YatSen OS: 一个支点撬动操作系统大山

——教学操作系统及实验设计

张钧宇 陈鹏飞 (指导老师)
Nelson Cheung Pengfei Chen(Adivisor)
zhangjunyu@nelson-cheung.cn chenpf7@mail.sysu.edu.cn
中山大学操作系统实验课程组

摘　要

“给一个支点，可以撬动操作系统这座大山”，操作系统的构建不再是遥不可及的事情。目前，rust、ARM、risc-v 以及国产芯片等新型编程语言和新的 CPU 架构的兴起给原有的操作系统方案带来了极大的挑战，亟需提出一种和操作系统发展相同步的新方案。经过细致而全面的调查，本项目认为完全颠覆旧方案，进行实验课程改革的方式不仅工作量巨大，而且在短时间内无法完成。因此，作为折中方案提出了“以点带面”的发展战略。首先在使用 C/CPP + x86 的实验模式下提出一种新方案。然后，以此为支点，可以并行地开展“64 位操作系统”、“rust + arm + 树莓派”，“rust + risc-v”等实验方案的探索，从而全面地推动中山大学操作实验课程改革。在完成“支点”的过程中，本项目提出了一种简明、全面的实验方案设计原则。在设计原则的指导下，编写了一套以递进演变方式叙述的实验教材，自底向上介绍了如何从零开始编写操作系统。同时，该方案已经应用到 2021 年中山大学操作系统实验课程中，并收获学生的一致好评。相比较于其他两个使用方案（16 位模式操作系统）和使用 ucore 的方案，学生对操作系统概念、操作系统开发和操作系统设计等重要知识的理解程度均有明显提高。通过提出新方案完成了中山大学操作系统实验课的初步改造方案。

关键词: 操作系统实验，保护模式，实验教材

1　引言

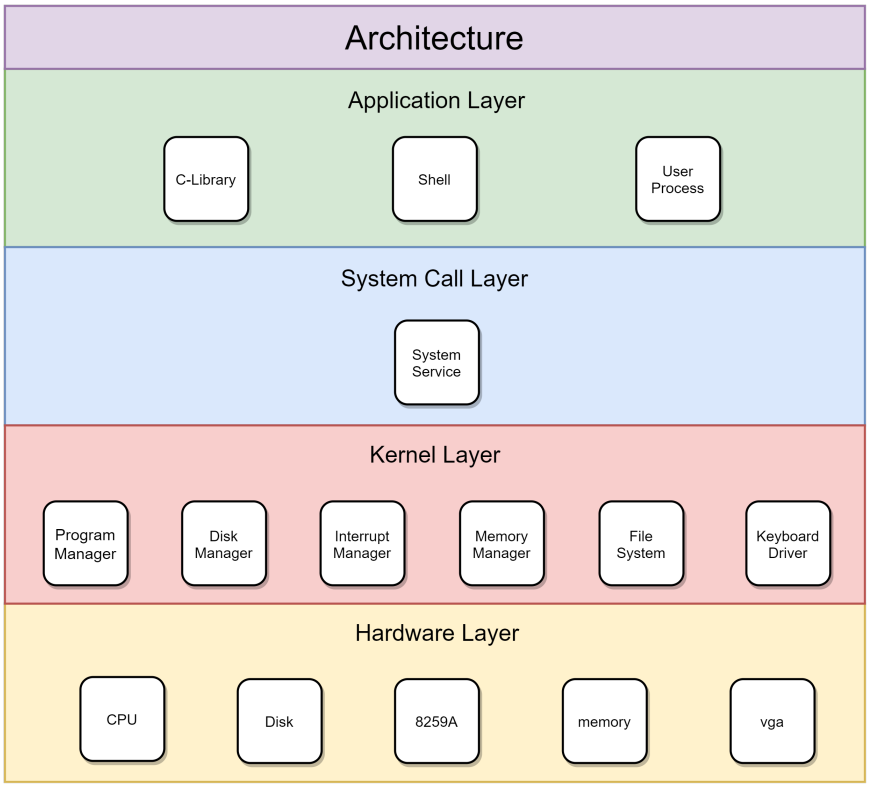
一直以来，从零开始编写操作系统是中山大学操作系统课程的一大特色。但是，随着操作系统的发展，rust，risc-v 和 ARM 等新技术开始不断涌现。在深入地调查目前操作系统发展现状后，本项目认为需要革新原有的操作系统实验方案，使方案能够保持和操作系统的现状相一致。这样的改革并不是一蹴而就，需要的工作量巨大。同时，改革并不是对旧方案的全盘否定，而是继承和发展。在分析和比较旧方案和操作系统现状后，本项目做了一个折中。首先使用 C/CPP + x86 的实验模式下提出一种新方案。新方案保留了中山大学一以贯之的特色——从零开始编写操作系统。但在此基础上，提出了一种简明、全面的实验方案设计原则。在设计方案的指导下，独立自主地编写了 32 位保护模式操作系统代码。基于本项目的代码，编写了一套以递进演变方式推进的的实验教材，全方面介绍了从零开始编写操作系统的知识。教材分为两部分，示例代码和指导材料。在示例代码中以递进的方式给出操作系统的一种简单的实现方式。指导材料分

为九章，每一章的结构为实验概述、实验要求、实验内容和课后思考题。从第零章到第八章，操作系统实验覆盖了 MBR、bootloader、x86 汇编、实模式、保护模式、中断、内核线程、用户进程、信号量和分页机制等主要的操作系统概念。同时，给出了关于新方案的使用方法。

目前，新方案已经成功地应用于 2021 年中山大学操作系统实验，并收集了大量的学生反馈。从纵向看，新的实验方案收获学生的一致好评。从横向看，相比较于其他方案，学生在完成新的方案后，其对操作系统概念、操作系统开发和操作系统设计等重要知识的理解程度均明显提高。

