实验八、磁盘移臂调度算法实验

8.1 实验目的

加深对于操作系统设备管理技术的了解，体验磁盘移臂调度算法的重要性；掌握几种重要的磁盘移臂调度算法，练习模拟算法的编程技巧，锻炼研究分析试验数据的能力。

8.2 实验说明

1. 示例实验程序中模拟两种磁盘移臂调度算法：SSTF 算法和 SCAN 算法
2. 能对两种算法给定任意序列不同的磁盘请求序列，显示响应磁盘请求的过程。
3. 能统计和报告不同算法情况下响应请求的顺序、移臂的总量。比较两种算法在给定条件下的优劣。
4. 为了能方便的扩充磁盘移臂调度算法，更好的描述磁盘移臂调度过程，示例实验程序采用了 C++语言用 DiskArm 类描述了磁盘移臂调度算法及其属性。

8.3 示例实验

1）在新建文件夹中建立以下 **dask.h** 文件：

```
/*
   *    Filename              :    dask.h
   *    copyright             : (C) 2006 by zhonghonglie
    * Function               : 声明磁盘移臂调度类
*/
#include <iostream>
#include <iomanip.h>
#include <malloc.h>

class DiskArm{
public:
      DiskArm();
      ~DiskArm();
      void InitSpace(char * MethodName); //初始化寻道记录
      void Report(void); //  报告算法执行情况
      void Fcfs(void); //先来先服务算法
      void Sstf(void);    //最短寻道时间优先算法
      void Scan(void); //电梯调度算法
      void CScan(void); //均匀电梯调度算法
      void Look(void); //LOOK 调度算法
private:
      int *Request ;        //磁盘请求道号
      int *Cylinder;        //工作柱面道号号
```

```
    int RequestNumber;          //磁盘请求数
    int CurrentCylinder;    //当前道号
    int SeekDirection;          //磁头方向
    int SeekNumber;         //移臂总数
    int SeekChang;          //磁头调头数
};
```

2) 在新建文件夹中建立以下 **dask.cc** 文件：

```
/*
   *   Filename            :   dask.cc
   *   copyright           : (C) 2006 by zhonghonglie
    * Function             : 磁盘移臂调度算法
*/

#include "dask.h"
DiskArm::DiskArm(){
  int i;
   //输入当前道号
   cout << "Please input Current cylinder :" ;
   cin >> CurrentCylinder;
   //磁头方向，输入 0 表示向小道号移动，1 表示向大道号移动
   cout << "Please input Current Direction (0/1) :" ;
   cin >> SeekDirection;
   //输入磁盘请求数，请求道号
   cout << "Please input Request Numbers :" ;
   cin >> RequestNumber;
   cout << "Please input Request cylinder string :";
   Request = new int[sizeof(int) * RequestNumber];
   Cylinder = new int[sizeof(int) * RequestNumber];
   for (i = 0; i < RequestNumber; i++)
       cin >> Request[i];
  }

DiskArm::~DiskArm(){

}
//初始化道号，寻道记录
void DiskArm::InitSpace(char * MethodName)
{
   int i;
   cout << endl <<    MethodName << endl;
   SeekNumber = 0;
   SeekChang = 0;
   for (i = 0; i < RequestNumber; i++)
          Cylinder[i] = Request[i];
   }
```

```
// 统计报告算法执行情况
void DiskArm::Report(void){
    cout << endl;
    cout <<   "Seek Number: " << SeekNumber << endl;
    cout <<   "Chang Direction: " << SeekChang << endl << endl;
}

//先来先服务算法
void DiskArm::Fcfs(void)
{
      int   Current = CurrentCylinder;
      int   Direction = SeekDirection;
      InitSpace("FCFS");

      cout << Current;
      for(int i=0;   i<RequestNumber; i++){
          if(((Cylinder[i] >= Current) && !Direction)
            ||((Cylinder[i] <   Current) &&   Direction)){
              //需要调头
                SeekChang++;   //调头数加 1
              Direction = !Direction ; //改变方向标志
              //报告当前响应的道号
              cout   <<   endl << Current << " -> " << Cylinder[i];
          }
          else //不需调头，报告当前响应的道号
               cout << " -> " << Cylinder[i] ;
          //累计寻道数，响应过的道号变为当前道号
          SeekNumber += abs(Current -Cylinder[i]);
          Current = Cylinder[i];
    }
     //报告磁盘移臂调度的情况
    Report();
}

//最短寻道时间优先算法
void DiskArm::Sstf(void)
{
      int   Shortest;
      int   Distance = 999999 ;
      int   Direction = SeekDirection;
      int   Current = CurrentCylinder;
      InitSpace("SSTF");
      cout << Current;
      for(int i=0; i<RequestNumber; i++){
          //查找当前最近道号
          for(int j=0; j<RequestNumber; j++){
             if(Cylinder[j] == -1) continue;   //-1 表示已经响应过了
```

```
            if(Distance    > abs(Current-Cylinder[j])){
                //到下一道号比当前距离近，下一道号为当前距离
                  Distance    = abs(Current-Cylinder[j]);
                  Shortest = j;
              }
            }
          if((( Cylinder[Shortest] >= Current) && !Direction)
             ||((    Cylinder[Shortest] < CurrentCylinder) && Direction)){
               //需要调头
                  SeekChang++;   //调头数加 1
                Direction = !Direction ;   //改变方向标志
                //报告当前响应的道号
                cout    <<    endl <<    Current << " -> " <<    Cylinder[Shortest];
           }
           else //不需调头，报告当前响应的道号
               cout << " -> " << Cylinder[Shortest] ;

          //累计寻道数，响应过的道号变为当前道号
            SeekNumber += abs(Current -Cylinder[Shortest]);
          Current = Cylinder[Shortest];
          //恢复最近距离，销去响应过的道号
          Distance = 999999;
          Cylinder[Shortest] = -1;
         }

       Report();

}

//电梯调度算法
void DiskArm::Scan(void){

}

//均匀电梯调度算法
void DiskArm::CScan(void){

}

//LOOK 调度算法
void DiskArm::Look(void)
{

}
//程序启动入口
int main(int argc,char *argv[]){
   //建立磁盘移臂调度类
```

```
    DiskArm *dask = new DiskArm();
    //比较和分析 FCFS 和 SSTF 两种调度算法的性能
    dask->Fcfs();
    dask->Sstf();
}
```

