北京邮电大学 操作系统课程设计项目文档	产品名称	产品版本	密级
	MiniOS	V1.0	公开
	提交日期： 2020 年 5 月 29 日		共 7 页

MiniOS Ver 1.0

操作系统课程设计

第二次会议纪要

学　　院：计算机学院

专　　业：计算机科学与技术

班　　级：2017211319 班

指 导 教 师：孟祥武老师

1 项目概述

1.1 工作内容

- **明确项目解决方案与路线**

团队明确解决方案具体内容，并构建系统主要框架。组长安排组员进行分工协作，进行具体内容的详细说明与展开；各成员完成方案的编写，由组长进行汇总后在团队内发布，所有成员仔细阅读并提出意见、统一修改。

- **进行计划跟踪与监督**

按照作业内容与作业时间安排，结合组员时间安排具体的项目开放计划。组员严格执行计划安排，由组长定期跟踪、统一项目进度，实行小组内成员互相监督的合作机制。

- **依据本方案中拟定的开发路线进行系统开发**

详细内容见本方案的“开发路线”部分，该计划根据课程安排由团队共同制定。

- **按要求向老师反馈开发进度与问题**

团队应做到及时与老师沟通开发过程中遇到的关于项目问题，或者是团队在开放过程中遇到的技术难点，做到按要求与老师交流项目开发进度及完成情况。

- **系统产品及时进行受控管理**

对我们的系统产品进行受控管理，根据老师提供的设计要求，以及团队对于操作系统所应具备功能的调研，对我们的需求管理进行及时的更改和完善，确保系统的完整性，规避技术和质量的风险。

- **按要求接受阶段性评审检查与相关文档的提交**
- **项目验收与总结**

1.2 团队成员基本信息与分工情况

本项目的开发团队由北京邮电大学计算机学院的六名三年级本科生组成，所有成员的分工情况经成员个人选择以及组内所有成员的共同商讨与协调，在第一次小组会议后，经过前三周的系统设计与开发阶段组内对各成员的分工进行了调整。当前小组内各成员的基本信息与分工安排如下表所示：

专业/班级	成员姓名	学号	成员分工	备注
计算机科学与技术/2017211319	陈斌	2017211661	系统框架、内核与 Shell	小组组长
	吴桐子	2017211710	文件管理模块	小组成员
	许浩然	2017212193	文件管理模块	
	张炜晨	2017211835	进程管理模块	
	郑苹	2017211934	进程管理模块	
	周雯笛	2017211769	内存管理模块	

其中每位成员需要负责对应模块的代码书写、文档拟定等工作，并与负责其他部分的成员保持密切的联系与交流。小组在将各个模块独立调试完成后进行组装测试，最后小组需要将系统的各部分源代码、文档进行整合，由组长进行进一步的统一管理与审核，并根据最新进度开展小组会议，作出下一步的工作指导与计划安排。此外，系统开发全过程将在老师的指导与评审下进行。

团队在开发过程中遇到的问题可通过互联网、书籍等途径查阅相关资料，也可通过电子邮件或社交应用向老师请求帮助。

2 开发路线

根据本次课程设计的相关要求，小组当前的整体工作与实现目标在于实现一个具有三大基本功能的操作系统模拟程序，该程序需要能够模拟现代操作系统所具备的三部分功能：

① 进程管理；
② 内存管理；
③ 文件管理。

该程序应能够支持在裸机上独立运行，或作为高层应用，能够支持多平台的运行（例如 Windows 操作系统环境、OpenEuler 操作系统环境）。除了程序外，小组还应当为本次完整的开发过程配备相应的说明性文档。

在第一次会议后，小组内部对本系统的开发路线以及设计目标进行了确定，即采用**模拟**的方式，将现代操作系统作为低层支撑架构，在其之上开发出精巧、高效的模拟操作系统程序，该程序的地位与其他应用程序是一致的。模拟操作系统应具备基本的用户交互能力，并将工作重心置于系统内部的具体实现算法（如进程调度算法、内存分配算法等）以及系统功能的完整性、多样性。

经过小组内部的协商，我们将采用 Python 3 作为模拟操作系统的开发语言。Python 语言具有简易、跨平台等优点，使得团队能够更加专注于算法与系统功能的设计。因此，以模拟系统为根本目标、将 Python 作为开发语言、注重算法与系统功能的设计是本项目团队的基本开发路线。

