

尝试用RUST写教学操作系统

向勇、陈渝
清华大学计算机系

20181123

背景

- 用什么语言写操作系统？
 - 汇编
 - C
 - C++
 - GO
 - RUST
 - ...

此题未设答案

各位老师所教的课程信息统计

- A 计算机组成原理
- B 编译原理
- C 操作系统
- D 其他

提交

面临的问题

- 教学的要求
 - 简洁
 - 实验环境: Nachos、XV6、ucore
 - CPU: X86、MIPS、ARM、RISC-V
 - 语言: 汇编、C、...
 - 真实
 - QEMU、开发板、真实系统
 - 开放
 - 树莓派、Edison、FPGA

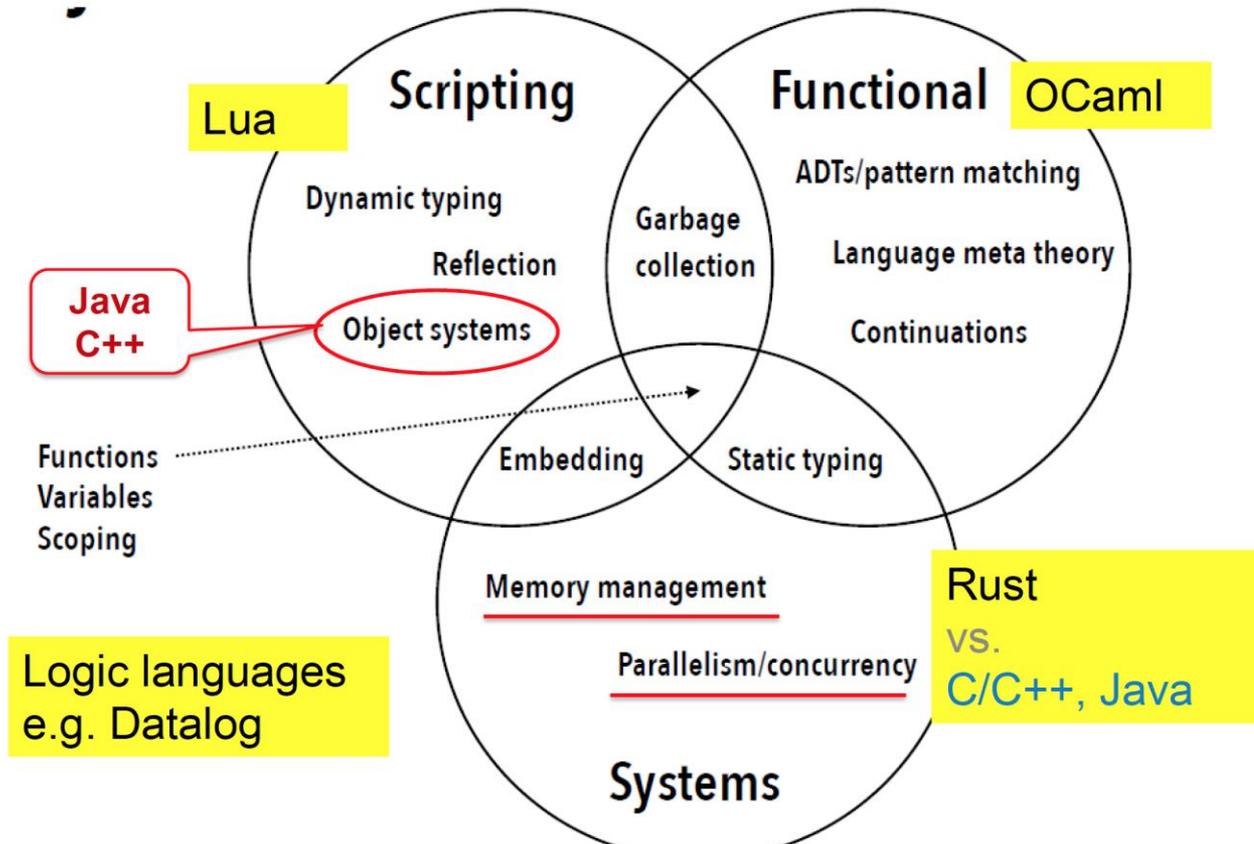
已有的尝试

- 用C语言写OS
 - MIT 6.828
 - Harvard cs161
 - Stanford cs140/140e
 - Univ. of Wisc. CS-537
- MIT: 用GO写OS
 - <https://github.com/mit-pdos/biscuit/>
- Standford: 用rust写OS
 - <https://web.stanford.edu/class/cs140e/>

