

**实 验 报 告**

**课程设计名称：** 操作系统原理

**专 业 班 级 ：** 物联网工程1902班

**学 生 姓 名 ：** 王源

**学 号 ：** 201916070216

**指 导 教 师 ：** 廖庆荣

**课程实验时间：** 2021年4月22日

**实验三 磁盘调度算法的实现**

# 一、设计目的

1. 能了解操作系统的磁盘调度算法。

2. 能实现操作系统的磁盘调度算法。

# 二、设计要求与内容

1. 能实现操作系统六种磁盘调度算法，包括FCFS、SSTF、SCAN、C-SCAN、LOOK、C-LOOK。

# 设计原理：包含程序想法和做法说明

说明：实验环境：Windows操作系统、VS 2019/CMD\_g++、C/C++语言

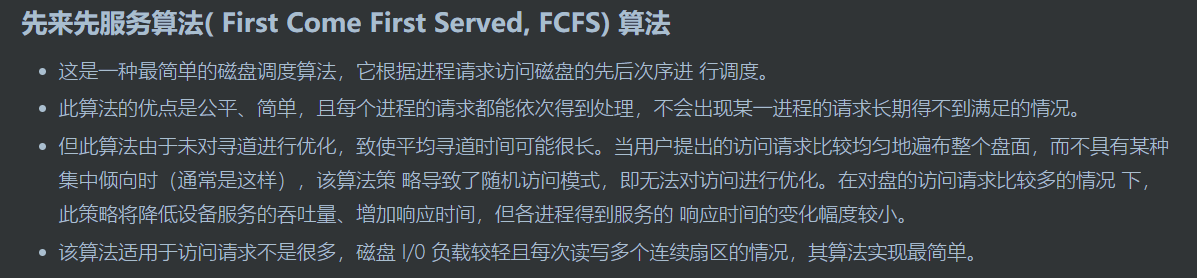
### 需求分析

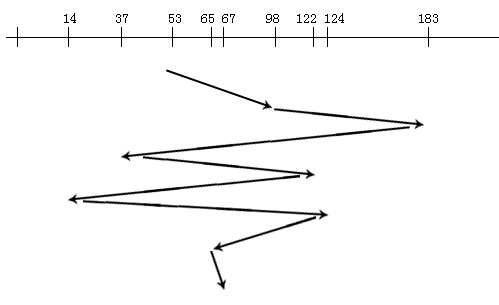
分别实现先到先服务调度(FCFS)磁盘调度算法、最短寻道时间优先算法(SSTF)、“电梯”调度算法（SCAN算法）、C-SCAN算法、LOOK调度算法和C-LOOK调度算法

概要设计

1.先到先服务调度(FCFS)磁盘调度算法

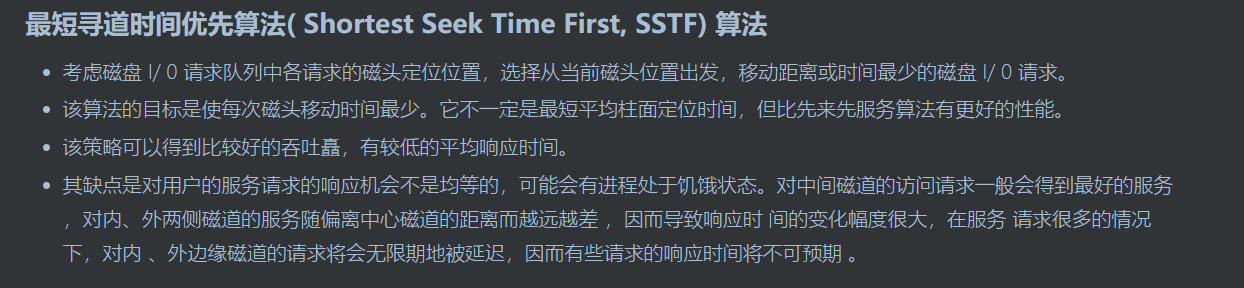
 按照磁道的访问顺序依次访问每个磁道。

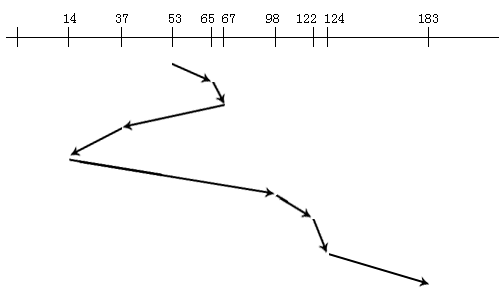




2.最短寻道时间优先算法(SSTF)

