**安徽科技学院**

**软件类实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 学 号： | 2701200421 |
| 姓 名： | 乔梁 |
| 班 级： | 计算机科学与技术204班 |
| 指导教师： | 李文才 |
| 开课学期： | 2022-2023-1 |
| 学 院： | 信息与网络工程学院 |

安徽科技学院教务处

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **进程调度算法** | **实验学时** | 4 |
| **实验场所** | **软件实验室九** | **实验性质** | **验证** |
| **一、实验目标**：  通过对进程调度算法的模拟加深对进程概念和进程调度算法的理解，了解进程调度算法的具过程和方式，对其有更深的认识。 | | | |
| **二、实验原理：**  使用不同的进程调度算法可以实现操作系统的各种目标和要求，譬如想要高相应度就要使用时间片轮转算法，而追求最小平均周转时间就可以使用短进程优先算法，所以集合很多算法为一体的操作系统也是非常有必要的。 | | | |
| **三、实验要求：**  1.设计进程控制块PCB表结构适用于优先权调度算法。  2.建立进程就绪队列。  3.编制时间片轮转调度算法。  4.编制先来先服务算法。  5.编制短进程优先算法。  6.调制进程的各项其他应用和辅助。 | | | |
| **四、实验步骤： （包括详细实验操作步骤、流程图、代码及重要语句注释）**  在原有的算法基础上添加**短进程**优先算法，只要调整好开始的准备队列顺序，随后调用开始的FCFS算法就可以完成短进程优先代码。  //短进程优先算法 **public void SJF() {  System.out.println("当前系统时间：" + systemTime);  newSequence.sort(new Comparator<PCB>() {  @Override  public int compare(PCB o1, PCB o2) {  return o1.needTime - o2.needTime;  }  });  FCFS(); //调制好顺序后使用FCFS即可。 }**  **case 3:  Found Process3 = new Found();  Process3.SFS(); //短进程优先算法  print();**  **完整代码**  package ch1;  import java.util.ArrayList; import java.util.Comparator;  public class Found {  ArrayList<PCB> newSequence;   PCB pcb[] = new PCB[5];   int systemTime = (int) (Math.random() \* 3) + 1;   public Found() {  newSequence = new ArrayList<PCB>();  for (int i = 0; i < 5; i++) {  pcb[i] = new PCB();  pcb[i].pcbId = i + 1;  newSequence.add(pcb[i]);  }  }   // 先来先服务算法  public void FCFS() {  PCB exeRun = null;  while (newSequence.size() > 0)  {  exeRun = getPcb();  System.out.println("当前系统时间：" + systemTime);  systemTime += exeRun.useTime;  ShowMessages(exeRun);  }  }   // 时间片轮换算法  public void RR() {  int time = systemTime;  while (newSequence.size() > 0) {  rrMethode(time);  }  }   //短进程优先算法  public void SJF() {  System.out.println("当前系统时间：" + systemTime);  newSequence.sort(new Comparator<PCB>() {  @Override  public int compare(PCB o1, PCB o2) {  return o1.needTime - o2.needTime;  }  });  FCFS(); //调制好顺序后使用FCFS即可。  }  //输出  public void ShowMessages(PCB p) {  System.out.println("当前运行进程：" + p.pcbId +  " " + "服务时间：" + p.useTime +  " " + "需要时间：" + p.needTime +  " " + "优先级：" + p.perior +  " " + "状态：" + p.status);  if (newSequence.size() > 0) {  System.out.println("当前就绪进程：");  for (PCB p1 : newSequence) {  System.out.println("进程编号：" + p1.pcbId +  " " + "服务时间：" + p1.useTime +  " " + "需要时间：" + p1.needTime +  " " + "优先级：" + p1.perior +  " " + "状态：" + p1.status);  System.out.println("--------------------------------------------------------------------------");  }  } else {  System.out.println("当前系统中已经没有就绪进程！");  }  System.out.println('\n');  }  private PCB getPcb() {  PCB exeRun;  exeRun = newSequence.remove(0);  exeRun.useTime = exeRun.needTime;  exeRun.needTime = 0;  exeRun.perior = 0;  return exeRun;  }  private void rrMethode(int time) {  PCB exeRun;  System.out.println("当前系统时间：" + systemTime);  exeRun = newSequence.remove(0);   //时间片用完结束  if (exeRun.needTime <= time) {  exeRun.useTime = exeRun.needTime;  exeRun.needTime = 0;  exeRun.perior = 0;  exeRun.status = "Finish";  systemTime += exeRun.useTime;  } else {  exeRun.useTime += time;  exeRun.needTime -= time;  exeRun.perior--;  exeRun.status = "Ready";  newSequence.add(exeRun);  systemTime += time;  }  ShowMessages(exeRun);  }  }  package ch1;  import java.util.Scanner;  //主界面菜单 public class Menu {  Scanner sc = new Scanner(System.in);   public int print() {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("进程调度算法演示");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(" 1.先来先服务（FCFS）算法");  System.out.println(" 2.时间片轮换（RR）算法");  System.out.println(" 3.短进程优先（SJF）算法");  System.out.println(" 4.退出该程序");  System.out.print("请选择所要采用的算法：");  int flag = sc.nextInt();  return flag;  }   public void select() {  int flag = print();  switch (flag) {  case 1:  Found Process1 = new Found();  Process1.FCFS(); // 先来先服务算法  print();  case 2:  Found Process2 = new Found();  Process2.RR(); // 时间片轮换算法  print();  case 3:  Found Process3 = new Found();  Process3.SJF(); //短进程优先算法  print();  case 4:  System.exit(0);  default:  break;  }  } }  package ch1;  public class PCB {  int pcbId;  int useTime;  int needTime;  int perior;  String status;  public PCB() {  pcbId++;  useTime = 0;  needTime = (int) Math.round(Math.random() \* 5) + 1;  perior = (int) Math.round(Math.random() \* 5) + 1;  status = "Ready";  } }  package ch1;  public class ProcessControl {  public static void main(String args[]) {  Menu application = new Menu();  application.select();  } } | | | |
| **五、实验结果：** | | | |
| **六、实验小结：**  本次实验让我对调度算法有了更深的认识和感受，对算法中的细节方面有更深的认识，算法上对其有更好的把握进行计算。自己实现调度算法在细节的计算中会更好的计算和细节。 | | | |