综上所述，基于实现的教学操作系统以及实验指导，本项目圆满完成了中山大学操作系统实验课的初步改造方案。

图 1: 操作系统架构图。



2 相关工作

ucore(13) 是清华大学的操作系统实验课程。ucore 以相对较少的代码实现了一个较为完备的操作系统，用清华大学陈渝老师话说，就是：“麻雀虽小五脏俱全。”在 ucore 的实验方案中，学生需要在 ucore 的代码框架下分析代码的执行逻辑或实现某些特定的函数。但是，本项目认为，这样的模式存在一些不足。例如在 ucore lab1 中，学生需要学习大量的知识，如 ELF 模式、实模式和保护模式、汇编和 C 等。学生还需要掌握许多工具如 gdb、qemu 等。这无疑会加剧学生对操作系统实验的畏难情绪。除此之外，ucore 的模式缺少一些变化，在结果评测方面存在漏洞。例如每一届学生的 ucore 的实验练习都是固定的，这就导致了前面完成实验的学生为后面的学生提供了充分的答案。

《操作系统真象还原》(3) 以一种通俗易懂的方式介绍了如何从零开始编写操作系统。为了满足通俗易懂的特性，作者极力以一种口语化的语言展开叙述。这样反而会导致语言繁琐，使读者找不到重点。而且，作者借鉴了早期 Linux 的代码，代码中不乏与中山大学程序设计课程相违背之处，如代码中会存在许多 goto 语句。

《x86 汇编语言：从实模式到保护模式》(2) 除了介绍如何从零开始编写操作系统外，还给出了许多课后习题来帮助读者理解。相比于《操作系统真象还原》，《从实模式到保护模式》的叙述方式更加简练、准确。但是，该书始终使用汇编语言来实现操作系统，不涉及到任何 C/CPP 等高级语言的内容。

《Orange's 一个操作系统的实现》(4) 同样是一本从零开始编写操作系统的教材。该书和《操作系统真象还原》恰恰相反，其叙述方式极具作者个人特点。而且，该书会使用许多编程语法上的技巧，这对初学者来说是不必要的。

xv6(14) 开创了操作系统实验的先河。自此，国内外许多优秀的操作系统的实验设计均借鉴于 xv6。在 xv6 的课程中，每个 lab 相对独立，其要求学生在系统的某个模块上，添加修改新功能。同时，xv6 提供了自动测评系统，可以让学生自行检查代码实现的正确性。但是，和 ucore 一样，xv6 并不是一本从零开始介绍操作系统实验的教材。学生对非自己实现的代码细节的感知是完全模糊的，所以很难真正理解它。

3 实验方案

3.1 想法来源

在本学期之前，学生使用 C 在 x86 架构下从零开始编写一个在 bochs 上模拟运行的 16 位操作系统。从零开始编写操作系统一直是中山大学操作系统实验课程一以贯之的传统和特色。为什么这样一份实验方案多年来受到老师和学生的追捧呢？主要有以下原因。

1. **契合实际**。在过去的十多年中，CPU 一直是 x86 架构的天下，历经了从 16 位的实地址模式到 32 位保护模式的变化。因此，以 x86 架构作为实验平台无疑是契合学生平时的使用实际的。同时，在 x86 架构下使用的术语如实模式、保护模式和 8259A 芯片等都是学生司空见惯的。因此，选择 x86 架构能够将学生对于硬件的畏难情绪降到最低。
2. **易于上手**。学生曾在大一接受了为期一年的 C 和 CPP 的深入学习，对于编写操作系统所必备的 Linux 环境下的工具的使用、C 语言的用法和项目管理等知识都有所了解。同时，学生们在大二上学习了计算机组成原理课程。在该课程中，学生们学习了 x86 汇编语言，了解了 C 语言和汇编语言的联系和区别，还使用 x86 汇编语言编写过程序。这些知识都为学生们在操作系统实验课程中奠定了坚实的基础，从而使得实验易于上手。
3. **收获感强**。从零开始编写操作系统并不是一件容易的事情，往往能够耗费学生许多精力。但是，学生们的收获比较显著。在实验课中，学生们通过具体的、独立的和深入的实践过程学习到许多知识，如计算机的启动过程、内存管理、进程管理和文件系统等。这些从实践中学习到的知识能够帮助学生加深对理论知识的理解。同时，许多学生会将自己的 Linux、Windows 和 MacOS 等功能“仿制”到自己的操作系统中，甚至编写了小游戏，从而形成了具有自己特色的操作系统。

天若有情天易老，人间正道是沧桑。在原有操作系统实验方案应用的十余年间，操作系统的发展也发生了翻天覆地的变化。例如，x86_64 架构主宰了 CPU 的话语权、arm 和 risc-v 也在异军突起、去 C/CPP，使用 rust 来编写更加安全的内核正见诸于计算机系统的各大顶会……因此，本项目作者愈发感觉到，改革操作系统实验方案，使其与时俱进的重任迫在眉睫。

为此，结合当下操作系统发展实际、学生的先修知识的广度和深度以及其他高校的操作系统实验设计等因素，重新审视了操作系统实验方案。在对以往的方案做出了批判性继承的基础上，提出了新的实验方案，新方案主要特点如下。

