

**课程设计(综合实验)报告**

( 2021-- 2022度第 2学期)

名 称： 操作系统综合实验

题 目： 小型操作系统仿真实验

院 系： 计算机系

班 级： 软件2001

学 号： 220201090123

学生姓名： 杨照

指导教师： 王平

设计周数： 分散进行

成 绩：

日期： 2022年 7月

**目 录**

**一、 综合实验的目的与要求3**

**二、实验正文** **3**

1. 系统总体结构设计3

1.1 设计思路3

1.2 模块层次 3

2. 功能模块实现4

2.1 处理机调度（进程调度）4

2.2 银行家算法5

2.3 主存储器空间的分配和回收7

2.4 独占设备的分配和回收8

2.5 文件系统9

3. 模块集成设计与实现11

3.1 设计步骤11

3.2 开始执行 11

**三、综合实验总结或结论14**

**四、参考文献14**

**附录（设计流程图、程序）15**

**一、 综合实验的目的与要求**

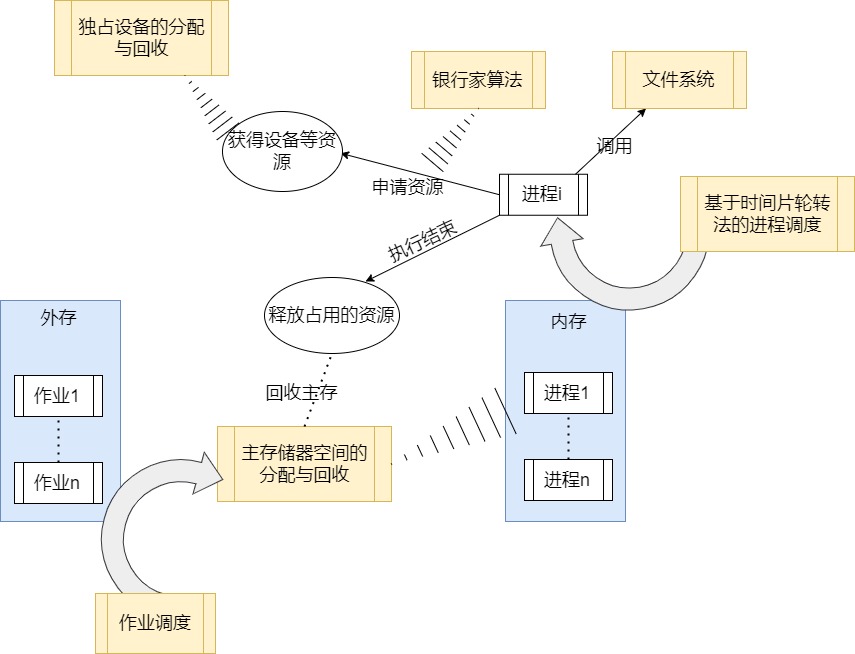
该实践环节是对学生掌握操作系统知识的综合检验，同时考察学生系统程序设计与编码能力。要求学生理解用户操作的流程，合理设计操作系统结构，明确操作系统各功能模块的联系，能够编码实现操作系统常用功能模块。

**二、实验正文**

1. 系统总体结构设计

1.1 设计思路

作业创建后存在于外存的后备队列中，等候操作系统的作业调度，将其读入内存，读入内存之前要满足主存储器空间的分配与回收原则，作业进入内存后，按照时间顺序排成循环队列，操作系统按时间片轮转法实现进程调度，被调度的进程当即获得一个单位的时间片，进程获得时间片后，即可申请相应的资源如设备资源，在通过银行家算法后进程即可获得对应的资源，同时，进程在运行过程可以调用文件系统进行相应的文件操作，在文件执行完要求运行时间后，即可释放设备等资源、同时从主存储器中回收对应的空间。

1.2 模块层次

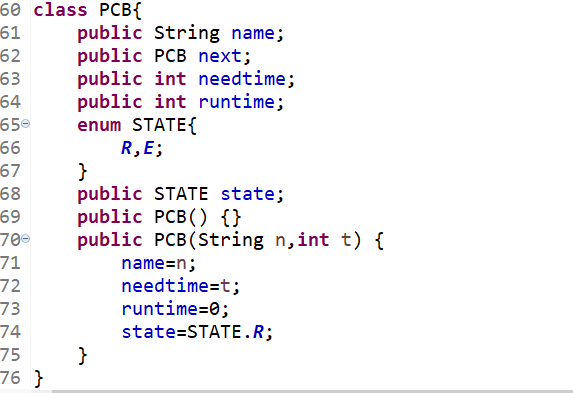
1. 功能模块实现

2.1 处理机调度（进程调度）

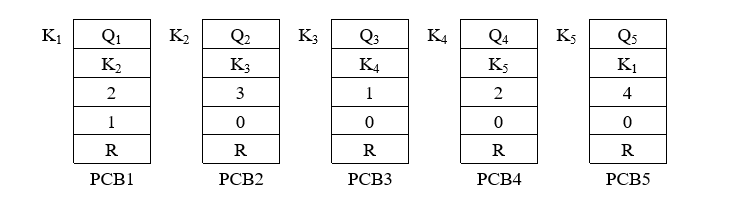
2.1.1 基本思路

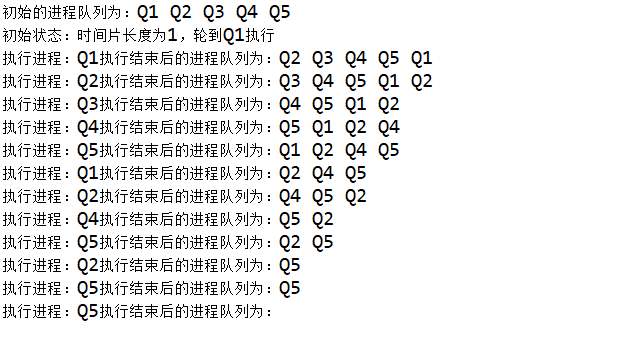
进程按调入主存的时间顺序排成循环队列，每个进程用一个进程控制块PCB来代表，PCB中包括进程名、指向下一个进程的指针、要求运行时间、已运行时间、状态（是否执行结束）等属性，按照循环队列各进程依次执行一个时间片单位，执行完成（已运行时间等于要求运行时间）时，该进程退出循环队列，同时上一个进程的指针直接执行该进程的后续进程，所有进程均执行完毕，进程调度结束。

