

操作系统

实 验1 报 告

学院： 计算机科学学院

专业： 网络工程1701

姓名： 胡玉琛41709040102

姓名： 何一鸣41709040107

## 1. 实验名称

进程调度算法FCFS/SJF

## 2.实验目的

理解FCFS和SJF的计算过程，掌握进程调度的具体算法。

## 3. 实验内容

(1) 输入的形式和输入值的范围；

#define MaxNum 100//允许的最大进程数

static int n=0,m=0;//进程数

static char ID[MaxNum];//各进程ID

static int ExecuteTime[MaxNum];//各进程的开始执行时间

static int ArrivalTime[MaxNum];//各进程的达到时间

static int ServiceTime[MaxNum];//各进程的服务时间

static int ServiceTime\_SJF[MaxNum];//各进程的服务时间(SJF的临时数组)

static int FinishTime[MaxNum];//各进程的完成时间

static int WholeTime[MaxNum];//各进程的周转时间

static double WeightWholeTime[MaxNum];//各进程的带权周转时间

static double AverageWT\_FCFS=0,AverageWT\_SJF=0;//FCFS和SJF的平均周转时间

static double AverageWWT\_FCFS=0,AverageWWT\_SJF=0;//FCFS和SJF的平均带权周转时间

static int SumWT\_FCFS=0,SumWT\_SJF=0;//FCFS、SJF中的周转时间总和

static double SumWWT\_FCFS=0,SumWWT\_SJF=0;;//FCFS、SJF中的带权周转时间总和

(2) 输出的形式；

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | ArrivalTime | ServiceTime | FinishTime | WholeTime | WeightWholeTime | ExecuteTime |
| A | 1 | 1 | 2 | 1 | 1 | 1 |

(3) 程序所能达到的功能；

输入到达时间、服务时间，自动根据FCFS或SJF算法计算出程序的完成时间、周转时间、带权周转时间、开始执行时间、周转时间总和、带权周转时间总和、平均周转时间、平均带权周转时间

(4) 测试数据，包括正确的输入及其输出结果和含有错误的输入及其输出结果。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | ArrivalTime | ServiceTime | FinishTime | WholeTime | WeightWholeTime | ExecuteTime |
| A | 1 | 1 | 2 | 1 | 1 | 1 |
| B | 3 | 2 | 5 | 2 | 1 | 3 |
| C | 5 | 2 | 7 | 2 | 1 | 5 |
| D | 6 | 3 | 10 | 4 | 1.3 | 7 |
| E | 8 | 4 | 14 | 6 | 1.5 | 10 |

## 4. 实验过程

实现程序模块的具体算法。

//输入进程数、各进程到达、服务时间

int Enter(int arr[], int a, int b) {

int i;

for(i=a;i<b;i++) {

cin>>arr[i];

}

}

//根据达时间排序

int sortArrivalTime(int a, int b) {

int i,j;

int t1,t2,t3;

for(i=a;i<b;i++) {

for(j=i+1;j<b;j++) {

if(ArrivalTime[i] > ArrivalTime[j]) {

t1 = ArrivalTime[i];

ArrivalTime[i] = ArrivalTime[j];

ArrivalTime[j] = t1;

t2 = ServiceTime[i];

ServiceTime[i] = ServiceTime[j];

ServiceTime[j] = t2;

t3 = ID[i];

ID[i] = ID[j];

ID[j] = t3;

}

}

}

}

//已到达的进程服务时间排序

int sortServiceTime(int a, int b) {

int i,j;

int t1,t2,t3;

for(i=a;i<b;i++) {

if(ArrivalTime[i] <= FinishTime[a-1])

ServiceTime\_SJF[i] = ServiceTime[i];

}

for(i=a;i<b;i++) {

for(j=i;j<b;j++) {

if(ServiceTime\_SJF[i] > ServiceTime\_SJF[j] && ServiceTime\_SJF[i] != 0 && ServiceTime\_SJF[j] != 0) {

t1 = ServiceTime\_SJF[i];

ServiceTime\_SJF[i] = ServiceTime\_SJF[j];

