

实验报告：磁盘调度模拟实验

姓名：陈晓彤 学号：PB14000556 时间：2016.5.21

实验要求：用 C 语言设计模拟一个磁盘调度算法，并测试其正确性。

输入：磁头所在的位置及对块的调度序列

输出：对块的调度访问序列

实验背景：基本的磁盘调度算法有 先来先服务算法 最短寻道时间优先算法 扫描算法 循环扫描算法等。

算法实现：本实验选择实现最短寻道时间优先算法，即每次寻找与当前磁头位置距离最近的磁道。先将需要访问的磁道排序，如果磁头在最左端或最右端，则只需一次遍历访问即可，否则先找到距离磁头最近的磁道，访问并输出，然后比较其左侧和右侧未被访问的磁道哪个距此更近，选择近者作为下一个访问的磁道，知道有一侧的磁道全部被访问，则向另一侧扫描逐个访问输出。

```
#include<stdio.h>
```

```
#include<time.h>
```

```
#include<stdlib.h>
```

```
int main()
```

```
{
```

```
    srand((unsigned)time(NULL));
```

```
    int pos[20],flag[20];
```

```
    int i,j;
```

```
    for(i=0;i<20;i++)
```

```
    {
```

```
        pos[i]=rand()%128;/*产生 20 个访问的磁道数*/
```

```
    }
```

```
    printf("磁道序列:\n");
```

```
    for(i=0;i<20;i++)
```

```
    {
```

```
        printf("%d ",pos[i]);
```

```
    }
```

```
    int temp;
```

```
    for(i=0;i<20;i++)
```

```
    {
```

```
        for(j=0;j<20-i-1;j++)
```

```
        {
```

```
            if(pos[j]>pos[j+1])
```

```
            {
```

```
                temp=pos[j];pos[j]=pos[j+1];pos[j+1]=temp;/*按磁道数排序*/
```

```
            }
```

```
        }
```

```
        flag[i]=0;
```

```
    }
```

```
    printf("\n");for(i=0;i<20;i++)
```

```
    {
```

```

        printf("%d ",pos[i]);
    }
    printf("\n 输入磁头位置:\n");
    int init;
    scanf("%d",&init);
    int nearest;
    int left_all=0,right_all=0,next1,next2;/*标记磁头是否已到达最左端或最右端*/
    if(pos[0]>init)
    {
        for(i=0;i<20;i++)
            printf("%d ",pos[i]);/*若磁头已到达最左端，则直接向右扫描输出*/
        return 0;
    }
    else if(pos[19]<init)
    {
        for(i=19;i>-1;i++)
            printf("%d ",pos[i]);/*若磁头已到达最右端，则直接向左扫描输出*/
        return 0;
    }
    else
    {
        for(i=0;i<20;i++)/*其余情况，比较当前磁头位置左右最近的未扫描过的磁道的远近，
选择近者*/
        {
            if(pos[i]<init && (init<pos[i+1] || init==pos[i+1]))
                break;
        }
        nearest=((init-pos[i])>(pos[i+1]-init))?(i+1):i;
        flag[nearest]=1;printf("%d ",pos[nearest]);/*找到与初始磁头最近的磁道，输出标记
并从此开始*/
        for(j=0;j<20;j++)
        {
            if(left_all)
            {
                for(i=nearest+1;i<20;i++)
                {
                    if(!flag[i])
                        printf("%d ",pos[i]);/*磁头左端磁道都已被访问，直接向右遍历扫描并访问输出*/
                }
            }
            return 0;
        }
        else if(right_all)
        {

```

```

        for(i=nearest-1;i>-1;i--)
        {
            if(!flag[i])
                printf("%d ",pos[i]);/*磁头右端磁道都被访问，直接向左遍历扫描并访问输出*/
        }
        return 0;
    }
    else
    {
        next1=nearest-1;next2=nearest+1;
        while(next1 && flag[next1])/*找距离最近的左侧的未访问磁道*/
            next1--;
        while(next2 && flag[next2])/*找距离最近的右侧的未访问磁道*/
            next2++;
        if(next1<0)
        {
            left_all=1;/*左端磁道都被访问，标记*/
        }
        else if(next2>19)
        {
            right_all=1;/*右端磁道都被访问，标记*/
        }
        else
        {
            nearest=((pos[next2]-pos[next1])>(pos[next2]-pos[next1]))?next2:next1;/*找到下一个访问磁道位置，输出并标记*/
            printf("%d ",pos[nearest]);flag[nearest]=1;
        }
    }
}
return 0;
}

```

```
"E:\lessons\cod\extra hw\hw3\disk_read\bin\Debug\disk_read.exe"
磁道序列:
123 81 52 9 127 109 36 47 108 76 81 1 34 94 65 4 106 85 42 30
1 4 9 30 34 36 42 47 52 65 76 81 81 85 94 106 108 109 123 127
输入磁头位置:
37
36 34 30 42 47 52 65 76 81 81 85 94 106 108 109 123 127 9 4 1
Process returned 0 (0x0)   execution time : 2.922 s
Press any key to continue.
```

微软拼音 半 :

可以看到访问序列与预期情况相同，结果正确