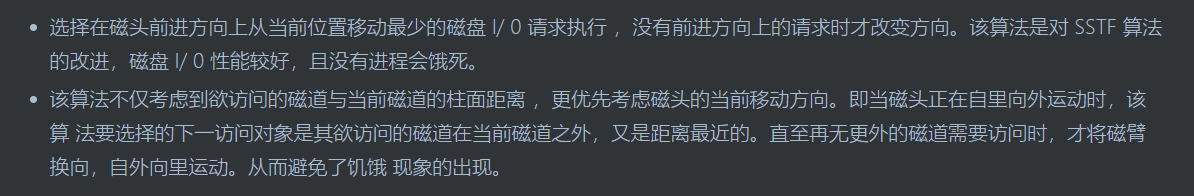
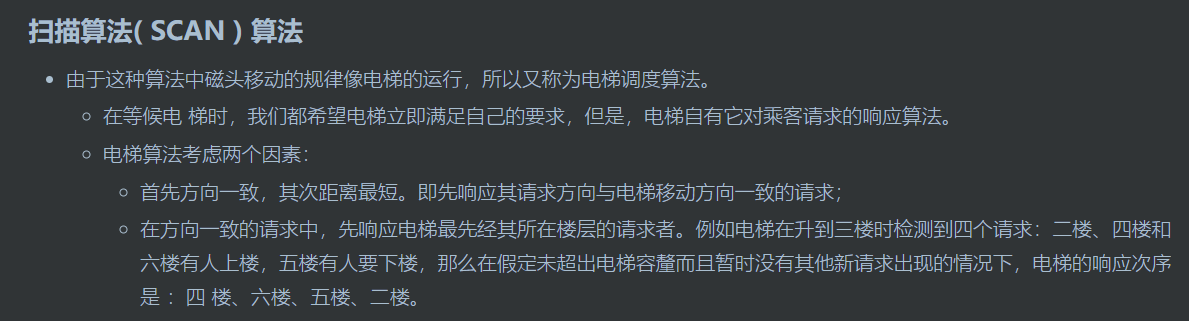
 从指定磁道，依次访问距离本磁道最近的磁道。

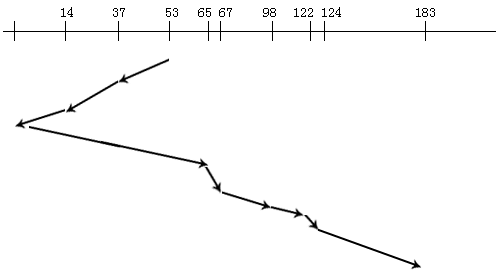




3.“电梯”调度算法（SCAN算法）

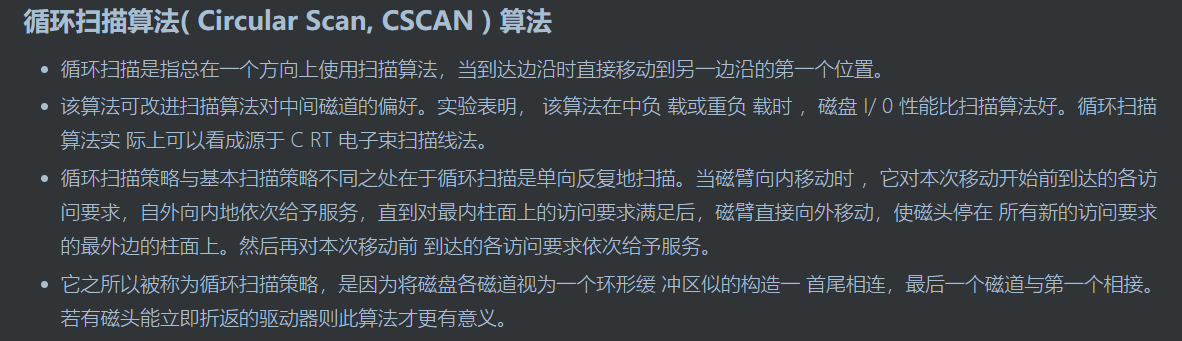
   从指定磁道，向指定方向单向移动，依次访问距离本磁道最近的磁道，访问到最终磁道后，转向访问初始磁道另一侧的每个磁道，规则和以上相同。