1. **从零开始**。坚持认为，从零开始编写操作系统不仅能够延续旧有的实验方案的传统，而且能够帮助学生加深对于操作系统的各个组成部分的理解，从而在学生的知识体系中建立起关于操作系统的全貌。因此，在新方案中，从零开始编写操作系统的原则依然是新方案的出发点和落脚点。
2. **代码自研**。本项目的操作系统代码是在《操作系统真象还原》和《从实模式到保护模式》的基础上开发的，但相较于二者做了几乎完全的改变。值得注意的是，代码并未参考任何高校的代码，如 xv6 和 ucore 等，也并未参考 Linux 的实现代码。同时，本项目的代码量只有 2000 余行。
3. **以点带面**。在对工作量和实验难度进行评估后，本项目认为在短时间内无法将许多新的内容引入到操作系统实验中。因此，采用了以点带面的方式来推进操作系统实验更新。首先在原有的实验的基础上作出改变，例如彻底抛弃实模式、引入 Linux + qemu + gdb + vscode 的统一开发调试环境、实验教材的编写等。由于本项目只是在原来的基础上迈出了试探性的一步，这样的改变称之为“点”。在“点”的工作中，本项目已经基本确定了操作系统实验新方案的设计思路，并逐步积累了自己的实践经验。在“点”确立的实验框架下，可以并行地开展“64 位操作系统”、“rust + arm + 树莓派”，“rust + risc-v”等实验方案的探索。由于可以并行地对多种方案进行探索，这样的探索称之为“面”。因此，可以说这个实验方案的改革过程称为“以点带面”。
4. **保护模式**。原有方案为学生提供了两种实验选择。学生可以在 16 位的实模式操作系统或 32 位保护模式操作系统中选择。相较于 16 位实模式，32 位保护模式需要学习的知识比较多。例如，实模式的 BIOS 中断囊括了读写磁盘，移动光标，输出输出等内容，学生直接调用即可。但是，在保护模式下，BIOS 中断无法使用，学生需要了解如何通过读写 IO 端口，编写系统调用来完成上述内容。这部分内容之多足以让学生望而却步。于是，绝大部分学生选择了编写实模式操作系统。这样就导致了学生无法通过实践了解到二级分页机制的实现，IO 端口，特权级保护等内容。这是旧方案的一大弊病，因为本项目认为实验的目的是通过具体的实验来帮助学生理解抽象的理论。结合了理论课教材后，本项目认为只有通过保护模式才能让学生实现理论课教材的大部分内容。因此，在新方案中，本项目彻底地抛弃了实模式，学生只会在 32 保护模式下完成实验。
5. **源码调试**。在原有方案中，使用 bochs 来作为开发和调试环境。但是，bochs 有一个最大的弊端是不支持源码级别的 debug。即使用 C 语言来编写操作系统，bochs 反汇编出来的代码是 C 语言对应的汇编代码。这给一些对汇编代码、C 和汇编等知识不熟悉的学生造成了极大的困扰。而且，直接使用汇编代码来 debug 大大降低了 debug 的效率。因此，在新方

案中，通过引入 qemu + gdb 实现了 16 位汇编代码，32 位汇编代码和 C/CPP 代码的源码级 debug，从而降低了学生的 debug 难度。

6. **实验教材**。新方案和原方案最大的不同点是为实验编写了一套实验教材，而原方案只是简单地为每次实验提供了实验需求。学生自己摸索是一件好事，但这不应该成为一门课程的设计主线。一门好的课程是能够给予学生以充分的路线指导，然后才能让学生沿着路线自行摸索。由于操作系统是硬件和软件之间的一层中间件，开发操作系统必然会涉及到许多硬件知识。如果不了解硬件是怎么工作的，那么学生的程序必定会出现不可解决的 bug。在以往的实验中，教师和学生将大量的时间耗费在找出因为先修知识不够而产生的 bug 上。因此，本项目为新方案编写了一套相对完整的教材。教材旨在使用最简单的方式来为学生阐明操作系统实验中许多精彩的设计思路，提高学生的学习效率。