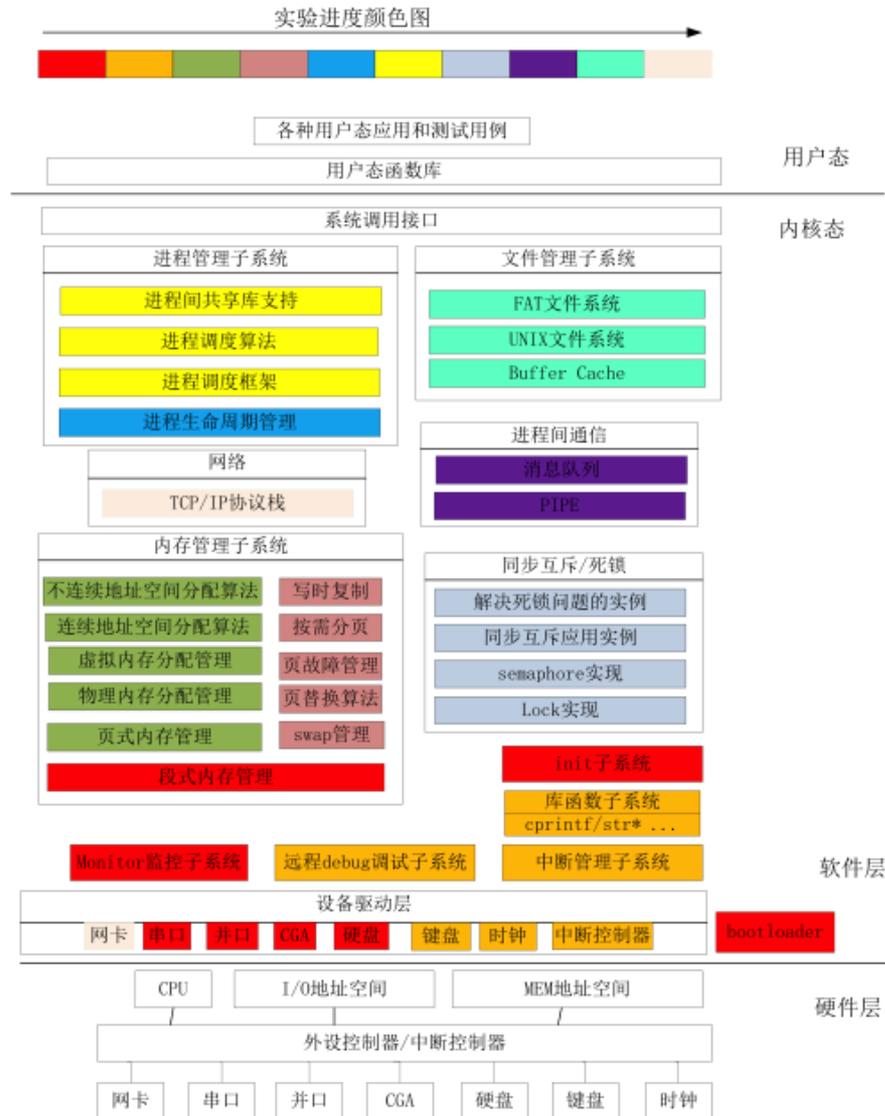
内核开发需要的程序设计语言

Selected PLs

