HOMOS系统设计手册

学院:计算机学院

专业:计算机科学与技术

班级:2020211304

组员:倪玮昊,缪奇志,何正豪,石亚行,施天勤,葛北鱼,钱锡锐

1.项目需求

根据本次课程设计的相关要求，团队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，团队还应当为本次完整的开发过程配备相应完整的说明性文档。

同时该系统应该具有较为合理的UI交互系统,便于用户的使用.

2.团队分配

1.组员任务

该项目由JAVA语言开发

姓名	分工
倪玮昊(组长)	kernel和shell的开发
缪奇志	进程系统开发
何正豪	UI系统设计与开发
石亚行	文件管理系统开发
施天勤	文件管理系统开发
葛北鱼	进程系统开发
钱锡锐	内存管理系统开发

2.团队要求

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，实行团队内成员互相监督的合作机制。

依据项目路线进行系统开发

详细内容见本文档的"项目开发路线"部分，该部分内容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

团队在开发过程中应根据课程要求接收评审或检查（如日常审查、中期审查等），并且

团队应注意在设计与编码阶段累积相关文档，在项目结束时做到产品、文档兼备。

3.开发计划

每位成员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。

团队在将各个模块独立调试完成后进行了组装测试，最后团队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.

进行严格的测试,得出存在的BUG,并且按照用户体验情况添加或者修改功能

根据中期结果二次开发.

二次开发完成以后进行测试,最终完成开发任务.

3.设计思路

1.系统概要

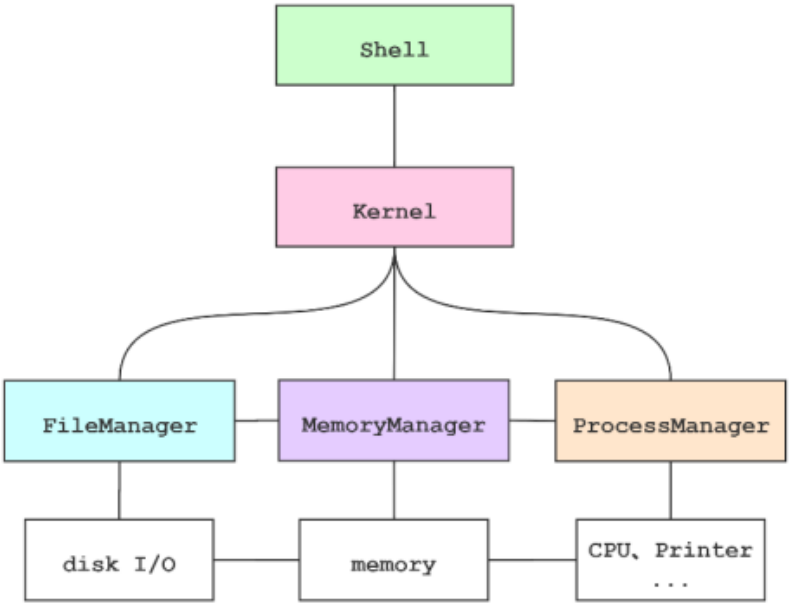
系统有四个不同的模块：Shell、Kernel、File Manager、Memory Manager、Process Manager。其中Shell 是用户与HOMOS沟通的桥梁，其在我们设计的系统中既是一种命令语言的存在，又是第一个在本系统之上运行的默认应用程序。

Shell 提供了一个界面，用户通过这个界面访问操作系统内核的服务；Kernel是MiniOS的核心部分，其实际上又由本身以及剩余的三个模块构成，负责管理系统的进程、内存、设备、文件等，决定着系统的性能和稳定性。

进一步划分，Kernel 实际上能够调用本系统剩余的三大核心模块。

其中 File Manager 负责对系统文件进行文件的逻辑组织和物理组织、文件树的结构和管理。所谓文件管理，就是操作系统中实现文件统一管理的一组软件，亦或是被管理的文件以及为实施文件管理所需要的一些数据结构的总称。在我们设计的系统中，该模块还负责与模拟磁盘设备进行交互，实现文件在物理存储体上的读写，并负责对空闲磁盘块的调度；另外，其具有驱动功能，能够对模拟磁盘设备进行直接控制，接管了与外存交互的所有事务。