ServiceTime\_SJF[j] = t1;

}

}

}

for(i=a;i<b;i++) {

if(ServiceTime[i] == ServiceTime\_SJF[a]) {

t1 = ArrivalTime[a];

ArrivalTime[a] = ArrivalTime[i];

ArrivalTime[i] = t1;

t2 = ServiceTime[a];

ServiceTime[a] = ServiceTime[i];

ServiceTime[i] = t2;

t3 = ID[a];

ID[a] = ID[i];

ID[i] = t3;

}

}

}

//计算第一个进程的各个时间

int calculationInitTime( int i) {

FinishTime[i] = ArrivalTime[i] + ServiceTime[i];

WholeTime[i] = FinishTime[i] - ArrivalTime[i];

WeightWholeTime[i] = (double)WholeTime[i]/(double)ServiceTime[i];

ExecuteTime[i] = ArrivalTime[i];

}

//计算完成时间、周转时间、带权周转时间、开始执行时间

int calculationTime(int i) {

if(ArrivalTime[i] <= FinishTime[i-1]) {

FinishTime[i] = FinishTime[i-1] + ServiceTime[i];

ExecuteTime[i] = FinishTime[i-1];

}

else {

FinishTime[i] = ArrivalTime[i] + ServiceTime[i];

ExecuteTime[i] = ArrivalTime[i];

}

WholeTime[i] = FinishTime[i] - ArrivalTime[i];

WeightWholeTime[i] = (double)WholeTime[i]/(double)ServiceTime[i];

}

//输出ID、到达时间、服务时间、完成时间、周转时间、带权周转时间、开始执行时间、周转时间总和、带权周转时间总和、平均周转时间、平均带权周转时间

int display(int a, int b) {

cout<<"ID"<<"\t"<<"ArrivalTime"<<"\t"<<"ServiceTime"<<"\t"<<"FinishTime"<<"\t"<<"WholeTime"<<"\t"<<"WeightWholeTime"<<"\t"<<"ExecuteTime"<<endl;

int i;

for(i=a;i<b;i++) {

cout<<ID[i]<<"\t"<<ArrivalTime[i]<<"\t"<<"\t"<<ServiceTime[i]<<"\t"<<"\t"<<FinishTime[i]<<"\t"<<"\t"<<WholeTime[i]<<"\t"<<"\t"<<setprecision(2)<<WeightWholeTime[i]<<"\t"<<"\t"<<ExecuteTime[i]<<endl;

}

if(a==0){

cout<<"SumWT\_FCFS: "<<SumWT\_FCFS<<endl<<"SumWWT\_FCFS: ";

cout<<setprecision(4)<<SumWWT\_FCFS<<endl<<"AverageWT\_FCFS: ";

cout<<setprecision(4)<<AverageWT\_FCFS<<endl<<"AverageWWT\_FCFS: ";

cout<<setprecision(4)<<AverageWWT\_FCFS<<endl;

}

else{

cout<<"SumWT\_SJF: "<<SumWT\_SJF<<endl<<"SumWWT\_SJF: ";

cout<<setprecision(4)<<SumWWT\_SJF<<endl<<"AverageWT\_SJF: ";

cout<<setprecision(4)<<AverageWT\_SJF<<endl<<"AverageWWT\_SJF: ";

cout<<setprecision(4)<<AverageWWT\_SJF<<endl;

}

}

//输出各时间状态

int outTimeState(int a, int b) {

int i,j;

for(i=0;i<=FinishTime[b-1];i++) {

cout<<"Time "<<i<<": ";

for(j=a;j<b;j++) {

if(ArrivalTime[j] == i && ExecuteTime[j] == i)

cout<<"ID "<<ID[j]<<" arrived and executing, ";

else if(ArrivalTime[j] == i)

cout<<"ID "<<ID[j]<<" arrived, ";

else if(ExecuteTime[j] == i)

cout<<"ID "<<ID[j]<<" executing, ";

else if(FinishTime[j] == i)

cout<<"ID "<<ID[j]<<" finished, ";

}

cout<<endl;

}

}

void FCFS(){

int i,j;

