**先来先服务FCFS和短作业优先SJF进程调度算法**

**1、实验目的**

通过这次实验，加深对进程概念的理解，进一步掌握进程状态的转变、进程调度的策略及对系统性能的评价方法。

**2、需求分析**

**(1) 输入的形式和输入值的范围**

输入值：进程个数Num 范围：0<Num<=100

依次输入Num个进程的到达时间 范围：

依次输入Num个进程的服务时间 范围：

输入要使用的算法（1-FCFS，2-SJF） 范围：1或者2

**(2) 输出的形式（X表示变量）**

时刻X：进程X开始运行。 其完成时间：X 周转时间：X 带权周转时间：X

…（省略（Num-1）个）

平均周转时间：X

平均带权周转时间：X

**(3) 程序所能达到的功能**

输入进程个数Num，每个进程到达时间ArrivalTime[i]，服务时间ServiceTime[i]。采用先来先服务FCFS或者短作业优先SJF进程调度算法进行调度，计算每个进程的完成时间、周转时间和带权周转时间，并且统计Num个进程的平均周转时间和平均带权周转时间。

**3、概要设计**

说明本程序中用到的所有抽象数据类型的定义、主程序的流程以及各程序模块之间的层次(调用)关系。

**4、详细设计**

**5、调试分析**

**(1)调试过程中遇到的问题以及解决方法，设计与实现的回顾讨论和分析**

**开始的时候没有判断进程是否到达，导致短进程优先算法运行结果错误，后来加上了判断语句后就解决了改问题。**

**基本完成的设计所要实现的功能，总的来说，FCFS编写容易，SJF需要先找到已经到达的进程，再从已经到达的进程里找到进程服务时间最短的进程，再进行计算。**

**(2)算法的改进设想**

**改进：即使用户输入的进程到达时间没有先后顺序也能准确的计算出结果。（就是再加个循环，判断各个进程的到达时间先后，组成一个有序的序列）**

**(3)经验和体会**

**通过本次实验，深入理解了先来先服务和短进程优先进程调度算法的思想，培养了自己的动手能力，通过实践加深了记忆。**

**6、用户使用说明**

（1）输入进程个数Num

（2）依次输入Num个进程的到达时间

（3）依次输入Num个进程的服务时间

（4）选择要使用的算法

进程调度算法

#include<iostream>

#include<iomanip>

using namespace std;

static const int Max=100;

int ArrivalTime[Max];//到达时间

int ServiceTime[Max];//服务时间

int FinishTime[Max];//完成时间

int WholeTime[Max];//周转时间

double WeightWholeTime[Max];//帯权周庄时间

double AverageWT\_FCFS,AverageWT\_SJF; //平均周转时间

double AverageWWT\_FCFS,AverageWWT\_SJF;//平均帯权周转时间

int ServiceTime\_SJF[Max];//在SJF算法中使用到

int Num=0;

int NowTime=0;//记录当前时间

double SumWT=0,SumWWT=0;//SumWT用来计算总的周转时间，SumWWT用来计算总的帯权周转时间

int i;

int choice;//记录选择

// 先到先服务算法

void FCFS()//找最早到达的。

{

cout<<"--------------------------------------------------------------"<<endl;

cout<<"-----------------------------FCFS-----------------------------"<<endl;

cout<<"--------------------------------------------------------------"<<endl;

for(i=0;i<Num;i++)

{

if(ArrivalTime[i]>NowTime)//假如进程到达的时间比现在已经运行的时间NowTime大，说明在NowTime时刻进程未到达

{

NowTime=ArrivalTime[i];//把进程的到达时间赋给NowTime

}

NowTime+=ServiceTime[i];//把进程的服务时间加到NowTime上

FinishTime[i]=NowTime;//计算完成时间

WholeTime[i]=FinishTime[i]-ArrivalTime[i];//计算周转时间=完成时间-到达时间

WeightWholeTime[i]=(double)WholeTime[i]/ServiceTime[i];//计算带权周转时间=周转时间/服务时间

SumWT+=WholeTime[i];//计算总的周转时间

SumWWT+=WeightWholeTime[i];//计算总的帯权周转时间

}

AverageWT\_FCFS=SumWT/Num;//平均周转时间

AverageWWT\_FCFS=SumWWT/Num;//平均帯权周转时间

for(i=0;i<Num;i++)//依次输出结果

{

cout<<"时刻"<<FinishTime[i]-ServiceTime[i]<<": 进程"<<i+1<<"开始运行。"<<" 其完成时间："<<FinishTime[i]<<" 周转时间："<<WholeTime[i]<<setprecision(3)<<" 帯权周转时间："<<WeightWholeTime[i]<<setprecision(3)<<endl;

