山东大学 计算机科学与技术 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名：张文浩 | | 班级： 19人工智能 |
| 实验题目：磁盘移臂调度算法实验 | | | |
| 实验学时：2 | | 实验日期： 5.24 | |
| 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。模拟SSTF和SCAN算法，比较算法在给定条件下的优劣。比较分析SCAN,C-SCAN,LOOK等算法的优劣，了解不同算法的寻道曲线图，分析各种算法适合怎样的寻道请求情况。 | | | |
| 硬件环境： 笔记本电脑 | | | |
| 软件环境：ubuntu20 | | | |
| 实验步骤与内容：   1. 根据指导书完成实例实验，编译运行，结果如下     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。   1. 独立实验: 补充SCAN，CSCAN，LOOK磁盘调度算法，源代码在最后。   每个算法的思路：  **FCFS：**  根据访问请求的先后次序选择先提出访问请求的为之服务。    程序运行结果如下：    寻道独立640，转向次数7.  **SSTF：**  以磁头移动距离的大小作为优先的因素，从当前磁头位置出发，选择离磁头最近的磁道为其服务。    程序运行结果如下：    寻道距离236，转向次数3.  **SCAN：**  是选请求队列中沿磁头臂前进方向最接近于磁头所在柱面的访问请求作为下一个服务对象。    程序运行结果如下：    寻道距离236，转向次数1.  **CSCAN：**  磁盘单向移动。当移动臂向内移动时，它对本次移动开始前到达的各访问要求自外向内地依次给予服务，直到对最内柱面上的访向要求满足后，然后移动臂直接向外移动，停在所有新的访问要求的最外边的柱面上。然后再对本次移动前到达的各访问要求依次给予服务。    程序运行结果如下：    寻道距离386，转向次数2  **LOOK：**    因为SCAN和C-SCAN中每次磁头都要走到磁道尽头，而实际过程中并不需要要求磁头走到尽头，而是到达该方向的最后一个请求后即可返回，这样可以避免一些不必要的磁头移动。  程序运行结果如下：  寻道距离208，转向次数1  **分析：**  **FCFS：**  优点:  公平、简单、每个进程请求都能依次得到处理，不会出现某一进程的请求长期得不到满足。  缺点:  平均寻道时间有点长，适用于磁盘I/O进程数目较少的场合  **SSTF：**  缺点:  优先级低的进程会发生“饥饿”现象。因为新进程请求到达，且其所要访问的磁道与磁头当前所在的磁道距离较近，必先优先满足。  **SCAN：**  优点:  不仅考虑到欲访问的磁道与当前磁道间的距离，更优先考虑了磁头当前的移动方向；避免了出现“饥饿”现象。被广泛用于大、中、小型机器和网络中的磁盘调度。  缺点  当磁道刚从里向外移动而越过了某一磁道时，刚好一进程请求访问此磁道，这时此进程会等待，待磁头继续从里向外，然后从外向里扫描完处于外面的所有要访问的磁道后，才处理此进程，致使该进程的请求被大大推迟。  **CSCAN：**  优点：  弥补扫描算法的不足  缺点：  因为会从最大磁道直接跳到最小磁道，或从最小磁道跳到最大磁道，所以寻道距离会比较大。  **LOOK：**  优点：  改进了电梯算法，避免了不必要的磁头移动，减少了寻道距离。 | | | |
| 结论分析与体会：  通过本次实验，我加深了对于操作系统设备管理技术的了解，体验了磁盘移臂调度算法的重要性；掌握了几种重要的磁盘移臂调度算法，练习模拟了算法的编程技巧，锻炼了研究分析试验数据的能力。模拟了SSTF和SCAN算法，比较了算法在给定条件下的优劣。比较了分析SCAN,C-SCAN,LOOK等算法的优劣，了解了不同算法的寻道曲线图，分析了各种算法适合怎样的寻道请求情况。总之，本次实验让我对磁盘调度问题有了更加深刻的理解，对几种调度算法进行了编程时间，对算法的原理有了更加清晰的认识，明白了不同算法的优缺点以及各自的特点和用途，收获很大。 | | | |

附源代码

**Dask.h**

#include <iostream>

#include <iomanip>

#include <malloc.h>

#include <cmath>

#include <stdlib.h>

using namespace std;

