 `main`（Windows 上是 `main.exe`）。

Step 5: 在命令行窗口执行上一步骤生成的 `main` (Windows 上是 `main.exe`) 就可以看到输出内容: `Hello world!`。

Hello Cargo

Cargo 是 Rust 的构建系统及项目包管理工具，它可以帮助我们做很多事情，包括下载项目所需要的依赖库、构建项目等。

Step 1: 使用 Cargo 创建一个项目

在命令行窗口中执行下面的命令：

```
1 $ cargo new hello_cargo
```

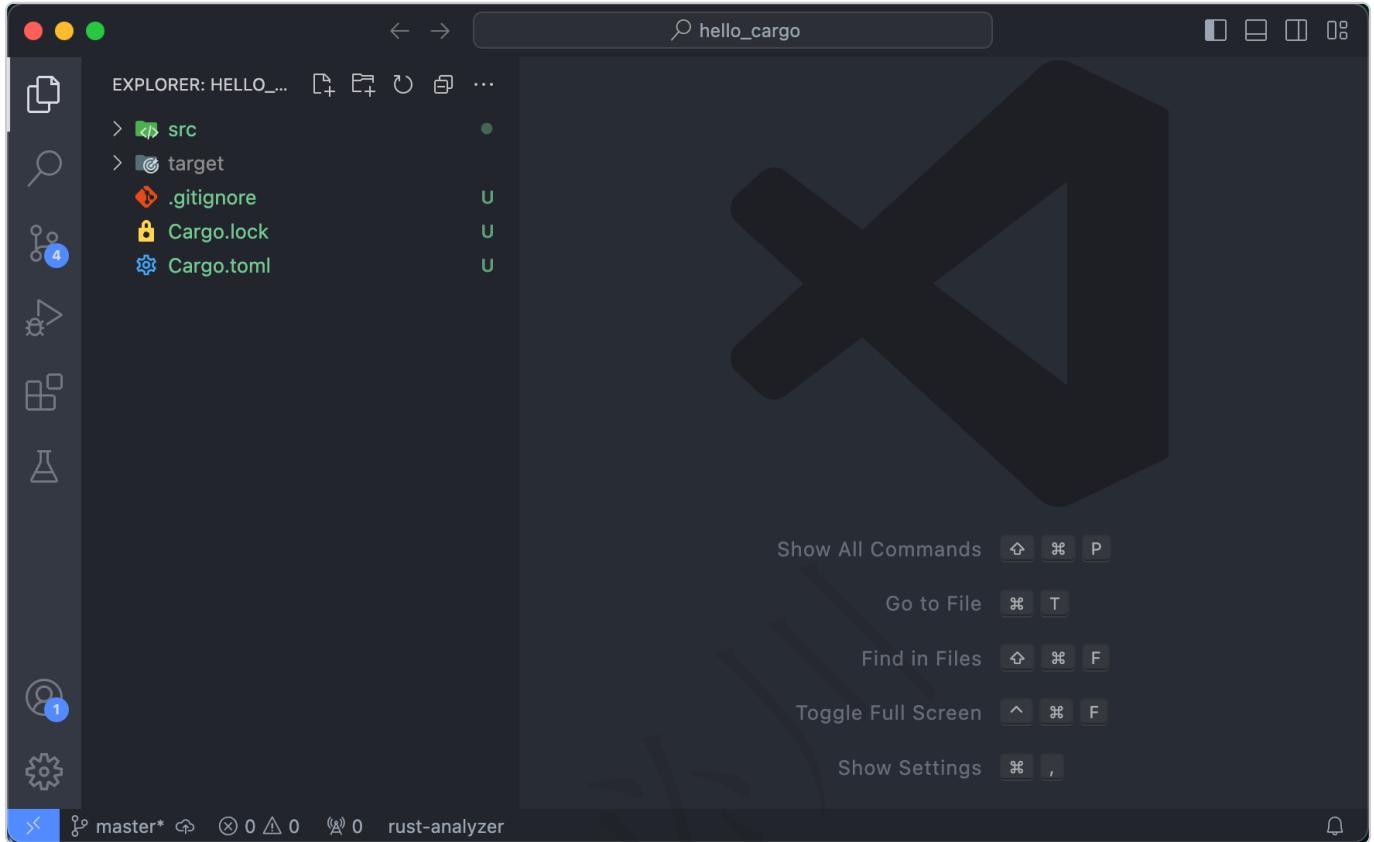
命令成功执行之后，Cargo 会帮助我们创建一个名为 `hello_cargo` 的文件夹，文件夹中的内容如下：

```
1 .
2 └── Cargo.toml
3   └── src
4     └── main.rs
```