图 2: 代码统计，总共代码只有 2000 余行。

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                             13            339            126           1529
Assembly                         4            124             38            459
C/C++ Header                    20            117            109            421
make                             1             17              1             50
Markdown                         1              4              0             17
-------------------------------------------------------------------------------
SUM:                            39            601            274           2476
-------------------------------------------------------------------------------
```

3.2 设计原则

新方案和旧方案，乃至和其他高校的操作系统实验设计方案相比，其最大的不同在于提出了一种简明、全面的操作系统实验设计原则，对设计原则的解释和理解如下。

- **简明的**。**“简明的”指的是设计一切从简**。由于实验方案是从零开始编写操作系统，操作系统中的中断、线程管理和进程管理都需要学生自行编写。因此，学生需要了解的东西非常多。如果不能够为学生删去一些不必要的东西，提供一种简明的指导方案，那么学生就容易迷失在知识的海洋中，从而对实验产生较大的抵触心理。本项目认为，大道至简，操作系统的设计亦是如此。每一部分的设计都会考虑再三，删去许多与主题相关性不大的内容，最终以一种最简单的方式来实现最全面的编程逻辑。例如应该让学生尽量的使用他们最熟悉的语言来做实验。因此，希望尽量简化汇编代码在编写操作系统中的数量，尽可能地使用高级语言 C/CPP。同时，认为内联汇编是不必要的。实际上，任何通过内联汇编实现的功能都可以通过汇编提供的函数接口实现。注意到内联汇编也有自己的一套语法，这也会给学生带来额外的学习成本。因此，不会介绍和使用任何内联汇编。
- **全面的**。**“全面的”指的是内容包罗万象**。实验方案囊括了操作系统理论中许多重要概念，其中包括线程/进程管理、内存管理、文件系统、I/O 管理、中断机制和系统调用机制等。通过完成实验，学生可以借助于具体的实践来理解抽象的理论。同时，新的实验方案不仅讲解操作系统重要概念的实现，还讲述了 toolchain 的使用和 debug 技巧。实验课后配有许多相应的课后思考题，学生借助于课后思考题可以理解实验中的重点内容。除此之外，通过

图 3: 目前的物理内存划分。其中，以 * 表示的地址表示任意地址，需要根据实际内存大小来确定。

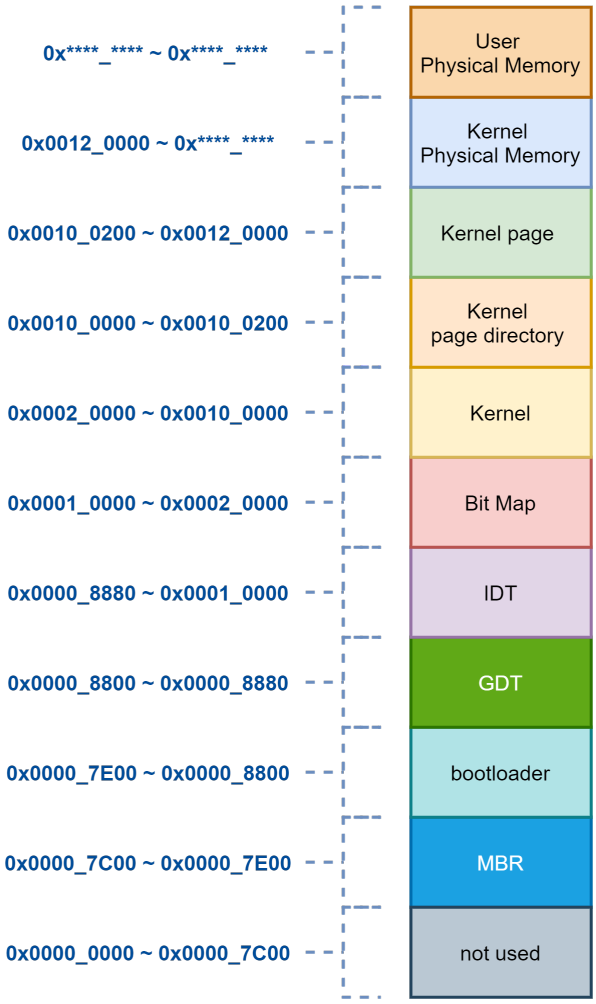
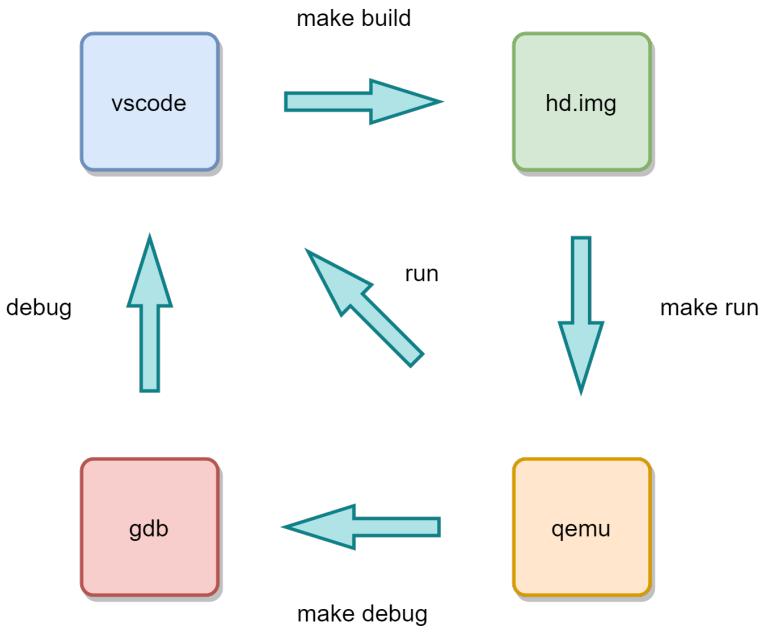


图 4: 闭环的实验环境。将学生的开发、编译和运行统一到 vscode 下，形成了闭环。学生在 vscode 上编写代码，使用 makefile 中给出的命令，在 vscode 的 Terminal 下分别使用 make build 来编译、make run 来调用 qemu 运行、make debug 来调用 gdb 进行源码级的 debug。学生根据运行结果和 debug 结果来修改代码。



本教程的学习后，学生将掌握操作系统核心概念，程序的编译执行过程，GCC 编译工具的使用，C/CPP 项目的构建和管理、类 Unix 环境等内容。

3.3 教材结构

实验教材可以划分为两部分，示例代码和指导材料。

由于采用了递进演变的方式来编写材料，示例代码也是递进演变的。一般来说，后面的代码是在前面的代码的基础上增加、删除或修改的某些部分的代码。也就是说，当学生完成到最后一个实验时，示例代码实际上就是一个具有最基本功能的操作系统代码。

