## 实验3 驱动调度 实验报告

信息安全1501沈鑫楠

3150604028

### 1.实验题目

模拟电梯调度算法，对磁盘进行移臂和旋转调度。

### 2.实验目的

磁盘是一种高速、大容量、旋转型、可直接存取的存储设备。它作为计算机系统的辅 助存储器，担负着繁重的输入输出任务、在多道程序设计系统中，往往同时会有若干个要求 访问磁盘的输入输出请求等待处理。系统可采用一种策略，尽可能按佳次序执行要求访问 磁盘的诸输入输出请求。这就叫驱动调度，使用的算法称为驱动调度算法。驱动调度能降低 为若干个输入输出请求服务所需的总时间，从而提高系统效率。本实验要求学生模拟设计一 个驱动调度程序，观察驱动调度程序的动态运行过程。通过实验使学生理解和掌握驱动调度 的职能。

### 3.程序中使用的数据结构及符号说明

请求I/O表使用结构体类型的数据结构来描述。

struct Table

{

string process\_name;//进程名

int cylinder\_num;//柱面号

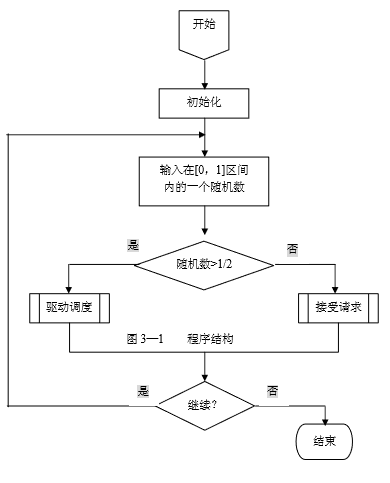
int track\_num;//磁道号

int record\_num;//物理记录号

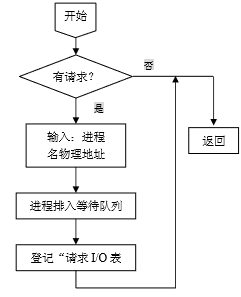
};

### 4.流程图

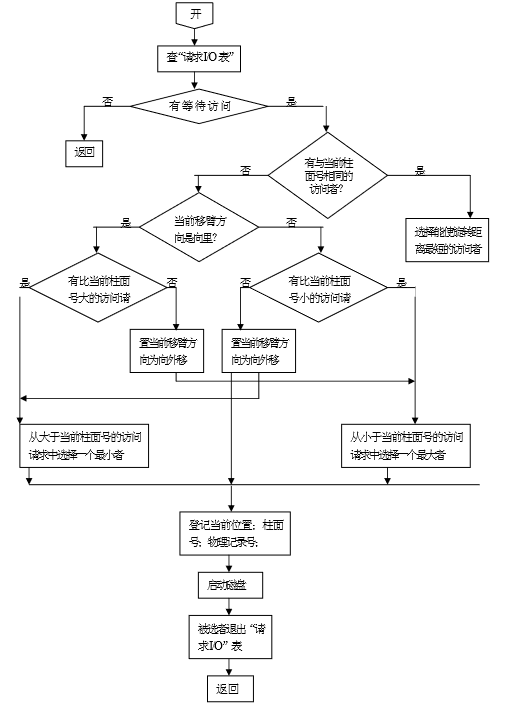
#### 4.1 程序的总体结构流程图



#### 4.2 接受请求模拟算法流程图



#### 4.3 电梯调度模拟算法流程图



### 5.源程序清单

#include <iostream>

#include <string>

#include <queue>

#include <math.h>

using namespace std;

enum Direction{UP,DOWN};

struct Table

{

string process\_name;//进程名

int cylinder\_num;//柱面号

int track\_num;//磁道号

int record\_num;//物理记录号

};

//queue<Table> q;

//初始化

Table t[100];//= { { "P1", 1, 2, 3 }, { "P2", 1, 3, 4 }, { "P3", 2, 4, 3 } };

int table\_num=0;//= 3;

Direction current\_direction = UP;//当前方向

int current\_cylinder = 0;//当前柱面号

//int current\_track = 0;//当前磁道号

int current\_record = 0;//当前记录号

/\*

\*/

void DriverModify()

{

int tag=-1;

int i = 0;

int num = -1;

if (table\_num>0)//请求I/O表有内容

{

for (i = 0; i < table\_num;i++)//遍历

{

if (t[i].cylinder\_num==current\_cylinder)//与当前柱面相同

{

int tem = abs(t[i].record\_num - current\_record);//旋转距离

if (num==-1||num>tem)//找出最短的旋转距离

{

tag = i;

num = tem;

}

}

}

if (num>=0)

{

current\_cylinder = t[tag].cylinder\_num;

current\_record = t[tag].record\_num;

cout << "请求IO表："<<endl;

for (i = 0; i < table\_num;i++)

{

cout << t[i].process\_name << " " << t[i].cylinder\_num << " " << t[i].track\_num << " " << t[i].record\_num << endl;

}

cout << "当前移臂方向:" << current\_direction << endl;

cout << "当前柱面号：" << current\_cylinder << endl;

cout << "当前物理记录号：" << current\_record << endl;

table\_num--;

for (i = tag; i < table\_num;i++)

{

t[i] = t[i + 1];

}

}

//没有与当前柱面相同的访问者

else if (current\_direction==UP)//移臂方向向里

{

for (i = 0; i < table\_num;i++)//遍历

{

if (t[i].cylinder\_num>current\_cylinder)//比当前柱面号大的请求

{

int tem = t[i].cylinder\_num;

if (num == -1 || num > tem)//找最小者

{

tag = i;

num = tem;

