

操作系统课程设计报告

学院名称： 京江学院

专业班级： J软件工程1601 \_

学生姓名： 郑 凯

学生学号： 4161169028

2019年01月11日

# 目 录

[一、 课程设计要求](#_Toc31860_WPSOffice_Level1) [1](#_Toc31860_WPSOffice_Level1)

[二、 程序总体设计](#_Toc25977_WPSOffice_Level1) [4](#_Toc25977_WPSOffice_Level1)

[三、 文件管理详细设计](#_Toc25816_WPSOffice_Level1) [5](#_Toc25816_WPSOffice_Level1)

[四、 进程管理详细设计](#_Toc12453_WPSOffice_Level1) [7](#_Toc12453_WPSOffice_Level1)

[五、 设备管理详细设计](#_Toc6646_WPSOffice_Level1) [8](#_Toc6646_WPSOffice_Level1)

### 课程设计要求

**1.1 课设题目**

实现一个模拟操作系统。

**1.2 课设目的**

通过模拟操作系统原理的实现，加深对操作系统工作原理理解，进一步了解操作系统的实现方法，并可练习合作完成系统的团队精神和提高程序设计能力。

**1.3 课设内容**

模拟采用多道程序设计方法的单用户操作系统，该操作系统包括进程管理、存储管理、设备管理、文件管理和用户接口四部分。

**1.4 具体要求**

**1.4.1 文件管理和用户接口**

文件管理和用户接口部分实现的主要是单用户的磁盘文件管理部分，包括文件的逻辑结构、物理结构、目录、磁盘分配回收、文件的保护和用户接口的实现。

**⑴文件的逻辑结构**

文件的逻辑结构采用流式结构；文件的内容均采用文本文件；

系统中有两种文件，一种是存放任意字符的文件，一种是可执行文件。可执行文件的内容就是模拟系统内进程的程序体。

文件中要有一种特定命令的“可执行”文件，该文件中的命令有：

x=?; 给x赋值一位数 x++; x加1 x--; x减1

!??； 第一个？为A,B,C中某个设备，第二个？为一位数，表示使用设备的时间；

end. 表示文件结束，同时将结果写入文件out，其中包括文件路径名和x的值。

**⑵磁盘模拟**

用一个文件disk1模拟磁盘c，用一个文件disk2模拟磁盘d。两个磁盘一样大小，磁盘的每个盘块64字节，模拟磁盘共有128块。第0、1块存放文件分配表，第2块存放根目录，其余存放子目录和文件。