对于 Process Manager，由于在HOMOS 中，进程是正在运行的程序实体，也包括这个运行的程序中占据的所有系统资源，比如说 CPU、打印机等。而这一模块能够实现对这些实体对系统核心资源使用的管理，如采用高效的调度算法（如优先级调度算法、高响应比调度算法等）以及通过维护 PCB（进程控制块）结构来实现对各进程实体关键信息的记录，以及使用资源的合理调度。



2.具体功能

模块	功能
Shell	为用户提供与系统进行交互的命令行形式的界面接口；对用户命令进行初步处理（如分割、转换等）；提供诸如缓存命令等机制，使得用户的使用体验以及使用效率能够得到大幅提升
Kernel	是整个系统最重要、最核心的程序；调度其之下所管辖的三大系统功能模块，处理计算、打印等任务请求，实现对系统资源的充分利用以及高效、统一的管理
File Manager	实现系统与外存的信息交换；构建文件系统，包括文件逻辑结构与树状存储结构，将系统中的所有文件进行组织；负责驱动并管理磁盘设备，指挥其进行指定磁道、扇区的块的读写；通过使用高效的算法实现对资源的合理利用；提供访问文件系统的应用接口；实现对磁盘等外设资源的监控
Memory Manager	围绕内核发出的分配或释放请求，对内存资源进行管理；实现虚拟内存机制，其中包括逻辑地址到物理地址的转换、运用多种算法对虚拟页进行替换操作等；对虚拟地址空间、物理内存等资源进行跟踪与管控
Process Manager	针对进程实体，通过 PCB、队列等数据结构实现对这类实体的内部及实体间的调配，使得计算机的核心功能资源（如CPU、打印机等）得到高效利用；对批处理任务进行解析与转换，使得各程序的任务能够有效地被分配到各功能部件上运行；提供多种批处理任务命令格式及系统调用（如 fork、access 等），使得程序的设计更加灵活，功能更丰富；对系统核心资源的使用情况进行跟踪监控
UI	完成和用户的交互以及对系统运行状况的监控

针对进程实体，通过 PCB、队列等数据结构实现对这类实体的内部及实体间的调配，使得计算机的核心功能资源（如CPU、打印机等）得到高效利用；对批处理任务进行解析与转换，使得各程序的任务能够有效地被分配到各功能部件上运行；提供多种批处理任务命令格式及系统调用（如 fork、access 等），使得程序的设计更加灵活，功能更丰富；对系统核心资源的使用情况进行跟踪监控

4.详细信息

1.process manager

1.需求分析

设计对应的批处理指令，完成对于批处理任务的解析与执行，使用PCB作为进程唯一存在的标志，并在PCB中存储进程相关信息。之后对内核创建的用户进程实体进行统一的调配与管理，实现系统核心资源与外设的充分利用，并通过时间片、优先级等机制，使得用户能够以更加灵活的方式与系统进行任务交互。

2.结构设计

PCB：采用哈希表来模拟实现，包括：

进程号(PID)：记录该进程的 ID 号

进程名(name)：记录该进程的名字

创建时间(create_time)：记录该 PCB 创建的时间

进程状态(status)：包括新建(new)、准备(ready)、运行(running)、等待(waiting)、终止(terminated)