//FCFS输入进程数

cout<<"FCFS"<<endl;;

cout<<"Enter n: ";

cin>>n;

//FCFS各进程到达时间

cout<<"Enter ArrivalTime: ";

Enter(ArrivalTime,0,n);

//FCFS各进程服务时间

cout<<"Enter ServiceTime: ";

Enter(ServiceTime,0,n);

//根据达时间排序

sortArrivalTime(0,n);

//FCFS计算各时间

calculationInitTime(0);

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

calculationTime(i);

SumWT\_FCFS += WholeTime[i];

SumWWT\_FCFS += WeightWholeTime[i];

}

SumWT\_FCFS += WholeTime[0];

SumWWT\_FCFS += WeightWholeTime[0];

AverageWT\_FCFS = (double)SumWT\_FCFS/(double)n;

AverageWWT\_FCFS = (double)SumWWT\_FCFS/(double)n;

//FCFS输出各时间

display(0,n);

//FCFS输出各时间状态

outTimeState(0,n);

}

void SJF(){

int i,j;

//SJF输入进程数

cout<<"SJF"<<endl;

cout<<"Enter m: ";

cin>>m;

//各进程到达时间

cout<<"Enter ArrivalTime: ";

Enter(ArrivalTime,n,n+m);

//各进程服务时间

cout<<"Enter ServiceTime: ";

Enter(ServiceTime,n,n+m);

//根据达时间排序

sortArrivalTime(n,n+m);

//SJF计算各时间

calculationInitTime(n);

for(i=n+1;i<n+m;i++) {

sortServiceTime(i,n+m);

calculationTime(i);

SumWT\_SJF += WholeTime[i];

SumWWT\_SJF += WeightWholeTime[i];

}

SumWT\_SJF += WholeTime[n];

SumWWT\_SJF += WeightWholeTime[n];

AverageWT\_SJF = (double)SumWT\_SJF/(double)m;

AverageWWT\_SJF = (double)SumWWT\_SJF/(double)m;

//SJF输出各时间

display(n,n+m);

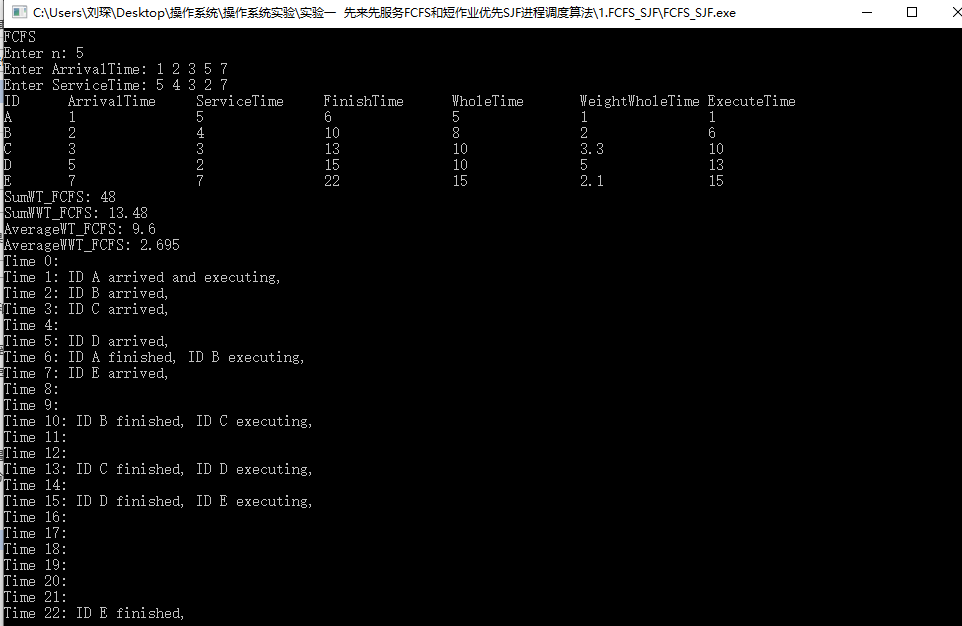
//SJF输出各时间状态

outTimeState(n,n+m);

