操作系统实验一（进程调度算法）

原创

GhostRiderQin 于 2019-12-07 20:16:44 发布 7939 收藏 145

分类专栏： 操作系统 文章标签： 进程调度 c语言 操作系统

本文链接：https://blog.csdn.net/qq_40986486/article/details/103439054

操作系统 专栏收录该内容

5 篇文章 1 订阅

订阅专栏

今日闲来无聊，发现很早之前写的操作系统实验还没有整理，再加上有很多人问，索性就发成博客吧。

实验一 进程调度算法

一、实验目的

用高级语言编写和调试一个进程调度程序，以加深对进程的概念及进程调度算法的理解.

二、实验指导

设计一个有 N个进程共行的进程调度程序。

进程调度算法：分别采用先来先服务算法、短作业优先算法、高响应比优先算法实现。

每个进程用一个进程控制块（ PCB）表示。进程控制块可以包含如下信息：进程名、优先级、到达时间、要求服务时间、进程状态等等。 其中到达时间和要求服务时间可以在程序中进行初始化或者在程序开始时由键盘输入。

每个进程的状态可以是就绪 W（Wait）、运行R（Run）、或完成F（Finish）三种状态之一。

每个进程完成后要打印该作业的开始运行时刻、完成时刻、周转时间和带权周转时间，这一组进程完成后要计算并打印这组进程的平均周转时间、带权平均周转时间。

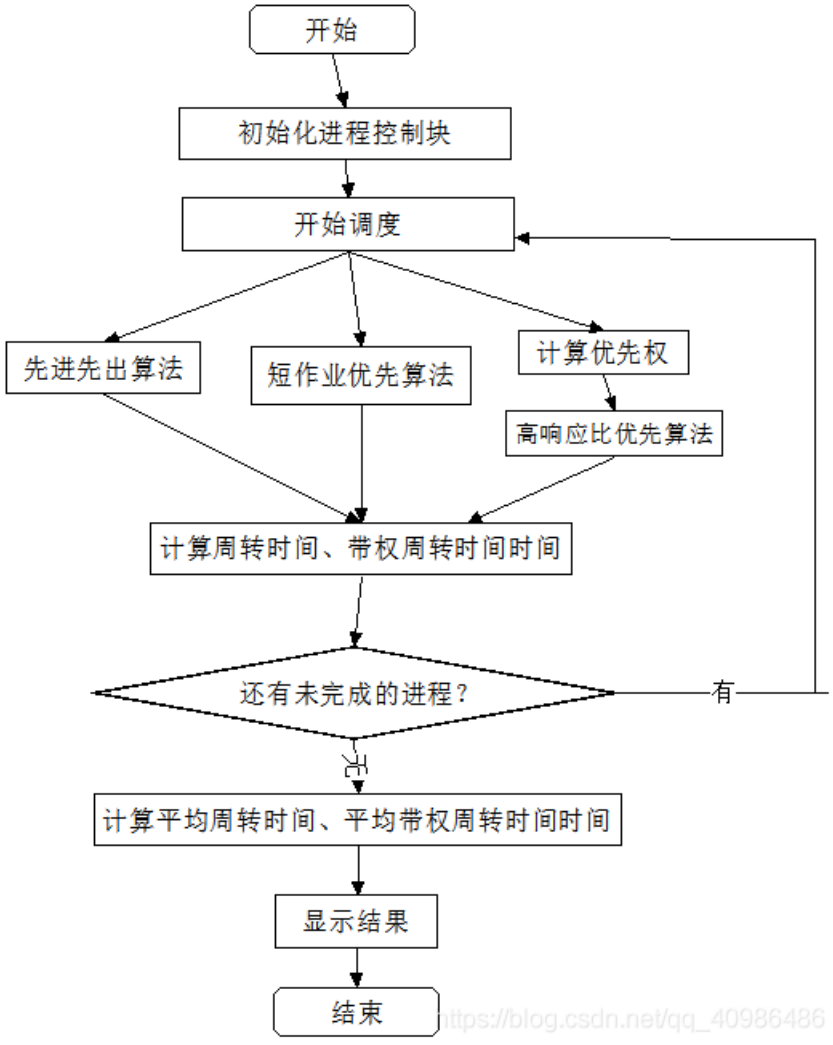
三、提示

1、在采用短作业优先算法和高响应比优先算法进行调度时应注意进程的到达时间，对于没有到达的进程不应参与调度。

2、注意在采用高响应比优先算法时计算优先权的时机，因为采用动态优先权，所以应在每次调度之前都重新计算优先权，高响

$$优先权=\frac{等待时间+要求服务时间}{要求服务时间}$$

应比优先算法采用下列公式计算优先权



进程调度算法流程图

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
struct PCB
{
```

```
 string name;
 int arrive_time;
 int service_time;
 int exit_time;
 int wait_time;
 double quan;
 bool ifrun;
 int RR_time;
} pcb[10];
queue<PCB> q;
PCB Que[100];
PCB MQ[50][50];
PCB fcfs[10];
PCB sjf[10];
PCB hrrn[10];
PCB rr[10];
PCB mfq[10];
int top=0;
bool cmpSJF(PCB a,PCB b)
{
 return a.service_time<b.service_time;
}

bool cmpHRRN(PCB a,PCB b)
{
 return a.quan>b.quan;
}
void menu()
{
 printf("进程调度模拟程序\n\n");
 printf("1. 输入作业情况\n\n");
 printf("2. 显示作业情况\n\n");
 printf("3. 先来先服务算法\n\n");
 printf("4. 短作业优先算法\n\n");
 printf("5. 高响应比优先算法\n\n");
 printf("6. 时间片轮转算法\n\n");
 printf("7. 多级反馈队列调度\n\n");
 printf("8. 算法结果对比\n\n");
 printf("0. 退出\n\n");
}
void init()
{
