Rust 框架介绍

梅骏逸 郭大玮

2024.9.2

概述

- 基于 orzcc 重构的框架
  - orzcc 未使用 LLVM IR；
  - 只实现了 RISC-V 后端；
  - ~~可能~~存在 Bug。
- 完成实验的代码量（基于 orzcc 估计）
  - 前端（Parser、AST、IRGen）：800 ~ 1200 行；
  - 中间代码：2000 ~ 3000 行；
  - 后端（框架、CodeGen、Regalloc）：2000 ~ 3000 行（仍在整理）；
  - 优化：基础设施 300 ~ 500 行，基本要求 200 ~ 500 行，进阶优化每一个 100 ~ 1000 行不等。
- Rust 框架代码量、实现难度、所需时间远大于 C++ 版本。
- 实验文档 & 项目地址

Why Rust?

> Rust 写图着色比 C++ 简单一个数量级。
>
> — GDW

- 更好的基础设施：`cargo`, `rust-analyzer`, `clippy`, `rustfmt`, etc;
- 类型、生命周期、所有权系统;
- 测试、文档、错误处理;
- ……

Why not Rust?

- 语言特性复杂，学习成本高;
- 许多特性面向大型项目开发，可能会存在许多不必要的复杂性;
- 参考资料少;
- 防呆不防傻。

注意事项

- 仍然要求实现完整的编译器；
- 如果中途切换 C++ 版本，则需要重新完成所有 C++ 框架的有关实验；
- 指导书中给出了一些编译器参考项目，可以学习；
- 遇到问题及时与助教联系
  - 提问前请先阅读指导书中“如何解决问题”一节；
- 注意保存 git commit 记录；
- 检查时会提问代码实现（可能会涉及一些 Rust 的特性）；
- 难度较高。

如果你希望使用 Rust 完成实验

为了尽可能防止实验中途从 Rust 切换到 C++ 而耽误后续实验的情况，我们要求选择 Rust 的同学在开始正式的编译器构建工作之前完成以下任务：

- 使用 Rust 实现一个 Dijkstra 或最小生成树算法：
  - 请不要使用第三方图论、算法或者数据结构库；
  - 请尽可能使用 Rust 的惯用写法；
  - 请保持一个良好的代码风格；
  - 请尽可能证明你代码的正确性：
    - 包括但不限于单元测试、模糊测试等；

如果你希望使用 Rust 完成实验（续）

- 建议在之后两周内完成，最迟于预备工作 1 完成时联系助教；
- 完成之后请联系助教（梅骏逸、郭大玮）进行检查，每一组只有一次检查机会；
- 不计入实验成绩，只是为了确保你们组具备了 Rust 的编程能力；
- 参考资料：Rust 实验指导文档；
- 如果你希望用其他方式表明你具备 Rust 的编程能力（例如已有的 Rust 项目，或者你是 rust-org 的一员），也可以联系助教；
  - 除非组内所有成员都能够表明自己具备了 Rust 编程能力，否则仍然需要完成上述任务；
- 如果无法在上述期限内按照要求完成这一任务，我们将默认你们选择 C++ 完成实验。