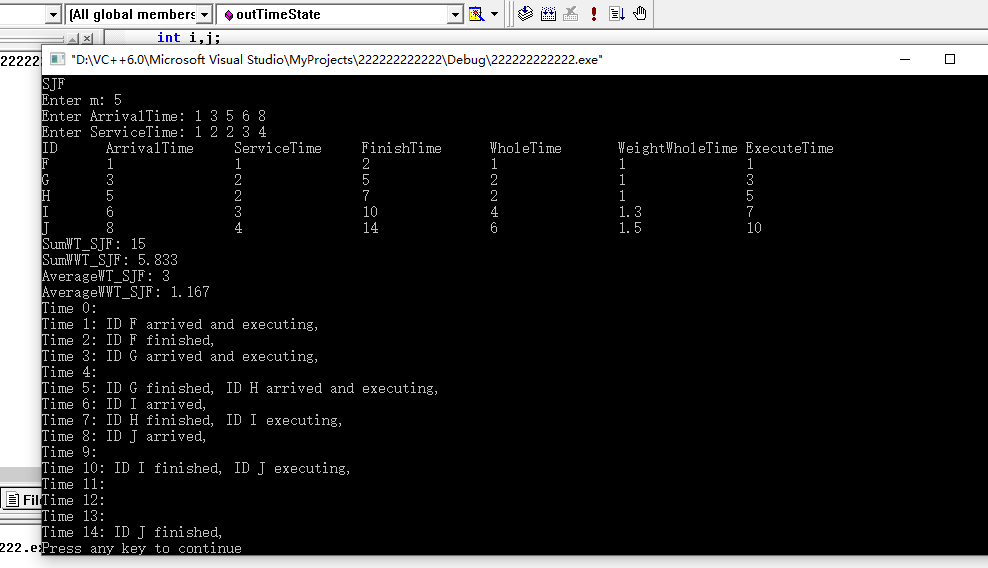
}

## 5. 实验结果截图

FCFS



SJF



## 6.实验总结

写程序各种报错，一开始是语法错误，一一改过来后，因为循环开始值错误所以结束时间计算不对，因为没有加（double）所以带权周转时间如法计算出小数，因为周转时间和/带权周转时间和加错了所以平均周转之间/平均带权周转时间计算错误等问题，在经过一一排查后都成功解决了，并将FCFS和SJF公用部分封装成函数调用。

## 附录

#include <iostream>

#include <iomanip>

using namespace std;

#define MaxNum 100//允许的最大进程数

static int n=0,m=0;//进程数

static char ID[MaxNum];//各进程ID

static int ExecuteTime[MaxNum];//各进程的开始执行时间

static int ArrivalTime[MaxNum];//各进程的达到时间

static int ServiceTime[MaxNum];//各进程的服务时间

static int ServiceTime\_SJF[MaxNum];//各进程的服务时间(SJF的临时数组)

static int FinishTime[MaxNum];//各进程的完成时间

static int WholeTime[MaxNum];//各进程的周转时间

static double WeightWholeTime[MaxNum];//各进程的带权周转时间

static double AverageWT\_FCFS=0,AverageWT\_SJF=0;//FCFS和SJF的平均周转时间

static double AverageWWT\_FCFS=0,AverageWWT\_SJF=0;//FCFS和SJF的平均带权周转时间

static int SumWT\_FCFS=0,SumWT\_SJF=0;//FCFS、SJF中的周转时间总和

static double SumWWT\_FCFS=0,SumWWT\_SJF=0;;//FCFS、SJF中的带权周转时间总和

int Enter();//输入进程数、各进程到达、服务时间

int sortArrivalTime(int a, int b);//根据达时间排序

int sortServiceTime(int a, int b);//已到达的进程服务时间排序

int calculationInitTime(int a, int b);//计算第一个进程的各个时间

int calculationTime(int i);//计算完成时间、周转时间、带权周转时间、开始执行时间

int display(int a, int b);//输出ID、到达时间、服务时间、完成时间、周转时间、带权周转时间、开始执行时间、周转时间总和、带权周转时间总和、平均周转时间、平均带权周转时间

int outTimeState(int a, int b);//输出各时间状态

void FCFS();

void SJF();

int main(){

int i,j;

