实验四：进程调度

一、 实验目的

1、理解进程调度的过程。

2、掌握各种进程调度算法的实现方法

3、通过实验比较各种进程调度算法的优劣。

进程调度算法是系统管理进程调度，提高系统性能的重要手段。通过本次实验理解进程调度的机制，在模拟实现先来先服务 FCFS、轮转 RR（q=1）、最短进程优先 SPN、最短剩余时间 SRT、最高响应比优先 HRRN 算法的基础上，比较各种进程调度算法的效率和优劣，从而了解系统进程调度的实现过程。

二、 实验内容

随机给出一个进程调度实例，如：

进程	到达时间	服务时间
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

模拟进程调度，给出按照算法先来先服务 FCFS、轮转 RR（q=1）、最短进程优先 SPN、最短剩余时间 SRT、最高响应比优先 HRRN 进行调度各进程的完成时间、周转时间、响应比的值。

三、 实验环境

PC + Linux Red Hat 操作系统

GCC

四、 实验原理及实验思路

1、**FCFS** 先来先服务也可以称为是 FIFO 先进先出。此策略是当前正在运行的进程停止执行时，选择在就绪队列中存在时间最长的进程运行。这种策略执行长进程比执行短进程更好。

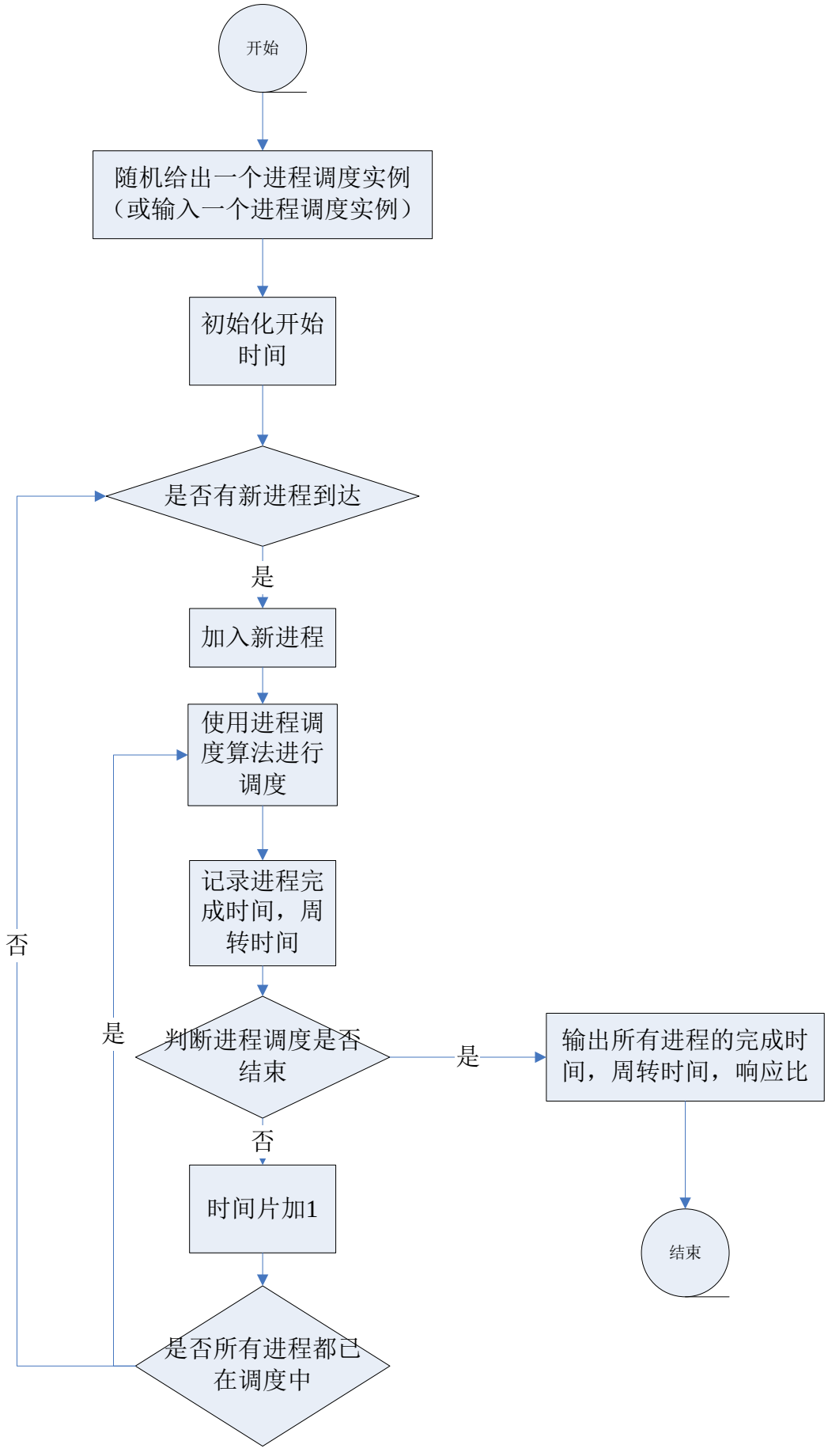
2、**轮转** 这种策略是以一个周期性间隔产生时钟中断，当中断发生时，当前正在运行的进程被置于就绪队列中，然后基于 FCFS 策略选择下一个就绪作业运行，目的是为了减少在 FCFS 策略下短作业的不利情况。

