

# 北京理工大学

# 本科生毕业设计（论文）开题报告

学院：计算机学院

专业：计算机科学与技术

班级：07111703

姓名：石文龙

指导教师：陆慧梅

二〇二二年 12 月 30 日

## 1. 毕业设计（论文）选题的内容

在实验室工作的基础上，实现对于 rcore 内核模块化的改进与优化。主要工作包括实现章节的模块化，使所有章节不再被 git 分支隔离开，而是只有逻辑上的隔离关系，后一章节能够以依赖库的形式继承前一章节的成果。

## 2. 研究方案

### 2.1 本选题的研究背景

操作系统是计算机的灵魂，目前国外操作系统品牌几乎垄断了巨大的中国市场，其中在桌面端、移动端的市占率分别超过 94.75%、98.86%。根据 Gartner 的统计数据，2018 年中国的操作系统市场容量在 189 亿以上，其中国外操作系统品牌几乎在中国市场处于垄断地位。截至 2019 年 8 月，在中国的桌面操作系统市场领域，微软 Windows 的市占率 87.66%，苹果 OSX 的市占率为 7.09%，合计为 94.75%；在中国的移动操作系统市场领域，谷歌 Android 的市占率为 75.98%，苹果 iOS 的市占率为 22.88%，合计为 98.86%。

虽然当前中国的操作系统市场依旧是微软的 windows+intel 占据了主导地位，但是 windows 的闭源架构正面临着以 linux 为代表的开源操作系统的挑战。中国的操作系统国产化浪潮起源与二十世纪末，目前正依托于开源操作系统的开源生态以及政策东风正在快速崛起，涌现出了中标麒麟、银河麒麟、深度 Deepin、华为鸿蒙等各种国产操作系统。

所以在这个时期，我们进行操作系统的学习和研究是非常可行的。但是由于随着操作系统的功能实现的增多，其复杂程度也在增加；并且在对操作系统的学习过程中，因为各个章节是被 git 分支隔离开的，所以完成前一个章节的实验后，在下一个章节有关前一个章节的实现内容需要复制代码过来，同样的代码用一次就需要写一次，并且如果改了某一章节的内容，后续的所有章节相同的地方都需要重新改一边，代码的复用性非常不好。

rust 的模块化编程就能很好的解决上述问题，之前做不同章节的课后实验需要切换到对应的 lab 分支去写代码，现在只需要封装一个 crate 加入 workspace 就可以了，之前每个实验都会有大量的代码需要重复的写在每一个章节中，模块化之后这些重复的代码就可以直接引用了，大大提高了代码的复用性。防止修改了某一模块的内容，但是后续章节没有修改造成的错误出现，减少了开发者出错的可能性。

## 2.2 本选题的主要任务

本文主要完善实验室现有的模块化的 rcore 操作系统内核，增加每一个章节的单元测试以及在用户态下的单元测试。

## 2.3 技术方案的分析、选择

### 2.3.1 已有工作

目前实现的模块化，主要有在所有的章节中复用的代码形成了单独的 package，各个章节对于这些复用的代码只需要在 cargo.toml 的依赖中添加上需要使用到的 package 就可以了，不需要再像之前一样需要将这些已经写过的代码在每一章节中都重新写一遍。

成功利用了 rust 语言的 workspace 和 crate，使得每一个章节是一个预期目标不同的 package，做不同的章节的实验的时候，只需要封装一个 crate 到对应的 package 中，然后在运行的时候运行指定的 package 就可以了，不需要像之前一样做不同章节的课后实验需要到不同的分支中写代码。

并且实现了系统调用接口的模块化，即系统调用的分发封装到一个 crate 中，这个 crate 就是 `syscall/src/kernel/mod.rs`。使得添加系统调用的模式不是为某个 `match` 增加分支，而是实现一个分发库要求的 `trait` 并将实例传递给分发库。

### 2.3.2 改进方向

首先，针对目前实验室的版本只是实现了操作系统内核模块化的基本功能的问题，增加了每个章节的模块化完成之后的单元测试。增加了单元测试之后，我们能够进行小而集中的测试能够在隔离的环境中一次测试一个模块或者测试模块的私有接口，能够更加方便的检测出来是代码的哪个模块甚至是哪个函数出了问题。