**安徽科技学院**

**软件类实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 学 号： | 2701200421 |
| 姓 名： | 乔梁 |
| 班 级： | 计算机科学与技术204班 |
| 指导教师： | 李文才 |
| 开课学期： | 2022-2023-1 |
| 学 院： | 信息与网络工程学院 |

安徽科技学院教务处

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **存储调度算法** | **实验学时** | 4 |
| **实验场所** | **软件实验室九** | **实验性质** | **验证** |
| **一、实验目标**：  通过这次的操作系统课程设计，我懂得了许多，首先是让我对操作系统磁盘调度策略有了更加深刻的认识，自己动手操作比光看书能更加了解磁盘调度的策略和原理，同时对磁盘调度的算法先来先服务算法(FCFS)有了更深刻的理解和掌握，使我能够为磁盘调度选择适当的算法，提高CPU工作效率。 | | | |
| **二、实验原理：**  使用合适的磁盘调度算法可以让操作系统更加高效。 | | | |
| **三、实验要求：**  设计page类和FIFO算法的细节和要求，优化和完整算法的结构。 | | | |
| 四、实验步骤： （包括详细实验操作步骤、流程图、代码及重要语句注释）  为了可以添加其他算法模式的代码，可以将main抽出并使用最简单的单例模式简化，使用抽象工厂模式来更好的添加调度算法代码。  package ch2;  import java.util.Scanner;  public class StartPage {  public static void main(String args[]) {  **FIFO fifo = FIFO.GetInstance();**  Scanner sc = new Scanner(System.in);  fifo.init();  int c = 16;  int pageNum, localTionPos, order;  while (c > 0) {  System.out.println("请输入指令页号：");  pageNum = sc.nextInt();   System.out.println("请输入偏移地址：");  localTionPos = sc.nextInt();   System.out.println("请选择指令：1、存指令 2、非存指令");  order = sc.nextInt();   fifo.input(pageNum, localTionPos, order);  fifo.running();  c--;  }  } }  package ch2;  import sun.security.jca.GetInstance; import testch2.Hungry;  import java.util.ArrayList; import java.util.Iterator;  public class FIFO {  **private final static FIFO fifo = new FIFO(); //饿汉式单例模式。  private FIFO(){  }  public static FIFO GetInstance(){  return fifo;  }**  private static final int PRO\_MEMORY = 4;  private static int countOldPoint;  private static int count = 4;  private static int lackTime;  private static Page[] pages = new Page[PRO\_MEMORY];  private ArrayList<Integer> usePageNumList = new ArrayList<Integer>();  public void init() {  for (int i = 0; i < pages.length; i++) {  pages[i] = new Page();  }  }  public void input(int pagenum, int localtion, int order) {  if (order == 1) {  setMod(pagenum);  }  usePageNumList.add(pagenum);  System.out.println("页面:" + pagenum);  pages[count - 1].setLocation(localtion);  }  public void running() {  // 列表置换  Iterator<Integer> it = usePageNumList.iterator();  while (it.hasNext()) {  // 对4取模，得到0，1，2，3，正好是下标  countOldPoint = countOldPoint % PRO\_MEMORY;  // 获取到输入的页号  int inPageId = it.next(); // 查找页表中是否存在  TestPageId(inPageId);  display();  }  System.out.println("缺页次数为：" + lackTime + "，缺页率是：" + (float) lackTime / usePageNumList.size());  }   private void TestPageId(int inPageId) {  if (search(inPageId)) {  // 如果为true，表示找到了指定的页号，说明数据在内存中，然后计算给出地址  System.out.println("页面" + inPageId + "已在内存中");  System.out.println(inPageId \* 1024 + searchLoc(inPageId));  } else if (count < PRO\_MEMORY) {  pages[count].setPagenum((Integer) inPageId);  System.out.println("页面" + pages[count].getPagenum() + "进入内存");  count++;  } else {  System.out.println("\*" + inPageId);  int tmp = searchMod(inPageId); // 正常情况下，为0  if (tmp == 1) {  System.out.println("OOTj");  } else {  System.out.println("INL");  }  replace(inPageId);  lackTime++;  countOldPoint++;  }  }   public boolean search(int pageId) {  // 在内存中查询给定的页码是否存在于内存  for (int i = 0; i < pages.length; i++) {  if (pages[i].getPagenum() == pageId) {  return true;  }  }  return false;  }  public int searchLoc(int pageId) {  for (int i = 0; i < pages.length; i++) {  if (pages[i].getPagenum() == pageId) {  return pages[i].getLocation();  }  }  return 0;  }   public void setMod(int pageId) {  for (int i = 0; i < pages.length; i++) {  if (pages[i].getPagenum() == pageId) {  pages[i].setModify(1);  }  }  }   public int searchMod(int pageId) {  for (int i = 0; i < pages.length; i++) {  if (pages[i].getPagenum() == pageId) {  return pages[i].getModify();  }  }  return 0;  }  public void replace(int pageId) {  // 被置换出的页号  int outPageId = -1;   // 在内存中，找到最先添加的页号，也就是在内存中存在时间最长的页号  outPageId = pages[countOldPoint].getPagenum();   // 把最长时间的页号换成最新的  pages[countOldPoint].setPagenum(pageId);   System.out.println("页面" + pageId + "进入内存，" + outPageId + "被置换出内存");  }  public void display() {  System.out.print("当前内存中的页面：");  for (Page page : pages) {  System.out.print((page.getPagenum() + " "));  }  System.out.println();  } }  package ch2;  public class Page {  private int pagenum = 0; //页号  private int flag = 0; //标志 是否在页框中  private int memory = 0; //页框号  private int modify = 0; //是否修改，1为修改  private int location = 0; // 地址偏移量   public int getPagenum() {  return pagenum;  }   public void setPagenum(int pagenum) {  this.pagenum = pagenum;  }   public int getModify() {  return modify;  }   public void setModify(int modify) {  this.modify = modify;  }   public int getLocation() {  return location;  }   public void setLocation(int location) {  this.location = location;  } }  **五、实验结果：** | | | |
| **六、实验小结：**  熟悉磁盘替换算法的操作，也可以添加和使用其他设计模式来完善代码，对个人的实战受益颇多。 | | | |