**⑶目录结构**

目录结构采用树型目录结构。

**⑷磁盘分配**

磁盘的分配采用链接结构（显式链接）的分配。系统采用文件分配表方式记录磁盘空间的使用情况和链接结构的指针。

**⑸磁盘空闲存储空间管理**

磁盘空闲存储空间管理采用位示图方法。

位示图和显示链接的指针合在一起组成文件分配表，占用磁盘空间第0、1块。

**⑹用户接口**

用户接口提供用户命令接口，要求实现以下命令：

创建文件：create 拷贝文件：copy 删除文件：delete 移动文件：move

显示文件：type 编辑文件 edit 改变文件属性：change 建立目录：makdir

删除目录：deldir 磁盘格式化命令：format

运行可执行文件：可执行文件的文件名（创建进程）。

**(7)屏幕显示**

如图1，屏幕显示要求包括：

用户命令接口，用于系统运行时用户输入命令；

磁盘目录显示，要求显示磁盘的树型目录结构；

磁盘使用情况，显示磁盘每一个磁盘块的空间是占用还是空闲。

**1.4.2 存储管理**

存储管理部分主要实现主存空间的分配和回收、存储保护。

(1)模拟系统中，内存部分分为两部分，一部分是系统区，这里只存放进程控制块和主存分配表，一部分是用户区，这里主要是对用户区的管理。

系统区包括PCB区域、主存空间分配表（位示图）；

用户区用数组模拟，大小为512字节，存储管理采用页式存储管理方式。

(2)主存用户区主存块大小为16字节，页表放在用户区内。

**1.4.3 设备管理**

设备管理主要包括设备的分配和回收。

(1)模拟系统中有A、B、C三种独占型设备，A设备3个，B设备2个，C设备1个。

(2)因为模拟系统比较小，因此只要设备表设计合理即可。

(3)采用先来先服务分配策略，采用设备的安全分配方式。

**1.4.4 进程管理**

进程管理主要包括进程调度，进程的创建和撤销、进程的阻塞和唤醒，中断作用的实现。

**⑴硬件工作的模拟**

①中央处理器的模拟

用函数CPU( )(该函数不能有参数)模拟中央处理器。

该函数主要负责解释“可执行文件”中的命令。

②中断的模拟

I．中断的发现应该是硬件的工作，这里在函数CPU中加检测PSW的方式来模拟。

在CPU（）函数中，每执行一条指令之前，先检查PSW，判断有无中断，若有进行中断处理，然后再运行解释指令。

CPU函数应该不断循环执行的。

Ⅱ．模拟中断的种类和中断处理方式：

* **I/O中断**（设备完成输入输出）：将输入输出完成的进程唤醒，将等待该设备的一个进程同时唤醒。

**⑵进程控制块**

进程控制块内容包括进程标识符、处理机状态信息（如主要寄存器内容）、进程调度信息（如进程状态、阻塞原因）、进程控制信息（如进程程序和数据的地址）。

PCB区域用数组模拟。

进程控制块根据内容的不同组成不同的队列：空白进程控制块链、就绪队列和阻塞队列，正在运行的进程只有一个，系统初始时只有空白进程控制块链。

**⑶进程调度**

①进程调度函数的主要工作是：

将正在运行的进程保存在该进程对应进程控制块中；从就绪队列中选择一个进程；

将这个进程中进程控制块中记录的各寄存器内容恢复到CPU各个寄存器内。

②闲逛进程

建立一个闲逛进程，当就绪队列为空时，系统调用该进程运行。当有进程就绪时，就调度就绪进程运行。

**⑷进程控制**

建立四个函数模拟进程创建、撤销、阻塞和唤醒四个原语。

### 程序总体设计

**2.1 编程语言**

JAVA语言

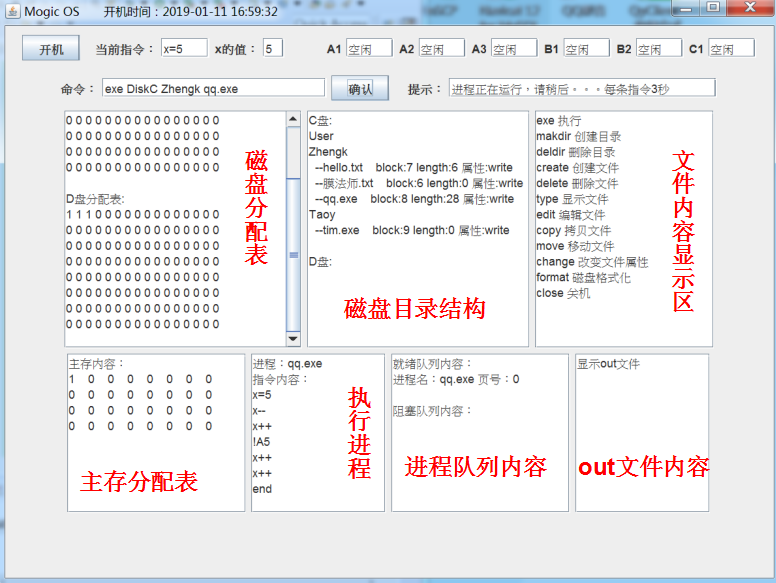
**2.2 总体思路**

模拟的操作系统一共分为四个模块，分别是进程管理、存储管理、设备管理、文件管理和用户接口四部分。其中文件管理部分与其他部分联系不是很紧密，只用一个exe执行命令与其他模块相连。模拟操作系统在运行时维护全局的进程就绪队列，进程阻塞链表，执行指令队列，设备队列这5个变量。exe命令执行时，系统将指令内容（文件里的内容）写入主存中，并创建进程控制块PCB，并将此PCB放入进程就绪队列中。