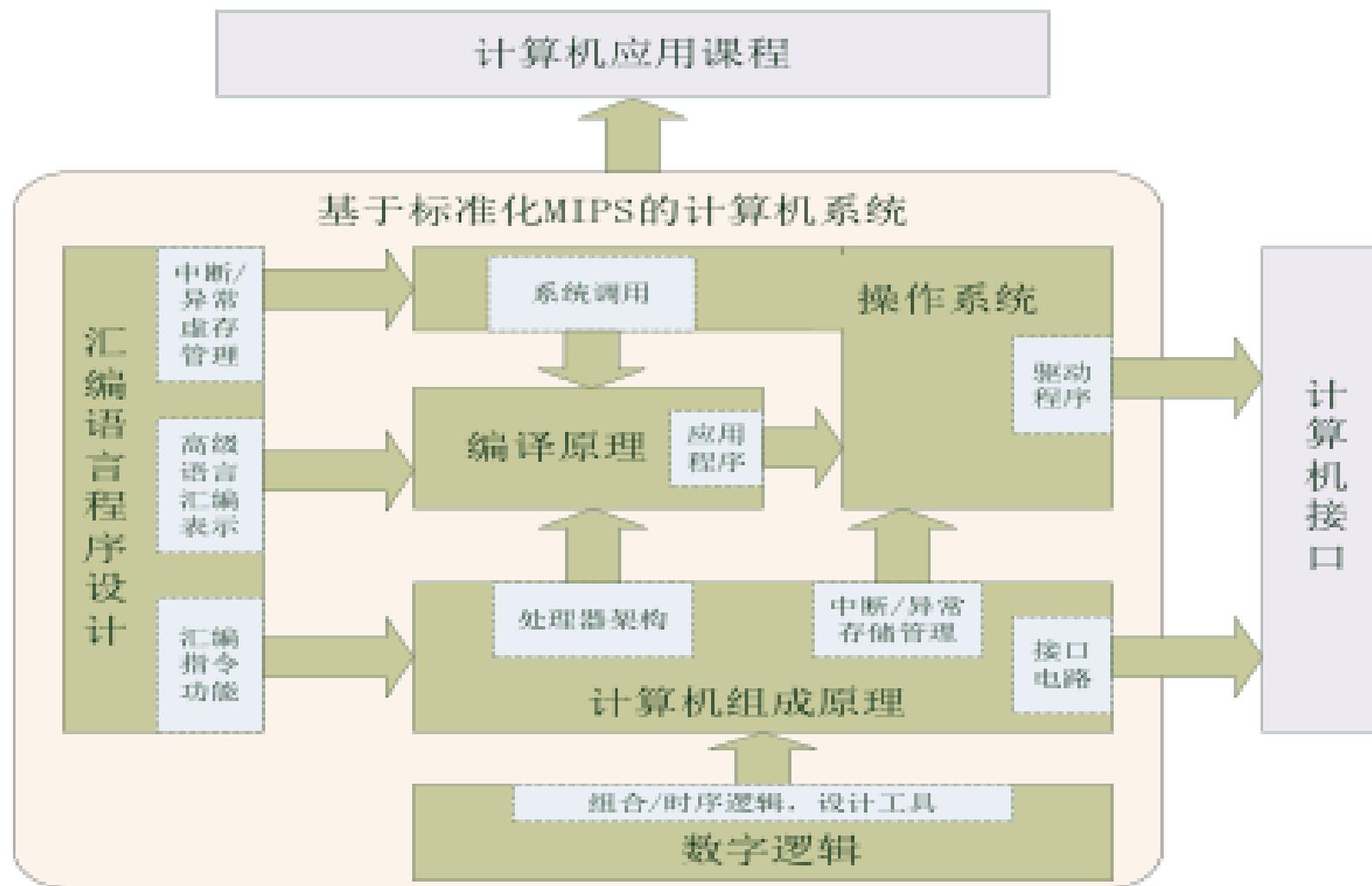
中国科学技术大学
University of Science and Technology of China