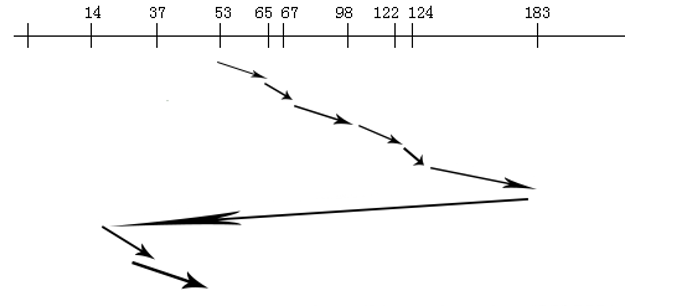




4.C-SCAN算法

  从指定磁道，向指定方向单向移动，依次访问距离本磁道最近的磁道，访问到磁盘最大（最小）容量后，转向访问初始磁道另一侧的每个磁道，到磁盘容量最小（最大），规则和以上相同。



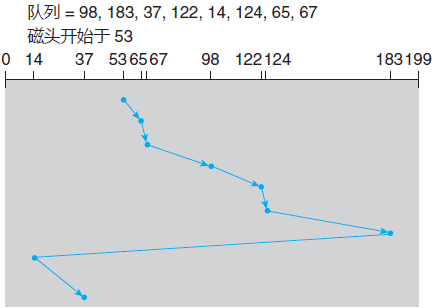


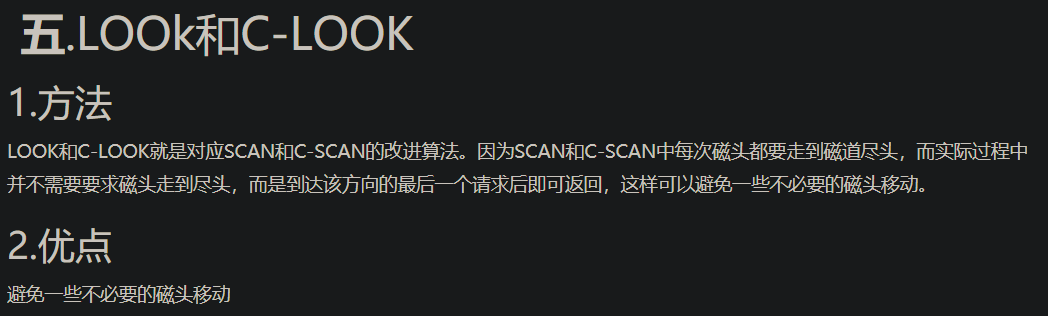
5.LOOK调度算法