示例代码的主要作用是给出操作系统的一种简单的实现方式。在指导材料中，借助于示例代码来讲解如何实现操作系统中的重要概念，以便学生理解实验的实现方式和思路来源。同时，会在课后思考题中设计一些分析示例代码的题目，分析方法包括 gdb 跟踪、测例验证和代码逻辑分析等。通过分析示例代码，学生可以加深自己对操作系统实现的理解。除此之外，还会设计一些个性化的题目。在这些题目中，学生或者可以在示例代码的基础上加入自己的个性化内容、或者自行实现一些功能、或者改进示例代码等。通过个性化的题目，学生可以发散自己的思维并动手实现自己的想法，从而加深自己对实验内容的理解。

指导材料可分为实验概述、实验要求、实验内容和课后思考题。

在实验概述中，按照实验内容的讲述顺序来对实验做一个简要的概括。通过阅读实验概述，学生可以迅速地了解每一次实验需要学习和完成的内容，从而在开始实验前就能够对实验内容产生一个初步的印象。

在实验要求中，针对实验内容中的主要内容来为学生设计实验要求。根据学生的实际情况

来确定实验要求中的题目数量和难度，这些题目是学生需要独立完成并提交结果和报告的。通过检查学生的题目完成情况，既可以即时地对教学成果进行评估，以便调整后续的实验内容，也可以引导学生朝着期望的方向发展。

实验内容是指导材料的主体部分。实验内容的长度被控制在 10,000 到 20,000 字之间。通过控制实验内容的长度，既可以保证实验的进度，又不会让学生一下子接触到大量的信息。正如前面所述，实验内容的编写方式是递进演变式的。逐步分析操作系统的一些重要概念的实现代码，如同搭积木一般，将一个操作系统实现的基本过程以递进的方式呈现在学生面前。以递进演变的方式来教授实验内容是符合学生的认知发展过程的。并不只局限于分析操作系统的实现，还会讲授一些预备知识。在预备知识的讲解中，往往会穿插若干个示例来帮助学生理解这部分的知识。将这些预备知识讲授清楚比单纯地分析示例代码是更重要的。调查后发现，学生不知道如何设计操作系统代码的根源不在于不懂操作系统的概念。事实恰好相反，学生实际上对操作系统的基本概念了解得十分清楚，但由于其他预备知识的缺失，导致了他们不知道如何去实现一个操作系统。而当讲授完预备知识后，学生往往能很快地理解操作系统的代码实现，甚至不用参考代码实现也能根据概念的定义来实现。

课后思考题主要是帮助学生自行理解实验的重点内容。在每一章的最后，精心地设计了 10 到 20 道课后思考题，以帮助学有余力的学生加深自己对于实验内容的理解。在设计课后思考题的过程中，划分了每一章的重点内容，然后根据这些重点内容来出题。也就是说，学生通过完成课后思考题就能够检验自己对实验中每一个重点内容的理解情况。

3.4 内容安排

结合理论课的教学思路，将从零开始实现操作系统的整个过程划分为一个个重要的阶段。根据重要阶段的划分方式编写了实验教材中的每一章，每一章的内容如下。

在第零章中，将熟悉现有 Linux 内核的编译过程和启动过程，以及在自行编译内核的基础上构建简单应用并启动。同时，利用精简的 Busybox 工具集构建简单的 OS，熟悉现代操作系统的构建过程。此外，还会尝试如何使用 gdb 实现内核远程调试。

在第一章中，将会学习到 x86 汇编、计算机的启动过程、IA-32 处理器架构和字符显存原理等知识。根据所学的知识，能自行在 MBR 下写入 16 位的汇编程序，然后让计算机在启动后加载运行。同时，将学习使用 gdb 来调试程序的基本方法。

在第二章中，介绍了如何从 16 位的实模式跳转到 32 位的保护模式，然后在平坦模式下运行 32 位程序。同时，将介绍如何使用 I/O 端口和硬件交互的基本方法，为后面保护模式编程打下基础。

在第三章中，首先介绍一份 C 代码是如何通过预编译、编译、汇编和链接生成最终的可执行文件。接着，提出了一种 C/CPP 项目管理方案。在做了上面的准备工作后，开始介绍 C 和汇编混合编程方法。然后，介绍了保护模式下的中断处理机制和可编程中断部件 8259A 芯片。最后，通过编写实时钟中断处理函数来将本章的所有内容串联起来。

在第四章中，将会学习到 C 语言的可变参数机制的实现方法。在此基础上，会揭开可变参数背后的原理，进而实现可变参数机制。实现了可变参数机制后，将实现一个较为简单的 printf 函数。接着，开始学习内核线程的实现。最后，会实现基于时钟中断的时间片轮转 (RR) 调度算

法。

在第五章中，首先使用硬件支持的原子指令来实现自旋锁，自旋锁将成为实现线程互斥的有力工具。接着，使用自旋锁来实现信号量，最后使用自旋锁和信号量来给出两个实现线程互斥的解决方案。

在第六章中，首先学习如何使用位图和地址池来管理资源。然后，将实现在物理地址空间下的内存管理。接着，将会学习并开启二级分页机制。在开启分页机制后，将实现在虚拟地址空间下的内存管理。