我们的尝试-教学操作系统ucore



我们的尝试-系统类课程的协调



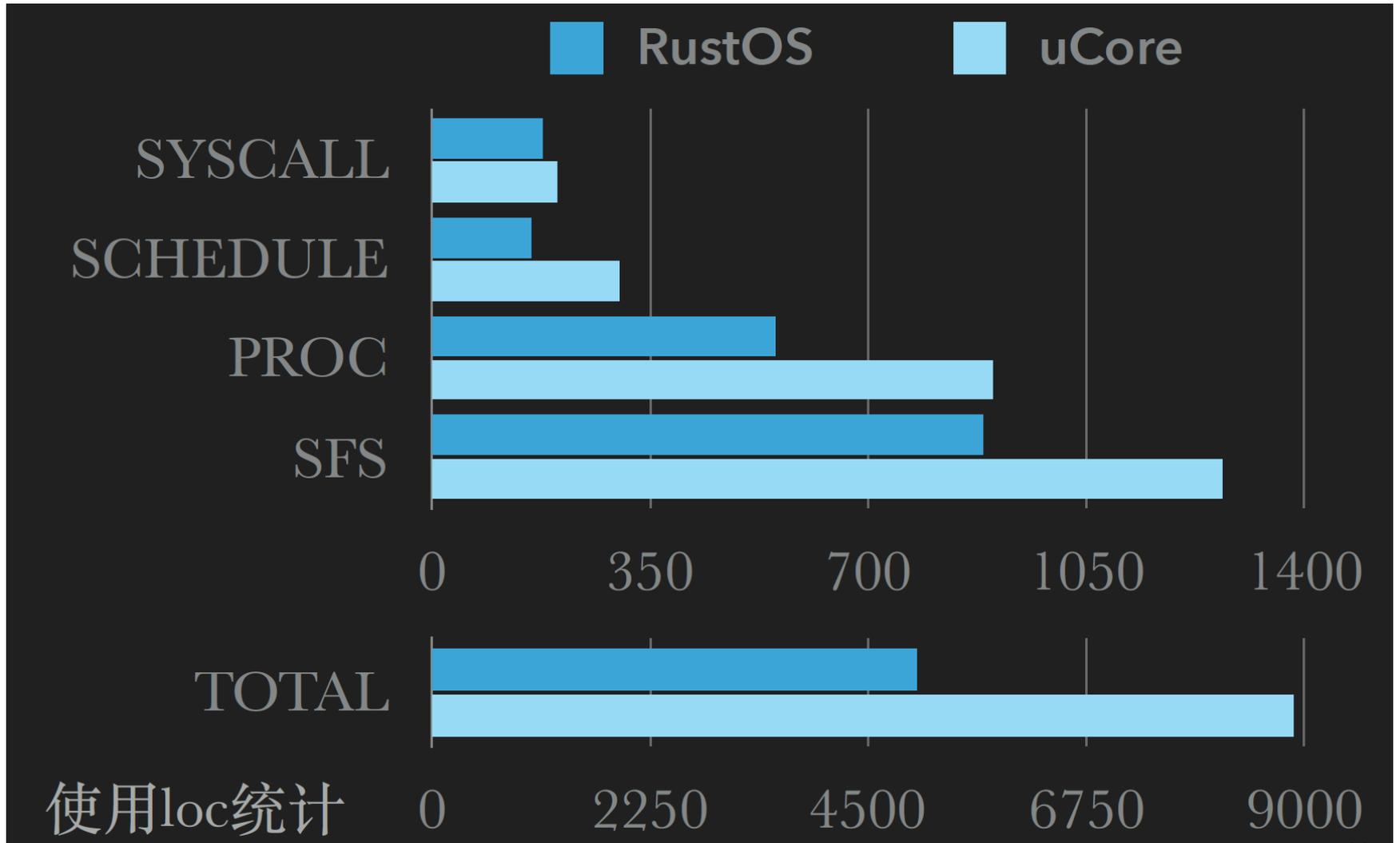
用RUST写操作系统： 2018年春季操作系统课

序号	选题方向	大实验题目	小组成员列表（姓名和学号）
2	op-cpu	轻量OS在“小脚丫”FPGA开发板上的实现	张天扬 2015011306 王延森 2015011285 戴臻暘 2015011296
11	rust	Rust OS for x86_64 SMP	王润基 2015011279 朱书聪 2015011322
13	rust	Rustable - ucore 在 arm 平台的 rust 移植	乔逸凡 2015013188 谭咏霖 2015011491
14	GUI	ucore with GUI	贾越凯 2015011335
15	rust	Rucore with LKM Drivers	乔一凡 2015011398 杨国炜2015011320