2.1.2 基本结构



2.1.3 运行结果

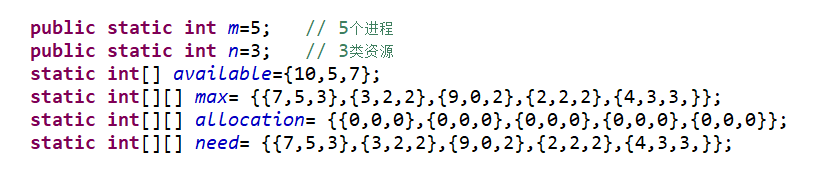
进程队列说明：目前进程队列中有五个进程，各进程的PCB如下

执行进程调度算法后，进程的执行情况：

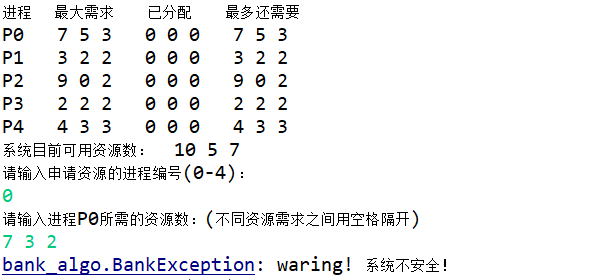
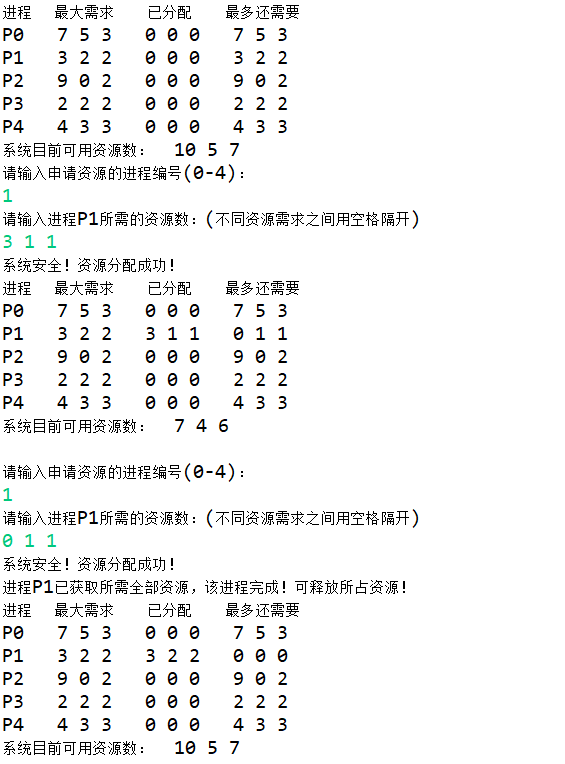
2.2 银行家算法

2.2.1 基本思路

首先确定系统资源类别数以及各资源数量，定义系统可用资源数向量AVAILABLE，然后根据进程个数和各进程对资源的需求量确定最大需求量矩阵MAX、进程已分配资源量矩阵ALLOCATION和进程还需要的资源量矩阵NEED，每次进程申请资源时，都要根据当前AVAILABLE矩阵判断资源量是否足够，若足够则试探性分配资源，检验分配后系统是否处于安全状态，若处于安全状态，则可以将资源分配给进程，否则输出异常信息。

 2.2.2 基本结构

2.2.3 运行结果

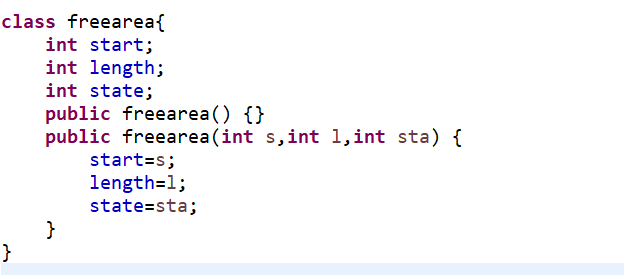


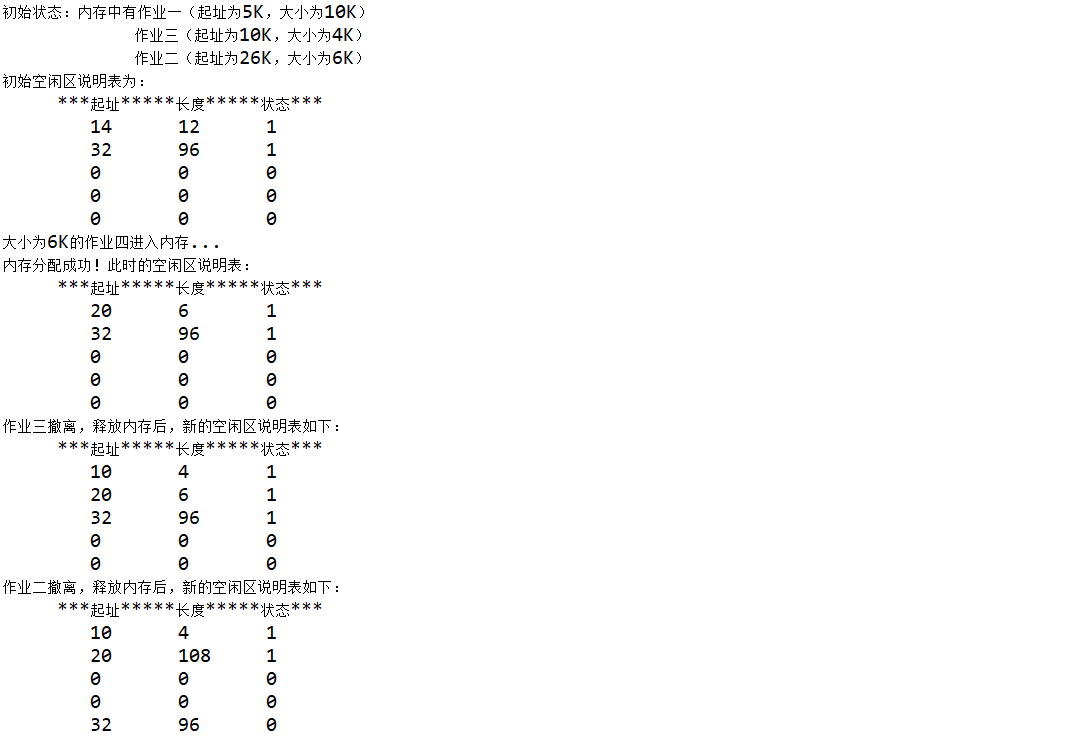
2.3 主存储器空间的分配和回收

2.3.1 基本思路

设置一个空闲区说明表，记录目前主存中的所有空闲区，包括其起始地址和长度，且表内空闲区按起始地址由大到小排序，当有进程请求调入主存时，会先在空闲区说明表依次查找，若无合适的空闲区则拒绝接受该进程，否则将空闲区分配给进程。当进程执行结束，要撤离主存，此时要释放对应的主存空间，并修改空闲区说明表的内容。