}

cout<<"平均周转时间："<<AverageWT\_FCFS<<endl;

cout<<"平均帯权周转时间："<<AverageWWT\_FCFS<<endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 短进程优先算法

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SJF()//找已经到达的且服务时间最短的进程（假定输入的进程是按照到达时间先后输入的）

{

cout<<"--------------------------------------------------------------"<<endl;

cout<<"-----------------------------SJF------------------------------"<<endl;

cout<<"--------------------------------------------------------------"<<endl;

int min=0;

NowTime=ArrivalTime[0]+ServiceTime[0];//计算第一次的NowTIme

FinishTime[0]=NowTime;//计算第一个进程的完成时间

ServiceTime\_SJF[0]=1000;//赋初值。

cout<<"时刻"<<FinishTime[0]-ServiceTime[0]<<": 进程"<<1<<"开始运行。";

int allin=0,j,k;

for(i=1;i<Num;i++)//进入循环，从第二个到达的进程开始

{

k=1;min=0;

if(allin==0)//找到已经到达的进程个数

{

j=0;

while(ArrivalTime[j]<=NowTime && j<Num)//已经到达的进程

{

j++;

if(j>=Num)

{

allin=1;

}

}

}

else

{

j=Num;

}

j=j-1;//j是已经到达的进程数

while(k<=j)//从已经到达的进程里找到服务时间最短的进程

{

if(ServiceTime\_SJF[k]==0)//进程的服务时间如果等于0，则跳过该进程

k++;

else

{

if(ServiceTime\_SJF[min]>ServiceTime\_SJF[k])//比较，找到服务时间最短的进程

min=k;

k++;

}

}

ServiceTime\_SJF[min]=0;//找完后置零，便于下一次循环时使用

NowTime+=ServiceTime[min];//累加当前时间

FinishTime[min]=NowTime;//完成时间

}

for(i=0;i<Num;i++)//计算周转时间，带权周转时间，总的周转时间和总的带权周转时间

{

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

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

SumWT+=WholeTime[i];

SumWWT+=WeightWholeTime[i];

}

AverageWT\_SJF=SumWT/Num;//平均周转时间

AverageWWT\_SJF=SumWWT/Num;//平均带权周转时间

cout<<" 其完成时间："<<FinishTime[0]<<" 周转时间："<<WholeTime[0]<<setprecision(3)<<" 帯权周转时间："<<WeightWholeTime[0]<<setprecision(3)<<endl;

for(i=1;i<Num;i++)//输出结果

{

cout<<"时刻"<<FinishTime[i]-ServiceTime[i]<<": 进程"<<i+1<<"开始运行。"<<" 其完成时间："<<FinishTime[i]<<" 周转时间："<<WholeTime[i]<<setprecision(3)<<" 帯权周转时间："<<WeightWholeTime[i]<<setprecision(3)<<endl;

}

cout<<"平均周转时间："<<AverageWT\_SJF<<endl;

cout<<"平均帯权周转时间："<<AverageWWT\_SJF<<endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 输入函数

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void input()

{

cout<<"请输入进程个数：";

cin>>Num;

while(Num>100||Num<=0)

{

cout<<"进程个数必须大于0且小于等于100！请重新输入进程个数：";

cin>>Num;

}

cout<<"-----------------------------------------"<<endl;

for(i=0;i<Num;i++)

{

cout<<"请输入第"<<i+1<<"个进程的到达时间：";

cin>>ArrivalTime[i];

}

cout<<"-----------------------------------------"<<endl;

for(i=0;i<Num;i++)

{

int data=0;

cout<<"请输入第"<<i+1<<"个进程的服务时间：";

cin>>data;

ServiceTime[i]=data;

ServiceTime\_SJF[i]=data;

}

cout<<"-----------------------------------------"<<endl;

cout<<"请选择要使用的算法(1-FCFS,2-SJF): ";

cin>>choice;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 主函数

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void main()

{

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

cout<<"\*\* 进程调度算法 \*\*"<<endl;