//初始化

for(i=0;i<MaxNum;i++) {

ID[i] = i+65;//自动将进程名字顺序编号为A、B、C、D、E等

}

FCFS();

SJF();

return 0;

}

//输入进程数、各进程到达、服务时间

int Enter(int arr[], int a, int b) {

int i;

for(i=a;i<b;i++) {

cin>>arr[i];

}

return 0;

}

//根据达时间排序

int sortArrivalTime(int a, int b) {

int i,j;

int t1,t2,t3;

for(i=a;i<b;i++) {

for(j=i+1;j<b;j++) {

if(ArrivalTime[i] > ArrivalTime[j]) {

t1 = ArrivalTime[i];

ArrivalTime[i] = ArrivalTime[j];

ArrivalTime[j] = t1;

t2 = ServiceTime[i];

ServiceTime[i] = ServiceTime[j];

ServiceTime[j] = t2;

t3 = ID[i];

ID[i] = ID[j];

ID[j] = t3;

}

}

}

return 0;

}

//已到达的进程服务时间排序

int sortServiceTime(int a, int b) {

int i,j;

int t1,t2,t3;

for(i=a;i<b;i++) {

if(ArrivalTime[i] <= FinishTime[a-1])

ServiceTime\_SJF[i] = ServiceTime[i];

}

for(i=a;i<b;i++) {

for(j=i;j<b;j++) {

if(ServiceTime\_SJF[i] > ServiceTime\_SJF[j] && ServiceTime\_SJF[i] != 0 && ServiceTime\_SJF[j] != 0) {

t1 = ServiceTime\_SJF[i];

ServiceTime\_SJF[i] = ServiceTime\_SJF[j];

ServiceTime\_SJF[j] = t1;

}

}

}

for(i=a;i<b;i++) {

if(ServiceTime[i] == ServiceTime\_SJF[a]) {

t1 = ArrivalTime[a];

ArrivalTime[a] = ArrivalTime[i];

ArrivalTime[i] = t1;

t2 = ServiceTime[a];

ServiceTime[a] = ServiceTime[i];

ServiceTime[i] = t2;

t3 = ID[a];

ID[a] = ID[i];

ID[i] = t3;

}

}

return 0;

}

//计算第一个进程的各个时间

int calculationInitTime( int i) {

FinishTime[i] = ArrivalTime[i] + ServiceTime[i];

WholeTime[i] = FinishTime[i] - ArrivalTime[i];

WeightWholeTime[i] = (double)WholeTime[i]/(double)ServiceTime[i];

ExecuteTime[i] = ArrivalTime[i];

return 0;

}

//计算完成时间、周转时间、带权周转时间、开始执行时间

int calculationTime(int i) {

if(ArrivalTime[i] <= FinishTime[i-1]) {

FinishTime[i] = FinishTime[i-1] + ServiceTime[i];

ExecuteTime[i] = FinishTime[i-1];

}

else {

FinishTime[i] = ArrivalTime[i] + ServiceTime[i];

ExecuteTime[i] = ArrivalTime[i];

}

WholeTime[i] = FinishTime[i] - ArrivalTime[i];

WeightWholeTime[i] = (double)WholeTime[i]/(double)ServiceTime[i];

return 0;

}

//输出ID、到达时间、服务时间、完成时间、周转时间、带权周转时间、开始执行时间、周转时间总和、带权周转时间总和、平均周转时间、平均带权周转时间

int display(int a, int b) {

cout<<"ID"<<"\t"<<"ArrivalTime"<<"\t"<<"ServiceTime"<<"\t"<<"FinishTime"<<"\t"<<"WholeTime"<<"\t"<<"WeightWholeTime"<<"\t"<<"ExecuteTime"<<endl;

int i;

for(i=a;i<b;i++) {

cout<<ID[i]<<"\t"<<ArrivalTime[i]<<"\t"<<"\t"<<ServiceTime[i]<<"\t"<<"\t"<<FinishTime[i]<<"\t"<<"\t"<<WholeTime[i]<<"\t"<<"\t"<<setprecision(2)<<WeightWholeTime[i]<<"\t"<<"\t"<<ExecuteTime[i]<<endl;