从指定磁道，向指定方向单向移动，依次访问距离本磁道最近的磁道，访问完最终磁道后，转向访问据本磁道最远的磁道，按如上规则依次访问至初始磁道。

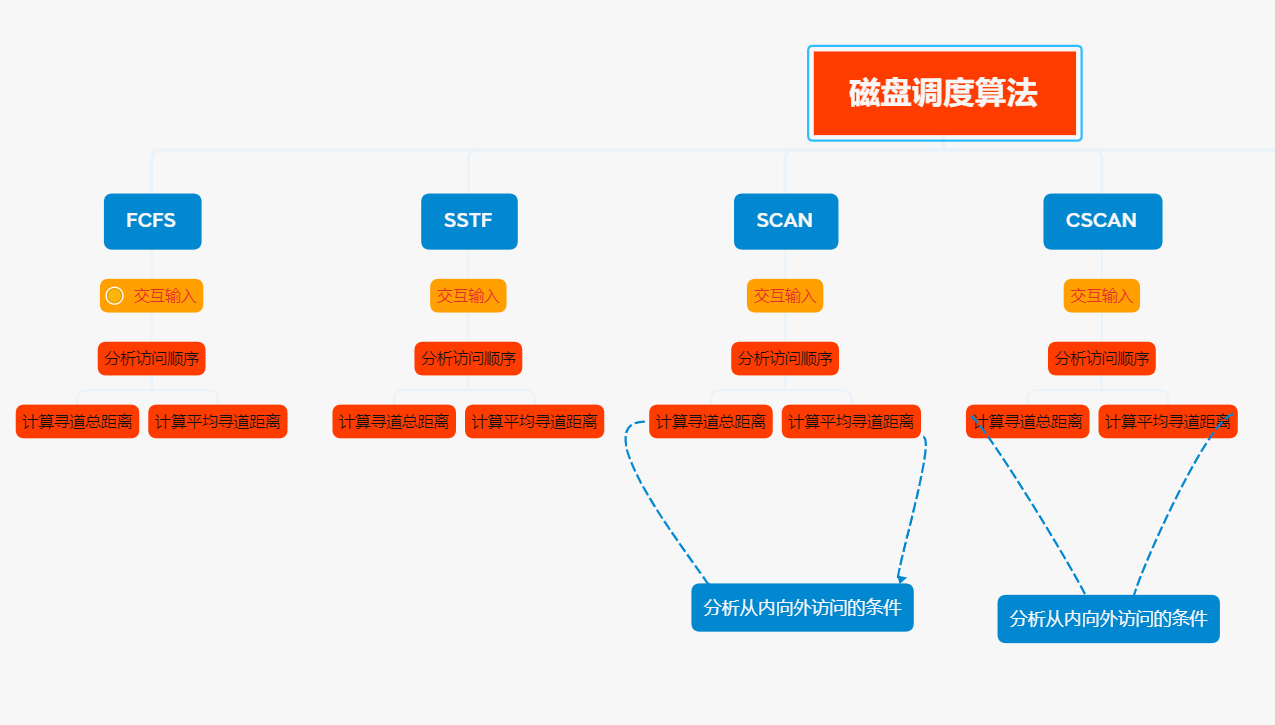
6.C-LOOK调度算法

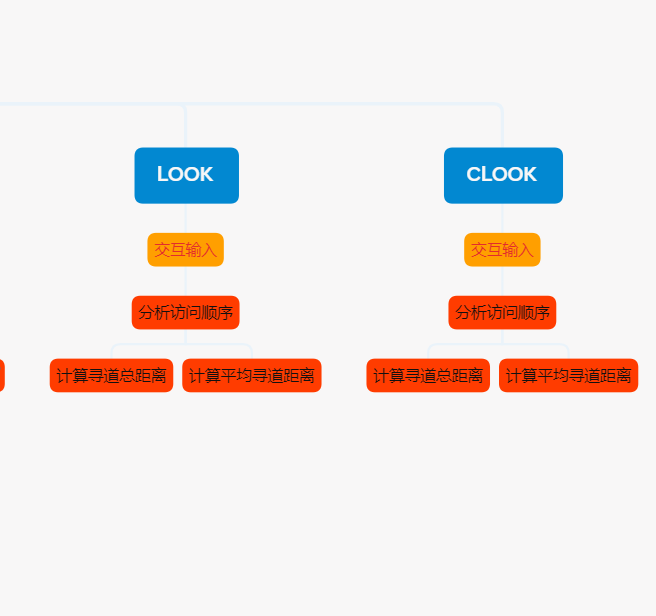
  从指定磁道，向指定方向单向移动，依次访问距离本磁道最近的磁道，访问至磁盘最大（最小）容量后，转向访问磁盘的最小（最大）容量磁道，按如上规则依次访问至初始磁道。