2.3.2 基本结构

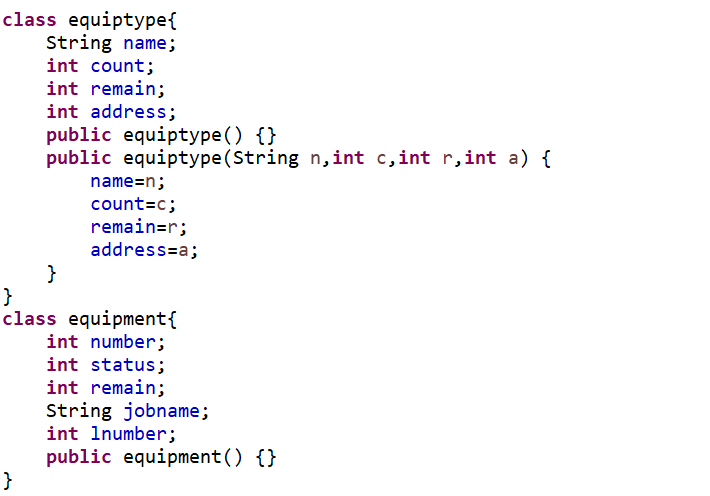


 2.3.3 运行结果

2.4 独占设备的分配和回收

2.4.1 基本思路

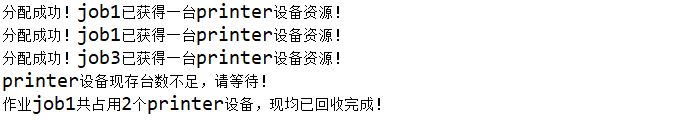
首先创建设备类表和设备表，设备类表记录系统内各设备的总台数、当前有几台设备空闲以及该类设备在设备表中的起始地址，设备表记录每一台设备的绝对号、设备的好坏情况、设备是否分配、设备被哪一个作业占用和设备相对号。当进程需要调用系统设备资源时，会现在设备类表中查询是否存在该类设备，若存在，则在设备表中查询是否有空闲的该类设备，若有，则分配给进程，否则输出异常提示信息。当进程使用完某一设备时，要根据设备类表读出该类设备在设备表中的起止地址，然后根据设备表中的作业占用记录，将该进程所占用的所有该类设备全部释放。

 2.4.2 基本结构

2.4.3 运行结果

说明：系统有设备printer、disk、input，分别有3、4、3台

执行流程：job1申请printer设备、job1申请printer设备、job3申请printer设备、

job1申请printer设备、job1释放printer设备

2.5 文件系统

2.5.1 基本思路

定义目录、目录项、文件等三个类型，其中每个目录内都有20个目录项，目录项根据type取值可以为目录也可以为文件，每个目录下都可以生成子目录或者文件，但是同一目录内文件（或子目录）不允许重名。文件系统共有CD（改变目录）、CREATE（创建文件）、DEL（删除文件）、LSALL（显示目录）、MD（创建目录）、RD（删除目录）等六类功能。初始化生成根目录Root和目录指针p，p指向根目录，后续功能实现均借助p。

CD子功能：输入一个目录名，在当前p目录下查找，若有对应名称的子目录，则p指向该子目录，否则输出异常提示信息。

CREATE子功能：输入一个文件名，以及对应的文件长度，在当前目录下若有同名文件则输出异常提示信息，若无则生成一个该文件。

DEL子功能：输入欲删除的文件名，在当前目录下查找，若有同名文件，则将其删除，若无，则输出异常提示信息。

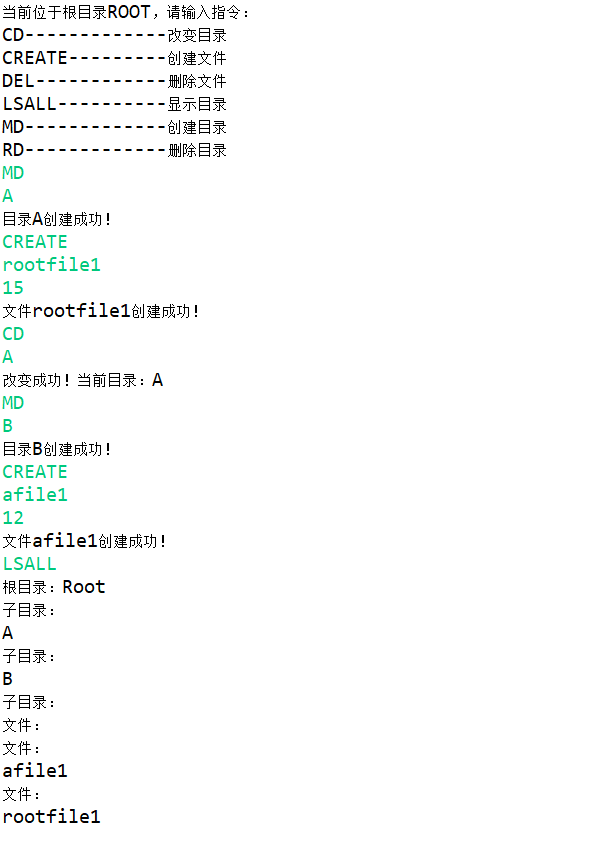
LSALL子功能：显示目录

MD子功能：输入一个目录名，在当前目录下查找，若无同名子目录，则创建该子目录，若有，则输出异常提示信息。

RD子功能：输入一个目录名，在当前目录下查找，若有同名子目录，检查该子目录内是否有文件（或子目录）若有，则输出提示信息确定是否删除，若无则直接删除该子目录，若无同名子目录则输出异常提示信息。

 2.5.2 基本结构



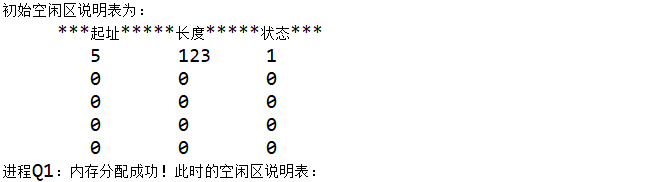
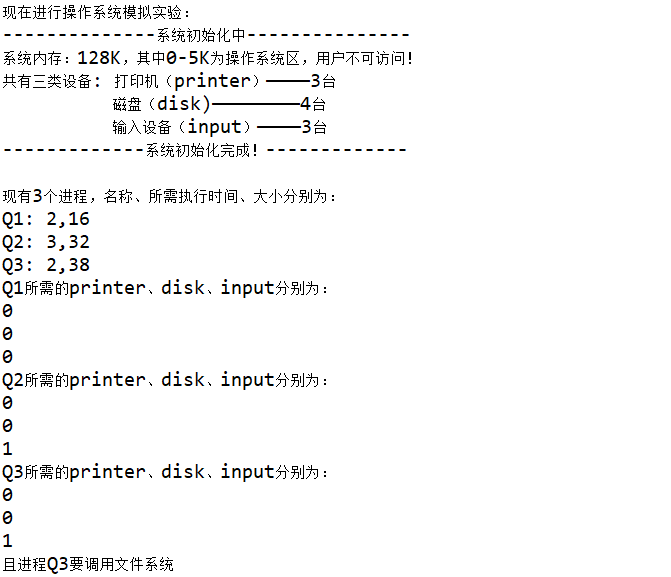
 2.5.3 运行结果