很简单，就一个项目配置文件 `Cargo.toml`，一个 `src` 文件夹，以及在 `src` 文件夹下有一个 `main.rs` 文件。默认情况下，Cargo 还顺便帮我们把这个项目初始化成了一个 git 仓库。如果你不需要这个功能，可以在创建项目的时候使用 `--vcs` 参数：`cargo new --vcs=none hello_cargo`。

Step 2: 编辑 `main.rs` 文件

使用 VS Code 打开 `hello_cargo` 文件夹，由于我们之前安装了 `rust-analyzer` 扩展，所以 VS Code 识别到这是一个 Rust 项目，所以在状态栏中显示了 `rust-analyzer` 字样。



编辑 `main.rs`，将其中的 `println!("Hello, world!");` 改成 `println!("Hello, cargo!");`。

Step 3: 使用 Cargo 构建项目

打开一个命令行窗口，进入到项目所在的目录（存放 `Cargo.toml` 文件的那个文件夹，以下简称 `%PROJECT_DIR%`），执行 `cargo build` 命令来构建项目：

```
1 $ cargo build
2     Compiling hello_cargo v0.1.0
3         (/Users/yuanyq/work/tmp/hello_cargo)
4             Finished `dev` profile [unoptimized + debuginfo] target(s) in
5                 0.18s
```

构建成功之后，编译后的可执行文件输出到 `%PROJECT_DIR%/target/debug` 目录下，文件名为 `hello_cargo`（Windows 上是 `hello_cargo.exe`）。

在命令行窗口中执行上面步骤输出的文件：

```
1 $ ./target/debug/hello_cargo  
2 Hello, cargo!
```

Step 4: 使用 Cargo 直接运行项目

除了可以使用 Cargo 来构建项目之外，也可以直接使用 Cargo 来运行项目。在命令行窗口中，进入到项目所在目录 %PROJECT_DIR%，执行： cargo run 命令，即可完成构建+执行的步骤。

认识 `Cargo.toml`

`Cargo.toml` 文件是 Rust Cargo 项目的核心文件，它的内容类似下面的：

```
1 [package]  
2 name = "hello_cargo"  
3 version = "0.1.0"  
4 edition = "2021"  
5  
6 [dependencies]
```

第 1 行 `[package]` 表示这个段落是针对包的配置

第 2 行 `name = "hello_cargo"` 是这个项目的名称。按照 Rust 的规范，项目名称、包名称、模块名称等，都采用全小写下划线分隔的格式；

第 3 行 `version = "0.1.0"` 是这个项目的版本号；

第 4 行 `edition = "2021"` 是这个项目采用的 Rust 语言的版本

第 6 行 `[dependencies]` 是这个项目所依赖的库的信息。目前由于这个项目没有依赖任何扩展的库，所以这个段落是空的。

几乎每种项目都有自己的项目配置文件，比如我们常见的：

| 项目类型 | 项目配置文件 |
|------------|---------------------------|
| Cargo | <code>Cargo.toml</code> |
| Java Maven | <code>pom.xml</code> |
| Node.js | <code>package.json</code> |
| Deno | <code>deno.json</code> |

Rust 生态中常用的库

在 Rust 生态中，常用的库简单列举一下：

- `serde` - 序列化/反序列化的库
- `tokio` - 高性能异步库
- `thiserror` - 异常处理库
- `clap` - 命令行参数解析库
- `actix-web`、`rocket` - Web 框架
- `diesel`、`sqlx` - 数据库
- `yew` - 前端项目库
- `tauri`、`egui` - GUI 库，可以使用 Rust 开发带有图形界面的应用

参考资料

- Rust 官网：<https://www.rust-lang.org>
- Crate 官网：<https://crates.io>
- 官方学习资料：<https://www.rust-lang.org/learn>

回顾

1. Rust 发展历史
2. Rust 的特点：高性能、可靠性、生产力

3. 安装 Rust 环境并配置开发工具。Windows 系统建议下载离线安装包
4. 设置 IDE 或编辑器。推荐 VS Code + rust-analyzer 扩展
5. Hello world 以及 Hello cargo
6. Cargo 项目控制文件：Cargo.toml