开题准备

基本情况介绍

- 专业课基础还算可以
- 有一定`Rust`基础, 对系统编程比较感兴趣
- 粗略地把`rCore`看了一遍, 不过后面几章的课后作业不太会
- 看了一篇关于`seL4`的总结性论文(L4 microkernels: The lessons from 20 years of research and deployment)

基于ReL4的用户态驱动模块设计

背景

ReL4是用Rust重写的支持seL4在RISCV上的系统调用的微内核，用户态中断允许应用不陷入内核直接接收信号。当前ReL4的系统中断通过notification机制代理给用户态。

目标

利用用户态中断改造ReL4的notification机制，使得接收线程无需主动陷入内核来阻塞等待中断到来，提升驱动线程的并发度。利用Rust语言的协程支持，设计并开发基于用户态中断的抢占式用户态驱动模块。

任务要求

- 理解微内核的相关概念，了解微内核的发展趋势和性能瓶颈。
- 理解seL4的notification机制，理解其中的优势和弊端。
- 学习理解用户态中断的相关知识。
- 学习使用Rust异步编程的相关技能。
- 设计用户态驱动模块，并能够部署在ReL4和seL4上，进行性能测试。

我的理解

什么是微内核?

操作系统核心功能被压缩得很小, 其他功能(文件管理, 网络等)被移动到用户进程, 这些进程被称为"服务器", 微内核只提供基本的进程间通信(IPC)等功能.

> 瓶颈: IPC开销.

`notification`

`notification`是一种用于在L4微内核中同步活动的方法，它提供了一种非阻塞的信号机制，可以在不等待其他操作的情况下设置和检查标志(类似`Linux C`的`select`函数)

```
#include <sys/select.h>

int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
struct timeval *timeout);
```

> `select()` 函数用于等待一组文件描述符上的 I/O 事件，并在有事件发生时进行处理。它可以监视读、写和异常事件，并提供超时机制。

用户态中断

通常情况下，中断处理程序在内核态下执行，因为它们需要直接访问硬件和操作系统的资源。但是，有些处理器架构允许在用户态下处理某些特定类型的中断，这就是用户态中断。

这个题目的目标应该是**设计一种用户态驱动程序**, 这个程序能够在用户态下处理中断, 并且是"抢占式"的, 以减少陷入内核的次数, 提高系统性能.