用RUST写操作系统： 2018年秋季操作系统专题训练课

序号	大实验题目	小组成员列表（姓名和学号）
2	Rust OS 上树莓派 USB 与 VideoCore IV 显卡驱动的移植	贾越凯 2015011335 寇明阳 2015011318 孔彦 2015011349
3	Rust OS 多核移植与基于PARD框架的线程级Label管理	王纪霆 2015011251
4	Rust OS wiki完善与教学lab实验的制作	陈秋昊 2015011283 刘辰屹 2015011277 朱书聪 2015011322
5	Rust OS 参考sv6的多核实现和优化	王润基 2015011279
6	Rust OS 移植到 rv64 及llvm编译器支持	戴臻暘 2015011296
7	Rust OS 树莓派网络及声卡支持	霍江浩 2015010611 吴昆 2015010625 范书沛 2015011202

效果比较-代码量统计



Rust的语言特征

- 类型推断->现代语言标配
- 面向对象特性->OO风格开发内核是符合思维习惯的
- 通过泛型实现高效多态
- 匿名函数->一部分的函数式特性
- 比C/C++ 更好的代码管理, 重用: `cargo` 和 `crate`
- 干净宏->简化静态分析

RUST的安全哲学

- 尝试把系统正确性证明整合到语言本身当中
- Rust有严格的安全约束，也可以把编译时约束转移到运行时（例如Mutex，RefCell），也允许程序员显式地指出不安全（unsafe块），并使用安全封装和管理不安全
- unsafe块是一个精妙的设计，在你想偷懒破坏安全性时给你带来小小的骚扰

Rust 和 C++ 有哪些优劣?



匿名用户

rust: 编译时想撞墙。

c++: 调试时想跳楼。

发布于 2016-01-20

▲ 102



● 6 条评论

Rust的安全特征

- 类型安全: 远离void* 保安全. 远离隐式cast
- 内存安全: 编译器自动推断变量的生命周期, 自动插入free,防止程序员忘记写free. (原则上) 不会有null-pointer-dereference, double free, use-after-free 等内存问题. rust 有runtime 检查ownership / buffer overflow -> 缓冲区溢出不会导致system compromise
- 并发安全: 对于全局变量, 它自带一个锁. 访问全局变量的过程必须取得锁 -> C里面全局变量和锁时分离的, 时常会忘记加锁, 当锁离开作用域时自动释放, 防止造成死锁
- Rust 提供了unsafe 这种块能够做不安全但必须的事情如指针算术. 但不安全的代码越少越好

RUST的运行时约束： 所有权机制和资源管理

- `RefCell<T>`
 - `borrow()`
 - `borrow_mut()`
- `Mutex<T>`
 - `lock()`
- `Dirty<T>`
 - `borrow()`
 - `borrow_mut()`
- `Rc<T>`
 - `clone()`
 - `drop()`

PageTable in Rust vs C

PageTable

P4Table

...

P1Table

Entry

Frame

Flags

Entry...

Entry...

P3Table...

PageTable

P4E

.....

.....

P4E

.....

.....

.....

.....

P1E

.....

.....