3、**SPN 最短进程优先** 这种策略是下一次选择所需处理时间最短的进程。是非抢占策略，目的也是为减少 FCFS 策略对长进程的偏向。

4、**SRT 最短剩余时间** 这种策略下调度器总是选择预期剩余时间最短的进程。是抢占策略。

5、**HRRN 最高响应比优先** 当当前进程完成或被阻塞时，选择响应比 R 最大的就绪进程，R=(w+s)/s 其中 w: 等待处理器的时间，s:期待的服务时间。这样长进程被饿死的可能性下降。

五、 流程图


开始
随机给出一个进程调度实例（或输入一个进程调度实例）
初始化开始时间
是否有新进程到达
是
加入新进程
使用进程调度算法进行调度
记录进程完成时间，周转时间
判断进程调度是否结束
是
输出所有进程的完成时间，周转时间，响应比
结束
否
时间片加1
是否所有进程都已在调度中
是
否

六、　源代码

```
#include <stdio.h>
#include <math.h>
void FCFS(float Atime[],float Stime[]);
void SJF(float Atime[],float Stime[]);
void RR(float Atime[],float Stime[]);
void HRN(float Atime[],float Stime[]);
main()
{
char c;
int i;
float Atime[5]={0,2,4,6,8};
float Stime[5]={3,6,4,5,2};
for(i=0;i<5;i++)
printf("%5.2f    ",Atime[i]);
printf("\n");
for(i=0;i<5;i++)
printf("%5.2f    ",Stime[i]);
printf("\n");

scanf("%c",&c);
switch(c)
{ case 'F':FCFS(Atime,Stime);break;
   case 'S':SJF(Atime,Stime);break;
   case 'R':RR(Atime,Stime);break;
   case 'H':HRN(Atime,Stime);break;
   default:printf("error");
}
}
void FCFS(float Atime[],float Stime[])
{
float Ctime[5];
float ZZtime[5];
float DQtime[5];
int i;
Ctime[0]=Stime[0];
for(i=1;i<5;i++)
Ctime[i]=Stime[i]+Ctime[i-1];
for(i=0;i<5;i++)
printf("%5.2f    ",Ctime[i]);
printf("\n");
for(i=0;i<5;i++){
ZZtime[i]=Ctime[i]-Atime[i];
```

```
printf("%5.2f    ",ZZtime[i]);}
printf("\n");
for(i=0;i<5;i++)
{DQtime[i]=ZZtime[i]/Stime[i];
printf("%5.2f    ",DQtime[i]);}
printf("\n");
}


void SJF(float Atime[],float Stime[])
{
float Ctime[5];
float ZZtime[5];
float DQtime[5];
int i,j;
float m,n;
Ctime[0]=Stime[0];
ZZtime[0]=Ctime[0]-Atime[0];
DQtime[0]=ZZtime[0]/Stime[0];
for(j=1;j<=3;j++)
for(i=1;i<=4-j;i++)
if(Stime[i]>Stime[i+1])
{
      m=Stime[i];Stime[i]=Stime[i+1];Stime[i+1]=m;
      n=Atime[i];Atime[i]=Atime[i+1];Atime[i+1]=n;
}
for(i=0;i<5;i++)
printf("%5.2f    ",Atime[i]);
printf("\n");
for(i=0;i<5;i++)
printf("%5.2f    ",Stime[i]);
printf("\n");
for(i=1;i<5;i++)
{
      Ctime[i]=Stime[i]+Ctime[i-1];
}
for(i=0;i<5;i++)
printf("%5.2f    ",Ctime[i]);
printf("\n");

for(i=1;i<5;i++)
ZZtime[i]=Ctime[i]-Atime[i];

for(i=0;i<5;i++)
```

```
printf("%5.2f    ",ZZtime[i]);
printf("\n");
for(i=1;i<5;i++)
DQtime[i]=ZZtime[i]/Stime[i];

for(i=0;i<5;i++)
printf("%5.2f    ",DQtime[i]);
printf("\n");
}

void RR(float Atime[],float Stime[])
{
float Ctime[5];
float ZZtime[5];
float DQtime[5];
int i,j,h,f,q;
float t[5],k;
scanf("%d",&q);
for(i=0;i<5;i++)
{
      if(q<Stime[i])
      {
for(j=0;j<5;j++)
{
      k=Stime[j]/q-1;
            t[j]=Atime[j]+k*(5/q)+1;
            for(h=0;h<5;h++)
            {
                  if(Stime[h]<k)
                  {
                  f=h;
                  }
            else
                  Ctime[j]=t[j];
            }
            Ctime[j]=t[j]-(k-Stime[f]);
      }
      }

      else
      {
      Ctime[0]=Stime[0];
      for(i=1;i<5;i++)
      Ctime[i]=Stime[i]+Ctime[i-1];
```

```
        }
for(i=0;i<5;i++)
        ZZtime[i]=Ctime[i]-Atime[i];

        for(i=0;i<5;i++)
DQtime[i]=ZZtime[i]/Stime[i];

for(i=0;i<5;i++)
printf("%5.2f    ",Ctime[i]);
printf("\n");
for(i=0;i<5;i++)
printf("%5.2f    ",ZZtime[i]);
printf("\n");
for(i=0;i<5;i++)
printf("%5.2f    ",DQtime[i]);
printf("\n");
}
}

void HRN(float Atime[],float Stime[])
{
float Ctime[5];
float ZZtime[5];
float DQtime[5];
int i,j,t=1;
float m,n,k,f;
Ctime[0]=Stime[0];
ZZtime[0]=Ctime[0]-Atime[0];
DQtime[0]=ZZtime[0]/Stime[0];
for(j=1;j<=3-t;j++)
for(i=1;i<=4-j-t;i++)
{
        k=Ctime[j-1]-Atime[i];
        f=Ctime[j-1]-Atime[i+1];
if(k/Stime[i]<f/Stime[i+1])
{
        m=Stime[i];Stime[i]=Stime[i+1];Stime[i+1]=m;
        n=Atime[i];Atime[i]=Atime[i+1];Atime[i+1]=n;
        t++;
}
}
for(i=0;i<5;i++)
printf("%5.2f    ",Atime[i]);
printf("\n");
```