优先级(priority)：进程的优先级

执行队列(commend_queue)：记录该进程仍需要执行的批处理指令

3.功能实现

预设批处理指令：

cpu[time]：模拟程序请求计算任务，执行时间time。

printer[time]：模拟程序请求打印任务，需要调度 Printer 部件来执行，执行时间time，先实现打印机设备，后期可添加其他其他设备的调度指令。

fork：在当前进程的基础上创建新的进程，需要调度CPU部件，执行时间1个单位。

access[address]：模拟进程访问访问对应逻辑地址（读或写操作），执行时间一个单位。

进程创建：

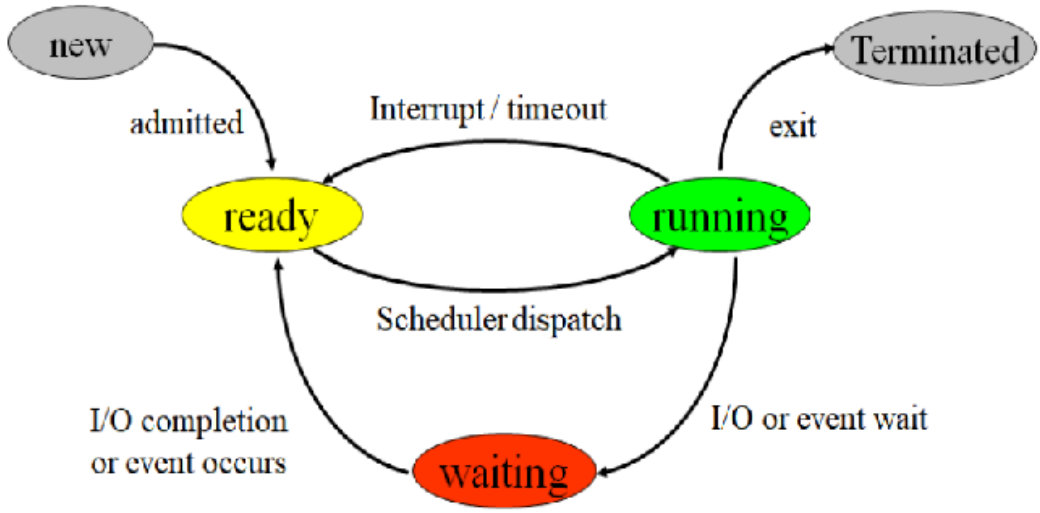
实现两种进程创建方式：

1.由操作系统创建，完成PCB的初始化。

2.由进程创建，标注父进程，分配好子进程号。

进程状态转移：

实现进程的五种状态转换，修改对应PCB的进程状态(status)。



进程调度算法：

预设实现先来先服务算法（FCFS）、最短CPU运行期优先调度算法（SJF）、优先级调度算法、最高响应比优先调度算法（HRRN）。

系统关键资源的监视与跟踪：

预设实现能通过输入对应的指令显示目前系统中仍在运行的程序的 PID 号，以及对应的程序名字，当前所处的运行状态还有创建该进程的时间，包括实现用图片绘制出近段时间系统资源占用率随时间的几何图形。

