**江苏科技大学**

**课程实验报告**

课 程： 操作系统原理与安全

课 题： 实验四·磁盘调度

学 院： 计算机学院

姓 名： 陈四贵

班 级： 1822107101

学 号： 182210710119

开发语言： Java

目 录

[一、 实验目的 1](#_Toc41664503)

[二、 实验内容 1](#_Toc41664504)

[三、 实验提示及要求 1](#_Toc41664505)

[四、 附图 2](#_Toc41664506)

[五、 实验代码及其说明 2](#_Toc41664507)

[1．总体结构 2](#_Toc41664508)

[2．分支结构 2](#_Toc41664509)

[⑴ 全局数据 2](#_Toc41664510)

[核心思想 2](#_Toc41664511)

[附·源代码 2](#_Toc41664512)

[⑵ init()函数 3](#_Toc41664513)

[核心思想 3](#_Toc41664514)

[附·源代码 3](#_Toc41664515)

[⑶DiskScheduled()函数 3](#_Toc41664516)

[核心思想 3](#_Toc41664517)

[附·源代码 4](#_Toc41664518)

[⑷seekB()函数 5](#_Toc41664519)

[核心思想 5](#_Toc41664520)

[附·源代码 5](#_Toc41664521)

[⑸seekS()函数 5](#_Toc41664522)

[核心思想 5](#_Toc41664523)

[附·源代码 5](#_Toc41664524)

[⑹printStatus()函数 6](#_Toc41664525)

[说明 6](#_Toc41664526)

[附·源代码 6](#_Toc41664527)

[⑺主函数 6](#_Toc41664528)

[说明 6](#_Toc41664529)

[附·源代码 6](#_Toc41664530)

[⑻I/O请求类 6](#_Toc41664531)

[说明 6](#_Toc41664532)

[附·源代码 6](#_Toc41664533)

[六、 运行截图 7](#_Toc41664534)

[七、 实验心得 8](#_Toc41664535)

# 实验目的

磁盘是高速、大容量、旋转型、可直接存取的存储设备。它作为计算机系统的辅助存储器，担负着繁重的输入输出工作，在现代计算机系统中往往同时会有若干个要求访问磁盘的输入输出要求。系统可采用一种策略，尽可能按最佳次序执行访 问磁盘的请求。由于磁盘访问时间主要受寻道时间 T 的影响，为此需要采用合适的寻道算法，以降低寻道时间。本实验要求学生模拟设计一个磁盘调度程序，观察调度程序的动态运行过程。通过实验让学生理解和掌握磁盘调度的职能。

# 实验内容

模拟电梯调度算法，对磁盘进行移臂操作。

# 实验提示及要求

1.假设磁盘只有一个盘面，并且磁盘是可移动头磁盘。

2.磁盘是可供多个进程共享的存储设备，但一个磁盘每个时刻只能为一个进 程服务。当有进程在访问某个磁盘时，其它想访问该磁盘的进程必须等待，直到磁盘一次工作结束。当有多个进程提出输入输出请求而处于等待状态时，可用电梯调度算法从若干个等待访问者中选择一个进程，让它访问磁盘。为此设置“驱动调度”进程。

3.由于磁盘与处理器是并行工作的，所以当磁盘在为一个进程服务时，占有 处理器的其它进程可以提出使用磁盘（这里我们只要求访问磁道），即动态 申请访问磁道，为此设置“接受请求”进程。

4.为了模拟以上两个进程的执行，可以考虑使用随机数来确定二者的允许顺

序，程序结构图参考附图：

5.“接受请求”进程建立一张“进程请求 I/O”表，指出等待访问磁盘的进程 要求访问的磁道，表的格式如下：

|  |  |
| --- | --- |
| 进程名 | 要求访问的磁道号 |
|  |  |
|  |  |

6.“磁盘调度”的功能是查“请求 I/O”表，当有等待访问的进程时，按电 梯调度算法（SCAN 算法）从中选择一个等待访问的进程，按其指定的要求 访问磁道。SCAN 算法参考课本第九章。算法模拟框图略。