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

char flag='y';

Loop:

NowTime=0;SumWT=0;SumWWT=0;//参数初始化

input();//输入

if(choice==1)

FCFS();//调用FCFS算法

else if(choice==2)

SJF();//调用SJF算法

else//输入有误，则重新选择

{

while(1)

{

cout<<"您的选择有误!请重新选择!"<<endl;

cout<<"请选择要使用的算法(1-FCFS,2-SJF): ";

cin>>choice;

if(choice==1)

{

FCFS();

break;

}

else if(choice==2)

{

SJF();

break;

}

}

}

cout<<endl<<"是否继续使用该程序，按'y'或'Y'键继续，按其他任意键退出：";

cin>>flag;

if(flag=='y' || flag=='Y')

goto Loop;

}

调试情况:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

\*\* 进程调度算法 \*\*"<<endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

输入进程个数:5

````````````````````````````````````````````````````````````````````````````````````````

输入第一个进程到达时间:0

输入第二个进程到达时间:1

输入第三个进程到达时间:2

输入第四个进程到达时间：3

输入第五个进程到达时间：4

输入第一个进程服务时间;4

输入第二个进程服务时间;3

输入第三个进程服务时间;5

输入第四个进程服务时间;2

输入第五个进程服务时间;4

````````````````````````````````````````````````````````````````````````````````````````````````````

请选择要使用的算法(1-FCFS,2-SJF):1

--------------------------------------------------------------

-----------------------------FCFS------------------------------

--------------------------------------------------------------

时刻0: 进程开始运行 其完成时间：4 周转时间：4 带权周转时间：1

时刻 4: 进程开始运行 其完成时间： 7 周转时间： 6 带权周转时间：2

时刻7: 进程开始运行 其完成时间：12 周转时间：10 带权周转时间：2

时刻12：进程开始运行 其完成时间： 14 周转时间：11 带权周转时间：5.5

时刻14：进程开始运行 其完成时间： 18 周转时间：14 带权周转时间：3.5

平均周转时间：9

平均带权周转时间：2.8

是否继续使用该程序，按'y'或'Y'键继续，按其他任意键退出：

输入Y

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

\*\* 进程调度算法 \*\*"<<endl;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

输入进程个数:5

````````````````````````````````````````````````````````````````````````````````````````

输入第一个进程到达时间:0

输入第二个进程到达时间:1

输入第三个进程到达时间:2

输入第四个进程到达时间：3

输入第五个进程到达时间：4

输入第一个进程服务时间;4

输入第二个进程服务时间;3

输入第三个进程服务时间;5

输入第四个进程服务时间;2

输入第五个进程服务时间;4

````````````````````````````````````````````````````````````````````````````````````````````````````

请选择要使用的算法(1-FCFS,2-SJF):1

--------------------------------------------------------------

-----------------------------SJF------------------------------

--------------------------------------------------------------

时刻0: 进程开始运行 其完成时间：4 周转时间：4 带权周转时间：1

时刻 6: 进程开始运行 其完成时间： 9 周转时间： 8 带权周转时间：2.67

时刻13: 进程开始运行 其完成时间：18 周转时间：16 带权周转时间：3.2

时刻4：进程开始运行 其完成时间： 6 周转时间：3 带权周转时间：1.5

时刻9：进程开始运行 其完成时间： 13 周转时间：9 带权周转时间：2.25

平均周转时间：8

平均带权周转时间：2.12

是否继续使用该程序，按'y'或'Y'键继续，按其他任意键退出：

**错误（进程个数错误）：**

请输入进程个数：0

进程个进程个数必须大于0且小于等于100！请重新输入进程个数：101

进程个数必须大于0且小于等于100！请重新输入进程个数：-1

进程个数必须大于0且小于等于100！请重新输入进程个数：1000

进程个数必须大于0且小于等于100！请重新输入进程个数：

**错误（选择算法错误）：**

请选择要使用的算法(1-FCFS,2-SJF):3

您的选择有误!请重新选择!

请选择要使用的算法(1-FCFS,2-SJF):0

您的选择有误!请重新选择!

请选择要使用的算法(1-FCFS,2-SJF):-1

您的选择有请选择要使用的算法(1-FCFS,2-SJF):50

您的选择有误!请重新选择!误!请重新选择!

您的选择有请选择要使用的算法(1-FCFS,2-SJF):