Rust 编程笔记

adamanteye

1. 文档及参考

1.1. 手册

- Rust 单页手册

1.2. 代码规范

- Rust API Guidelines

1.2.1. 命名

`crate` 的命名应当为 `snake_case`,更多讨论参考 Naming convention for crates · rust-lang/api-guidelines · Discussion 29

1.3. 性能

- The Rust Performance Book

2. 对象安全

3. 异步

无脑 tokio 全家桶就对了:

- tokio-rs/tokio

支持并发读写的容器:

- xacrimon/dashmap: Blazing fast concurrent HashMap for Rust

参考

- Async Rust 的实现 | 夜天之书

4. 并行

多线程并行化 rayon-rs/rayon: Rayon: A data parallelism library for Rust

5. 解析器

- Nom — Parser
- zesterer/chumsky: Write expressive, high-performance parsers with ease

5.1. chumsky

`chumsky` 是一个 PEG 解析器.

- `choice` 选择第一个成功匹配的,注意安排优先级
- `to_slice` 会把整个解析器的输入范围都捕获为原始字符串,所以应该放在可能有的 `then_ignore` 之前

6. 网页服务

自己一直习惯用 `axum` 来着,虽然上手会费劲一点,但感觉逻辑还是很自洽的,其他著名的库还有 `actix-web`, `poem`, `rocket`,前者开发不顺,后两者没有试过.

此外,支持 OpenAPI 规范的 `utoipa` 也是简化文档编写的利器.

关于后端服务性能调优,可以参考:

- Structural changes for +48-89% throughput in a Rust web service - Sander Saares

7. 系统调用

rustix - Rust

8. 包管理

crate 是 Rust 编译的最小单元,即便是使用 `rustc` 编译的单文件,也是一个 crate. 更高的一层是 **package**,只要包含 `Cargo.toml`,就是一个 package.单个 package 最多有一个库 crate,以及任意多个二进制 crate.

8.1. feature

Cargo 会将同一个包的 features 进行并集操作,例如 A,B 都依赖了 C 包的不同 feature,那么为了避免产生两个版本的 C,会对 C 启用的 features 取并集.

这意味着,添加 feature 理应增加功能,同样,feature 也不应该互斥.

9. 调试技巧

dbg!宏用于打印到 `stderr`. 可以在 `cargo test` 下使用,也可以调试 release 构建下出现的问题.

10. 超好用的库

10.1. 错误处理

- dtolnay/anyhow: Flexible concrete Error type built on std::error::Error

10.2. 输入输出

- kkawakam/rustyline: Readline Implementation in Rust
- Nukesor/comfy-table: Build beautiful terminal tables with automatic content wrapping

- clap-rs/clap: A full featured, fast Command Line Argument Parser for Rust

10.3. 内存分配

- purpleprotocol/mimalloc_rust: A Rust wrapper over Microsoft's MiMalloc memory allocator

11. 嵌入式开发

- Introduction - Rust Embedded Drivers Book

便利蜂 | 门店网络与 Rust 落地实践 - Rust 精选