本项目的目标产出及成果的相关信息如下所示：

- **系统名称**
  MiniOS Ver 1.0
- **系统属性**
  跨平台、多功能的模拟操作系统程序
- **编程语言**
  Python 3
- **系统功能**

对于用户而言，本系统应具有良好的交互功能，包括对文件进行操作、模拟提交批处理任务（如打印任务、计算任务等）或运行普通程序的功能；对于系统开发人员而言，本系统应具备完整的进程管理、内存管理与文件管理功能，其中这三大功能中不仅包含对模拟硬件设备采用高效算法进行统一管理，而且能够支持对系统各功能组件的使用情况进行日志记录与分析。此外，本系统要能够支持对外开放清晰、友好的交互界面。

- **系统风格**

本系统应参考 Linux、Windows 等优秀的现代操作系统源代码或实现成果。经组内讨论，我们将把系统设计工作聚焦于内核功能集成开发，并对外提供简约、清晰的界面，形成自己独特、精巧的风格。

- **相关文档**

本项目在开发过程中除程序外，还应根据客户需求与实际情况提供相关文档，具体文档内容需要根据要求进一步拟定。团队在全过程中应注意文档的累积，为项目的顺利进行和后期的维护工作打下良好的基础。

本项目的实施过程将严格按照软件工程方法对系统程序进行设计与开发，团队的开发过程将主要分为前期准备、需求分析与系统设计、编码与测试、部署与维护这四个阶段，其中在第二个阶段团队将进行系统的概要设计与详细设计。

3 项目当前进展

在本系统开发的第一阶段，团队已经完成了系统的初步功能模块设计，并予以编码实现。团队的整体思路在于首先开发出具备相对完整的基本功能的系统程序，并在第二个阶段中在这一版本的程序之上进行交互能力、算法、系统功能层面上的增添、优化与改进。

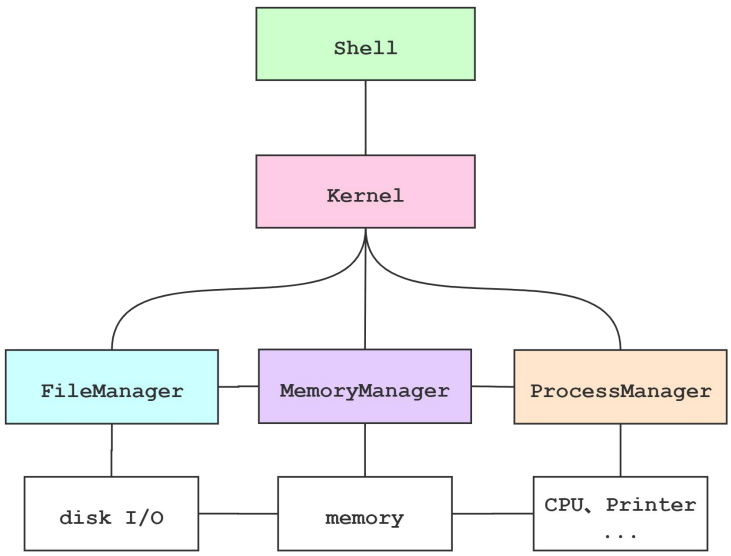
因此到目前为止，我们已经成功地完成了团队第一阶段目标，即完成了系统的基本框架设计并予以编程实现，其中各成员在第一阶段进一步根据自己选择的模块进一步划分为四个小组进行独立设计。由于在正式进行编码实现前，小组内部已经对各基本功能模块的接口进行了规定（如对象方法或全局函数的输入与输出）并明确了代码风格与编程规范，因此小组在完成了各基本模块的第一阶段独立测试后非常顺利地进行了组装测试，形成了具备基本功能的模拟操作系统程序。

下面是各小组成员在系统第一阶段开发过程中所完成的工作内容与当前待解决问题的归纳与总结：

- **陈斌同学（小组组长，负责部分：Kernel、Shell 模块）**

本阶段我的工作可以分为三个部分：提出路线、系统结构设计、编程开发。我组织了团队会议，召集小组成员们共同商讨，与小组成员们在前期对不同实现方式进行调研后提出当前的开发路线，即以模拟的方式进行操作系统功能实现，并以 Python 3 作为编程语言，专注于系统性能、算法与功能的设计上。