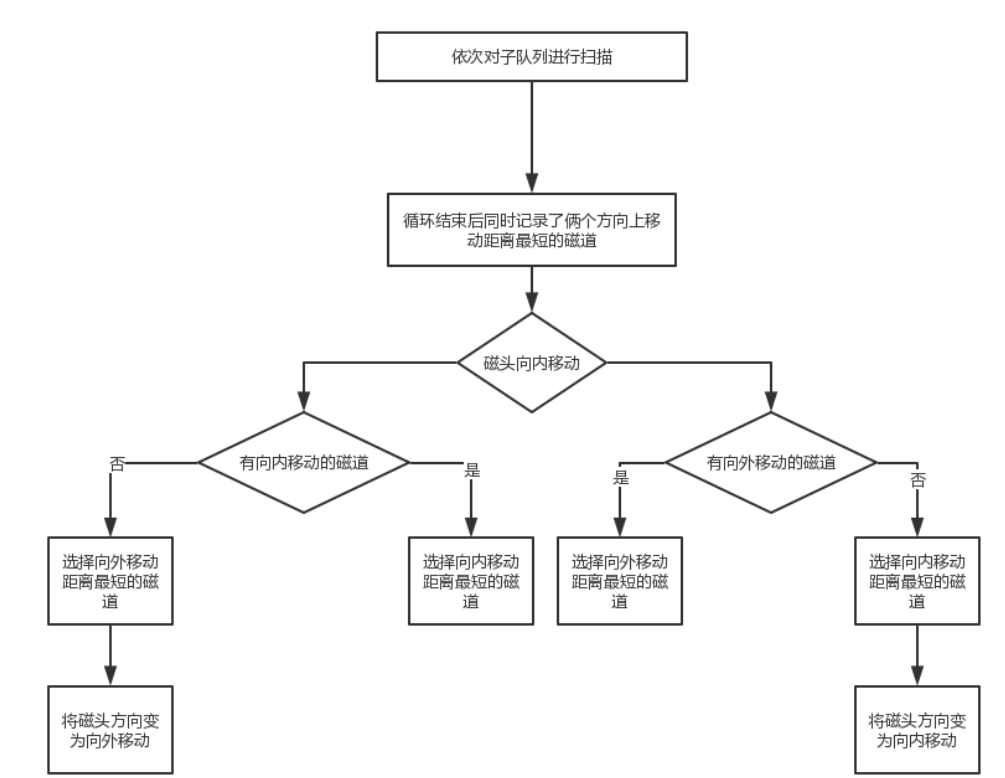




# 流程图







# 源代码：程序一定要加上批注 /\* … \*/

**C\_look.cpp**

#include"c\_look.h"

const int maxn06 = 1010;

int track06[maxn06]; //设置最大轨道数

priority\_queue<int, vector<int>, greater<int>>l6; //设置队列

priority\_queue<int, vector<int>, greater<int>>r6; //设置队列

void run06(int index, int n)

{

int sum = 0;

int order[maxn06]; //存放在磁道口

int cur = 1;

while (!r6.empty())

{

sum += abs(r6.top() - index);

index = r6.top();

r6.pop();

order[cur++] = index;

}

while (!l6.empty())

{

sum += abs(l6.top() - index);

index = l6.top();

l6.pop();

order[cur++] = index;

}

cout << endl << "\*\*\*\*\*\*\*\*\*C\_LOOK\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为：";

for (int i = 1; i <= n; i++) //依次输出下一个磁道距离

{

cout << order[i];

if (i != n)

{

cout << "->";

}

}

cout << endl<<endl;

cout << "寻道总距离为：" << sum << endl<<endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n << endl<<endl;

}

void input06(int index, int n) //输入磁道信息

{

cout << "请依次输入需要访问的磁道号>:" << endl;

for (int i = 1; i <= n; i++)

{

cin >> track06[i];

if (track06[i] >= index)

r6.push(track06[i]);

else

l6.push(track06[i]);

}

}

void CLOOK\_PRO() //打印计算后的磁道信息

{

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

input06(index, n);

run06(index, n);

}

C\_look.h

#ifndef \_\_C\_LOOK\_\_

#define \_\_C\_LOOK\_\_

#include <bits/stdc++.h>

using namespace std;

void run06(int index, int n); //初始化

void input06(int index, int n); //输入信息

void CLOOK\_PRO(); //打印信息

#endif

C\_scan.cpp

#include"c\_scan.h"

const int maxn05 = 1010;

int track05[maxn05]; //设置最大轨道数

int maxindex05 = 0;

int minindex05 = 0;

priority\_queue<int, vector<int>, greater<int>>l5; //设置队列

priority\_queue<int, vector<int>, greater<int>>r5; //设置队列

void run05(int index, int n, int capacity)

{

int sum = 0;

int order[maxn05]; //存放在磁道口

int cur = 1;

while (!r5.empty())

{

sum += abs(r5.top() - index);

index = r5.top();

r5.pop();

if (index != capacity || (index == capacity && maxindex05 == 1))

{

order[cur++] = index;

}

}

while (!l5.empty())

{

sum += abs(l5.top() - index);

index = l5.top();

l5.pop();

if (index != 0 || (index == 0 && minindex05 == 1))

order[cur++] = index;

}

cout <<endl<< "\*\*\*\*\*\*\*\*\*\*CLOOK\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为:";

for (int i = 1; i <= n; i++)

{

cout << order[i]; //依次输出下一个磁道距离

if (i != n)

{

cout << "->";

}

}

cout <<endl<< endl;

cout << "寻道总距离为：" << sum <<endl<< endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n <<endl<< endl;

}

void input05(int index, int n, int capacity) //输入磁道信息

{

cout << "请依次输入需要访问的磁道号:>" << endl;

for (int i = 1; i <= n; i++)

{

cin >> track05[i];

if (track05[i] >= index)

{

if (track05[i] == capacity)

maxindex05 = 1;

r5.push(track05[i]);

}

else

{

if (track05[i] == 0)

minindex05 = 1;

l5.push(track05[i]);

}

}

if (!maxindex05)

r5.push(capacity);

if (!minindex05)

l5.push(0);

}

void CSCAN\_PRO() //打印计算后的磁道信息

{

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入磁盘容量>:" << endl;

int capacity;

cin >> capacity;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

input05(index, n, capacity);

run05(index, n, capacity);