2.UI

UI在该系统中负责提供界面交互，并对每个功能模块提供可视化的状态监测。

1.UI构成

UI主要由以下部分组成：

Desktop:UI的主界面，模拟操作系统桌面的基础功能，其他组件都将在该界面内进行操作。

Win:模拟窗口，实现窗口的最大化、最小化、关闭、改变大小和拖动功能，具体功能的可视化交互将在其基础上进行。

Terminal:终端的显示窗口，在该窗口中进行指令操作。

TaskManager:任务管理器，通过该窗口检测cpu和内存的使用情况以及各进程的运行状态。

FileManager:文件管理器，通过该窗口可查看磁盘的使用状态以及以及文件目录。

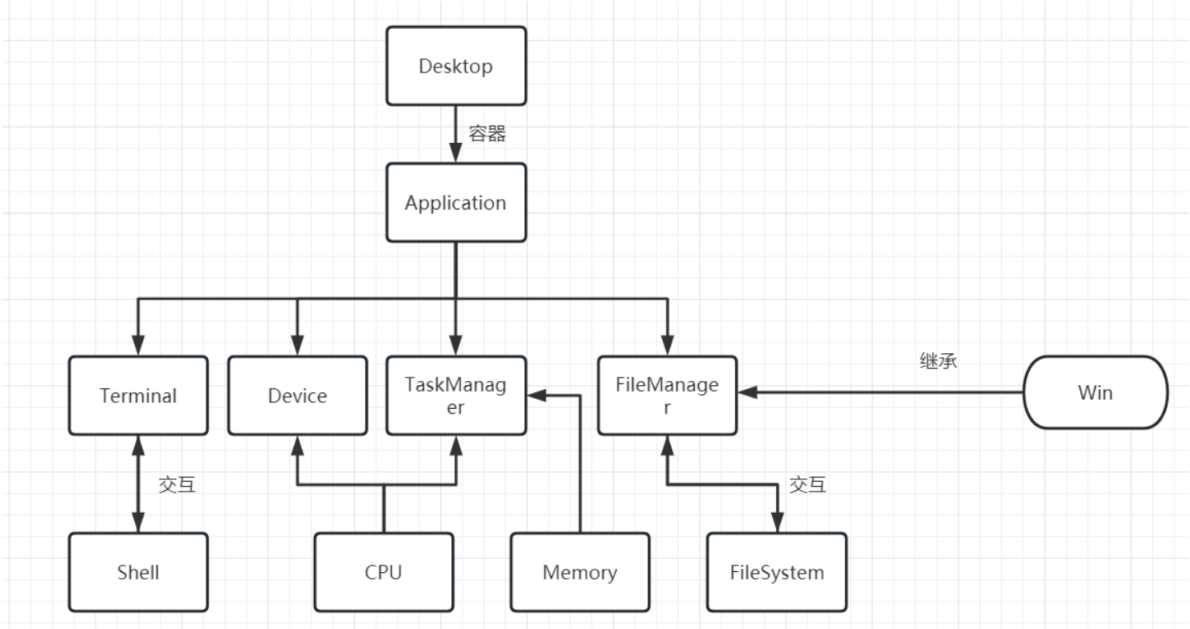
Application:为各系统应用提供图标。

Device:设备列表，显示设备使用情况。

Controller:该部分为UI界面提供交互逻辑的实现，上述每个部分都拥有一个对应的控制类。

2.UI逻辑

UI组件通过对应的controller类实现与用户的交互，并通过各系统功能提供的接口获取需要的信息。



3.File Manager

文件管理模块在该系统中主要负责管理文件系统中的文件和目录，同时是系统与外存之间交互控制的主要接口，负责对外存设备进行读写。

1.文件结构

我们准备以json文本来进行模拟文件内容。文本将有文件名(name)，文件属性(type)，占有磁盘的空间大小(size)，文件的内容(content)。其中文件名与原生操作系统的文件名一致，文件属性由一个字符串来表示文件类型、文件的可执行属性等，size的单位为byte。

```
{
   "name": "file_001",
   "type": "xxxxx",
   "size": "1024",
   "content": [.....]
}
```

2.文件组织结构

文件管理模块将采用多级树形目录结构对文件进行管理，并以map的形式存储。其中键对应文件名，值则对应文件的属性，如果元素的键对应的是一个目录的名称，则值对应着一个包含该目录下所有文件和目录的map。

3.文件管理基本功能

以下为该模块提供的一些方法及其功能。

- ls：列出目标路径下的文件。
- cd：改变当前工作目录。
- mkdir：新建文件夹。
- mkf：创建文件。
- rm：删除文件。
- tidy_disk：磁盘碎片整理。

在文件的读、写、创建与删除过程中，将以外存块block为对象进行操作。我们将为block声明一个类，其中的方法包括创建文件时block的写入、block信息的展示、删除文件时对block的清除。

4.文件数据空间的组织方式

我们准备采取连续分配的方式对文件进行存储，并建立文件目录，以记录文件的起始地址和长度，此处目录预计采用map的方式来进行索引，键对应文件或目录的名称，值对应存放起始地址和长度的长度为2的数组。同时为减少磁盘碎片，我们也准备提供紧凑的功能（tidy_disk）以整理磁盘碎片，提高存储空间利用率。同时，我们准备提供first-fit、best-fit、worst-fit 三种策略用于磁盘块的分配，用户可以从中进行选择。其中，对于block是否空闲我们将用bitmap进行表示，1为空闲，0为占用。