在提出开发路线的基础上，我与成员们协商探讨，确立了当前系统的基本结构与模块划分。其中为了使得不同功能的实现能够被清晰地划分并独立设计与测试，本系统尽可能地将不同功能划分到不同的模块中：将用户命令的接收与处理工作划分至 Shell 模块，对于底层外设的使用并实现文件、内存、进程管理的功能分别划分到三大管理模块 FileManager、MemoryManager、ProcessManager 中，并由 Kernel 模块对整个操作系统的其他模块进行统一调度与管理。其中三大管理模块之间不存在耦合关系，它们都是可以独立设计与测试的。本系统的基本结构与模块划分可由下图进行简单示意：



在完成了前两部分的工作后，我提出采用面向对象的设计方法对上示结构进行设计，并编写了代码框架，规定各变量的含义、各功能模块接口规范，并将任务分配予对应的小组成

员，其中我主要负责 Shell 与 Kernel 模块的开发。以此将团队内部的 6 个成员进一步划分为 4 个不同的分组开始并行开发，在各模块第一阶段开发完成后进行测试，与成员进行问题反馈与修复，并将模块进行了初步整合，形成一个基本功能相对完整的版本。

- **吴桐子同学（负责部分：文件管理模块）**

本阶段我的工作是完成了文件管理 FileManager 中对于外存文件块的管理。为了尽快完成最小版本开发，我们采用了连续分配的策略。我们为外存块 block 单独声明为一个类 Block。具体的外存块管理操作包括：初始化、新建文件时对 block 的写入、删除文件时对 block 的清除、展示各 block 的信息。其中，又涉及到对于 block 的重要操作，具体说明如下：

- 寻找空闲的连续 n 个 block。在本次作业中，我们采用了 first-fit 策略，即第一次找到足够的连续空间后，直接写入其中。这个方法将会引入外部碎片；
- 将文件填充进 block。指定文件大小和文件路径后，此操作将文件存放于连续的若干块中，并将更新 block 目录（其中每小项的格式为文件名：起始 block 号，使用的 block 数量）。这个操作在初始化 block 信息时会被使用，即读取目录下的所有文件，并放入的 block 中。此外，在新建文件时也会被使用。文件新建时会指定文件所占的空间，当外存空间中无法找到足够的空闲连续块时，将会报告空间不足的错误；
- 删除文件后清除 block。指定文件路径后，系统将从 block 目录中检索该文件所占有的空间，并将相应的外存空间释放。

正如上述文字所说，我们在外存块管理中会引入外部碎片，所以在后续的内核版本中，我们将考虑引入整理磁盘碎片的功能。此外，在判断 block 是否空闲的操作中，我们采用的策略是对每个块的剩余空间进行判断，这个操作较为不经济，而实际上可以采用 bit map 的方式，在后续的版本中，我们将考虑引入这个结构。

- **许浩然同学（负责部分：文件管理模块）**

本阶段我负责与吴桐子同学一起完成文件管理模块的功能分析，并迭代地、由易到难地实现这些功能。在其中，我主要负责各个文件操作命令的实现。我们和小组组长进行探讨，并明确了文件管理模块中针对文件存储体系的模拟方式：

模拟系统的文件存储结构在程序之外以对应文件夹的形式存在，在模拟系统程序启动后，该系统程序会依据外部的文件存储结构，通过复刻该结构形成本系统内部的文件体系；而对于文件内容，系统也将会把其读取到模拟磁盘中。这一模拟方式较为简易，其核心在于如何对文件树进行高效、合理的组织。

我本阶段开发工作的第一部分在于构建文件树字典。字典中的一个键值对可以代表一个目录, 以目录名为键, 以目录子字典为值。以此方式遍历这个字典结构，就可以获取到所有文件的信息。

在第二部分，我们参考了 Linux 系统中的多个文件操作命令：ls(显示目录下的内容)，cd(切换当前工作目录)，mkdir(创建目录)，mkf(创建普通文件)，rm(删除文件)，chmod(更改文件属性)，pwd(显示当前工作目录)等并进行相应的功能实现。

在第三部分，我们完善了文件操作命令的参数。从仅支持文件名，到支持相对路径，再到支持绝对路径；从无选项，到支持部分命令行选项功能，如 rm 命令中的“-f”(强制删除)，“-r”(递归地删除文件夹)等具有多种功能的命令选项。