}

if(a==0){

cout<<"SumWT\_FCFS: "<<SumWT\_FCFS<<endl<<"SumWWT\_FCFS: ";

cout<<setprecision(4)<<SumWWT\_FCFS<<endl<<"AverageWT\_FCFS: ";

cout<<setprecision(4)<<AverageWT\_FCFS<<endl<<"AverageWWT\_FCFS: ";

cout<<setprecision(4)<<AverageWWT\_FCFS<<endl;

}

else{

cout<<"SumWT\_SJF: "<<SumWT\_SJF<<endl<<"SumWWT\_SJF: ";

cout<<setprecision(4)<<SumWWT\_SJF<<endl<<"AverageWT\_SJF: ";

cout<<setprecision(4)<<AverageWT\_SJF<<endl<<"AverageWWT\_SJF: ";

cout<<setprecision(4)<<AverageWWT\_SJF<<endl;

}

return 0;

}

//输出各时间状态

int outTimeState(int a, int b) {

int i,j;

for(i=0;i<=FinishTime[b-1];i++) {

cout<<"Time "<<i<<": ";

for(j=a;j<b;j++) {

if(ArrivalTime[j] == i && ExecuteTime[j] == i)

cout<<"ID "<<ID[j]<<" arrived and executing, ";

else if(ArrivalTime[j] == i)

cout<<"ID "<<ID[j]<<" arrived, ";

else if(ExecuteTime[j] == i)

cout<<"ID "<<ID[j]<<" executing, ";

else if(FinishTime[j] == i)

cout<<"ID "<<ID[j]<<" finished, ";

}

cout<<endl;

}

return 0;

}

void FCFS(){

int i,j;

//FCFS输入进程数

cout<<"FCFS"<<endl;;

cout<<"Enter n: ";

cin>>n;

//FCFS各进程到达时间

cout<<"Enter ArrivalTime: ";

Enter(ArrivalTime,0,n);

//FCFS各进程服务时间

cout<<"Enter ServiceTime: ";

Enter(ServiceTime,0,n);

//根据达时间排序

sortArrivalTime(0,n);

//FCFS计算各时间

calculationInitTime(0);

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

calculationTime(i);

SumWT\_FCFS += WholeTime[i];

SumWWT\_FCFS += WeightWholeTime[i];

}

SumWT\_FCFS += WholeTime[0];

SumWWT\_FCFS += WeightWholeTime[0];

AverageWT\_FCFS = (double)SumWT\_FCFS/(double)n;

AverageWWT\_FCFS = (double)SumWWT\_FCFS/(double)n;

//FCFS输出各时间

display(0,n);

//FCFS输出各时间状态

outTimeState(0,n);

}

void SJF(){

int i,j;

//SJF输入进程数

cout<<"SJF"<<endl;

cout<<"Enter m: ";

cin>>m;

//各进程到达时间

cout<<"Enter ArrivalTime: ";

Enter(ArrivalTime,n,n+m);

//各进程服务时间

cout<<"Enter ServiceTime: ";

Enter(ServiceTime,n,n+m);

//根据达时间排序

sortArrivalTime(n,n+m);

//SJF计算各时间

calculationInitTime(n);

for(i=n+1;i<n+m;i++) {

sortServiceTime(i,n+m);

calculationTime(i);

SumWT\_SJF += WholeTime[i];

SumWWT\_SJF += WeightWholeTime[i];

}

SumWT\_SJF += WholeTime[n];

SumWWT\_SJF += WeightWholeTime[n];

AverageWT\_SJF = (double)SumWT\_SJF/(double)m;

AverageWWT\_SJF = (double)SumWWT\_SJF/(double)m;

//SJF输出各时间

display(n,n+m);

//SJF输出各时间状态

outTimeState(n,n+m);

}