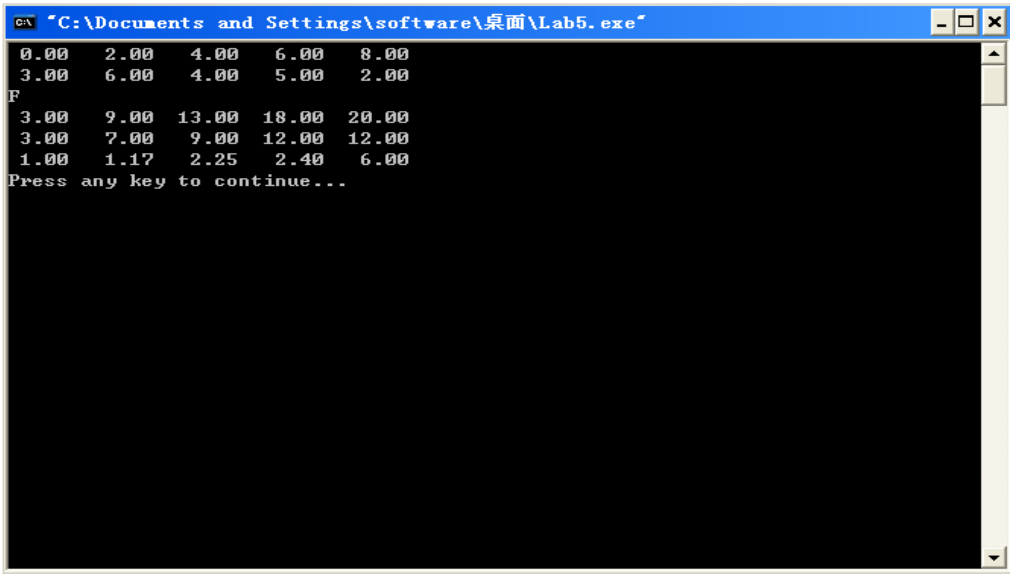
```
for(i=0;i<5;i++)
printf("%5.2f    ",Stime[i]);
printf("\n");
for(i=1;i<5;i++)
{
      Ctime[i]=Stime[i]+Ctime[i-1];
}
for(i=0;i<5;i++)
printf("%5.2f    ",Ctime[i]);
printf("\n");

for(i=1;i<5;i++)
ZZtime[i]=Ctime[i]-Atime[i];

for(i=0;i<5;i++)
printf("%5.2f    ",ZZtime[i]);
printf("\n");
for(i=1;i<5;i++)
DQtime[i]=ZZtime[i]/Stime[i];

for(i=0;i<5;i++)
printf("%5.2f    ",DQtime[i]);
printf("\n");
}
```