在完成了以上三个部分的系统开发后，我们经过大量测试，发现系统中仍存在一些问题，如 chmod 未设置格式错误检测、绝对路径不允许以多个“\”结尾等等。因此我们下一步的工作在于致力解决这些问题，并尝试将系统对存储体系的模拟方式从“完全模拟”方式转化为“部分模拟”方式，即通过采用更加底层的方法对系统的文件存储体系进行二次设计与开发实现。

- **张炜晨同学（负责部分：进程管理模块）**

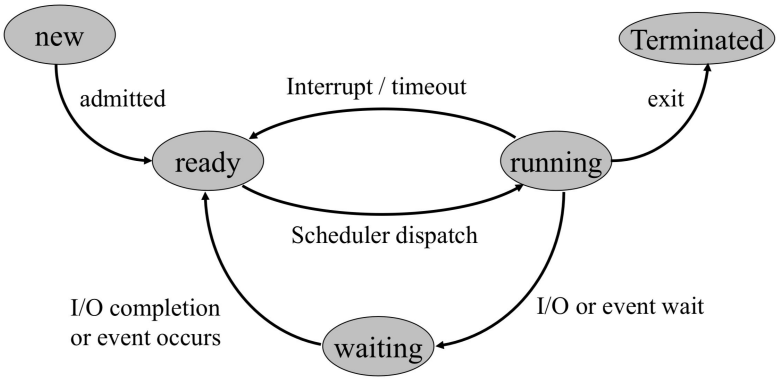
本阶段我主要负责进程管理模块的基础功能实现并与最后其他模块的对接工作，我将我的工作内容总结如下：

1．通过 exec 命令执行可执行文件，创建进程，创建后的进程在其整个生命周期中可以以 PCB 形式存在。每个进程用一个 PCB 表示，进程的属性在 PCB 中保存；

2．使用 ps 命令查看进程状态，rs 命令查看目前硬件资源使用状态，和预计被释放的时间（目前硬件资源仅支持打印机 Printer）；

3．使用 kill 命令杀死当前运行的进程，杀死进程后会释放其占用的 CPU，硬件 IO 等资源；

4．进程根据其执行情况在不同队列(就绪队列,阻塞队列,运行队列)间迁移。进程从 new 后在 ready、waiting、running 之间状态转换，直到最后进程执行完毕或者被 kill 后进入 terminated 状态，同时也模拟了时钟中断，硬件 I/O 中断等功能：



5．实现使用单处理器进程调度功能，目前的 CPU 的调度算法只设置了一个，即优先级调度与时间片相结合的调度算法。考虑到硬件资源的特殊性，对于硬件资源我们采用非抢占的先到先服务的调度算法。

当前关于进程管理模块依然存在的问题有：

1．CPU 调度算法仍是采用静态优先级和时间片相结合的策略，未使用动态优先级；

2．在进程进入 terminated 状态后，该进程 PCB 仍会保留，不会被删除；

3．未对内存资源进行竞争使用；

4．操作时间以一秒（时间片）为单位，若该时间片还未结束，该进程就结束 CPU 使用的话，并不会立即调度，而是会等到时间片结束后统一执行。

- **郑莘同学（负责部分：进程管理模块）**

本阶段我负责了四个方面的工作：关于实现裸机运行方法的调研、内核代码研究、进程管理模块程序设计与代码审阅、系统程序的跨平台测试。

我先研究学习了操作系统的启动过程。首先，计算机启动 BIOS，再加载位于硬盘引导扇区的引导程序(bootloader)至内存，并跳转开始运行引导程序。引导程序初始化 CPU 运行状态，再调用引导主程序将操作系统内核加载进内存，并将控制权交给它。最后，操作系统设置页表，初始化内存管理、进程管理、中断控制、文件管理，并启动第一个用户进程，操作系统启动完毕。我还成功在 QEMU 硬件模拟器以及 VMWare 虚拟机上启动运行了 xv6 操作系统。此外，我制作了 haribote OS 的 U 盘启动盘，学习研究了 BIOS U 盘启动的操作步骤。裸机启动的实现方法较为复杂，且实现难度大，这些前期调研工作的结果为后续团队开发路线的确定提供了较大的参考。

然后，我查阅了 linux 0.96、ucore、xv6 等小型操作系统的内核源码，重点研究了他们的进程管理模块，认真学习了代码风格，了解到进程管理模块需要设计的部分分别为：进程

控制块(PCB)、初始化进程(init)、创建子进程(fork)、进程退出(exit)、进程等待(wait)、进程调度(scheduler)、进程睡眠(sleep)等，对进程管理模块构建了总体框架。