在第七章中，首先会简单讨论保护模式下的特权级的相关内容。然后，介绍了系统调用的概念和如何通过中断来实现系统调用。在实现了系统调用后，介绍了用户进程的创建和管理的基本方法。最后，介绍了 fork/wait/exit 的一种简洁的实现思路。

在第八章中，给出了若干个课程项目并不再提供示例代码。

对于一些和操作系统本身无关但和开发操作系统有关的内容，如 IDE 的选择、使用 gdb 来进行源码级的 debug、使用 make 来进行项目管理等，将其一并放入到附录中。

3.5 使用方法

为了便于理解新方案的整体设计，在这里给出中山大学操作系统实验课程组关于新方案的使用方法。

会在课前根据每一章的实验内容制作 PPT，然后在上实验课的时候花费 0.5 到 1 个小时来为学生讲解其中的重点内容。讲解完毕后，剩下的实验课时间就留给学生自行阅读并消化实验教材中的内容，教师和助教负责答疑。在课后，或者从课后思考题中选取若干道习题并作修改，或者根据学生的学习情况来针对性地命题，学生需要在课后完成并提交这些题目。

值得注意的是，正因为给出的是一种简明的实现思路，就可以为学生留出更多的发挥空间。由于不会限定学生必须完成的内容，在后续的实验中，每一次为学生颁布的习题都可以不同。这就为教师的教学提供了极大的灵活性。

4 方案效果

4.1 横向比较

在 2021 年操作系统实验课程中，有另外两个班级的实验方案与本方案并不相同。一个使用的是旧方案，一个使用的是 ucore。

对于使用旧方案的班级，之前陈述的问题依然存在，并未解决。绝大部分学生依然选择只在 16 位实模式下完成实验，而不会去学习 32 位保护模式。根据这些班级的学生反映，他们还存在实验环境不统一、实验方案陈旧和缺少一份指导材料的问题。而这些问题均在新的方案中被重点解决了。

对于使用 ucore 的班级，由于是第一次使用 ucore，他们对 ucore 的认识还不够充分。同时，这个班级选择了先讲述进程管理，再讲述内存管理。而这与 ucore 的实验顺序相违背，从而导致了理论和实验不统一的现象。最重要的是，学生都能够在网上找到以往完成了 ucore 的学生的答

案，部分学生只是做了简单的复制粘贴，并未做深入的思考。由于 ucore 一开始就把整个操作系统的基本框架搭建起来了，部分学生反映内容较多，理解起来难度过大。

综合了学生的评价后，可以发现，相比较于其他两个使用旧方案和使用 ucore 的班级，学生在完成了新方案后，其对操作系统概念、操作系统开发和操作系统设计等重要知识的理解程度均明显高于另外两个班级。

4.2 纵向比较

在新方案应用的班级中，实验方案受到了学生的一致好评，这里给出了一些学生的评价 (5, 6, 7) 和报告 (8,9,10,11) 截图。

图 5: 学生评价。

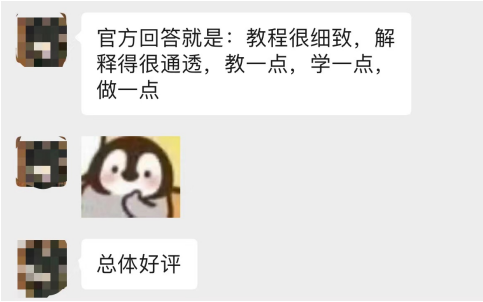


图 6: 学生评价。

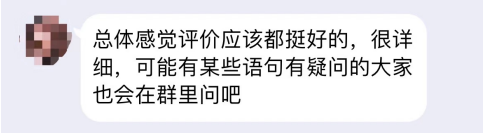
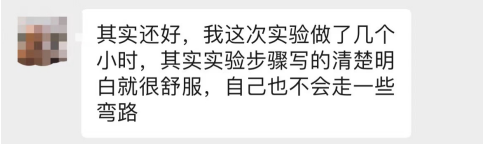


图 7: 学生评价。



5 未来发展

前面已经提到，新方案起到的只是“点”的作用。新方案最大的贡献在于为后面的方案的提出奠定了基础。例如，通过 Linux、vscode、make、gdb、qemu、gcc 和 g++ 为学生统一了从开发到编译，从编译到运行，从运行到 debug 的道路。但并不会止步于此，最终的目的还是希望能够通过这个“点”将“面”带出来。在下一代的方案中，将会彻底抛弃 C/CPP + x86 这种操作系统实验模式，然后引入 rust，在 risc-v，arm 甚至是开发板上构建操作系统实验。应当注意到，下一代方案并不是对当前方案的完全推翻。恰恰相反，当前方案提出的统一实验环境、设计原则和教材结构等重要思想依然会被下一代方案所继承和发展。

图 8: 学生报告。

四、 实验心得

这次实验我掌握了 C 语言的可变参数机制的实现方法，并且学会了 printf 函数的实现。此外，还掌握了内核线程的实现，理解了线程之间是怎样切换的。

1. 在 Assignment 1 中我是在示例代码的基础上增加了 printf 的一些其他功能，但因为时间有限我也只实现了一部分，在实现%u 的功能时我的想法是如果要打印的数字为负数，那么求出负数的补码，然后补码的最高位不再是符号位，根据这样，求出它的十进制，那么我们就可以得到自己想要的结果了。但后来发现不用这么麻烦，因为示例代码中进度转换函数的声明是这样的：int itos(char *numStr, uint32 num, uint32 mod,byte ch)，uint32 就是 unsigned int，num 是我们要转换的数，就是负数在传进来时就已经转换成我所需要的数，因此不用像我想的那么麻烦。