4.memory mananger

内存管理模块在该系统中主要负责对系统内存资源的同意调配与管理，Kernel会向内存管理模块提出请求，通过虚拟内存机制，对可执行文件在内存里进行有效的装载与回收

1.管理模式与算法的选择

1.动态分区分配（BF、FF、WF）

Best-fit算法，当Kernel申请存储区时，内存管理模块会选中一个满足要求的最小的空闲区分配给Kernel

First-fit算法，当Kernel申请存储区时，内存管理模块会选中一个最靠前的空闲区分配给Kernel

Worst-fit算法，当Kernel申请存储区时，内存管理模块会选中一个满足要求的最大的空闲区分配给Kernel

2.页式分配（FIFO、LRU）

采用“按需分页”，只有当进程的某一页被访问时，才会将这一页调入物理内存中。存储一个页表，标志位初始值为-1，即未被引用过

FIFO，先进先出算法，当Kernel申请存储区，总是淘汰最先进入内存的页面,即选择在内存中驻留时间最久的页面予以淘汰

LRU，最久未使用算法，当Kernel申请存储区，以过去预测未来，选择之前最长时间未使用的页面置换

2.内存结构

动态分区分配

用数据结构（arrayList或者linkedlist）记录分配分区的起始地址及初始地址，查找满足要求的空闲块，将该空闲块分配

页式分配

用数据结构（数组或者arraylist）来模拟多个页面，再添加标志位等信息

3.内存管理基本功能

1查看当前内存模式与调度算法

2自定义当前的内存大小（页大小、虚拟内存大小、物理内存大小等）

3查看缺页率

4内存碎片清理（可选）

4.其他

两种模式的本质几乎是完全不同的，应写两套代码，几乎没有重复的地方

用户可以选择管理的模式及其算法，可以自定义内存大小、页大小、虚拟内存大小等

5.shell与kernel

1.指令设计

命令	格式	功能
re	re [command]	对 re 之后字段的命令中的路径字段采用正则表达式进行解析与替换
man	man [command1] [command2] ...	展示单条或多条命令的帮助信息，若未携带具体命令，则默认对所有命令进行展示
ls	ls [-a|-l-al] [path]	列出指定路径 path 的内容，-a 选项列出全部内容（包括隐藏文件或目录），-l 选项列出文件或目录的详细信息，使用-al 同时指明以上两个选项。若未提供 path 参数，则默认 path 为当前目录
cd	cd [path]	修改当前工作目录，若未提供 path 参数，则默认 path 为系统根目录
rm	rm [-e|-f|-rf] path	删除文件或目录。其中待删除的文件或目录的路径必须提供，-r 选项能够递归地删除目录，-f 选项对应于强制删除功能，使用-rf 同时指明以上两个选项
mkf	mkf path type size	创建具有指定大小与权限的文件
mkdir	mkdir path	创建目录
dss	dss	展示系统外存各磁盘块的占用状态
dms	dms	展示系统物理内存占用状态
exec	exec path	执行指定路径下的文件，该文件须为可执行文件
ps	ps	展示当前系统所有进程状态
rs	rs	展示当前系统所有资源的使用状态
mon	mon [-o]	开始监控系统资源使用情况，将监视结果以图片的方式实时输出；-o 选项停止监控
td	td	整理系统磁盘外部碎片，即“紧凑”操作
kill	kill pid	强制结束指定进程
exit	exit	退出

2.调用流程

开机阶段,Kernel 负责对三个系统核心功能模块进行初始化操作，并通过这三大模块，实现对整个系统所有资源的统一、高效的管控。

Shell 支持用户在单个命令行内键入多条命令，如：① man ls; man cd，② mkdir new_dir; cd new_dir 等。在系统获取到用户输入的命令后，首先该命令会经过 Shell 的初步处理，如进行分割操作后，将其传递至内核.

内核会对命令进行初步检查，若命令名称、基本格式正确无误，则将调用指定的子程序来完成功能；否则系统将报告错误信息。特别地，对于携带过多参数的命令，如ls path1 path2 path3，则内核将取合法前缀进行处理，而对其他内容进行忽略，因此上条命令所完成的功能实际上是列出 path1 的内容。

如果用户输入了运行程序的指令,Kernel 需要首先引导磁头从磁盘上读取文件数据，将其内容调,入内存中，并指挥进程管理模块将其安插至队列结构中，参与调度与执行；最后，当该程序运行完毕后，Kernel 需要指导其余模块对该程序所占用的资源进行释放与回收