系统启动时，三个JAVA线程同时执行，CPU（）一直访问进程就绪队列（readyqueue），当队列不为空时，读取进程并模拟执行指令过程；OrderThread（）一直访问执行指令队列（orderqueue），当队列不为空时，读取当前指令并显示在界面上；Equip（）一直访问设备队列（euqipqueue），当队列不为空时，模拟使用设备过程。

执行指令时，每执行一条指令，就把指令名和X的值放入执行指令队列中，当指令为”!??”时要使用设备，就将进程调入进程阻塞队列并将进程放入设备队列，Equip（）创建JAVA线程执行使用设备指令。

**2.3 界面显示**

****

**2.4 程序结构**

**2.4.1 总体目录结构**

资源文件（src）下分为Windows（界面），InterFace（接口），FileManage，ProcessManage，MemoryManage，EquipManage六个包。

**2.4.2 目录结构详解**

**⑴FileManage包**

包里有3个Java文件，FileBlock.java存放文件/目录项数据内容，如文件名，扩展名，文件属性等；FileOperation.java存放具体接口功能方法，例如创建文件，删除文件，移动文件，磁盘格式化等；Tools.java内容都为静态函数，存放可复用的方法，例如找盘块号，文件同名判断等，以减少主类的代码量。

**⑵ProcessManage包**

包里有3个Java文件，PCB.java存放进程控制块；Process.java存放创建，销毁，阻塞，唤醒进程原语；OrderOpreation.java存放具体指令操作的方法函数。

**⑶InterFace包**

包里有4个Java文件，FileOperation.java，CPUListen.java是监视器类，前者是文件接口，用户输入命令由类具体调用函数执行，后者是CPU接口，当系统开机时，CPUListen.java，OrderThread.java，Equip.java3个线程同时启动，一直运行。

### 文件管理详细设计

**3.1 磁盘模拟**

用一个文件夹DiskC模拟磁盘c，用一个文件夹DiskD模拟磁盘D。两个磁盘一样大小，磁盘的每个盘块64字节，模拟磁盘共有128块。第0、1块存放文件分配表，第2块存放根目录，其余存放子目录和文件。

用DiskC.txt，DiskD.txt，FATC.txt，FATD.txt文件分别存放C、D盘的文件分配表，用盘块号.txt文件存放具体内容。例如2.txt存放盘块号为2的内容，即根目录的目录项内容。

磁盘空闲存储空间管理用位示图法：已分配磁盘置为1，未分配磁盘置为0，初始化时前三个盘块都为1。

磁盘分配方法用显式链接分配：FATC.txt，FATD.txt存放链接表，每一个表项存放指向的下一个盘块号。

系统运行时从文件读入空闲表和FAT表放入数组中，有修改时写回文件中。

static char[][] AllocationTable = new char[2][128]; //空闲表

static char[][] FATable = new char[2][128]; //FAT表

**3.2 目录项数据结构**

String filename; //文件名

String extension; //扩展名

int startblock; //起始块号

int filelength; //文件长度

//文件属性

String fd = null; //目录or文件

String wr = "write"; //可写or只读

String show = "show"; //显示or隐藏

**3.3 接口函数**

**3.3.1 makdir 创建目录**

顺序查找空闲表，得到一个空闲盘块号，分配给目录，并创建 盘块号.txt 文件；创建一个FileBlock（目录/文件项）；从根目录起，顺序查找目录，得到要写创建目录的所在目录的盘块号，在 目录.txt 文件中找是否有同名目录，若有，则无法创建，否则将FileBlock存入。

**3.3.2 deldir 删除目录**

找到目录的目录项，得到目录的起始盘块号，根据盘块号查找FAT表，得到存储目录的所有盘块号；修改空闲表，将盘块号置为0；根据目录中所存的文件项内容，删除文件；修改要删除的目录所在目录的目录项内容，删除此目录的FileBlock（目录/文件项）。

**3.3.3 create 创建文件**

顺序查找空闲表，得到一个空闲盘块号，分配给文件，并创建 盘块号.txt 文件；创建一个FileBlock（目录/文件项），写入属性；从根目录起，顺序查找目录，得到文件所在目录的盘块号，在 目录.txt 文件中找是否有同名文件，若有，则无法创建，否则将FileBlock存入。