7.附图中的“初始化”工作包括：初始化“请求 I/O”表，设置置当前移臂方 向；当前磁道号。并且假设程序运行前“请求 I/O”表中已有若干进程（4～ 8 个）申请访问相应磁道。

# 附图

# 实验代码及其说明

## 1．总体结构

整个实验采用面向对象的编程技术，按功能将各个部分模块化，分为init()函数（初始化）、DiskScheduled()函数（模拟Scan方法进行磁盘调度）、seekB()函数（向大方向寻找合适调度磁盘）、seekS()函数（向小方向寻找合适调度磁盘）、printStatus()函数（打印磁盘调度状态信息）、IO类（构建磁盘I/O请求对象）。

## 2．分支结构

### ⑴ 全局数据

核心思想：存放全局都会用到的数据，方便使用，增加程序可读性。

附·源代码：

1. **static** **int** Now\_number,index,temp,variable;
2. **static** **char** Now\_direction;
3. **static** IO p;
4. **static** List<IO> list= **new** LinkedList<>();
5. **static** Scanner scanner=**new** Scanner(System.in);

### ⑵ init()函数

核心思想：初始化I/O请求表，初始化移臂状态。

附·源代码：

1. **public** **static** **void** init() **throws** IOException{//初始化I/O请求表
2. System.out.println("我们需要您初始化I/O请求表，需要您的配合，谢谢。");
3. System.out.print("请您输入访问磁道的进程数量(范围：4~8)：");
4. **int** num=scanner.nextInt();
5. **while**(num<4||num>8){
6. System.out.print("进程数过大或过小，请重新输入：");
7. num=scanner.nextInt();
8. }
9. **int** pid,trackNum;
10. System.out.println("请您依次输入访问进程的Pid和访问磁道号（例：1  2）：");
11. **for**(**int** i=0;i<=num-1;i++){
12. pid=scanner.nextInt();
13. trackNum=scanner.nextInt();
14. p=**new** IO(pid,trackNum);
15. list.add(p);
16. }
17. System.out.print("请您输入当前所在磁道号：");
18. Now\_number=scanner.nextInt();
19. System.out.print("请您输入当前移臂方向(B/b代表向大的方向前进，S/s代表向小的方向前进）:");
20. Now\_direction = (**char**)System.in.read();
21. **while**(Now\_direction!='B'&&Now\_direction!='b'&&Now\_direction!='S'&&Now\_direction!='s'){
22. System.out.print("输入格式错误，请您重新输入移臂方向：");
23. Now\_direction = (**char**)System.in.read();
24. }
25. }

### ⑶DiskScheduled()函数

核心思想：模拟Scan电梯调度磁盘，控制流程，条件成熟时在seekB()和seekS()中轮换寻找。

附·源代码：

1. **public** **static** **void** DiskScheduled() **throws** IOException{
2. **do**{
3. System.out.print("现需要您随机输入一个[0,1]的随机数（模拟进程调度，便于控制流程）控制程序流程（<=0.5则进行磁盘调度，>0.5则接受请求）：");
4. Scanner scanner=**new** Scanner(System.in);
5. **double** choi=scanner.nextDouble();
6. **if**(choi<=0.5) {//磁盘调度
7. index =temp=variable=0; //variable记录最近值
8. **if** (Now\_direction == 'B' || Now\_direction == 'b') {//向大的方向走
9. seekB();
10. //若没有找到（悬臂到头啦），则调转过来，向另一方向进军
11. **if**((list.get(index).isControl)||((!list.get(index).isControl)&&(list.get(index).TrackNum<Now\_number))){//已被控制，或者是未被控制但不满足调度
12. Now\_direction='s';
13. seekS();
14. }
15. } **else** {//向小的方向走
16. seekS();
17. **if**((list.get(index).isControl)||((!list.get(index).isControl)&&(list.get(index).TrackNum>Now\_number))){
18. Now\_direction='b';
19. seekB();
20. }
21. }
22. list.get(index).isControl=**true**;
23. Now\_number=list.get(index).TrackNum;
24. printStatus();
25. }
26. **else**{//接受请求
27. System.out.print("请输入新的进程名和磁道号：");
28. **int** pid,trackNum;
29. pid=scanner.nextInt();
30. trackNum=scanner.nextInt();
31. p=**new** IO(pid,trackNum);
32. list.add(p);
33. printStatus();
34. }
35. System.out.print("是否继续？(y/n)：");
36. **char** dir=(**char**) System.in.read();
37. **while**(dir!='y'&&dir!='Y'&&dir!='n'&&dir!='N'){
38. System.out.print("格式错误，请您重新输入：");
39. dir=(**char**) System.in.read();
40. }
41. **if**(dir=='y'||dir=='Y');
42. **else** **break**;
43. }**while**(**true**);
44. }

