



开题准备汇报

董若扬

2024/2/22



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

目录 / CONTENTS

01

基本情况

02

选题

03

理解

04

学习方向



PART ONE

基本情况



- 408考研（数据结构，组成原理，操作系统，计算机网络），对操作系统有基础的理解。
- rCore 实验进行到 第七章——进程间通信与I/O重定向。
- 第一次接触 Rust 语言，能写基础的 Rust 程序，能理解 Rust 项目代码。
- 写过 RISC-V 架构的流水线CPU外设简单驱动，对 RISC-V 架构的设计思想有基础的了解。



PART TWO

选题



研究背景：

由于内核代码的复杂性和快速的开发速度，加上使用不安全的低级编程语言，导致内核中经常出现错误和漏洞。尽管已经有许多静态和动态机制来保护内核代码的执行，但攻击者仍然能够找到绕过这些保护措施的新方法。现代内核需要一种实际的手段来限制单个漏洞的影响，即内核子系统的隔离。



目标:

大多数隔离技术限定在特定的硬件平台之上，不仅性能受限，扩展性也很差，无法广泛部署。通过**语言机制**实现与**硬件支持**相同的子系统隔离效果，在保证性能的同时达到**更好的扩展性和适用性**。



任务要求:

- 1.理解为什么需要对操作系统进行隔离
- 2.熟悉常见的**硬件隔离手段**
- 3.具备对内核子系统的分析和拆解能力
- 4.具备较强的编程实现能力
- 5.设计并实现基于语言机制的隔离措施并应用到内核子系统上



PART THREE

理解

为什么需要对操作系统进行隔离

- 操作系统中的不同部分可能存在漏洞或错误，一个部分的故障可能会影响到整个系统的稳定性和安全性。通过隔离不同的内核子系统，可以限制单个漏洞的影响范围，提高系统的安全性和稳定性。
- 设计上与微内核操作系统几分相似，避免局部的异常造成整个系统的崩溃。其与微内核的区别之一是它并不是把一部分功能交给用户空间管理，而是在内核中实现**更细粒度的隔离**，利用 **内存安全**和**并发模型**等特性降低单个模块的故障对系统整体的影响。

常见的硬件隔离手段：

- 虚拟化技术**：如虚拟机、容器等，可以在不同的虚拟环境中运行不同的操作系统或应用程序，从而实现隔离。
- 内存保护**：通过硬件支持的内存管理单元（MMU）等机制，实现内存隔离，防止内存越界访问等问题。
- 安全扩展指令集**：一些现代处理器提供了安全扩展指令集，可以在硬件层面提供更多的安全保障。

为什么选用Rust语言：

- 安全性：** Rust语言提供内存安全和线程安全的编程环境。通过 Rust 的所有权系统、借用检查器和类型系统等特性，可以在编译时捕获内存安全和数据竞争等常见错误，减少了内核中出现漏洞和错误的可能性。
- 性能：** Rust 语言的性能与 C/C++ 相媲美，甚至在某些情况下性能更好。其在实现高性能的内核模块隔离方案时表现优异。
- 可维护性：** Rust 的模块化设计和包管理工具 Cargo，有助于管理项目的依赖关系和版本控制，提高了代码的可维护性。

PART FOUR

学习方向



想法：

- 编写基于 Rust 的库，提供一套接口，用于管理和隔离不同的模块。
- 基于 rCore 实验的操作系统上编写管理模块。
- 创建 Rust 项目，包含模块管理器和模块加载器等组件，用于维护所有加载模块直接的隔离和通信。



RedLeaf操作系统：

- RedLeaf 是一个用 Rust 从零开始的操作系统，旨在探索语言安全对操作系统架构的影响。
- RedLeaf 不依赖硬件地址空间进行隔离，而只使用 Rust 语言的类型和内存安全。
- 脱离昂贵的硬件隔离机制，可以探索轻量级细粒度隔离系统的设计空间。
- 开发了一种新的基于轻量级语言的隔离域抽象，它提供了一个信息隐藏和故障隔离单元。
- 域可以动态加载和彻底终止，一个域中的错误不会影响其他域的执行。
- 在 RedLeaf 隔离机制的基础上，展示了实现设备驱动程序端到端零拷贝、故障隔离和透明恢复的可能性。



谢谢!