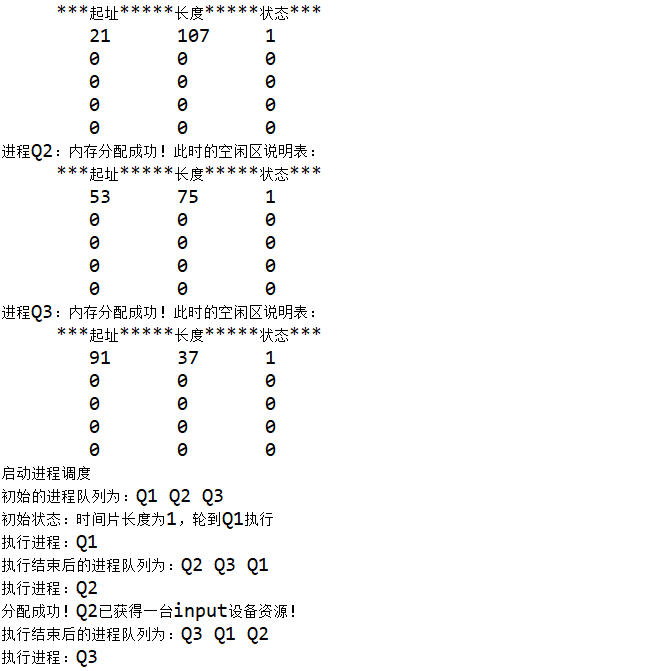
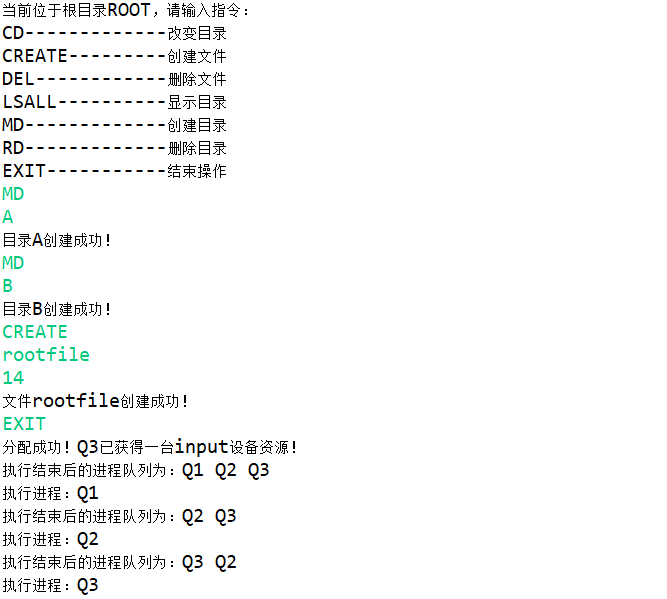
1. 模块集成设计与实现

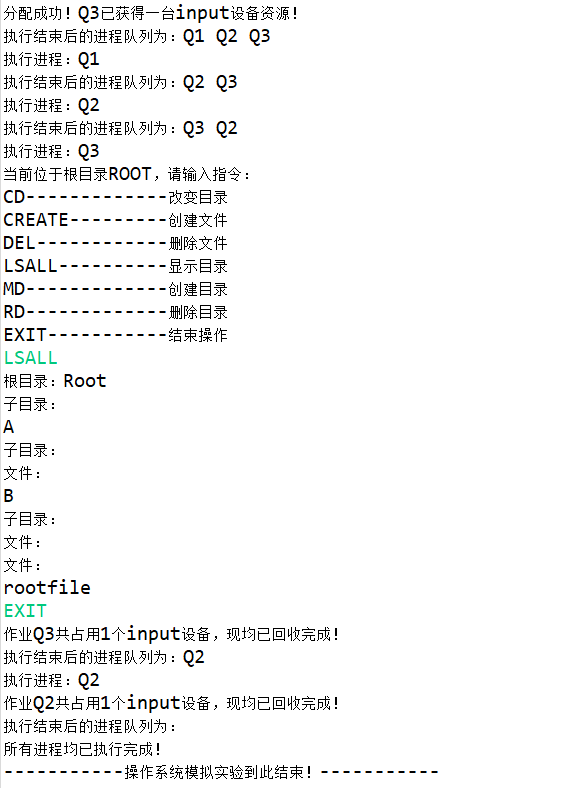
3.1 设计步骤

（1）初始化系统：系统内存共128K，其中0-5K为操作系统区，用户不可访问，系统共有三类设备：printer（3台）、disk（4台）、input（3台），预定义三个进程，名称为Q1、Q2、Q3，三个进程的要求运行时间、大小、对各类设备的需求量均由随机数函数生成，其中进程Q3需要使用文件系统。空闲区说明表初始状态有一个空闲区项，起始地址为5K，大小123K。

（2）开始执行：进程调度根据时间片轮转法依次调度进程，被调度的进程会申请调用自己所需的设备，若设备有空闲的，则调用成功，否则提示进程等待，进程获得所有所需设备后就不在申请设备，直到执行结束，就释放所有占用的设备。进程使用文件系统时，需要用户输入相应的文件系统指令完成相应的功能。

3.2 运行过程





**三、综合实验总结或结论**

本次综合实验难度较大，对综合素质的要求较高，一共五个模块实验与一个综合集成实验，其中模块实验虽然有些复杂，但实验指导书给出了很详细的思路，所以实现起来也没有遇到过多困难。但是最终的集成综合实验，由于缺少指导，我也是走了许多弯路，开始时没有头绪，不知如何下手，实现过程中也由于内容太多太乱太杂，而不断出错重试。但是最终程序的成功运作也让人倍感激动与兴奋，通过这次综合实验，我巩固了本学期对于操作系统这门课程的学习，强化了代码编写能力与分析问题的能力，最重要的是亲身感受到了操作系统的强大之处，操作系统如同一座坚挺的桥梁，联系着用户与计算机设备，它统筹协调着计算机内部的几乎一切运转，让计算机能够高效有序的完成功能，极大的提高了计算机工作效率，简化了人工操作成本，方便了用户使用。