}

C\_scan.h

#ifndef \_\_C\_SCAN\_\_

#define \_\_C\_SCAN\_\_

#include <bits/stdc++.h>

using namespace std;

void run05(int index, int n, int capacity); //初始化

void input05(int index, int n, int capacity); //输入信息

void CSCAN\_PRO(); //打印信息

#endif

Fcfs.cpp

#include"fcfs.h"

const int maxn02 = 1010; //设置最大轨道数

void run02(int track02[], int index, int n) //计算访问顺序，寻道距离，平均距离

{

int sum = 0;

for (int i = 1; i <= n; i++)

{

sum += abs(index - track02[i]);

index = track02[i];

}

cout << endl << "\*\*\*\*\*\*\*\*\*\*FCFS\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为:";

for (int i = 1; i <= n; i++)

{

cout << track02[i];

if (i != n)

{

cout << "->";

}

}

cout <<endl<< endl;

cout << "寻道总距离为：" << sum << endl<< endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n <<endl<< endl;

}

void input02(int track02[], int n) //输入信息

{

cout << "请依次输入需要访问的磁道号>:" << endl;

for (int i = 1; i <= n; i++)

cin >> track02[i];

}

void FCFS\_PRO() //输入信息

{

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

int track02[maxn02];

input02(track02, n);

run02(track02, index, n);

}

Fcfs.h

#ifndef \_\_FCFS\_\_

#define \_\_FCFS\_\_

#include <iostream>

#include <cmath>

using namespace std;

void run02(int track02[], int index, int n); //初始化

void input02(int track02[], int n); //输入信息

void FCFS\_PRO(); //打印信息

#endif

Look.cpp

#include"look.h"

using namespace std;

const int maxn01 = 1010; //设置最大轨道数

int track01[maxn01];

priority\_queue<int>l1; //设置队列

priority\_queue<int, vector<int>, greater<int>>r1; //设置队列

void run01(int index, int n)

{

int sum = 0;

int order[maxn01]; //存放在磁道口

int cur = 1;

while (!r1.empty())

{

sum += abs(r1.top() - index);

index = r1.top();

r1.pop();

order[cur++] = index;

}

while (!l1.empty())

{

sum += abs(l1.top() - index);

index = l1.top();

l1.pop();

order[cur++] = index;

}

cout << endl << "\*\*\*\*\*\*\*\*\*\*LOOK\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为";

for (int i = 1; i <= n; i++)

{

cout << order[i];

if (i != n) //依次输出下一个磁道距离

{

cout << "->";

}

}

cout << endl << endl;

cout << "寻道总距离为：" << sum << endl<<endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n <<endl<< endl;

}

void input01(int index, int n)

{ //输入磁道信息

cout << "请依次输入需要访问的磁道号>:" << endl;

for (int i = 1; i <= n; i++)

{

cin >> track01[i];

if (track01[i] >= index)

r1.push(track01[i]);

else

l1.push(track01[i]);

}

}

void LOOK\_PRO() //打印计算后的磁道信息

{

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

input01(index, n);

run01(index, n);

}

Look.h

#ifndef \_\_LOOK\_\_

#define \_\_LOOK\_\_

#include<bits/stdc++.h>

using namespace std;

void run01(int index, int n); //初始化

void input01(int index, int n); //输入信息

void LOOK\_PRO(); //打印信息

#endif

Scan.cpp

#include"scan.h"

const int maxn04 = 1010;

int track04[maxn04]; //设置最大轨道数

int maxindex04 = 0;

int minindex04 = 0;

priority\_queue<int>l4; //设置队列

priority\_queue<int, vector<int>, greater<int>>r4; //设置队列

void run04(int index, int n, int capacity)

{

int sum = 0; //存放在磁道口

int order[maxn04];

int cur = 1;

while (!r4.empty())

{

sum += abs(r4.top() - index);

index = r4.top();

r4.pop();

if (index != capacity || (index == capacity && maxindex04== 1))

{

order[cur++] = index;

}

}

while (!l4.empty())

{

sum += abs(l4.top() - index);

index = l4.top();

l4.pop();

if (index != 0 || (index == 0 && minindex04 == 1))

order[cur++] = index;

}

cout << endl << "\*\*\*\*\*\*\*\*\*\*SCAN\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为";

for (int i = 1; i <= n; i++)

{

cout << order[i]; //依次输出下一个磁道距离

if (i != n)

{

cout << "->";

}

}

cout << endl<<endl;

cout << "寻道总距离为：" << sum << endl<<endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n << endl<<endl;