继续完善 `rcore` 操作系统内核的模块化，对已经存在的模块化进行自己的改进，可以在分析了已有模块接口的优缺点的基础上，只进行模块接口定义的改进；也可以对模块的实现方法进行改进，并且通过测试来分析自己的实现与已有模块的实现相比有哪些优缺点。

### 2.3.3 各章节单元测试

在 Rust 中一个测试函数的本质就是一个函数，只是需要使用 `test` 属性进行标注或者叫做修饰，测试函数被用于验证非测试代码的功能是否与预期一致。在测试的函数体里经常会进行三个操作，即准备数据/状态，运行被测试的代码，断言结果。

在各章节的 `src` 目录下，每个文件都可以创建单元测试。标注了 `#[cfg(test)]` 的模块就是单元测试模块，它会告诉 [Rust] 只在执行 `cargo test` 时才编译和运行代码。在 `library` 项目中，添加任意数量的测试模块或者测试函数，之后进入该目录，在终端中输入 `cargo test` 运行测试。

但是，我们的内核是一个 `no_std` 应用，没有标准库，而 rust 的测试框架会隐饰的调用 `test` 库，而 `test` 库使依赖标准库的，所以我们的内核无法使用默认框架。

因此我们需要自定义测试框架，并且幸运的使 Rust 支持通过使用不稳定的自定义测试框架来代替默认的测试框架，而且该功能不需要额外的依赖库，因此在我们的 `no_std` 内核中可以使用。它的工作原理是收集所有标注了 `#[test_case]` 属性的函数，然后将这个

测试函数的列表作为参数传递给用户指定的 runner 函数。因此，它实现了对测试过程的最大控制。

完成了自定义测试框架之后，我们现在在 `_start` 函数结束之后进入了一个死循环，在每次运行完 `cargo test` 后需要我们手动退出 QEMU，非常麻烦。幸运的是 QEMU 支持一种名为 `isa-debug-exit` 的特殊设备，它提供了一种从客户系统（guest system）里退出 QEMU 的简单方式。我们可以通过将配置关键字 `package.metadata.bootimage.test-args` 添加到我们的 `Cargo.toml` 中来达到目的。

要在控制台上查看测试输出，我们还需要以某种方式将数据从内核发送到宿主系统，发送数据的一个简单的方式是通过串行端口，QEMU 可将通过串口发送的数据重定向到宿主机的标准输出或是外部文件中。并且为了查看 QEMU 的串行输出，我们需要使用 `-serial` 参数将输出重定向到 `stdout`。同时为了在 `panic` 时使用错误信息来退出 QEMU，我们可以使用条件编译在测试模式下使用不同的 `panic` 处理方式。由于我们使用 `isa-debug-exit` 设备和串行端口来报告完整的测试结果，所以我们不再需要 QEMU 的窗口了。我们可以通过向 QEMU 传递 `-display none` 参数来隐藏弹出的窗口。

完成上述功能后我们已经有了一个可以工作的测试框架了，我们可以为我们的 VGA 缓冲区实现创建一些测试。

## 2.4 实施技术方案所需的条件

Ubuntu 虚拟机

Rust 版本管理器 `rustup` 和 Rust 包管理器 `cargo`

QEMU 7.0 模拟器

VSCode

## 2.5 存在的主要问题和关键技术关键

针对现有的 rcore 操作系统内核模块化，提出了针对每一个章节模块的独立测试增加单元测试用例和在用户态对每一个模块进行单元测试的优化方向，。

技术关键：rust 的 workspace、crates 和 traits

## 2.6 预期能够达到的研究目标

### 软件成果

完成当前出 rcore 操作系统内核的模块化改进，使后一章节能够利用前一章节的成果。

### 论文成果

完成一篇论文

## 3. 课题计划进度表

12.23-12.30	了解课题研究内容，撰写开题报告
12.30-1.10	实现现有的模块化 rcore 内核
1.10-1.30	完成各章节内核态的单元测试
1.30-2.28	完成各章节用户态的单元测试
2.28-3.20	分析各模块接口的优缺点后修改模块的接口
3.20-4.30	分析改进各模块的实现
5.1-6 月初	完成毕业论文，参加答辩

## 4. 参考文献

[1]杨德睿 单核环境的模块化 Rust 语言参考实现

[2]guomeng.2021 年操作系统的商业化应用 国内操作系统现状与前景分析.//2021 年 10 月 20 日中研网

[3]洛佳 使用 Rust 编写操作系统（四）：内核测试.//2019 年 11 月 07 日知乎