

# 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。
- [参考文档](#) & [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。