例：创建DiskC\User\Java\zhengk.txt 首先到2.txt里找目录项名为User，得到盘块号为5，再到5.txt里找目录项名为Java，得到盘块号为8，再到8.txt里找文件项名为zhengk.txt，没有则写入FileBlock。

**3.3.4 delete 删除文件**

找到文件的文件项，得到文件的起始盘块号，根据盘块号查找FAT表，得到存储文件的所有盘块号；修改空闲表，将盘块号置为0，删除所有的 盘块号.txt 文件；修改文件所在目录的文件项内容，删除此文件的FileBlock（目录/文件项）。

**3.3.5 type 显示文件**

找到文件的文件项，得到文件的起始盘块号，根据盘块号查找FAT表，得到存储文件的所有盘块号；依次读出盘块内的内容，并显示在文件内容区。

**3.3.6 edit 编辑文件**

首先显示文件，若文件属性为write，则将编辑内容写回。

**3.3.7 copy 拷贝文件**

首先读取原文件内容，记录原文件所有内容，然后创建新文件，最后将原文件内容写入新文件中。

**3.3.8 move 移动文件**

首先读取原文件内容，记录原文件所有内容，接着删除原文件，然后创建新文件，最后将原文件内容写入新文件中。

**3.3.9 change 改变文件属性**

首先查找文件的FileBlock（目录/文件项），并根据属性修改文件项。

**3.3.10 format 磁盘格式化**

格式化文件分配表，将前三项置为1，其余都为0，删除所有 盘号.txt 文件。

### 进程管理详细设计

**4.1 PCB数据结构**

String name; //进程名

int page; //进程所在页

PCB next = null; //下一条进程地址

//中断信息

public int time = 0; //阻塞时间

public String equip = null; //设备名

int x = 0; //x的内容

Queue<String> order = new LinkedList<String>(); //指令内容

**4.2 静态变量**

volatile Queue<PCB> readyqueue = new LinkedList<PCB>() //就绪队列

volatile LinkedList<PCB> waitqueue = new LinkedList<PCB>() //阻塞链表

volatile Queue<String> orderqueue = new LinkedList<String>() //指令队列

CPU()执行时一直读取就绪队列内容，读到PCB时开始根据PCB内的指令队列执行指令，若遇到中断，则根据保存PCB信息放入阻塞链表，用链表的原因是阻塞链表不一定先进先出。

**4.3 进程原语**

**4.3.1 create 创建进程**

首先创建PCB（进程控制块），从内存获得内存页号和指令集，放入PCB中，将此PCB放入进程就绪队列中。

**4.3.2 destroy 销毁进程**

让内存删除放进程的内容页号，然后将PCB从就绪队列移除。

**4.3.3 wait 阻塞进程**

将PCB从就绪队列移除，并放入阻塞队列中。

**4.3.4 again 唤醒进程**

将PCB从阻塞队列移除，并放入就绪队列中。

**4.4 指令执行**

逐条读取PCB中的指令队列中所存指令内容poll（）并执行，若指令是x=? x++ x--，则继续获取；若指令是!??，则产生中断，送给设备执行，并将此进程加入阻塞队列；若指令是end，则将运行结果写入outfile并销毁进程。

### 设备管理详细设计

**5.1 总体思路**

static volatile Queue<PCB> euqipqueue = new LinkedList<PCB>(); //执行设备队列

从执行设备队列里读取PCB（进程控制块），根据PCB里调用设备名，创建相对应的线程，模拟设备操作。

**5.2 设备模拟**

private volatile static LinkedList<Object> buffer = new LinkedList<Object>();

设备类维护一个静态链表，大小为设备个数，当使用设备时，创建一个设备类线程，若buffer.size()>=设备个数，则此线程wait（）；若buffer.size()<设备个数，则往链表中添加一个对象，接着用sleep（pcb.time）模拟使用设备，睡眠完毕后将对象从buffer中移除，接着notify（）吵醒等待的线程，并将此PCB从进程阻塞队列移到进程就绪队列。