 pcb[0]= {"A",0,3,0,0,0,false,0};
 pcb[1]= {"B",2,6,0,0,0,false,0};
 pcb[2]= {"C",4,4,0,0,0,false,0};
 pcb[3]= {"D",6,5,0,0,0,false,0};
 pcb[4]= {"E",8,2,0,0,0,false,0};
}
void input()
{
 for(int i=0; i<5; i++)
 {
  printf("进程%d:\n",i);
  printf("请输入进程名称:");
  cin>>pcb[i].name;
  printf("请输入到达时间:");
  cin>>pcb[i].arrive_time;
  printf("请输入服务时间:");
  cin>>pcb[i].service_time;
```

```
  printf("\n");
 }
}
void display()
{
 printf("名称");
 printf("     ");
 for(int i=0; i<=4; i++)
  cout<<pcb[i].name<<" ";
 printf("\n到达时间 ");
 for(int i=0; i<=4; i++)
  cout<<pcb[i].arrive_time<<" ";
 printf("\n服务时间 ");
 for(int i=0; i<=4; i++)
  cout<<pcb[i].service_time<<" ";
 printf("\n");
 system("pause");
}

void output(PCB a[])
{
 int sumzhou = 0;
 double sumdai = 0;
 printf("名称\t到达时间\t服务时间\t完成时间\t周转时间\t带权周转时间\n");
 for(int i=0; i<=4; i++)
 {
  cout<<a[i].name<<"\t";
  printf("%d\t\t%d\t\t%d\t\t%d\t\t%.2f\n",a[i].arrive_time,a[i].service_time,a[i].exit_time,a[i].exit_time-a[i].
arrive_time,(a[i].exit_time*1.0-a[i].arrive_time)/a[i].service_time*1.0);
 }
 for(int i=0; i<=4; i++)
 {
  sumzhou+=a[i].exit_time-a[i].arrive_time;
  sumdai+=(a[i].exit_time*1.0-a[i].arrive_time)/a[i].service_time*1.0;
 }
 printf("平均周转时间：%.2f   平均带权周转时间：%.2f",sumzhou*1.0/5,sumdai/5);
 system("pause");
}
void outputQue(int front,int rear)
{

 printf("\t当前就绪队列： " );
 for(int i=front; i<rear; i++)
 {
  cout<<Que[i].name;
 }
 cout<<endl;
}
void outputMQ(int index,int front[],int rear[])
{
 cout<<endl;
 for(int i=0; i<index; i++)
 {

  printf("\t%d级队列： ",i+1);
  for(int j=front[i]; j<rear[i]; j++)
  {
   cout<<MQ[i][j].name;