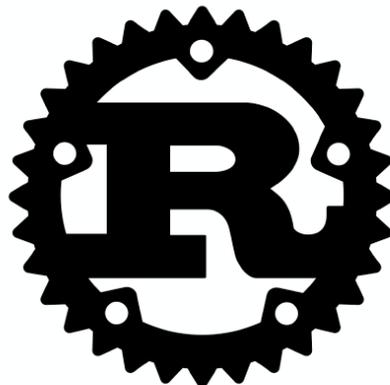
P1E

Entry

Frame

Flags

Pros



Cons

- 线程安全: Mutex
- 所有权:
 - 无需 free
 - 变量值安全
- 智能指针
 - Box
 - Rc

- 学习曲线陡峭
- 与编译器作斗争
 - 所有权
 - 生命周期
- 大量 unsafe
 - 野指针访问
 - mut static

RUST对Linux安全漏洞的安全性增强(1)

有人分析 Linux 的安全漏洞, 将它们大致分为 10 类¹, 下面大致地分析, 中间一列两个 + 表示是直接被 Rust 语言完全解决, 一个 + 表示可以通过编程风格/框架的约束解决

bug 类型	是否解决	备注
未检查用户态传入的指针/下标	+	定义安全/不安全类型
未检查权限	+	使用函数式操作, 并且 执行操作前先取得 token 检查
缓冲溢出	++	
整数溢出	+	可以设置 runtime 溢出 就发生 panic

¹Linux kernel vulnerabilities: State-of-the-art defenses and open problems, Haogang Chen et al., APSys '11

RUST对Linux安全漏洞的安全性增强(2)

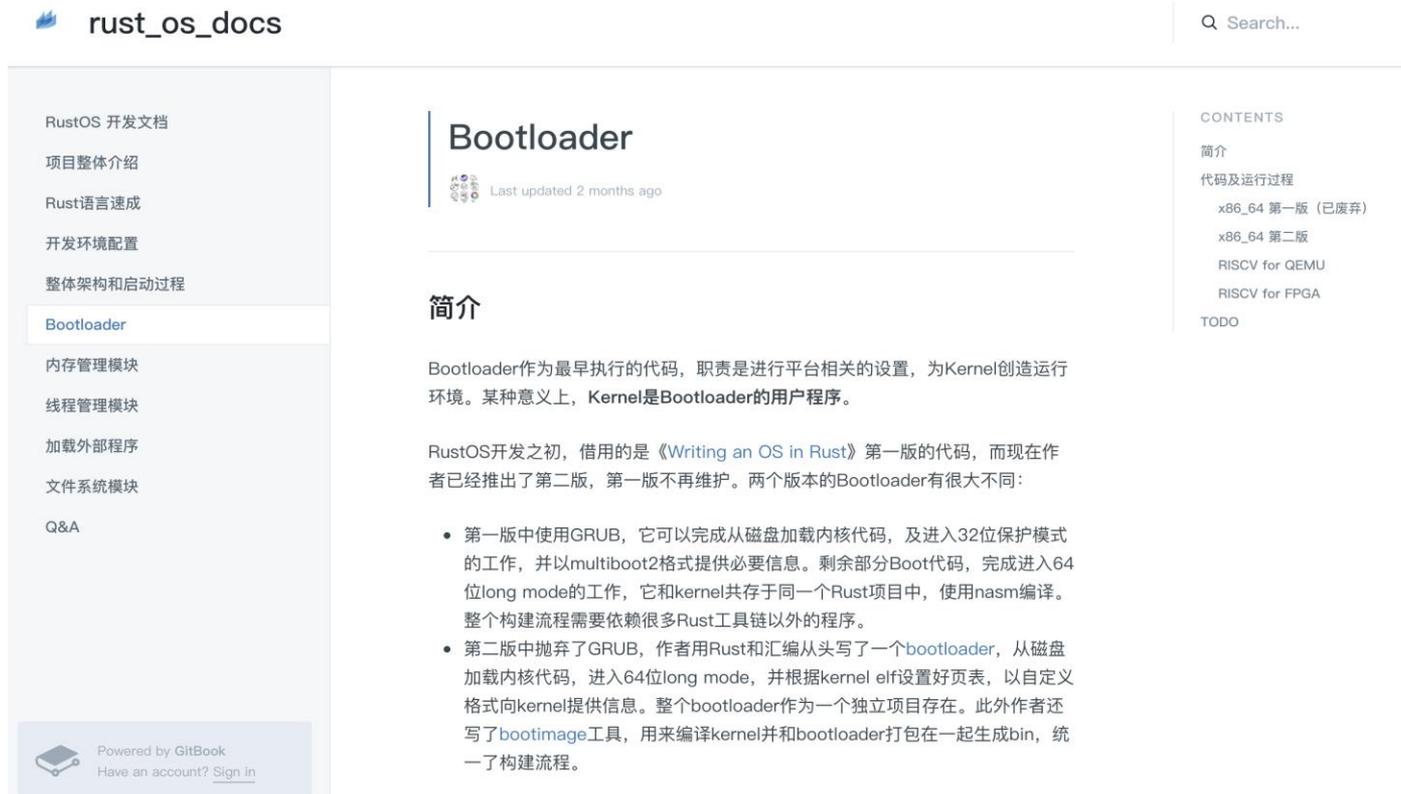
bug 类型	是否解决	备注
不再使用的内核内存中包含敏感信息	+	页框的析构函数中加上清零
空指针解引用	++	
除以 0		
死循环		一般是高层语义出了问题, 不能光靠 Rust 解决
内存管理问题, 如 double free, use after free	++	不考虑内存泄漏的话...实际上 Rust 的自动 free 导致内存泄漏也比 C 要难出现