**安徽科技学院**

**软件类实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 学 号： | 2701200421 |
| 姓 名： | 乔梁 |
| 班 级： | 计算机科学与技术204班 |
| 指导教师： | 李文才 |
| 开课学期： | 2022-2023-1 |
| 学 院： | 信息与网络工程学院 |

安徽科技学院教务处

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **文件管理实验** | **实验学时** | 4 |
| **实验场所** | **软件实验室九** | **实验性质** | **验证** |
| **一、实验目标**：  模拟系统中的用户控制管理建立文件，删除文件，打开文件，修改文件权限，读出文件，写入文件，关闭文件，不同用户之间的权限管理，使学生理解文件系统的基本结构和文件的管理。 | | | |
| **二、实验原理：**  文件通常存放在外存(如磁盘、磁带),上， 可以作为一个独立单位存放和实施相应的操作(如打开、关闭、读、写等)。为了加快对文件的检索，往往将文件控制块集中在一-起进行管理。这种文件控制块的有序集合称为文件目录。文件控制块就是其中的目录项。 | | | |
| **三、实验要求：**  使用程序模拟实现文件和用户的管理。 | | | |
| 四、实验步骤： （包括详细实验操作步骤、流程图、代码及重要语句注释）  主函数代码可以抽调成部分函数。  package ch3;  import java.io.FileInputStream; import java.io.FileOutputStream; import java.io.IOException; import java.util.Scanner;  public class Demo {  public static void main(String[] args) throws IOException {  MFD mfd = new MFD();  mfd.setSize(4);//设定用户数量。  SumUser(mfd);//增加用户  ControlMethod(mfd);//用户随后的操作输入。  }   private static void ControlMethod(MFD mfd) throws IOException {  boolean flag1 = true;//定义逻辑控制标记  while (flag1) {  System.out.print("请输入用户名：");  Scanner scanner = new Scanner(System.in);  String name = scanner.next();  if ((mfd.Contains(name) == null)) {  System.out.println("用户名不存在");  continue;  }  User user = mfd.Contains(name);   System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(user.getFilefold().get(i).getFile().getName() + '\t' + "权限" + user.getFilefold().get(i).getRwx() + '\t'  + "长度" + user.getFilefold().get(i).getFile().length() + "字节");  }   boolean flag = true;//定义逻辑控制标记  TestMeun(scanner, name, user, flag);//根据输入来选择操作。  System.out.print("是否退出系统：0)退出");  int jixu = Integer.parseInt(scanner.next());  System.out.println();  if (jixu == 0) {  flag1 = false;  System.out.println("系统退出");  }  }  }   private static void SumUser(MFD mfd) {  while (mfd.getSize() < 4) {  System.out.print("新增用户：");  Scanner scanner = new Scanner(System.in);  String username = scanner.next();  User user1 = new User(username);  mfd.addUser(user1);  }  }   private static void TestMeun(Scanner scanner, String name, User user, boolean flag) throws IOException {  while (flag) {  System.out.printf("%-14s%-15s%-12s\n", "1)新建文件", "2)删除文件", "3)打开文件");  System.out.printf("%-12s%-15s%-12s\n", "4)修改文件权限", "5)读出文件", "6)写入文件");  System.out.printf("%-14s%-13s%-12s\n", "7)关闭文件", "8)保存打印并退出", "0)不保存退出");  System.out.print("请输入指令: ");  int target = Integer.parseInt(scanner.next());  if (target < 0 || target > 8) {  System.out.println("输入有误(0-8):");  continue;  }  //根据指令运行相应功能  switch (target) {  //直接退出不保存  case 0: {  System.out.print("确定退出？(Y/N):");  String exit = scanner.next();  if (exit.equals("Y") || exit.equals("y")) {  flag = false;  System.out.println("已退出");  System.out.println();  user.remove();//删除用户目录里所有文件  break;  } else {  break;  }  }  //新建文件  case 1: {  if (user.getFilefold().size() == user.getFoldsize()) {  System.out.println("文件数目达到上限！");  break;  } else {  System.out.print("请输入文件名:");  String filename = scanner.next();  System.out.print("请输入文件保护码（读写执行）1表示允许，0表示不允许：");  String rwx = scanner.next();  java.io.File filetemp = new java.io.File("E:\\IDEACode\\算法\\src\\文件管理\\文件\\" + name + "\\" + filename);  filetemp.createNewFile();//创建文件  File file = new File(filetemp, rwx);  System.out.println("文件创建成功！");  