  }
```

```
 cout<<endl;
 }
 cout<<endl<<endl;
}
void outsum()
{
 cout<<"先来先服务:"<<endl;
 output(fcfs);
 cout<<"短作业优先:"<<endl;
 output(sjf);
 cout<<"高响应比:"<<endl;
 output(hrrn);
 cout<<"时间片轮转:"<<endl;
 output(rr);
 cout<<"多级反馈队列:"<<endl;
 output(mfq);
}
void FCFS()
{
 top=0;
 PCB p= {"0",0,0,0};
 int now = 0;
 int flag = 0; //0空闲 1运行
 int r=0;
 while(1)
 {
  //printf("%d\n",now);
  //如果当前空闲，并且不是第一次空闲，说明有进程执行完毕
  if(flag == 0&&p.name!="0")
  {
   cout<<now<<" "<<p.name<<"执行完毕"<<endl;
   p.ifrun = true;
   fcfs[top++]=p;
  }
  //如果进程全部执行完
  if(top == 5)
   break;
  //查看这一秒是否有进程到达
  for(int i=r; i<5; i++)
  {
   if(pcb[i].arrive_time == now)   //此时间有进程到达
   {
    q.push(pcb[i]); //入队
    cout<<now<<" "<<pcb[i].name<<"到达"<<endl;
    r=i+1;
   }
  }
  if(flag == 0&&!q.empty())   //空闲
  {
   //取队头执行
   p = q.front();
   q.pop();
   cout<<now<<" "<<p.name<<"开始执行"<<endl;
   p.exit_time = p.service_time+now;
   flag = p.service_time;
  }
  //下一秒
  now++;
  if(flag>0)
```

```
   flag--;
  Sleep(500);
 }
 output(fcfs);
}

void SJF()
{
 top = 0;
 int front = 0;
 int rear = 0;
 PCB p= {"0",0,0,0};
 int now = 0;
 int flag = 0; //0空闲 1运行
 int r=0;
 while(1)
 {
  //printf("%d\n",now);
  //如果当前空闲，并且不是第一次空闲，说明有进程执行完毕
  if(flag == 0&&p.name!="0")
  {
   cout<<now<<" "<<p.name<<"执行完毕"<<endl;
   p.ifrun = true;
   sjf[top++]=p;
  }
  //如果进程全部执行完
  if(top == 5)
   break;
  //查看这一秒是否有进程到达
  for(int i=r; i<5; i++)
  {
   if(pcb[i].arrive_time == now)   //此时间有进程到达
   {
    Que[rear++]=pcb[i]; //入队
    cout<<now<<" "<<pcb[i].name<<"到达"<<endl;
    sort(Que+front,Que+rear,cmpSJF);
    r=i+1;
   }
  }
  if(flag == 0&&front!=rear)   //空闲
  {
   //取队头执行
   p = Que[front++];
   cout<<now<<" "<<p.name<<"开始执行"<<endl;
   p.exit_time = p.service_time+now;
   flag = p.service_time;
  }
  //下一秒
  now++;
  if(flag>0)
   flag--;
  Sleep(500);
 }
 output(sjf);
}