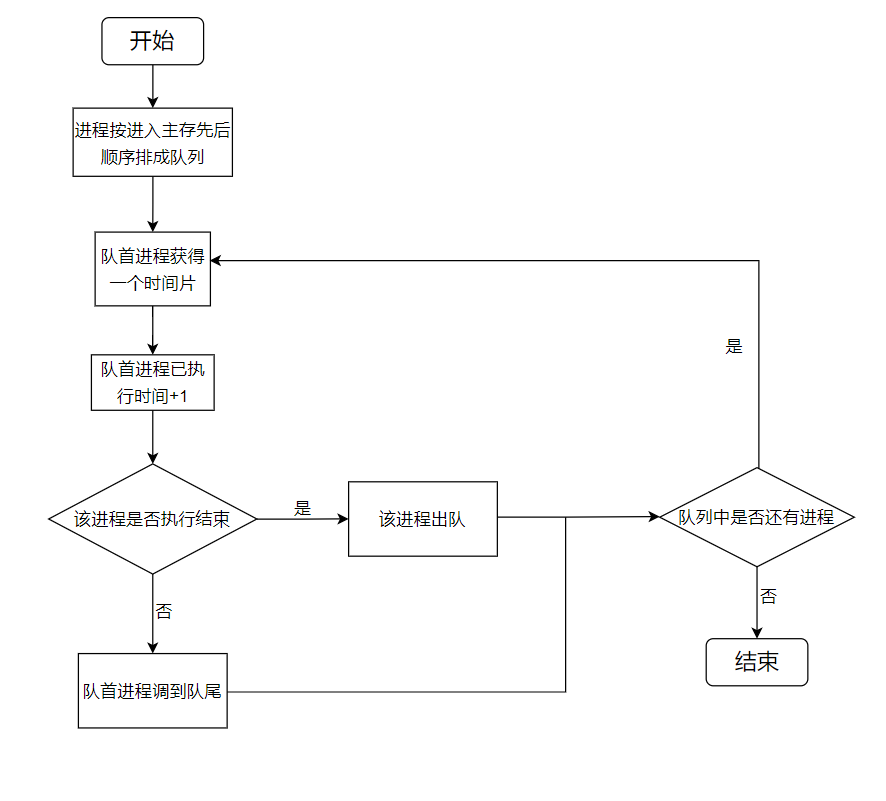
**四、参考文献**

【1】操作系统综合实验实验指导书（2022）

**附录（设计流程图、程序）**

一、进程调度

流程图：

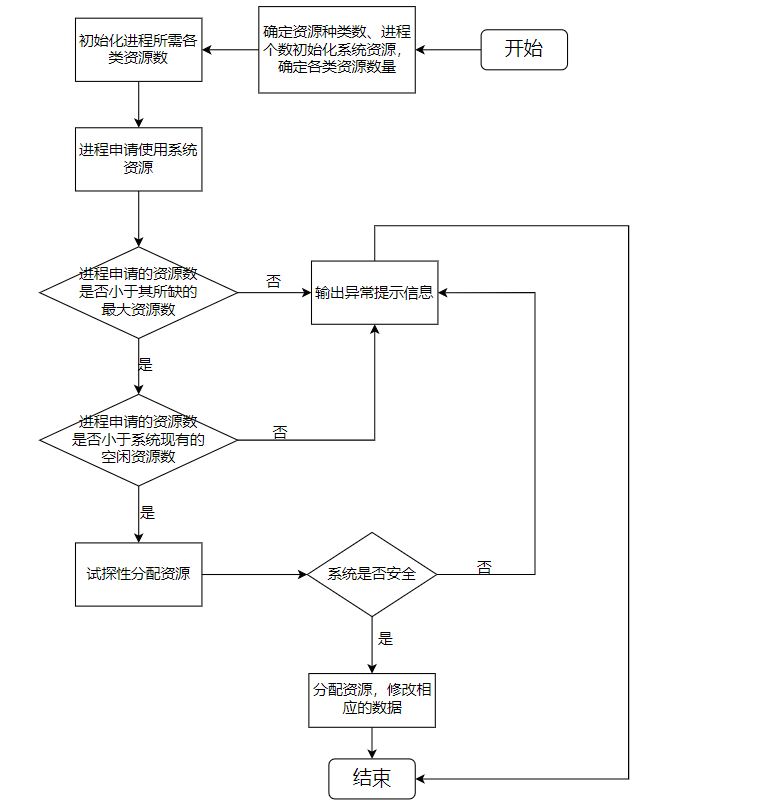


程序：

|  |
| --- |
| public class Test {  public static void main(String[] abs) {  Scanner scn=new Scanner(System.*in*);  int n=5;  PCB[] P={new PCB("Q1",2),new PCB("Q2",3),new PCB("Q3",1),new PCB("Q4",2),new PCB("Q5",4)};  for(int i=0;i<5;i++) {  if(i==4) {  P[i].next=P[0];  }  else{  P[i].next=P[i+1];  }  }  PCB p=new PCB();  p=P[0];  System.*out*.print("初始的进程队列为：");  *print*(p);  System.*out*.println();  System.*out*.println("初始状态：时间片长度为1，轮到Q1执行");  while(*NotEnd*(P)){  while(p.state==STATE.*E*) {  p=p.next;  }  System.*out*.print("执行进程："+p.name);  p.runtime++;  if(p.runtime==p.needtime) {  p.state=STATE.*E*;  }  p=p.next;  System.*out*.print("执行结束后的进程队列为：");  *print*(p);  System.*out*.print('\n');  }  }    public static void print(PCB p) {  for(int i=0;i<5;i++) {  if(p.state==STATE.*E*) {  p=p.next;  continue;  }  System.*out*.print(p.name+" ");  p=p.next;  }  }    public static boolean NotEnd(PCB[] p) {  for(int i=0;i<p.length;i++) {  if(p[i].state==STATE.*R*)  return true;  }  return false;  }  }  class PCB{  public String name;  public PCB next;  public int needtime;  public int runtime;  enum STATE{  *R*,*E*;  }  public STATE state;  public PCB() {}  public PCB(String n,int t) {  name=n;  needtime=t;  runtime=0;  state=STATE.*R*;  }  } |