### ⑷seekB()函数

核心思想：往悬臂方向（大）上寻找最近的磁盘进行调度。若磁盘已被调度，则应跳过该磁盘；磁道号需比当前磁盘号大，且为其中最小的，未被调度，才能被调度。

附·源代码：

1. **public** **static** **void** seekB(){
2. **for** (**int** i = 0; i < list.size(); i++) {//遍历查找
3. **if** ((!list.get(i).isControl) && list.get(i).TrackNum >= Now\_number) {//未被调用（访问）且在旋臂方向上
4. temp=list.get(i).TrackNum-Now\_number;
5. **if** ((temp>=0)&&((variable!=0&&temp<=variable)||(variable==0))) {//磁道号最小
6. variable = temp;
7. index = i;
8. }
9. }
10. }
11. }

### ⑸seekS()函数

核心思想：往悬臂方向（小）上寻找最近的磁盘进行调度。若磁盘已被调度，则应跳过该磁盘；磁道号需比当前磁盘号小，且为其中最大的，未被调度，才能被调度。

附·源代码：

1. **public** **static** **void** seekS(){
2. **for** (**int** i = 0; i < list.size(); i++) {//遍历查找
3. **if** ((!list.get(i).isControl) && list.get(i).TrackNum <= Now\_number) {//未被调用（访问）且在旋臂方向上
4. temp=Now\_number-list.get(i).TrackNum;
5. **if** ((temp>=0)&&((variable!=0&&temp<=variable)||(variable==0))) {
6. variable =  Now\_number-list.get(i).TrackNum;
7. index = i;
8. }
9. }
10. }
11. }

### ⑹printStatus()函数

说明：打印当前进程表调度信息。

附·源代码：

1. **public** **static** **void** printStatus(){
2. System.out.println("进程id\t磁道\t是否调度");
3. **for**(**int** i=0;i<list.size();i++){
4. System.out.println(list.get(i).Pid+"\t\t"+list.get(i).TrackNum+"\t\t"+list.get(i).isControl);
5. }
6. }

### ⑺主函数

说明：启动程序，调用函数。

附·源代码：

1. **public** **static** **void** main(String[] args) **throws** IOException {
2. init();
3. DiskScheduled();
4. }

