# 答辩PPT讲稿

## 介绍

各位评审老师、同学大家好，我是计算机科学与技术专业陈伟豪，下面将由我来进行本科毕业设计的答辩。我毕业设计的题目为《ReL4操作系统中基于用户态中断的异步系统调用设计》，我的指导老师为陆慧梅老师。

下面是我的论文检测结果和盲审结果，均符合答辩要求。

我的答辩分为五个部分：第一部分介绍毕业设计的研究背景，第二部分介绍毕业设计的开发平台ReL4微内核，第三部分介绍异步系统调用的设计与实现，第四部分介绍实验与结果分析，第五部分为总结和展望。

## 第一部分：研究背景

首先进入第一部分，研究背景。

由于微内核结构的特殊性，低IPC与系统调用成本至关重要。

因此，优化IPC的相关工作主要分为三个部分：提升多核利用率、减少内核路径与减少特权级切换开销。其中，由于前两部分的优化已经趋于极限，减少特权级切换开销成为了优化的主要方向。在这样的现状下，我结合用户态中断提出了改进方案。用户态中断是一种新兴的硬件机制，用户态中断通过硬件的方式，无需陷入内核即可发送信号，且仅需在通信注册过程中陷入内核分配硬件资源，避免了用户态和内核态的上下文切换，具有了更好的通信性能。

所以，本次毕设的内容在于利用用户态中断机制改进后的Notification机制，对系统调用进行异步化改造，减少内核陷入和上下文切换次数，以此减少开销，提升系统性能。

## 第二部分：ReL4微内核介绍

下面进入第二部分，ReL4微内核介绍。

ReL4是Rust重写seL4支持RISC-V接口的微内核，提供基本的内核支持，它的结构如图1所示。其中，Async Runtime模块内实现了异步系统调用的相关方法，是本设计工作的重点，它以基于用户态中断的Notification机制为基础。

因此，本设计的代码组成如下：首先是Root-Task-Demo，它是ReL4的用户态根任务实现，负责注册发起异步系统调用，其次是ReL4-Kernel，它是ReL4的内核完整实现，最后是Rust-seL4，它在编译后暴露ReL4内核的同步系统调用接口，供用户态线程封装调用。它们的github代码仓库如下所示。

## 第三部分：设计与实现

下面进入第三部分，设计与实现。

首先是异步系统调用的框架图，这部分是本设计的核心内容。在图2中可以看到，用户态、内核态都配备异步运行时模块提供制交互方法。用户态与内核态系统调用请求与结果的交互通过Rust协程进行，交互内容写入共享内存。用户态具有一个接收响应的协程与多个提交协程，内核态具有一个处理响应的协程。用户态提交协程唤醒内核协程采用系统调用陷入的形式，该系统调用会选择空闲核心来执行内核处理协程；内核态唤醒用户态接收协程采用基于用户态中断的Notification机制。

异步系统调用工作流程图如下三张图所示。如图3所示，注册异步系统调用的过程也包含了注册用户态中断的过程，4、5内核处理协程与用户态接收协程的工作类似，都需要循环读取待处理的请求或响应，处理完后阻塞让权。不同点在于在内核态处理完后可能需要发送用户态中断唤醒用户态协程。

共享内存Syscall Buffer的实现如图6所示，其中包含两个维护协程状态的原子布尔变量以及两个IPC Queue分别用于储存请求和响应，IPC Queue的读写特性如图6上标注所示。IPC Queue通过互斥锁Mutex实现用户态与内核态的访问控制。

系统调用的区分与参数的传递采用IPCItem数据结构进行，如图7所示，MessageInformation成员用于存储表示不同系统调用的枚举变量，而ExtendMessage成员用于传递系统调用参数。

协程的实现如图8所示，两类协程中都包含了处理方法以及必要的数据结构来进行系统调用参数传递、解析与协程交互。

在本次毕业设计中，实现的异步系统调用主要分为四类，其中输出类系统调用用于输出系统信息，通用类系统调用用于生成内核对象，Notification机制类系统调用用于TCB与Notification对象的注册绑定，内存映射类系统调用用于实现虚拟地址空间到物理地址空间的映射。

# 第四部分：实验与结果分析

下面进入第四部分，实验与结果分析。

在系统的功能测试中，异步化改造的的四类系统调用都可以进行正常的工作，功能性测试将在演示环节中展示分析。

而系统的性能测试选择在不同并发的环境下提交一定量的系统调用请求并比较异步实现与原有实现的处理速度和陷入频率。

首先，从每轮测试耗时的指标上看，在低并发环境下异步实现性能低于原有的同步实现，这是因为在这样的情况下，内核协程处理显著快于用户态提交速度，导致用户协程需要不断陷入唤醒内核协程。随着页框数（并发度）提升，异步实现的性能逐渐超过同步实现。在高并发环境下异步实现性能可以达到同步实现的233.35%，具有很高的性能优势。

其次，从陷入频率的指标上看，同步系统调用的陷入频率恒为1，而异步实现的陷入频率在任何测试环境都低于同步实现，这体现出本设计的工作达到了减少陷入频率的目的。

# 第五部分：总结与展望

下面进入第四部分，总结与展望。

在本次毕业设计中，我所做工作为分析ReL4同步系统调用实现方式与瓶颈并设计实现基于用户态中断的异步系统调用框架。在未来的发展中，首先可以通过时钟占比分析方法优化异步系统调用框架性能；其次可以通过增加内核协程数量进一步提升并发度；最后，可以为ReL4微内核实现调度器根据需求动态选择同步与异步系统调用接口以获得最好的系统性能。

# 结尾

以上就是我的毕设工作，感谢各位专家老师，请您批评指正！