user.addFile(file);  System.out.println();  break;  }  }  //删除文件  case 2: {  if (user.getFilefold().isEmpty()) {  System.out.println("目录为空！");  break;  } else {  System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(user.getFilefold().get(i).getFile().getName() + '\t' + "权限" + user.getFilefold().get(i).getRwx() + '\t'  + "长度" + user.getFilefold().get(i).getFile().length() + "字节");  }  System.out.print("请输入要删除的文件名：");  String filename = scanner.next();  user.delFile(filename);  System.out.println();  break;  }  }  //打开文件  case 3: {  if (user.getFilefold().isEmpty()) {  System.out.println("目录为空！");  break;  } else {  System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(i + "." + user.getFilefold().get(i).getFile().getName() + '\t' + "权限" +  user.getFilefold().get(i).getRwx() + '\t' + "长度" + user.getFilefold().get(i).getFile().length() + "字节");  }  System.out.print("请选择要打开的文件：（输入index）：");  String in = scanner.next();  int index = Integer.parseInt(in);  if (user.getFilefold().get(index).CanXing()) {  System.out.println("将选中的文件放入AFD中");  System.out.println("正在打开文件.....");  //创建一个文件输入流  FileInputStream is = new FileInputStream(user.getFilefold().get(index).getFile());  byte[] buf = new byte[1024];  int length = 0;  //循环读取文件内容，输入流中将最多buf.length个字节的数据读入一个buf数组中,返回类型是读取到的字节数。  //当文件读取到结尾时返回 -1,循环结束。  while ((length = is.read(buf)) != -1) {  System.out.print(new String(buf, 0, length));  }  //最后记得，关闭流  is.close();  } else {  System.out.println("没有该文件的运行权限");  }  System.out.println();  break;  }  }  //修改文件权限  case 4: {  if (user.getFilefold().isEmpty()) {  System.out.println("目录为空！");  break;  } else {  System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(i + "." + user.getFilefold().get(i).getFile().getName() + '\t' + "权限"  + user.getFilefold().get(i).getRwx() + '\t' + "长度" + user.getFilefold().get(i).getFile().length() + "字节");  }  System.out.print("请选择要修改权限的文件：（输入index）：");  String in = scanner.next();  int index = Integer.parseInt(in);  System.out.print("请输入文件新的权限（RWX）：");  String rwx = scanner.next();  user.getFilefold().get(index).setRwx(rwx);  System.out.println("修改成功！");  System.out.println();  break;  }  }  //读出文件  case 5: {  if (user.getFilefold().isEmpty()) {  System.out.println("目录为空！");  break;  } else {  System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(i + "." + user.getFilefold().get(i).getFile().getName() + '\t' + "权限" + user.getFilefold().get(i).getRwx());  }  System.out.print("请选择要读取的文件：（输入index）：");  String in = scanner.next();  int index = Integer.parseInt(in);  if (user.getFilefold().get(index).CanRead()) {  System.out.println("将选中的文件放入AFD中");  System.out.println("正在读取文件.....");  FileInputStream is = new FileInputStream(user.getFilefold().get(index).getFile());  byte[] buf = new byte[1024];  int length = 0;  //循环读取文件内容，输入流中将最多buf.length个字节的数据读入一个buf数组中,返回类型是读取到的字节数。  //当文件读取到结尾时返回 -1,循环结束。  