rCore 的基本开发环境的搭建

罗崚骁　　宋香君

2020 年 3 月 29 日

课程设计目标

- **C/C++ 开发**：基于 musl libc 的 gcc 工具链
- **工程构建**：GNU make, CMake
- **Rust 开发**：Rust 工具链
- **版本控制系统**：git
- **编辑器**：vi, vim
- **软件包管理器**：cargo

课程设计背景

rCore 已经具备了运行一些软件的能力。但是，rCore 的软件开发能力还稍显不足。

- 编译器支持不足，目前仅支持 musl-gcc
- C/C++，Rust 标准库实现支持不详，缺少测试，目前仅支持 musl-libc
- 没有版本控制系统
- 缺少项目构建工具
- 编辑器只能说勉勉强强（vi）
- 没有软件包管理器，甚至不能从源码安装软件

已有工作

- rCore 已经实现了相当数量的系统调用，能够支持 rcore-user 中的软件，
- 考虑到编译器是我们的课程设计中重要的一环，尤其可以参考此前 rCore 对 musl-gcc 的支持历程
- 可以设法获取的各种满足我们需求的开源软件，这一点决定了我们的工作思路是对 rCore 针对软件进行完善，而非针对 rCore 开发软件

工作计划及目前进展

工作计划按如下顺序开展：

- 完善对 gcc 的支持，测试 C/C++ 标准库实现
- 支持 make，使 rCore 具备从源码安装软件的能力
  - 还可考虑支持 CMake
- 支持 Git
- 支持 Rust 工具链，主要包括：
  - rustc
  - cargo
  - 测试 rust 标准库实现
- 支持更多的编辑器
  - nano
  - vim

软件支持过程

安装

- 寻找官方发布的预编译版本
- 外部源码构建
- rCore 内部源码构建（目前估计难以达到较好效果）

软件支持过程

测试

- 对于 C/C++,Rust 标准库的测试，似乎没有找到官方公开的测例，运行大规模工程
- 对于各种软件，决定选取其重要功能进行测试
  - make 是否能顺利构建项目、是否支持增量构建等
  - Git 的 add,commit,log,status,branch 等功能
  - 当然如果能找到测例的话就最好了

软件支持过程

系统调用

- 利用 rCore 的 log 信息

```
ode: 0o666
[ INFO][0,1] => Err(ENOENT)
[ INFO][0,1] close: fd: 18446744073709551615
[ INFO][0,1] => Err(EBADF)
[ INFO][0,1] close: fd: 18446744073709551615
[ INFO][0,1] => Err(EBADF)
```

小组分工

- 罗崚骁
  - 实现和完善系统调用
  - 完成对 Git 支持
  - 完成对 cargo 离线版的支持
  - 测试 C++ 和 Rust 项目
- 宋香君
  - 实现和完善系统调用
  - 完成对 make,CMake 的支持
  - 测试 C 项目

结束

谢谢观看