}

void input04(int index, int n, int capacity) //输入磁道信息

{

cout << "请依次输入需要访问的磁道号>:" << endl;

for (int i = 1; i <= n; i++)

{

cin >> track04[i];

if (track04[i] >= index)

{

if (track04[i] == capacity)

maxindex04 = 1;

r4.push(track04[i]);

}

else

{

if (track04[i] == 0)

minindex04 = 1;

l4.push(track04[i]);

}

}

if (!maxindex04)

r4.push(capacity);

if (!minindex04)

l4.push(0);

}

//打印计算后的磁道信息

void SCAN\_PRO()

{

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入磁盘容量>:" << endl;

int capacity;

cin >> capacity;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

input04(index, n, capacity);

run04(index, n, capacity);

}

Scan.h

#ifndef \_\_SCAN\_\_

#define \_\_SCAN\_\_

#include <bits/stdc++.h>

using namespace std;

void run04(int index, int n, int capacity); //初始化

void input04(int index, int n, int capacity); //输入信息

void SCAN\_PRO(); //打印信息

#endif

Sstf.cpp

#include"sstf.h"

const int maxn03 = 1010; //设置最大轨道数

int track03[maxn03];

priority\_queue<int>l3; //设置队列

priority\_queue<int, vector<int>, greater<int>>r3; //设置队列

void run03(int index, int n)

{

int sum = 0;

int order[maxn03]; //存放在磁道口

int cur = 1;

while (!l3.empty() && !r3.empty())

{

if (abs(l3.top() - index) <= abs(r3.top() - index))

{

sum += abs(l3.top() - index);

index = l3.top();

l3.pop();

order[cur++] = index;

}

else

{

sum += abs(r3.top() - index);

index = l3.top();

l3.pop();

order[cur++] = index;

}

}

while (!l3.empty())

{

sum += abs(l3.top() - index);

index = l3.top();

l3.pop();

order[cur++] = index;

}

while (!r3.empty())

{

sum += abs(r3.top() - index);

index = r3.top();

r3.pop();

order[cur++] = index;

}

cout << endl << "\*\*\*\*\*\*\*\*\*SSTF\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "访问磁道顺序为";

for (int i = 1; i <= n; i++)

{

cout << order[i]; //依次输出下一个磁道距离

if (i != n)

{

cout << "->";

}

}

cout <<endl<< endl;

cout << "寻道总距离为：" << sum <<endl<< endl;

cout << "平均寻道距离为：" << sum \* 1.0 / n << endl<<endl;

}

void input03(int index, int n) //输入磁道信息

{

cout << "请依次输入需要访问的磁道号>:" << endl;

for (int i = 1; i <= n; i++)

{

cin >> track03[i];

if (track03[i] >= index)

r3.push(track03[i]);

else

l3.push(track03[i]);

}

}

void SSTF\_PRO()

{ //打印计算后的磁道信息

cout << "请输入当前磁道号>:" << endl;

int index;

cin >> index;

cout << "请输入需要访问的磁道数目>:" << endl;

int n;

cin >> n;

input03(index, n);

run03(index, n);

}

Sstf.h

#ifndef \_\_SSTF\_\_

#define \_\_SSTF\_\_

#include <bits/stdc++.h>

using namespace std;

void run03(int index, int n); //初始化

void input03(int index, int n); //输入信息

void SSTF\_PRO(); //打印信息

#endif

Test.cpp

/\*

操作系统实验课OS\_03

物联网1902班

学号：201916070216

姓名：王源

\*/

#include"c\_look.h"

#include"c\_scan.h"

#include"fcfs.h"

#include"look.h"

#include"scan.h"

#include"sstf.h"

//包含各算法头文件

using namespace std;

void ShowUI() //show ui,让界面更美观

{

cout << "\*\*\*\*\*\*\*物联网1902班王源\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*201916070216\*\*\*\*\*\*\*\*" << endl;

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 算法选择 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 1. FCFS算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 2. SSTF算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 3. SCAN算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 4. CSCAN算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 5. LOOK算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 6. CLOOK算法 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 0. 退出该程序 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" << endl;

}

int main()

{

//int flag = 1;

int key = 0;

out:

do

{

ShowUI();

cout << "请选择您要执行的算法>:";

cin >> key;

switch (key) //switch语句实现交互式选择

{

case 1: FCFS\_PRO(); break;

case 2: SSTF\_PRO(); break;

case 3: SCAN\_PRO(); break;

case 4: CSCAN\_PRO(); break;

case 5: LOOK\_PRO(); break;

case 6: CLOOK\_PRO(); break;

case 0: break;

default: cout << endl << "您的输入有误，请重新输入>:"; //增加容错性

goto out;