System.out.println("文件内容如下");  while ((length = is.read(buf)) != -1) {  System.out.print(new String(buf, 0, length));  }  //最后记得，关闭流  is.close();  } else {  System.out.println("没有该文件的读取权限");  }  System.out.println();  break;  }   }  //写入文件  case 6: {  if (user.getFilefold().isEmpty()) {  System.out.println("目录为空！");  break;  } else {  System.out.println("当前目录：");  for (int i = 0; i < user.getFilefold().size(); i++) {  System.out.println(i + "." + user.getFilefold().get(i).getFile().getName() + '\t' + "权限" +  user.getFilefold().get(i).getRwx() + '\t' + "长度" + user.getFilefold().get(i).getFile().length() + "字节");  }  System.out.print("请选择要写入的文件：（输入index）：");  String in = scanner.next();  int index = Integer.parseInt(in);  if (user.getFilefold().get(index).CanWrite()) {  System.out.print("请输入要写入的内容：");  String content = scanner.next();  FileOutputStream os = new FileOutputStream(user.getFilefold().get(index).getFile());  byte[] t = new byte[1024];  t = content.getBytes();//将字符串编码为 byte 序列，并将结果存储到一个新的 byte 数组中  os.write(t, 0, t.length);//即向文件写入一个byte数组的内容  //关闭资源  os.close();  System.out.println("将选中的文件放入AFD中,AFD的写指针指向选中文件");  System.out.println("正在写入文件...");  System.out.println("写入成功");  } else {  System.out.println("没有该文件的写入权限");  }  System.out.println();  break;  }  }  //关闭文件  case 7: {  System.out.println("文件已关闭！");  break;  }  //保存目录并打印  case 8: {  //打印目录  user.display();  flag = false;  break;  }  default:  continue;  }  }  } }  package ch3;  public class File {  private java.io.File file;  private int[] rwx = new int[3];// 安全标记    public File(java.io.File file, String rwx) {  this.file = file;  this.setRwx(rwx);  }   public String getRwx() {  return Integer.toString(rwx[0]) + Integer.toString(rwx[1]) + Integer.toString(rwx[2]);  }   public void setRwx(String rwx) {  this.rwx[0] = Integer.parseInt(String.valueOf(rwx.charAt(0)));   this.rwx[1] = Integer.parseInt(String.valueOf(rwx.charAt(1)));   this.rwx[2] = Integer.parseInt(String.valueOf(rwx.charAt(2)));  }   public java.io.File getFile() {  return file;  }   public boolean CanRead() {  if (this.rwx[0] == 1) {  return true;  }  return false;  }   public boolean CanWrite() {  if (this.rwx[1] == 1) {  return true;  }  return false;  }   public boolean CanXing() {  if (this.rwx[2] == 1) {  return true;  }  return false;  } }  package ch3;  import java.util.ArrayList;  public class User {  private String username;// 用户名  private ArrayList<File> filefold = new ArrayList();//用户目录  private int foldsize = 10;// 目录大小   public User() {  }   public User(String username) {  this.username = username;  java.io.File ff = new java.io.File("E:\\IDEACode\\code\\" + username);  ff.mkdir();  }   public String getUsername() {  return username;  }   public int getFoldsize() {  return foldsize;  }   //未选择保存则清空目录  public void remove() {  this.filefold.removeAll(filefold);  /\*java.io.File ff = new java.io.File("E:\IDEACode\code\"+username);  ff.delete();\*/  }   //添加文件  public void addFile(File file) {  filefold.add(file);  }   public ArrayList<File> getFilefold() {  return filefold;  }   //删除文件  public void delFile(String filename) {  java.io.File file = new java.io.File("E:\\IDEACode\\code\\" + username + "\\" + filename);  if (filefold.isEmpty()) return;  for (int i = 0; i < filefold.size(); i++) {  if (this.filefold.get(i).getFile().getName().equals(filename)) {  this.filefold.remove(i);  }  }  file.delete();  System.out.println("文件删除成功！");  }   public void display() {  if (filefold.isEmpty()) {  System.out.println("目录为空！");  }  for (int i = 0; i < this.getFilefold().size(); i++) {  System.out.println(this.getFilefold().get(i).getFile().getName());  }  } }  **五、实验结果：** | | | |
| **六、实验小结：**  对文件和用户的管理操作更有实感，对其的代码程序理解更具体，更高效。 | | | |