3）在新建文件夹中建立以下 **Makefile** 文件

```
head = dask.h
srcs = dask.cc
objs = dask.o
opts = -w -g -c
all:        dask
dask:       $(objs)
            g++    $(objs) -o dask
dask.o:     $(srcs) $(head)
            g++ $(opts) $(srcs)
clean:
            rm dask *.o
```

4) 执行 **make** 命令编译连接，生成可执行文件 **dask**

```
$ g make
g++    -w -g -c dask.cc
g++    dask.o -o dask
```

5) 执行 **dask** 命令，输入当前道号，当前寻道方向，当前请求寻道数，当前请求寻道的道号串：

```
./dask
Please input Current cylinder : 53
Please input Current Direction (0/1) : 0
Please input Request Numbers : 8
Please input Request cylinder string : 98 183 37 122 14 124 65 67

FCFS
53
53 -> 98 -> 183
183 -> 37
37 -> 122
122 -> 14
14 -> 124
124 -> 65
65 -> 67
Seek Number: 640
Chang Direction: 7
SSTF
53
53 -> 65 -> 67
67 -> 37 -> 14
14 -> 98 -> 122 -> 124 -> 183
Seek Number: 236
```

```
Chang Direction: 3
$
```

可以看到以上程序的执行演示了 FCFS 和 SSTF 两种磁盘移臂调度算法响应磁盘请求的次序(其中每换一行表示磁头发生调头)。统计出了这两种算法的调度性能，从中看出在以上磁盘柱面请求序列下 SSTF 调度算法所产生的磁头移动为 236 柱面，约为 FCFS 调度算法所产生的磁头移动数量 640 柱面的三分之一稍多一点。磁头调头数 3 次也比 FCFS 调度算法的 7 次少了 4 次。因此对于以上磁盘柱面请求序列,SSTF 调度算法将比 FCFS 调度算法大大提高了磁盘的响应速度。

8.4 独立实验

请在以上示例实验程序中补充 SCAN，C-SCAN，LOOK 磁盘移臂调度算法的模拟程序。输入不同的磁盘柱面请求序列，观察和分析其调度效果和性能，并将其与 FCFS 和 SSTF 算法进行比较。改进以上示例实验程序，使之能够随机的产生磁盘柱面请求序列，以便能动态的观测各种调度算法的性能。

8.5 实验要求

1. 说明您做了哪些磁盘请求序列的测试，发现了哪些现象？
2. 选择一些典型磁盘请求序列的响应结果，画出不同算法中的寻道曲线图。
3. 说明您的程序是怎样模拟 SCAN 等算法的？
4. 综合分析实验结果中各种算法各适应于怎样的磁盘柱面寻道请求情况。
5. 根据实验程序、调试过程和结果分析写出实验报告