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

};

**Dask.cc**

#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();

}

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){

int Current = CurrentCylinder;

int Direction = SeekDirection;

InitSpace("SCAN");

cout << Current;//汇报一开始在哪个磁道

for(int i=0; i<RequestNumber; i++){

int index=-1;

int Distance = 999999;

for(int j=0;j<RequestNumber;j++){

if(Cylinder[j]==-1)//已经扫描过该磁道了

continue;

else if((Direction==0&&Cylinder[j]<Current&&(Current-Cylinder[j])<Distance)

||(Direction==1&&Cylinder[j]>Current&&(Cylinder[j]-Current)<Distance)){//找到一个同方向最近的一个磁道

index=j;

Distance=abs(Current-Cylinder[j]);

}

}

if(Direction==0){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Current-Cylinder[index];

Current=Cylinder[index];

Cylinder[index]=-1;

}else{//如果同方向上没有请求的磁道了，就跳到0号磁道并转向

cout<<"->"<<0<<endl;

SeekNumber+=Current;

Direction=!Direction;

//cout<<0;

Current=0;

SeekChang++;

i--;

}

}

else if(Direction==1){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Cylinder[index]-Current;

Current=Cylinder[index];

Cylinder[index]=-1;

}else{

cout<<"->"<<199<<endl;

SeekNumber+=199-Current;

Direction=!Direction;

//cout<<199;

Current=199;

SeekChang++;

i--;

}

}

}

//报告磁盘移臂调度的情况

Report();

}

//Look调度算法

void DiskArm::Look(void){

int Current = CurrentCylinder;

int Direction = SeekDirection;

InitSpace("Look");

cout << Current;

for(int i=0; i<RequestNumber; i++){

int index=-1;

int Distance = 999999;

for(int j=0;j<RequestNumber;j++){

if(Cylinder[j]==-1)

continue;

else if((Direction==0&&Cylinder[j]<Current&&(Current-Cylinder[j])<Distance)

||(Direction==1&&Cylinder[j]>Current&&(Cylinder[j]-Current)<Distance)){

index=j;

Distance=abs(Current-Cylinder[j]);

}

}

if(Direction==0){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Current-Cylinder[index];

Current=Cylinder[index];

Cylinder[index]=-1;

}else{

//与scan算法的区别在于如果当前方向前方没有请求的磁道了，就掉头，不再走到尽头了

//cout<<Current<<endl;

Direction=!Direction;

SeekChang++;

i--;

}

}

else if(Direction==1){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Cylinder[index]-Current;

Current=Cylinder[index];

Cylinder[index]=-1;

}else{

//cout<<Current<<endl;

Direction=!Direction;

SeekChang++;

i--;

}

}

}

//报告磁盘移臂调度的情况

Report();

}

//CScan 调度算法

void DiskArm::CScan(void)

{

int Current = CurrentCylinder;

int Direction = SeekDirection;

InitSpace("CScan");

cout << Current;

for(int i=0; i<RequestNumber; i++){

int index=-1;

int Distance = 999999;

for(int j=0;j<RequestNumber;j++){

if(Cylinder[j]==-1)

continue;

else if((Direction==0&&Cylinder[j]<Current&&(Current-Cylinder[j])<Distance)

||(Direction==1&&Cylinder[j]>Current&&(Cylinder[j]-Current)<Distance)){

index=j;

Distance=abs(Current-Cylinder[j]);

}

}

if(Direction==0){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Current-Cylinder[index];

Current=Cylinder[index];

Cylinder[index]=-1;

}else{

cout<<"->"<<0<<endl;

SeekNumber+=Current+199;

Current=199;

cout<<"0->199";

SeekChang+=2;

i--;

}

}

else if(Direction==1){

if(index!=-1){

cout<<"->"<<Cylinder[index];

SeekNumber+=Cylinder[index]-Current;

Current=Cylinder[index];

Cylinder[index]=-1;

}else{

cout<<"->"<<199<<endl;

SeekNumber+=199-Current+199;

Current=0;

SeekChang+=2;

i--;

}

}

}

Report();

}

//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();

dask->Scan();

dask->CScan();

dask->Look();

}

**Makefile**

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