**安徽科技学院**

**软件类实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 学 号： | 2701200421 |
| 姓 名： | 乔梁 |
| 班 级： | 计算机科学与技术204班 |
| 指导教师： | 李文才 |
| 开课学期： | 2022-2023-1 |
| 学 院： | 信息与网络工程学院 |

安徽科技学院教务处

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **磁盘管理** | **实验学时** | 4 |
| **实验场所** | **软件实验室九** | **实验性质** | **验证** |
| **一、实验目标**：  掌握磁盘管理使用的FCFS,SSTF,SCAN,CSCAN算法，对算法本身的运行细节有更深的理解和认识。 | | | |
| **二、实验原理：**  磁盘读写原理，先来先服务原理，对磁道总数的概念理解。 | | | |
| **三、实验要求：**  使用程序模拟和实现磁盘管理使用的各项算法和操作。。 | | | |
| 四、实验步骤： （包括详细实验操作步骤、流程图、代码及重要语句注释）  鉴于算法代码多次复用排序算法，可以将其换成快速排序来加快代码速度，并将其放入主函数内减少代码冗余，并将算法类代码继承于主类，使得排序代码只需要写一遍即可。  另外为了方便的使用提供的sort方法排序，将int类修为其应用类Integer。  package ch4;  import java.util.Arrays; import java.util.Comparator; import java.util.Scanner;  public class DiskManage {  **public Integer[] Sorts(Integer[] a, int n) {  Arrays.sort(a, new Comparator<Integer>() {  @Override  public int compare(Integer o1,Integer o2) {  return o2-o1;  }  });  return a;  }**  public static void main(String[] args) {  Integer numS[] = new Integer[20];  int n = 0;  int m = 0;  double sum = 0;  double temp = 0;  int position = 0;  SSTF sstf = new SSTF();  SCAN scan = new SCAN();  CSCAN cscan = new CSCAN();  System.out.println("请输入磁盘请求的个数：");  try {  Scanner reader = new Scanner(System.in);  n = reader.nextInt();  } catch (Exception e) {  System.out.println("null");  }  System.out.println("请输入磁盘请求序列：");  Scanner reader = new Scanner(System.in);  for (int i = 1; i <= n; i++) {  try {  numS[i] = reader.nextInt();  } catch (Exception e) {  }  }  System.out.print("请输入磁头的初始位置：");   // try{  Scanner re = new Scanner(System.in);  numS[0] = re.nextInt();  position = numS[0]; // }  // catch(for (int i = 1; i <= n; i++){Exception e){ }  // System.out.print(a[i] + " "); }  Post();  Scanner read = new Scanner(System.in);  try {  m = read.nextInt();  } catch (Exception e) {  }  System.out.println("m的值为" + m);  read.close();  switch (m) {  case 1:  sum = 0;  System.out.println("FCFS算法序列：");  for (int i = 1; i <= n; i++) {  System.out.print(numS[i] + " ");  temp = numS[i] - numS[i - 1];  if (temp >= 0) ;  else if (temp < 0) {  temp = -temp;  }  sum = sum + temp;  }  System.out.println();  System.out.println("sum的值为" + sum);  System.out.println("平均寻到长度为" + sum / n);  break;  case 2:  sstf.Calculate(numS, n, position);  break;  case 3:  scan.Check(numS, n, position);  break;  case 4:  cscan.Check(numS, n, position);  break;  }  }   private static void Post() {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("提示：磁头的序列范围1~200！磁头从小号到大号寻道！");  System.out.println("请输入相应的选择操作：：");  System.out.println("1、先来先服务算法（FCFS）");  System.out.println("2、最短寻道时间优先算法（SSTF）");  System.out.println("3、扫描算法（SCAN）");  System.out.println("4、循环扫描算法（CSCAN）");  } }  package ch4;  public class CSCAN extends DiskManage{  public int m = 0; //用来存放磁头的初始位置  public boolean Run = true;  public int sum = 0;   public void Check(Integer[] a, int n, int position) {  a = Sorts(a, n);//排序  while (Run) {  for (int i = 0; i <= n; i++) {  if (a[i] == position) {  m = i;  Run = false;  }  }  }  System.out.println("CSCAN算法序列：");  for (int i = m + 1; i <= n; i++) { //磁头向大号移动  sum = sum + a[i] - a[i - 1];  System.out.print(a[i] + " ");  }  sum = sum + 200 - a[n];  sum = sum + 199;  sum = sum + a[0] - 1;  for (int i = 0; i <= m - 1; i++) {  if (i != m - 1) {  sum = sum + a[i + 1] - a[i];  }  System.out.print(a[i] + " ");  }  System.out.println();  System.out.println("寻道长度为" + sum);  System.out.println("平均寻道长度为" + sum / n);  } }  package ch4;   public class SCAN extends DiskManage{  public int m = 0; //用来存放磁头的初始位置  public boolean Run = true;  public int sum = 0;   public void Check(Integer[] a, int n, int position) {  a = Sorts(a, n);//排序  while (Run) {  //此循环用来寻找磁头的初始位置被排到了什么位置  for (int i = 0; i <= n; i++) {  if (a[i] == position) {  m = i;  Run = false;  }  }  }  System.out.println("SCAN算法序列：");  for (int i = m + 1; i <= n; i++) {//磁头向大号移动  sum = sum + a[i] - a[i - 1];  System.out.print(a[i] + " ");  }  sum = sum + 200 - a[n];  sum = sum + 200 - a[m - 1];  for (int i = m - 1; i >= 0; i--) {  if (i != 0) {  sum = sum + a[i] - a[i - 1];  }  System.out.print(a[i] + " ");  }  System.out.println();  System.out.println("寻到长度为" + sum);  System.out.println("平均寻到长度为" + sum / n);  } }  package ch4;  public class SSTF extends DiskManage{  public int m; // 用来判断排序后 磁头所在的初始位置的下标  public int b[];  public int rflag = 0, lflag = 0;  public double SUM = 0;  public int mleft, mright;   public SSTF() {  b = new int[20];  }   public void Calculate(Integer[] a, int n, int position) {  a = Sorts(a, n);//排序  if (a[n] == position) {//如果磁头的初始位置在最大号  System.out.println("SSTF序列为：");  for (int i = 0; i <= n; i++) {  System.out.print(a[i] + " ");  }  } else if (a[0] == position) {  System.out.println("SSTF序列为：");  for (int i = n; i >= 0; i--) {  System.out.print(a[i] + " ");  }  } else {  System.out.println("SSTF序列为：");  Check(a, n, position);  }  }   public void Check(Integer[] a, int n, int position) {  boolean Run = true, Running = true;  while (Run) {//此循环用来寻找磁头的初始位置被排到了什么位置  for (int i = 0; i <= n; i++) {  if (a[i] == position) {  m = i;  System.out.println("m的值为" + m);  mleft = m - 1;  mright = m + 1;  Run = false;  }  }  }  while (Running) {  if (a[m] - a[mleft] >= a[mright] - a[m] && rflag != 1) {  SUM = SUM + (a[mright] - a[m]); //用来统计寻到长度  m = mright;  mright++;  if (mright > n) {  rflag = 1;  }  System.out.print(a[m] + " ");  } else if (a[m] - a[mleft] < a[mright] - a[m] && lflag != 1) {  SUM = SUM + (a[m] - a[mleft]);  m = mleft;  mleft--;  if (mleft < 0) {  lflag = 1;  }  System.out.print(a[m] + " ");  }  if (mleft != 0 && mright > n) {  Running = false;  SUM = SUM + a[n] - a[mleft];  for (int j = mleft; j >= 0; j--) {  if (j != 0) {  SUM = SUM + a[j] - a[j - 1];  }  System.out.print(a[j] + " ");  }  }  if (mright != n && mleft < 0) {  Running = false;  SUM = SUM + a[mright] - a[0];  for (int k = mright; k <= n; k++) {  if (k != 0) {  SUM = SUM + a[k + 1] - a[k];  }  System.out.print(a[k] + " ");  }  }  }  System.out.println();  System.out.println("sum的值为" + SUM);  System.out.println("平均寻到长度为" + SUM / n);  } }    **五、实验结果：** | | | |
| **六、实验小结：**  对JAVA各个类的使用和排序都有了很深刻的认识，对磁盘管理的各项算法有了更加细节上的了解和认知，在面对细节问题时仔细思考代码的使用和意义，就能很好的了解这些算法。 | | | |