之后，我和张炜晨同学共同变成实现了进程管理模块，并对代码进行审阅，初步实现了创建进程、进程调度、创建子进程的功能。其中，创建子进程的具体步骤为：复制父进程PCB，设置父节点进程号，分配子进程号，加入等待队列。下一步，我还计划添加进程间通信的功能设计，例如管道、虚拟网络环路等。

最后，我对本系统进行了跨平台运行测试，测试结果表明本次设计的系统程序在当前主流的操作系统 Windows 10 与 Ubuntu 20.04 LTS 下都能够正常运行，再次验证了系统程序的跨平台能力。

- **周雯笛同学（负责部分：内存管理模块）**

本阶段我的工作内容总结如下：

本阶段我主要负责内存管理模块，包括设计实现内存的分配与回收模式，作为其他模块的调用部分。内存分配大体分为连续型分配与不连续型分配，于是在本阶段我针对两大类型实现了两种具体内存分配方式：固定页面大小的页式分配及应用 best-fit 算法的连续分配，使用者可以在初始化时选择采用的分配模式，也针对这两种内存分配算法实现了两种内存回收方式，可以实现内存的正确分配及回收。除此之外，实现了针对不同内存存储方式的跟踪程序，可以在调用时格式化的打印出当前内存所处的状态。

当前关于内存管理模块依然存在的问题有：

① 针对不同的内存分配方式，我设计了不同的内存存储结构，但我认为在后期应当尝试设计出统一的存储结构，可以对多种算法进行兼容；

② 当前还未实现从配置文件中读取对内存分配的设置功能；

③ 还未尝试通过多线程的形式，通过调用跟踪程序来跟踪打印内存状态的功能。

4 后续开发计划

经团队第二次会议中成员们的共同商讨，我们对团队在第一阶段开发完成的系统程序进行了归纳总结，并针对当前各模块中存在的问题以及当前的系统状态进行了探究，提出并部署了接下来在第二阶段中本团队的开发计划，其可大致总结如下：

① 将各部分功能模块当前已经发现的问题予以全部解决；

② 在此版本的系统程序之上考虑新增以下功能或系统特性：

1. 对于文件管理模块，我们考虑将当前所采用的完全模拟方法修改为部分模拟策略，即尝试将磁盘设备进行更进一步的仿真模拟，对文件系统作出扩展和改进；

2. 在当前基础之上通过参考相关文献，尝试向三大功能模块中引入更加高效的算法与机制，例如引入管道、通信环路等进程间通信机制，并实现对磁盘、内存、CPU 等资源的使用情况进行实时追踪与统计分析的功能；

3. 参考当前主流操作系统，向本系统程序中引入更多、更丰富的命令集合。

③ 团队在第二阶段的开发过程应高效且有序，具体细节将在此次总结工作完成之后立即再次召开团队会议进行讨论与明确，与前一阶段类似，我们将在第二阶段的开发过程中对系统各功能模块及新增接口的输入输出作出统一规定，达成组内共识，并在此阶段的开发工作完成后对系统程序进行完整的测试与性能评估，将结果用于指导下一阶段的项目开发。

5 其他问题及风险分析

本小节将给出团队在接下来的开发过程中可能会遇到各类不同的问题。在此，本文将可

能遇到的关键问题以及开发风险归纳为如下三个方面，对每个风险均给予描述并给出初步的缓解方案，具体问题仍需要团队在开发过程中进行分析与处理。

风险排序	风险项名称	风险描述	缓解方案
1	专业基础知识不牢	本次项目开发过程中涉及的知识较多，如调度算法、分配算法等，以及程序实现细节，包括多线程并发、面向对象程序设计等，给项目的顺利进行带来一定的困难	团队成员进行相互学习与交流，并通过互联网等有效途径获取资源，对专业知识作出进一步的巩固，并可参考最新文献了解当前本领域的优秀算法与机制
2	经验欠缺	团队成员从未参与过操作系统程序的完整开发，使项目质量难以保证	只有通过不断的思考与实践
3	缺乏良好的开发环境	由于此次课程设计情况较为特殊，团队进行项目开发的软硬件设施较普通，缺乏集中的研发场所，团队在开发过程中难以进行高效的协调与沟通	尽量选择合适的软件与硬件设备，配置高效的集成开发环境，并尽可能通过社交媒体平台进行高效的组内交流与协作，及时反馈问题