二、银行家算法

流程图：

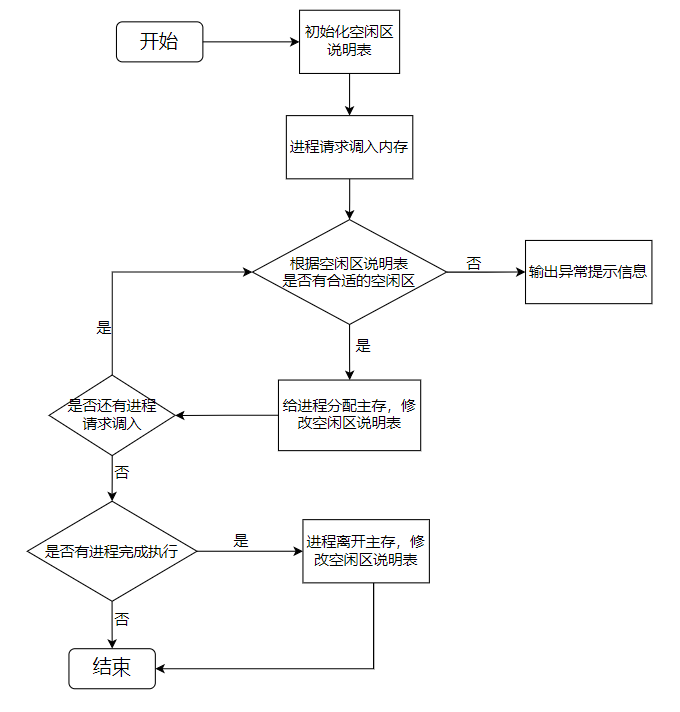


程序：

|  |
| --- |
| public class Test {  public static int *m*=5; // 5个进程  public static int *n*=3; // 3类资源  static int[] *available*={10,5,7};  static int[][] *max*= {{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3,}};  static int[][] *allocation*= {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0}};  static int[][] *need*= {{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3,}};    public static void main(String[] abs) throws BankException {  Scanner scn=new Scanner(System.*in*);  *print*();  while(true) {  System.*out*.println("请输入申请资源的进程编号(0-4)：");  int x=scn.nextInt();  System.*out*.println("请输入进程P"+x+"所需的资源数：(不同资源需求之间用空格隔开)");  int[] request=new int[*n*];  for(int i=0;i<*n*;i++) {  request[i]=scn.nextInt();  }  try{  *IsNormal*(request, x);  }catch(BankException e) {  System.*out*.println(e.toString());  continue;  }  try {  *CanOffer*(request, x);  }catch(BankException e) {  System.*out*.println(e.toString());  continue;  }  for(int i=0;i<*n*;i++) {  *available*[i]-=request[i];  *allocation*[x][i]+=request[i];  *need*[x][i]-=request[i];  }  int temp=0;  for(int i=0;i<*n*;i++) {  if(*allocation*[x][i]==*max*[x][i]){  temp++;  if(temp==*n*) {  System.*out*.println("进程P"+x+"已获取所需全部资源，该进程完成！可释放所占资源！");  for(int j=0;j<*n*;j++) {  *available*[j]+=*allocation*[x][j];  }  }  }  }  *print*();  System.*out*.println();  }  }    public static void print() {  //打印当前资源信息  System.*out*.println("进程 最大需求 已分配 最多还需要");  for(int i=0;i<*m*;i++) {  System.*out*.print("P"+i+" ");  for(int j=0;j<*n*;j++) {  System.*out*.print(*max*[i][j]+" ");  }  System.*out*.print(" ");  for(int j=0;j<*n*;j++) {  System.*out*.print(*allocation*[i][j]+" ");  }  System.*out*.print(" ");  for(int j=0;j<*n*;j++) {  System.*out*.print(*need*[i][j]+" ");  }  System.*out*.println();  }  System.*out*.print("系统目前可用资源数： ");  for(int j=0;j<*n*;j++) {  System.*out*.print(*available*[j]+" ");  }  System.*out*.println();  }    public static void IsNormal(int[] req,int x) throws BankException {  for(int i=0;i<*n*;i++) {  if(req[i]>*need*[x][i])  throw new BankException("Waring！ 进程P"+x+"申请了过量资源！");  }  for(int i=0;i<*n*;i++) {  if(req[i]>*available*[i])  throw new BankException("Waring！ 系统资源不足！");  }  }    public static void CanOffer(int[] req,int x) throws BankException {  int[] work=new int[*n*];  int[][] allocation1=new int[*m*][*n*];  int[][] need1=new int[*m*][*n*];  int[] finish=new int[*m*];  for(int i=0;i<*m*;i++) {  finish[i]=0;  }  for(int i=0;i<*n*;i++) {  work[i]=*available*[i];  }  for(int i=0;i<*m*;i++) {  for(int j=0;j<*n*;j++) {  allocation1[i][j]=*allocation*[i][j];  need1[i][j]=*need*[i][j];  }  }  for(int i=0;i<*n*;i++) {  work[i]-=req[i];  allocation1[x][i]+=req[i];  need1[x][i]-=req[i];  }  for(int i=0;i<*m*;i++) {  int apply=0;  for(int j=0;j<*n*;j++) {  if(finish[i]==0 && *need*[i][j]<=work[j]) {  apply++;  if(apply==*n*) {  for(int k=0;k<*n*;k++) {  work[k]=work[k]+allocation1[i][k];  }  finish[i]=1;  i=-1;  }  }  }  }  for(int i=0;i<*m*;i++) {  if(finish[i]==0)  throw new BankException("waring! 系统不安全！");  }  System.*out*.println("系统安全！资源分配成功！");  }  }  class BankException extends Exception {  public String ab;  public BankException() {}  public BankException(String abs) {  super(abs);  ab=abs;  }  public void printexception() {  System.*out*.println(ab);  }  } |