2. 在 Assignment 2 中，添加优先级后，在实现的过程我加入了优先级抢占，但我的优先级抢占的程序设计的有问题：就是在线程调度函数 schedule 中如果被切换的线程还在运行的，那么就要把它加入就就绪队列，加入就绪队列的位置要根据优先级来确定。但我在判断它是 RUNNING 并将它初始化为 READY 直接根据优先级加入就绪队列。但是后面要替代这个线程进入处理器执行的线程还未取出，那么这就造成了一个问题。我们知道第一个线程是不返回到 program_exit 的，也就是它的状态要么是 READY 要么是 RUNNING（这是按照我们的程序设计而言）。假设在第一个线程之后创建的线程的优先级都不高于第一个线程，那么第一个线程在和其他线程进行切换时，因为第一个线程的状态为 RUNNING，所以重新进入就绪队列且排在就绪队列的最前面（因为它的优先级最高）或排在和它优先级相同的线程的后面，除了第一个线程外，其他线程执行完后都会被释放，到最后，就会出现一种情况，第一个线程和自己（因为目前只有它的优先）进行切换，其他比它优先级低的线程一直在等待，产生饥饿。这个问题是我在做 Assignment 的时候忽然意识到的，但在 Assignment2 的时候没意识到，所以最后把优先级抢占去掉了，实现的非抢占的优先级。
3. 在 Assignment 3 中，我对如何用 gdb 调试有了深刻的了解，但这个调试过程还是很繁琐，要多次调试，查看寄存器的值等等，还很容易一不小心错过自己所需要的，还要重来才行。但是，不得不说通过这个调试过程，我对线程的运行机制有了一个很清晰的认知。
4. Assignment 4 我只实现了两个算法，所以还好。

总之，我觉得这次实验收获非常大。

图 9: 学生报告。

总结

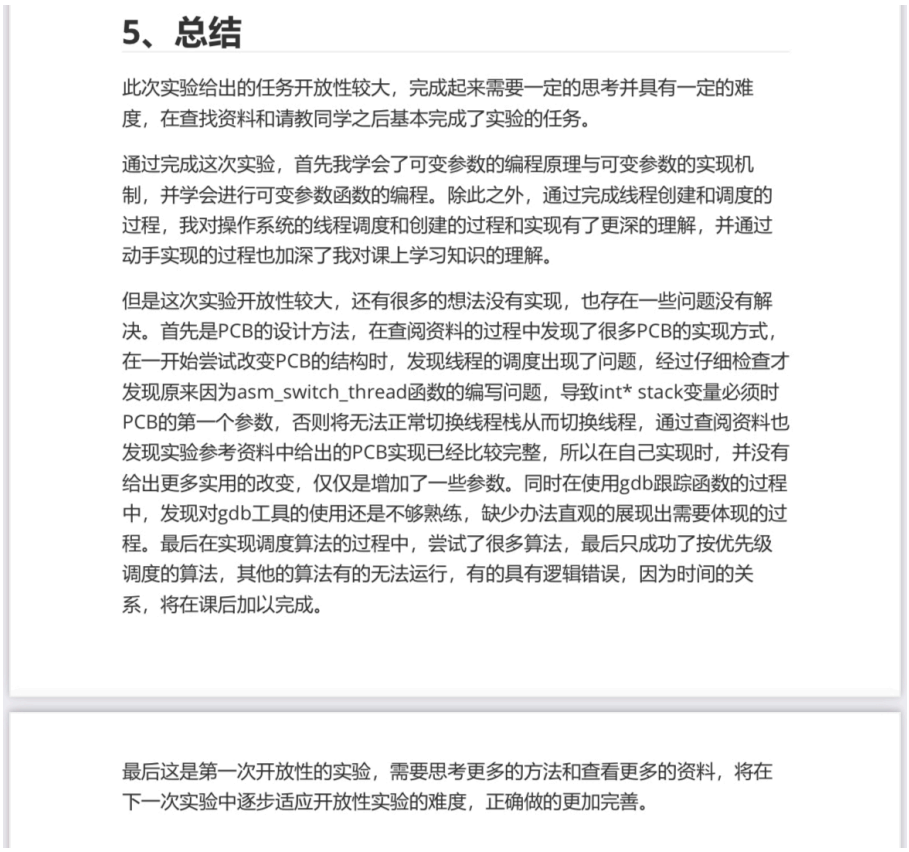
这一次实验比上一次实验的开放度高了很多，没有以前的"傻瓜教程"了，所以许多地方都需要自己进行摸索。

一开始做printf的时候，感觉很多地方只需要稍微修改一下，也不是那么困难，但是在过程中还是遇到了一些bug（前面的实验过程中有具体讲到）。其实很多bug只是细节上的问题，但是在对细节问题的处理中我对这些知识的理解也更深刻了。我学习了可变参数函数的用法，并且以此为依据设计了自己的printf。

之后在对PCB和调度算法的设计中，我对线程的创建、线程的调度都有了一个更深的认识和理解，在自己设计调度算法的过程中，更是对线程调度的原理有了一个更深刻的掌握。

下次实验继续努力！

图 10: 学生报告。



5、总结

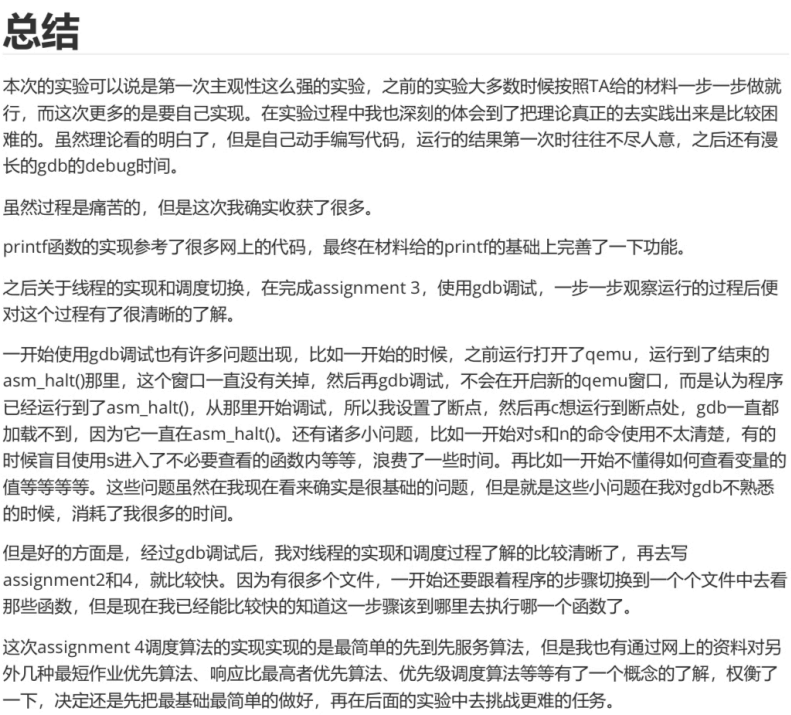
此次实验给出的任务开放性较大，完成起来需要一定的思考并具有一定的难度，在查找资料和请教同学之后基本完成了实验的任务。

通过完成这次实验，首先我学会了可变参数的编程原理与可变参数的实现机制，并学会进行可变参数函数的编程。除此之外，通过完成线程创建和调度的过程，我对操作系统的线程调度和创建的过程和实现有了更深的理解，并通过动手实现的过程也加深了我对课上学习知识的理解。

但是这次实验开放性较大，还有很多的想法没有实现，也存在一些问题没有解决。首先是PCB的设计方法，在查阅资料的过程中发现了很多PCB的实现方式，在一开始尝试改变PCB的结构时，发现线程的调度出现了问题，经过仔细检查才发现原来因为asm_switch_thread函数的编写问题，导致int* stack变量必须时PCB的第一个参数，否则将无法正常切换线程栈从而切换线程，通过查阅资料也发现实验参考资料中给出的PCB实现已经比较完整，所以在自己实现时，并没有给出更多实用的改变，仅仅是增加了一些参数。同时在使用gdb跟踪函数的过程中，发现对gdb工具的使用还是不够熟练，缺少办法直观的展现出需要体现的过程。最后在实现调度算法的过程中，尝试了很多算法，最后只成功了按优先级调度的算法，其他的算法有的无法运行，有的具有逻辑错误，因为时间的关系，将在课后加以完成。

最后这是第一次开放性的实验，需要思考更多的方法和查看更多的资料，将在下一次实验中逐步适应开放性实验的难度，正确做的更加完善。

图 11: 学生报告。



总结

本次的实验可以说是第一次主观性这么强的实验，之前的实验大多数时候按照TA给的材料一步一步做就行，而这次更多的是要自己实现。在实验过程中我也深刻的体会到了把理论真正的去实践出来是比较困难的。虽然理论看的明白了，但是自己动手编写代码，运行的结果第一次时往往不尽人意，之后还有漫长的gdb的debug时间。

虽然过程是痛苦的，但是这次我确实收获了很多。