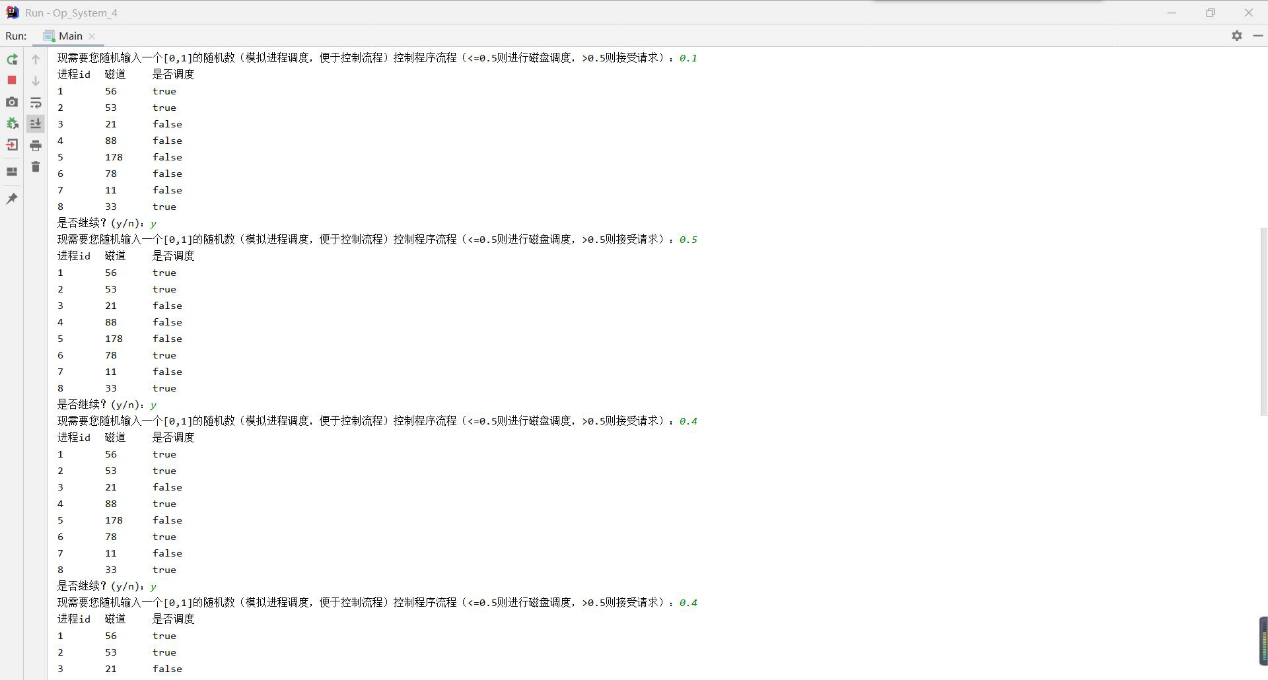
### ⑻I/O请求类

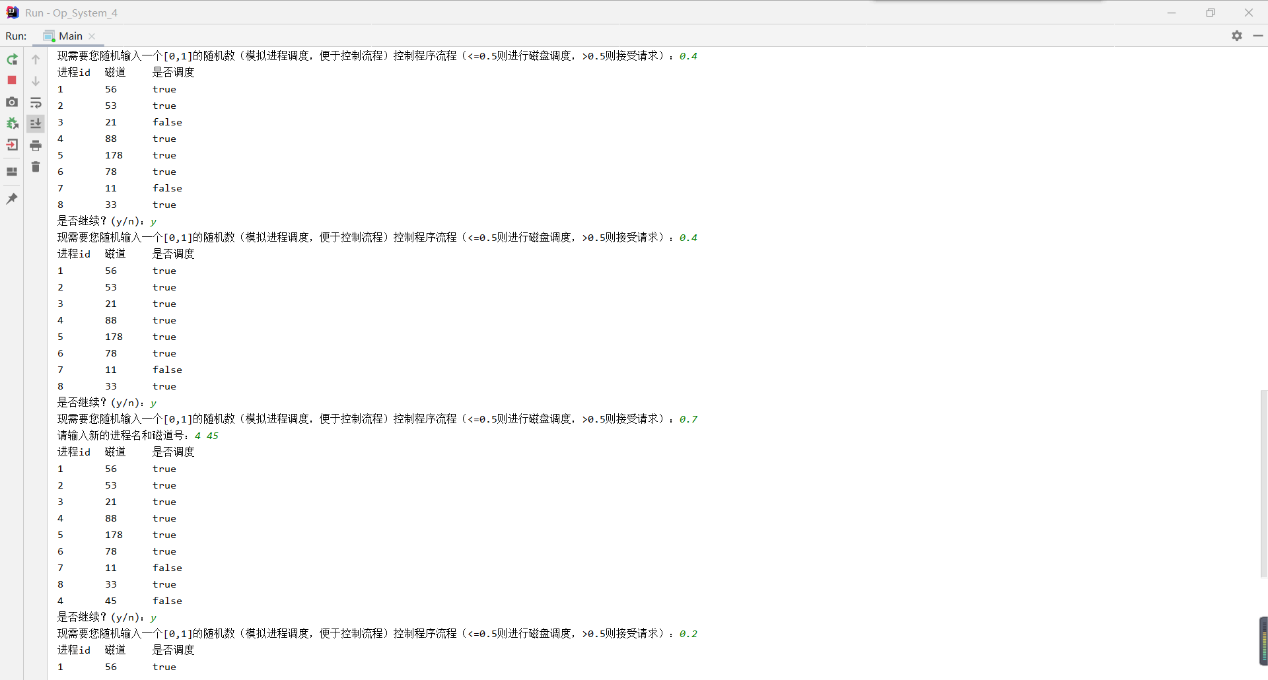
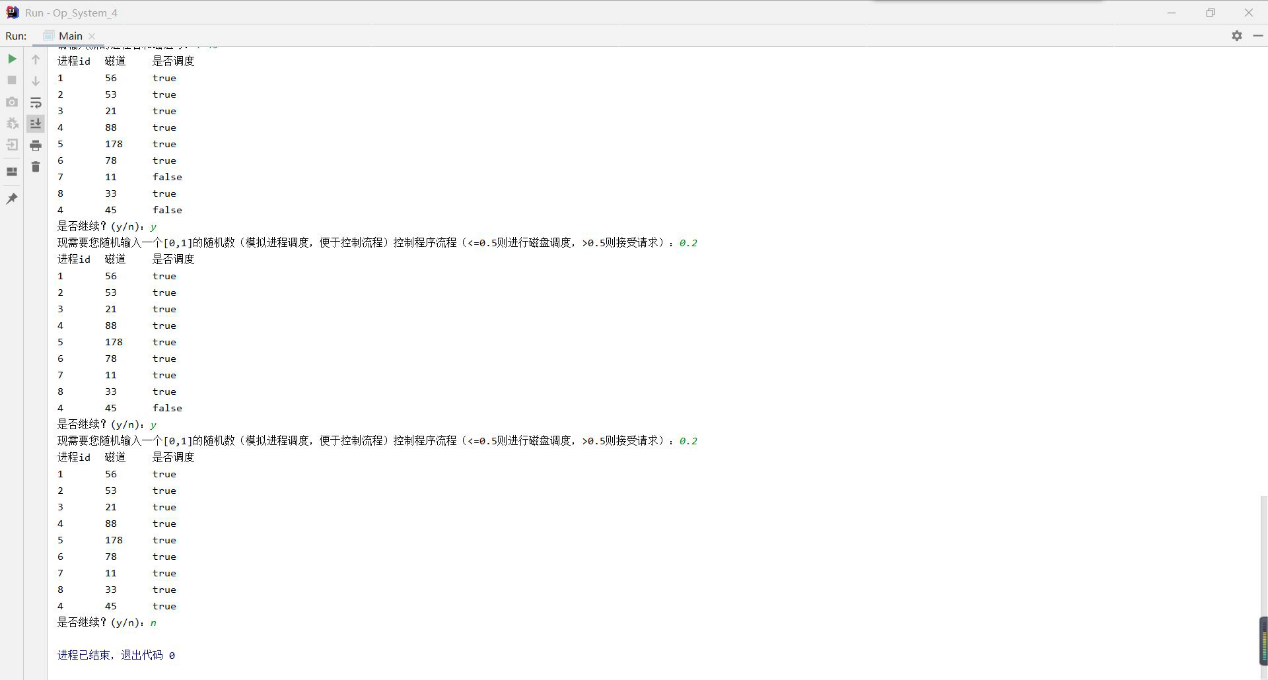
说明：封装了基本的I/O请求的基本信息，为构造I/O请求表作铺垫。

附·源代码：

1. **public** **static** **class** IO{
2. **public** **int** Pid;  //进程名
3. **public** **int** TrackNum;   //请求磁道号
4. **public** **boolean** isControl;  //调度状态量，true表示正在被调度，false表示未被调度
6. IO(**int** pid,**int** trackNum){
7. **this**.Pid=pid;
8. **this**.TrackNum=trackNum;
9. **this**.isControl=**false**;
10. }
11. }

# 运行截图





# 实验心得

1.为了提高计算机的响应速度，提升效率，我们有必要选择一个较高效的磁盘调度算法。其中，scan电梯调度算法是比较实际且比较常见的一种算法。它的同方向就近原则没有违背磁盘的读取方式，是较为高效的读取方式。

2.在生活中，这与部分电梯的响应楼层方式很像，同方向就近原则使得电梯的运输效率大大提升，磁盘调度也是如此。