三、主存储器空间的分配和回收

流程图：

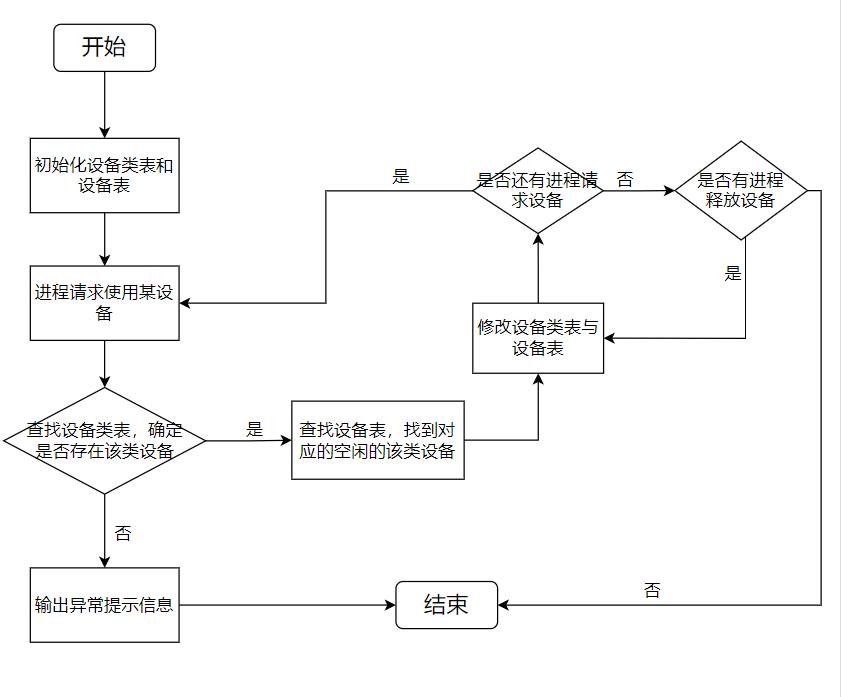


程序：

|  |
| --- |
| public class Test {  public static void main(String[] abs) {  freearea[] free= {new freearea(14,12,1),new freearea(32,96,1),new freearea(0,0,0),new freearea(0,0,0),new freearea(0,0,0)};  System.*out*.println("初始状态：内存中有作业一（起址为5K，大小为10K）"  + "\n 作业三（起址为10K，大小为4K）"  + "\n 作业二（起址为26K，大小为6K）");  System.*out*.println("初始空闲区说明表为：");  *sort*(free);  System.*out*.println(" \*\*\*起址\*\*\*\*\*长度\*\*\*\*\*状态\*\*\* ");  for(int i=0;i<5;i++) {  System.*out*.println("\t"+free[i].start+"\t"+free[i].length+"\t"+free[i].state);  }  System.*out*.println("大小为6K的作业四进入内存...");  if(!*allocation*(free, 6)) {  System.*out*.println("无合适的空闲区！");  }else {  System.*out*.println("内存分配成功！此时的空闲区说明表：");  *sort*(free);  System.*out*.println(" \*\*\*起址\*\*\*\*\*长度\*\*\*\*\*状态\*\*\* ");  for(int i=0;i<5;i++) {  System.*out*.println("\t"+free[i].start+"\t"+free[i].length+"\t"+free[i].state);  }  }  System.*out*.println("作业三撤离，释放内存后，新的空闲区说明表如下：");  *recycle*(free,10,4);  *sort*(free);  System.*out*.println(" \*\*\*起址\*\*\*\*\*长度\*\*\*\*\*状态\*\*\* ");  for(int i=0;i<5;i++) {  System.*out*.println("\t"+free[i].start+"\t"+free[i].length+"\t"+free[i].state);  }  System.*out*.println("作业二撤离，释放内存后，新的空闲区说明表如下：");  *recycle*(free,26,6);  *sort*(free);  System.*out*.println(" \*\*\*起址\*\*\*\*\*长度\*\*\*\*\*状态\*\*\* ");  for(int i=0;i<5;i++) {  System.*out*.println("\t"+free[i].start+"\t"+free[i].length+"\t"+free[i].state);  }  }    public static boolean allocation(freearea[] f,int n) {  for(int i=0;i<5;i++) {  if(f[i].state==1&&f[i].length>n) {  f[i].start+=n;  f[i].length-=n;  return true;  }  if(f[i].state==1&&f[i].length==n) {  f[i].state=0;  return true;  }  }  return false;  }    public static void sort(freearea[] free) {  freearea f;  for(int i=0;i<5;i++) {  for(int j=0;j<4;j++) {  if(free[j].start>free[j+1].start) {  f=free[j];  free[j]=free[j+1];  free[j+1]=f;  }  }  }  for(int i=0;i<5;i++) {  for(int j=0;j<4;j++) {  if(free[j].state==0 && free[j+1].state==1)  {  f=free[j];  free[j]=free[j+1];  free[j+1]=f;  }  }  }  }    public static void recycle(freearea[] free,int start,int length) {  int f1=0;  int f2=0; // 与上、下空闲区是否有相连  int i=0,j=0;  for(i=0;i<5;i++) {  if(free[i].start+free[i].length==start&&free[i].state==1) {  f1=1;  for(j=0;j<5;j++) {  if(free[j].start==start+length&&free[j].state==1) {  f2=1;  break;  }  }  break;  }  }  if(f1==1&&f2==1) {  free[i].length=free[i].length+length+free[j].length;  free[j].state=0;  }  if(f1==1&&f2==0) {  free[i].length+=length;  }  if(f1==0) {  int k=0;  for(k=0;k<5;k++) {  if(free[k].start==start+length && free[k].state==1) {  f2=1;  break;  }  }  if(f2==1) {  free[k].start=start;  free[k].length+=length;  }else {  for(j=0;j<5;j++) {  if(free[j].state==0) {  free[j].start=start;  free[j].length=length;  free[j].state=1;  break;  }  }  }  }  }  }  class freearea{  int start;  int length;  int state;  public freearea() {}  public freearea(int s,int l,int sta) {  start=s;  length=l;  state=sta;  }  } |

四、独占设备的分配和回收

流程图：

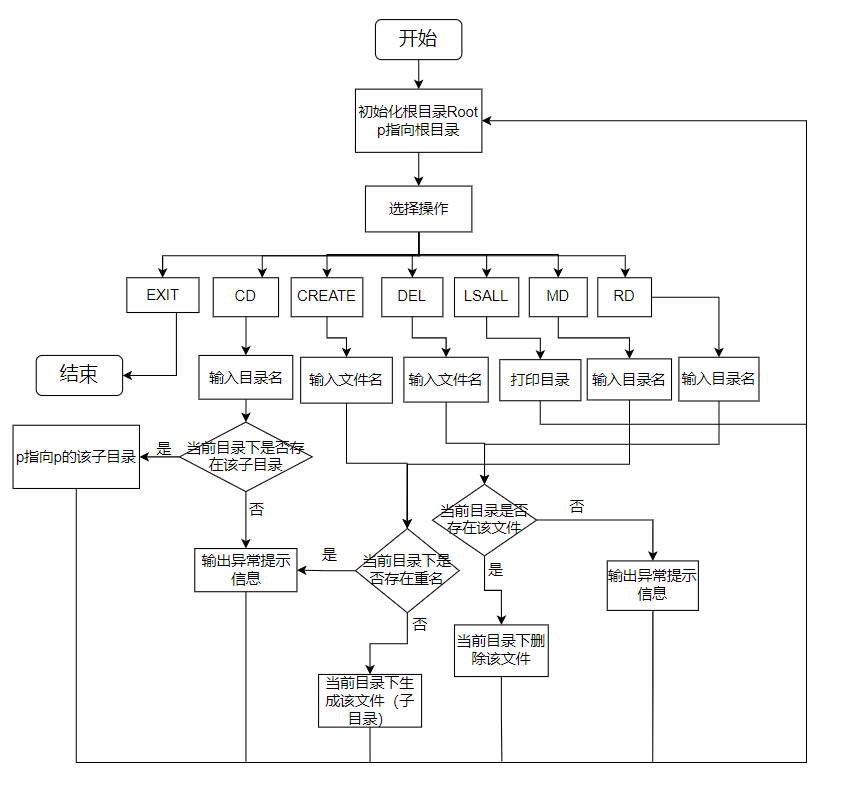


程序：

|  |
| --- |
| public class Test {  static int *N*=3;  static int *M*=10;  public static void main(String[] abs) {  equiptype[] equiptype=new equiptype[*N*];  equipment[] equipment=new equipment[*M*];  equiptype[0]=new equiptype("printer",3,3,0);  equiptype[1]=new equiptype("disk",4,4,3);  equiptype[2]=new equiptype("input",3,2,7);  for(int i=0;i<*M*;i++) {  equipment[i]=new equipment();  equipment[i].number=100+i;  equipment[i].status=1;  equipment[i].remain=0;  }  *allocate*(equiptype,equipment,"job1","printer",1000);  *allocate*(equiptype,equipment,"job1","printer",2);  *allocate*(equiptype,equipment,"job3","printer",3);  *allocate*(equiptype,equipment,"job1","printer",2);  *recycle*(equiptype,equipment,"job1","printer");  }    public static void allocate(equiptype[] e,equipment[] e1,String jname,String ename,int lnum) {  int i=0;  for(i=0;i<*N*;i++) {  if(e[i].name.equalsIgnoreCase(ename)){  if(e[i].remain<1) {  System.*out*.println(ename+"设备现存台数不足，请等待！");  return;  }  if(e[i].remain>=1){  int a=e[i].address;  int c=e[i].count;  for(int j=a;j<a+c;j++) {  if(e1[j].remain==0&&e1[j].status==1) {  e1[j].jobname=jname;  e1[j].remain=1;  e1[j].lnumber=lnum;  e[i].remain--;  System.*out*.println("分配成功！"+jname+"已获得一台"+ename+"设备资源！");  return;  }  }  }  }  }  System.*out*.println("没有设备"+ename);  }    public static void recycle(equiptype[] e,equipment[] e1,String jname,String ename) {  for(int i=0;i<*N*;i++) {  if(e[i].name.equals(ename)) {  int a=e[i].address;  int c=e[i].count;  int k=0;  for(int j=a;j<a+c;j++) {  if(e1[j].remain==1&&e1[j].jobname.equals(jname)) {  e1[j].remain=0;  k++;  }  }  if(k>0) {  e[i].remain+=k;  System.*out*.println("作业"+jname+"共占用"+k+"个"+ename+"设备，现均已回收完成！");  return;  }  System.*out*.println("作业"+jname+"并未使用"+ename+"设备！");  return;  }  }  System.*out*.println("没有"+ename+"设备！");  return;  }  }  class equiptype{  String name;  int count;  int remain;  int address;  public equiptype() {}  public equiptype(String n,int c,int r,int a) {  name=n;  count=c;  remain=r;  address=a;  }  }  class equipment{  int number;  int status;  int remain;  String jobname;  int lnumber;  public equipment() {}  } |