printf函数的实现参考了很多网上的代码，最终在材料给的printf的基础上完善了一下功能。

之后关于线程的实现和调度切换，在完成assignment 3，使用gdb调试，一步一步观察运行的过程后便对这个过程有了很清晰的了解。

一开始使用gdb调试也有许多问题出现，比如一开始的时候，之前运行打开了qemu，运行到了结束的asm_halt()那里，这个窗口一直没有关掉，然后再gdb调试，不会在开启新的qemu窗口，而是认为程序已经运行到了asm_halt()，从那里开始调试，所以我设置了断点，然后再c想运行到断点处，gdb一直都加载不到，因为它一直在asm_halt()。还有诸多小问题，比如一开始对s和n的命令使用不太清楚，有的时候盲目使用s进入了不必要查看的函数内等等，浪费了一些时间。再比如一开始不懂得如何查看变量的值等等等等。这些问题虽然在我现在看来确实是很基础的问题，但是就是这些小问题在我对gdb不熟悉的时候，消耗了我很多的时间。

但是好的方面是，经过gdb调试后，我对线程的实现和调度过程了解的比较清晰了，再去写assignment2和4，就比较快。因为有很多个文件，一开始还要跟着程序的步骤切换到一个个文件中去看那些函数，但是现在我已经能比较快的知道这一步骤该到哪里去执行哪一个函数了。

这次assignment 4调度算法的实现实现的是最简单的先到先服务算法，但是我也有通过网上的资料对另外几种最短作业优先算法、响应比最高者优先算法、优先级调度算法等等有了一个概念的了解，权衡了一下，决定还是先把最基础最简单的做好，再在后面的实验中去挑战更难的任务。

为了简化操作系统实验难度，本项目提出了一种简明的、优雅的和全面的设计原则。设计原则决定了给出的操作系统并不是一个功能十分完善的操作系统，而只是一个包含了操作系统的许多重要概念的 toy OS。但这样的结果也是符合课程期望的。因为实验的根本目的在于通过具体的实践来帮助学生理解抽象的理论。学生的学习成果并不需要在当下完全展现，而是能够通过完成 toy OS 来熟读操作系统重要概念、了解开发操作系统的基本方法、学习 Linux 下的开发工具等。通过实验课学到的知识，学生能够在后面的科研、工作中，遇到开发操作系统的问题，能够将这部分知识做出迁移和发展。这才是实验课的最终目的。

实验评价体系还没有完全建立，在下一代方案中，需要构建出一套自动化的评价系统。

6 结论

在刚刚过去的一年半中，综合比较了旧有的实验方案和操作系统发展现状之间的差距，从而在旧方案的基础上提出关于中山大学操作系统实验课改革的设想。为了减少目前的工作量，本项目提出了“以点带面”的发展战略，在使用 C/CPP + x86 的实验模式下提出一种新方案。在新方案中，提出了一种简明、全面的实验方案设计原则。在设计原则的指导下，编写了一套以递进演变方式叙述的实验教材，自底向上介绍了如何从零开始编写操作系统。同时，成功地将新方案应用到 2021 年中山大学操作系统实验课程中，并收获学生的一致好评。相比较于其他两个使用旧方案和使用 ucore 的班级，学生在完成新方案后，其对操作系统概念、操作系统开发和操作系统设计等重要知识的理解程度均明显提高。

参考文献

[1] kip R Irvine. Intel 汇编语言程序设计第五版. 电子工业出版社.

[2] 李忠, 王晓波, 余洁. x86 汇编语言：从实模式到保护模式. 电子工业出版社.

[3] 郑钢. 操作系统真象还原. 中国工信出版社.

[4] 于渊. Orange’ s 一个操作系统的实现. 电子工业出版社.

[5] 蒋静, 徐志伟. 操作系统实验: 原理、技术与编程. 机械工业出版社.

[6] William Stallings. 操作系统: 精髓与设计原理第八版. 中国工信出版社.

[7] 谢煜波.PYOS. 纯 C 论坛

[8] 陆勇. 中山大学. 考试酷

[9] 获取内存容量 https://zhuanlan.zhihu.com/p/35776128

[10] 可变参数函数详解 https://www.cnblogs.com/clover-toeic/p/3736748.html

[11] C 语言-从代码到程序的过程理解 https://www.cnblogs.com/linzworld/p/13690620.html

[12] x86 汇编 (Intel 汇编) 入门 https://www.cnblogs.com/jiftle/p/8453106.html

[13] ucore https://chyyuu.gitbooks.io/ucore_os_docs/content/

[14] xv6 https://pdos.csail.mit.edu/6.828/2012/xv6.html