七、　运行结果及其分析

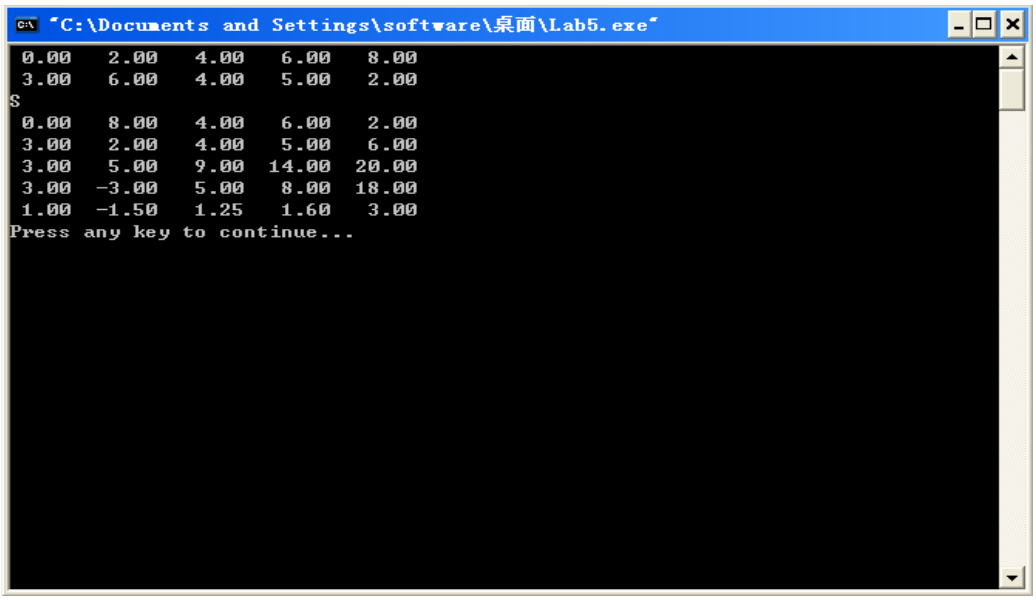
当程序运行，打印出 5 个进程到达时间：0 2 4 6 8 和服务时间：3 6 4 5 2 时，然后再输入要选择的调动服务算法 F :FCFS　S :SJF　R :RR　H :HRN　可得到不同的完成情况。

5 个进程完成时间，周转时间，响应比如下面各图中每行所示：

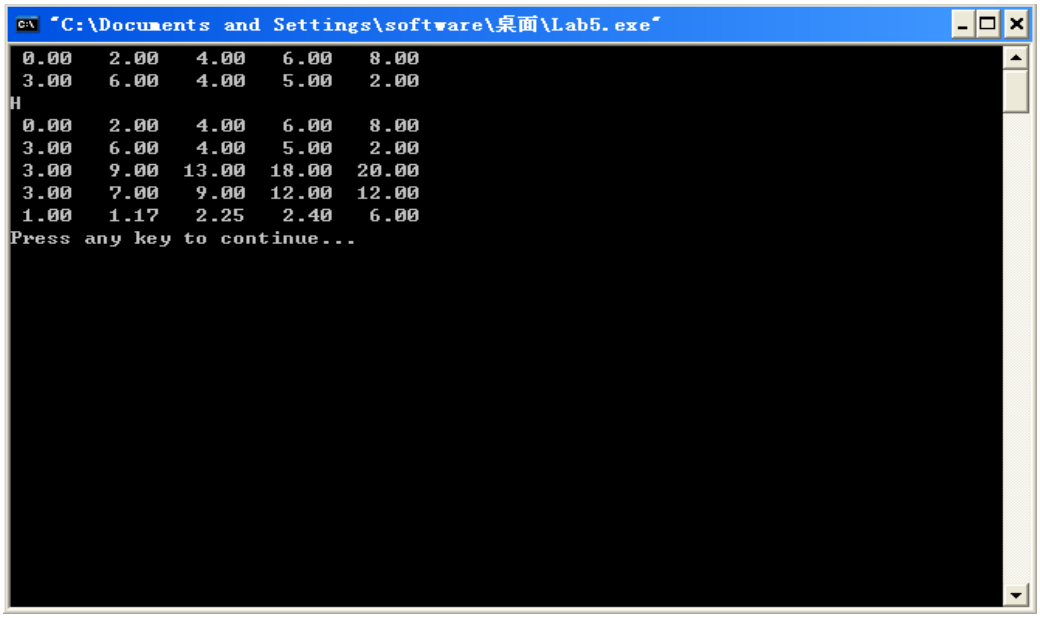
FCFS 的完成情况：



SJF 的完成情况



HRN 的完成情况:



在 RR 的选择运行过程中出现未知问题不能显示，所以暂时没有图片