五、文件系统

流程图：



程序：

|  |
| --- |
| public class Test {  static int *j*=1;  static Scanner *scn*=new Scanner(System.*in*);  public static void main(String[] abs) {  DirectoryFile p;  DirectoryFile Root=new DirectoryFile("Root");  p=Root;  System.*out*.println("当前位于根目录ROOT，请输入指令：");  System.*out*.println("CD-------------改变目录"  + "\nCREATE---------创建文件"  + "\nDEL------------删除文件"  + "\nLSALL----------显示目录"  + "\nMD-------------创建目录"  + "\nRD-------------删除目录"  + "\nEXIT-----------结束操作");  boolean bool=true;  while(bool) {  String op=*scn*.next();  switch(op){  case"CD":  String dir=*scn*.next();  int temp=0;  try {  for(int i=0;i<20;i++) {  if(p.d[i].type==1&&p.d[i].d.name.equals(dir)) {  p=p.d[i].d;  temp=1;  break;  }  }  if(temp==0)  throw new FileException("该目录不存在！");  }catch(FileException e) {  System.*out*.println(e.toString());  }  if(temp==1) {  System.*out*.println("改变成功！当前目录："+p.name);  }  break;  case"CREATE":  *create*(p);  break;  case"DEL":  *del*(p);  break;  case"LSALL":  System.*out*.println("根目录："+Root.name);  *lsall*(Root);  break;  case"MD":  *md*(p);  break;  case"RD":  *rd*(p);  break;  case"EXIT":  bool=false;  break;  default:  }  }  }  public static void cd(DirectoryFile p) {  }  public static void create(DirectoryFile p) {  String dir=*scn*.next();  int len=*scn*.nextInt();  int temp=0;  try {  for(int i=0;i<20;i++) {  if(p.d[i].type==2&&p.d[i].f.name.equals(dir)) {  temp=1;  throw new FileException("文件重名！");  }  }  }catch(FileException e) {  System.*out*.println(e.toString());  }  if(temp==0) {  for(int i=0;i<20;i++) {  if(p.d[i].type==0) {  p.d[i].settype(2);  p.d[i].f.name=dir;  p.d[i].f.size=len;  System.*out*.println("文件"+dir+"创建成功！");  break;  }  }  }  }  public static void del(DirectoryFile p) {  String dir=*scn*.next();  int temp=0;  try {  for(int i=0;i<20;i++) {  if(p.d[i].type==2&&p.d[i].f.name.equals(dir)) {  temp=1;  }  }  if(temp==0) {  throw new FileException("文件不存在！");  }  }catch(FileException e) {  System.*out*.println(e.toString());  }  if(temp==1) {  for(int i=0;i<20;i++) {  if(p.d[i].type==2&&p.d[i].f.name.equals(dir)) {  p.d[i].settype(0);  System.*out*.println("文件"+dir+"删除成功！");  break;  }  }  }    }  public static void lsall(DirectoryFile p) {  DirectoryFile p1=p;  System.*out*.println("子目录：");  for(int i=0;i<20;i++) {  if(p1.d[i].type==1) {  System.*out*.println(p1.d[i].d.name);    *lsall*(p1.d[i].d);  }  }  System.*out*.println("文件：");  for(int i=0;i<20;i++) {  if(p1.d[i].type==2) {  System.*out*.println(p1.d[i].f.name);  }  }  }  public static void md(DirectoryFile p) {  String dir=*scn*.next();  int temp=0;  try {  for(int i=0;i<20;i++) {  if(p.d[i].type==1&&p.d[i].d.name.equals(dir)) {  temp=1;  throw new FileException("目录重名！");  }  }  }catch(FileException e) {  System.*out*.println(e.toString());  }  if(temp==0) {  for(int i=0;i<20;i++) {  if(p.d[i].type==0) {  p.d[i].settype(1);  p.d[i].d.name=dir;  System.*out*.println("目录"+dir+"创建成功！");  break;  }  }  }  }  public static void rd(DirectoryFile p) {  String dir=*scn*.next();  int temp=0;  try {  for(int i=0;i<20;i++) {  if(p.d[i].type==1&&p.d[i].d.name.equals(dir)) {  temp=1;  break;  }  }  if(temp==0) {  throw new FileException("目录不存在！");  }  }catch(FileException e) {  System.*out*.println(e.toString());  }  if(temp==1) {  for(int i=0;i<20;i++) {  if(p.d[i].type==1&&p.d[i].d.name.equals(dir)) {  for(int j=0;j<20;j++) {  if(p.d[i].d.d[j].type!=0) {  temp=2;  break;  }  }  if(temp==2) {  System.*out*.println("该目录下有文件，是否确定删除(请输入Y或N)");  char op=*scn*.next().toCharArray()[0];  if(op=='Y') {  p.d[i].settype(0);  System.*out*.println("目录"+dir+"删除成功！");  }  }else {  p.d[i].settype(0);  System.*out*.println("目录"+dir+"删除成功！");  }  }  }  }  }    }  class File{  String name;  int size;  DirectoryFile up;  }  class DirectoryFile{  DirectoryFile up;  String name;  DirectoryNode[] d=new DirectoryNode[20];  public DirectoryFile() {  for(int i=0;i<20;i++) {  d[i]=new DirectoryNode();  }  }  public DirectoryFile(String name) {  this.name=name;  for(int i=0;i<20;i++) {  d[i]=new DirectoryNode();  }  }  }  class DirectoryNode{  int type;  File f;  DirectoryFile d;  public DirectoryNode() {}  public void settype(int i){  this.type=i;  if(type==1)  d=new DirectoryFile();  if(type==2)  f=new File();  }  }  class FileException extends Exception{  public FileException (String msg){  super(msg);  }  } |