

# 实验报告二

题目：用JAVA语言实现双磁头引臂调度SCAN算法。

目的：理解硬件提供的多部件支持，并设计算法利用硬件支持提高系统性能。

要求：同实验1。

方法：同实验1。

说明：仍假定盘面上共有200个磁道，由外向内依次编号0,...,199，同一引臂上两个磁头head1和head2，二者相距100个磁道，复位时head1位于磁道0，head2位于磁道100，head1负责0..99磁道上的I/O请求，head2负责100..199磁道上的I/O请求。例如，当head1位于磁道35时，head2位于磁道135。

结果：

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

- Package Explorer:** Shows the project structure with packages JRE System Library [JavaSE-13], OperateSystem, and test.
- Disk.java:** The active code editor window contains Java code for a Disk class. It includes methods for handling I/O requests and printing log messages about head movement and current position.
- Console:** The output window displays the execution results of the code. It shows the disk's current position at 100, followed by a series of seek and print operations. The log entries indicate the head moving between tracks 49 and 132, and then returning to track 41 before finishing at track 108.
- Task List:** A panel on the right side of the interface.

```
110     System.out.println("释放: "+position+"磁道");
111 }
<terminated> Main [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (2020年6月24日下午5:41):
当前磁道: 100 移向: -1
申请: 49磁道
申请: 152磁道
申请: 158磁道
申请: 98磁道
申请: 88磁道
正在访问磁道: 49 (移动了51个磁道)
释放: 49磁道
正在访问磁道: 152 (移动了3个磁道)
释放: 152磁道
正在访问磁道: 158 (移动了6个磁道)
释放: 158磁道
正在访问磁道: 88 (移动了30个磁道)
释放: 88磁道
正在访问磁道: 88 (移动了0个磁道)
释放: 88磁道
申请: 132磁道
申请: 41磁道
正在访问磁道: 132 (移动了56个磁道)
申请: 43磁道
释放: 132磁道
正在访问磁道: 41 (移动了9个磁道)
释放: 41磁道
正在访问磁道: 43 (移动了2个磁道)
释放: 43磁道
申请: 64磁道
申请: 106磁道
正在访问磁道: 64 (移动了21个磁道)
释放: 106磁道
申请: 64磁道
正在访问磁道: 108 (移动了56个磁道)
释放: 108磁道
访问结束
```

代码：

```
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    package test;
21
22    public class Device implements Runnable {
23        Disk disk;
24
25
26        public Device(Disk disk) {
27            this.disk = disk;
28        }
29
30
31        @Override
32        public void run() {
33            while (disk.getTimes()>0)
34            {
35                int max=199,min=0;
36                int randomDest = (int) (Math.random()*(max-min)+min);
37                disk.require(randomDest);
38                try {
39                    Thread.sleep(1500);
40                } catch (InterruptedException e) {
41                    e.printStackTrace();
42                }
43            }
44        }
45    }
```

```

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=0;
85        if((dest-(dests.length/2))*(position-(dests.length/2))<0)
86        {
87            if(dest>(dests.length/2))
88            {
89                position=position+(dests.length/2);
90            }
91            else
92            {
93                position=position-(dests.length/2);
94            }
95        }
96        d=Math.abs(position-dest);
97        position=dest;
98        System.out.println("正访问"+dest+" 磁道 (移动了"+d+"个磁道)");
99        try {
100            Thread.sleep(1000);
101        }catch (Exception e)
102        {
103        }
104        return;
105    }
106
107    private synchronized void release()
108    {
109        dests[position]--;
110        System.out.println("释放: "+position+"磁道");
111    }
112
113    private int hasNext()
114    {
115        int head1;
116        int head2;
117        if(position<dests.length/2)
118        {
119            head1=position;
120            System.out.println("释放: "+position+"磁道");
121        }
122    }
123    private int hasPrev()
124    {
125        int head1;
126        int head2;
127        if(position<dests.length/2)
128        {
129            head1=position;
130            head2=position+(dests.length/2);
131        }
132        else
133        {
134            head1=position-(dests.length/2);
135            head2=position;
136        }
137        for(head2<dests.length&&head1>=0;head1+=flag,head2+=flag)
138        {
139            if(dests[head1]>0)
140            {
141                return head1;
142            }
143            else if ([dests[head2]>0]
144            {
145                return head2;
146            }
147        }
148        return -1;
149    }
150}

```