

实验报告一

题目：用JAVA同步方法实现磁头引臂调度问题，采用SCAN算法。

目的：深入理解设备调度算法，能用同步方法解决调度问题。

要求：(1) 给出核心调度解法，用JAVA类实现，其中包含require(dest)和release()两个同步方法；(2) 创建若干线程或进程，分别提出某一磁道上某个磁盘块的访问请求，给出调度结果。

说明：(1)假定盘面上共有200个磁道，由外向内依次编号0,...,199，盘面只有一个移动磁头；(2)模拟访问磁盘时打印出磁道编号，并延迟一段时间以表示磁盘I/O操作时间；(3)假定所有块的I/O操作均为同步I/O。

结果：

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages `OperateSystem`, `JRE System Library [JavaSE-13]`, and source files `Device.java`, `Disk.java`, `Main.java`, and `module-info.java`.
- Code Editor:** Displays the `Disk.java` file with the following code:

```
84     int d=Math.abs(position-dest);
85     position=dest;
```
- Console Output:** Shows the execution results of the `Main` class:

```
<terminated> Main [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (2020年6月24日下午5:24:07 - 下午5:24:20)
当前磁道： 108 移向： -1
申请： 111磁道
申请： 131磁道
申请： 149磁道
申请： 34磁道
申请： 118磁道
正访问： 34 磁道（移动66个磁道）
释放： 34磁道
正访问： 111 磁道（移动77个磁道）
释放： 111磁道
正访问： 118 磁道（移动7个磁道）
释放： 118磁道
正访问： 131 磁道（移动13个磁道）
释放： 131磁道
正访问： 149 磁道（移动18个磁道）
释放： 149磁道
申请： 22磁道
申请： 179磁道
申请： 165磁道
正访问： 22 磁道（移动127个磁道）
释放： 22磁道
正访问： 105 磁道（移动83个磁道）
释放： 105磁道
正访问： 179 磁道（移动74个磁道）
释放： 179磁道
申请： 125磁道
申请： 117磁道
正访问： 125 磁道（移动54个磁道）
释放： 125磁道
正访问： 117 磁道（移动8个磁道）
释放： 117磁道
访问结束
```

```
1 package test;
2
3 public class Device implements Runnable {
4     Disk disk;
5
6     public Device(Disk disk) {
7         this.disk = disk;
8     }
9
10    @Override
11    public void run() {
12        while (disk.getTimes()>0)
13        {
14            int max=199,min=0;
15            int randomDest = (int) (Math.random()*(max-min)+min);
16            disk.require(randomDest);
17            try {
18                Thread.sleep(1500);
19            } catch (InterruptedException e) {
20                e.printStackTrace();
21            }
22        }
23    }
24 }
25 }
```

```
1 package test;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         int threadNumber=6;
7         int deviceNumber=threadNumber-1;
8         Disk disk=new Disk(100,-1);
9         Thread[]threads=new Thread[threadNumber];
10        Device[]devices=new Device[deviceNumber];
11        threads[0]=new Thread(disk,Integer.toString(0));
12        for (int i=1;i<threadNumber;i++)
13        {
14            devices[i-1]=new Device(disk);
15            threads[i]=new Thread(devices[i-1],Integer.toString(i));
16        }
17
18        for (int i=0;i<threadNumber;i++)
19        {
20            threads[i].start();
21        }
22    }
23 }
```

代码：

```
1 package test;
2
3
4 public class Disk implements Runnable {
5     int[]dests;
6     int position;
7     int flag;
8     int times;
9
10    public synchronized int getTimes() {
11        return times;
12    }
13
14    public Disk(int position, int flag) {
15        this.position = position;
16        this.flag = flag;
17        dests=new int[200];
18        times=10;
19        System.out.println("当前磁道: "+position+" 移向: "+flag);
20    }
21
22    @Override
23    public void run() {
24        try {
25            Thread.sleep(1000);
26        } catch (InterruptedException e) {
27            e.printStackTrace();
28        }
29        boolean result=false;
30        while (times>0)
31        {
32            try {
33                result=manage();
34            } catch (InterruptedException e) {
35                e.printStackTrace();
36            }
37            if (result)
38            {
39                times--;
40            }
41        }
42        System.out.println("访问结束");
43    }
44
45    private synchronized boolean manage() throws InterruptedException {
46        int nextDest= scan();
47        if(nextDest!=-1)
48        {
49            using(nextDest);
50            release();
51            return true;
52        }
53        else
54        {
55            return false;
56        }
57    }
58    void require(int dest)
59    {
60        System.out.println("申请: "+dest+"磁道");
61        dests[dest]++;
62    }
63    private int scan()
64    {
65        int next=-1;
66        next=hasNext();
67        if (next!=-1)
68        {
69            return next;
70        }
71        else
72        {
73            flag=-flag;
74            next=hasNext();
75            if(next!=-1)
76            {
77                return next;
78            }
79        }
80        return -1;
81    }
82
83    private synchronized void using(int dest) throws InterruptedException {
84        int d=Math.abs(position-dest);
85        position=dest;
86        System.out.println("正访问: "+dest+" 磁道 (移动"+d+"个磁道)");
87        //模拟访问过程
88        try {
89            Thread.sleep(1000);
90        }catch (Exception e)
91        {
92        }
93        return;
94    }
95
96    private synchronized void release()
97    {
98        dests[position]--;
99        System.out.println("释放: "+position+"磁道");
100    }
101
102    private int hasNext()
103    {
104        for(int i=position;i<dests.length&&i>0;i+=flag)
105        {
106            if(dests[i]>0)
107            {
108                return i;
109            }
110        }
111        return -1;
112    }
113 }
```