}

}

}

if (num>=0)

{

current\_cylinder = t[tag].cylinder\_num;

current\_record = t[tag].record\_num;

cout << "请求IO表："<<endl;

for (i = 0; i < table\_num; i++)

{

cout << t[i].process\_name << " " << t[i].cylinder\_num << " " << t[i].track\_num << " " << t[i].record\_num << endl;

}

cout << "当前移臂方向:" << current\_direction << endl;

cout << "当前柱面号：" << current\_cylinder << endl;

cout << "当前物理记录号：" << current\_record << endl;

table\_num--;

for (i = tag; i < table\_num; i++)

{

t[i] = t[i + 1];

}

}

else//没有比当前柱面号大的请求

{

num = -1;

current\_direction = DOWN;//移臂方向置为外移

for (i = 0; i < table\_num; i++)

{

if (t[i].cylinder\_num<current\_cylinder)//比当前柱面号小的请求

{

int tem = t[i].cylinder\_num;

if (num == -1 || num < tem)//找最大者

{

tag = i;

num = tem;

}

}

}

if (num>=0)

{

current\_cylinder = t[tag].cylinder\_num;

current\_record = t[tag].record\_num;

cout << "请求IO表："<<endl;

for (i = 0; i < table\_num; i++)

{

cout << t[i].process\_name << " " << t[i].cylinder\_num << " " << t[i].track\_num << " " << t[i].record\_num << endl;

}

cout << "当前移臂方向:" << current\_direction << endl;

cout << "当前柱面号：" << current\_cylinder << endl;

cout << "当前物理记录号：" << current\_record << endl;

table\_num--;

for (i = tag; i < table\_num; i++)

{

t[i] = t[i + 1];

}

}

else;

}

}

else//当前移臂方向向外

{

num = -1;

for (i = 0; i < table\_num; i++)//遍历

{

if (t[i].cylinder\_num < current\_cylinder)//比当前柱面号小的访问请求

{

int tem = t[i].cylinder\_num;

if (num == -1 || num < tem)//找最大者

{

tag = i;

num = tem;

}

}

}

if (num>=0)

{

current\_cylinder = t[tag].cylinder\_num;

current\_record = t[tag].record\_num;

cout << "请求IO表："<<endl;

for (i = 0; i < table\_num; i++)

{

cout << t[i].process\_name << " " << t[i].cylinder\_num << " " << t[i].track\_num << " " << t[i].record\_num << endl;

}

cout << "当前移臂方向:" << current\_direction << endl;

cout << "当前柱面号：" << current\_cylinder << endl;

cout << "当前物理记录号：" << current\_record << endl;

table\_num--;

for (i = tag; i < table\_num; i++)

{

t[i] = t[i + 1];

}

}

else//没有比当前柱面号小的访问请求

{

num = -1;

current\_direction = DOWN;//移臂方向置为外移

for (i = 0; i < table\_num; i++)

{

if (t[i].cylinder\_num > current\_cylinder)//寻找比当前柱面号大的访问请求

{

int tem = t[i].cylinder\_num;

if (num == -1 || num > tem)//寻找最小者

{

tag = i;

num = tem;

}

}

}

if (num >= 0)

{

current\_cylinder = t[tag].cylinder\_num;

current\_record = t[tag].record\_num;

cout << "请求IO表："<<endl;

for (i = 0; i < table\_num; i++)

{

cout << t[i].process\_name << " " << t[i].cylinder\_num << " " << t[i].track\_num << " " << t[i].record\_num << endl;

}

cout << "当前移臂方向:" << current\_direction << endl;

cout << "当前柱面号：" << current\_cylinder << endl;

cout << "当前物理记录号：" << current\_record << endl;

table\_num--;

for (i = tag; i < table\_num; i++)

{

t[i] = t[i + 1];

}

}

else;

}

}

}

}

void RequestReceive()

{

cout << "是否有请求？(Y/N)";

char c;

cin >> c;

if (c=='y'||c=='Y')//有请求

{

cout << "请输入进程名、柱面号、磁道号、物理记录号：";

string name;

int num1, num2, num3;

cin >> name >> num1 >> num2 >> num3;//输入进程名、物理地址

Table temp = { name, num1, num2, num3 };

//q.push(temp);

t[table\_num++] = temp;//登记请求I/O表

}

return;

}

int main(\_In\_ int \_Argc, \_In\_reads\_(\_Argc) \_Pre\_z\_ char \*\* \_Argv, \_In\_z\_ char \*\* \_Env)

{

double input\_num = 0.0;

while (cout<<"请输入一个随机数："&&cin>>input\_num)//输入随机数

{

if (input\_num>0.5)//大于0.5

{

DriverModify();//驱动调度

}

else//小于0.5

{

RequestReceive();//接受请求

}

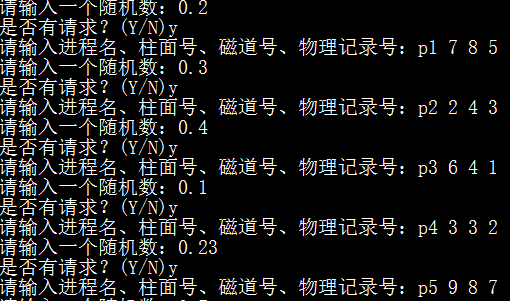
}

return 0;

}

### 6.实验测试结果分析

输入数据：



结果：（注：移臂方向中0为UP，1为DOWN）