简单结论

安全语言和好的编程范式/框架/...能够减少很多非语义 bug. 不过光靠 Rust 还不太够.

正在进行的工作

- 形成一组基于RustOS的操作系统课实验



The screenshot shows the 'rust_os_docs' website. The left sidebar contains a navigation menu with items: 'RustOS 开发文档', '项目整体介绍', 'Rust语言速成', '开发环境配置', '整体架构和启动过程', 'Bootloader' (highlighted), '内存管理模块', '线程管理模块', '加载外部程序', '文件系统模块', and 'Q&A'. The main content area is titled 'Bootloader' and includes a '简介' (Introduction) section. The introduction text states that the bootloader is the earliest code to execute, responsible for platform-related settings and creating a runtime environment for the kernel. It mentions that the first version used GRUB, while the second version is written in Rust and assembly. A list of bullet points follows, detailing the differences between the two versions. The right sidebar contains a 'CONTENTS' section with links to '简介', '代码及运行过程', 'x86_64 第一版 (已废弃)', 'x86_64 第二版', 'RISC-V for QEMU', 'RISC-V for FPGA', and 'TODO'. At the bottom of the page, there is a footer with the text 'Powered by GitBook' and a link to 'Sign in'.

rust_os_docs

Q Search...

Bootloader

Last updated 2 months ago

简介

Bootloader作为最早执行的代码，职责是进行平台相关的设置，为Kernel创造运行环境。某种意义上，Kernel是Bootloader的用户程序。

RustOS开发之初，借用的是《Writing an OS in Rust》第一版的代码，而现在作者已经推出了第二版，第一版不再维护。两个版本的Bootloader有很大不同：

- 第一版中使用GRUB，它可以完成从磁盘加载内核代码，及进入32位保护模式的工作，并以multiboot2格式提供必要信息。剩余部分Boot代码，完成进入64位long mode的工作，它和kernel共存于同一个Rust项目中，使用nasm编译。整个构建流程需要依赖很多Rust工具链以外的程序。
- 第二版中抛弃了GRUB，作者用Rust和汇编从头写了一个bootloader，从磁盘加载内核代码，进入64位long mode，并根据kernel elf设置好页表，以自定义格式向kernel提供信息。整个bootloader作为一个独立项目存在。此外作者还写了bootimage工具，用来编译kernel并和bootloader打包在一起生成bin，统一了构建流程。

CONTENTS

- 简介
- 代码及运行过程
- x86_64 第一版 (已废弃)
- x86_64 第二版
- RISC-V for QEMU
- RISC-V for FPGA
- TODO

Powered by GitBook
Have an account? [Sign in](#)

后续工作

- 适合操作系统开发的语言剪裁
 - 汇编
 - C
 - Rust
 - ...
 - 什么是适合内核开发的语言特征？

您认为系统软件教学应该选择什么程序设计语言？

正常使用主观题需2.0以上版本雨课堂

作答

讨论时间