}

} while (key);

system("pause"); //暂停

return 0;

}

# 实验结果与说明：包含程序运行结果截图，结果要先显示出你自己的班级、学号、姓名。

1. FCFS算法：



2.SSTF算法：



3．SCAN算法：



4.CSCAN算法：



5.LOOK算法：

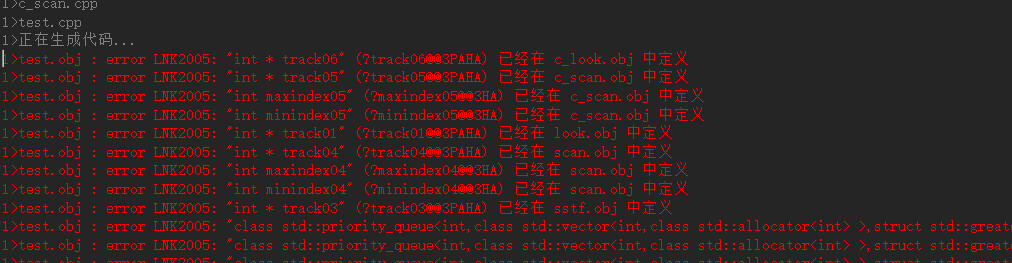


6.CLOOK算法：



# 我所遭遇的问题

1. 各个算法的距离该如何计算，是否要把读取的移动和最大位置到最小位置的不读取移动作区分。
2. 在实现扫描算法时，当输入的当前磁道号不在当前磁盘请求序列中时，程序可以正常执行。
3. 我实现的是分文件编写，在编写的时候误把全局变量写在了对应的.h文件中，导致程序崩溃，自己当时并没有意识到是这个小错误导致的，在反复调试许久后仍不能解决。后来查阅了资料以及请教了我的好朋友，才了解到全局变量不能写在.h文件中，会显示该.obj与test.obj重定义。



之后把全局变量写在对应的.cpp中就不报错了。算是给自己的分文件编写提升了一点能力吧。

# 收获、体会和建议：包含撰写程序的心得

磁盘调度的主要思想是：设备的动态分配算法与进程调度相似，也是基于一定的分配策略的。常用的分配策略有先请求先分配、优先级高者先分配等策略。在多道程序系统中，低效率通常是由于磁盘类旋转设备使用不当造成的。操作系统中，对磁盘的访问要求来自多方面，常常需要排队。这时，对众多的访问要求按一定的次序响应，会直接影响磁盘的工作效率，进而影响系统的性能。访问磁盘的时间因子由3部分构成，它们是查找（查找磁道）时间、等待（旋转等待扇区）时间和数据传输时间，其中查找时间是决定因素。因此，磁盘调度算法先考虑优化查找策略，需要时再优化旋转等待策略。平均寻道长度（L）为所有磁道所需移动距离之和除以总的所需访问的磁道数（N），即：

L=（M1+M2+……+Mi+……+MN）/N

其中Mi为所需访问的磁道号所需移动的磁道数。

总的来说，整个实验中最麻烦的就是整个程序模块的划分和各模块之间接口设计，编程中经常犯想当然的错误，编程中出现了不少奇怪的错误。再调试中尝试使用了分割法,对错误模块进行定位,再进行排查.

在算法实现上要有一定的思路要更能体现设计的目的。同时上机调试也是十分重要的，在调试的过程中能够不断的发现在编写算法时应该注意的一些细节和算法语句的非法使用，在调试过程中通过对算法的不断测试、更正、扩充功能、修饰细节，使算法程序不断的得到完善。

通过本次实验，学习了解磁盘调度六种调度算法的工作原理以及六种调度算法之间的差异和共性，并且在当中发现了自己的不足，对以前所学过的知识理解得不够深刻，掌握得不够牢固，看到了自己的实践经验还是比较缺乏，实践能力还需要提高。

九．附件



# 主要参考文献

1.程序员大本营：[图解五种磁盘调度算法, FCFS, SSTF, SCAN, C-SCAN, LOOK - 程序员大本营 (pianshen.com)](https://www.pianshen.com/article/72471017753/)

2.csdn：<https://blog.csdn.net/Baiyi_destroyer/article/details/107197307>

3.汤小丹，梁红兵等 《计算机操作系统》(磁盘调度)

**截止日期：2021.05.06 Thr. 23:00**

**档案命名方式：OS03\_班级学号姓名**