void HRRN()
{
 top = 0;
 int front = 0;
```

```
int rear = 0;
PCB p= {"0",0,0,0};
int now = 0;
int flag = 0; //0空闲 1运行
int r=0;
while(1)
{
 //printf("%d\n",now);
 //如果当前空闲，并且不是第一次空闲，说明有进程执行完毕
 if(flag == 0&&p.name!="0")
 {
  cout<<now<<" "<<p.name<<"执行完毕"<<endl;
  p.ifrun = true;
  hrrn[top++]=p;
 }
 //如果进程全部执行完
 if(top == 5)
  break;
 //查看这一秒是否有进程到达
 for(int i=r; i<5; i++)
 {
  if(pcb[i].arrive_time == now)   //此时间有进程到达
  {
   cout<<now<<" "<<pcb[i].name<<"到达"<<endl;
   Que[rear++]=pcb[i]; //入队
   r=i+1;
  }
 }
 if(flag == 0&&front!=rear)   //空闲
 {
  //计算优先权并排序
  for(int j=front; j<rear; j++)
   Que[j].quan = (Que[j].wait_time*1.0+Que[j].service_time)/Que[j].service_time;
  sort(Que+front,Que+rear,cmpHRRN);
  //取队头
  p = Que[front++];
  cout<<now<<" "<<p.name<<"开始执行"<<endl;
  p.exit_time = p.service_time+now;
  flag = p.service_time;
 }
 //下一秒
 now++;
 for(int i = front; i<rear; i++)
 {
  Que[i].wait_time++;
 }
 if(flag>0)
  flag--;
 Sleep(500);
}
output(hrrn);
}

void RR()
{
 int t=0;
 top = 0;
 int front = 0;
 int rear = 0;
```

```
 PCB p= {"0",0,0,0};
 int now = 0;
 int flag = 0; //0空闲 1运行
 int r=0;
 int time;
 printf("请输入时间片间隔:");
 scanf("%d",&time);
 while(1)
 {
  if(t == time)//时间片用完
   t=0;
  //扫描入队
  for(int i=r; i<5; i++)
  {
   if(pcb[i].arrive_time == now)   //此时间有进程到达
   {
    cout<<now<<" "<<pcb[i].name<<"到达"<<endl;
    Que[rear++]=pcb[i]; //入队
    outputQue(front,rear);
    r=i+1;
   }
  }
  if(p.name!="0") //不是第一次
  {
   if(p.RR_time >= p.service_time)//如果已经完成服务
   {
    cout<<now<<" "<<p.name<<" "<<"结束!"<<endl;
    p.exit_time=now;
    t=0;
    rr[top++]=p;
    if(top == 5)
     break;
    p.name="0";
   }
   else if(t == 0) //如果没有完成并且时间片用完了
   {
    cout<<now<<" "<<p.name<<" "<<"时间片用完!"<<endl;
    Que[rear++]=p;
    outputQue(front,rear);
   }

  }
  if(t == 0 && front!=rear)
  {
   p = Que[front++];
   cout<<now<<" "<<p.name<<"开始执行"<<endl;
   outputQue(front,rear);
  }
  now++;
  t++;
  p.RR_time++;
  Sleep(500);
 }
 output(rr);
}

void MFQ()
{
 int t=0;
 top = 0;
```

```
int num;
int front[10] ;
int rear[10] ;
memset(front,0,sizeof(front));
memset(rear,0,sizeof(rear));
int time[10]= {1,2,4,8,16};
printf("请输入队列的个数(最多为5): ");
scanf("%d",&num);
PCB p= {"0",0,0,0};
int now = 0;
int r=0;
int index=0;
t=0;
while(1)
{
 if(t == time[index])//时间片用完
  t=0;
 //扫描入队
 for(int i=r; i<5; i++)
 {
  if(pcb[i].arrive_time == now)   //此时间有进程到达
  {
   cout<<now<<" "<<pcb[i].name<<"到达"<<endl;
   MQ[0][rear[0]++]=pcb[i]; //入队
   outputMQ(num,front,rear);
   r=i+1;
  }
 }
 if(p.name!="0") //不是第一次
 {
  if(p.RR_time >= p.service_time)//如果已经完成服务
  {
   cout<<now<<" "<<p.name<<" "<<"结束!"<<endl;
   p.exit_time=now;
   t=0;
   mfq[top++]=p;
   if(top == 5)
    break;
   p.name="0";
  }
  else if(t == 0) //如果没有完成并且时间片用完了
  {
   cout<<now<<" "<<p.name<<" "<<"时间片用完!"<<endl;
   if(index+1<=num-1)
    MQ[index+1][rear[index+1]++]=p;
   else
    MQ[index][rear[index]++]=p;
   outputMQ(num,front,rear);
  }
 }
 if(t == 0)
 {
  int ff=0;
  for(int i=0; i<5; i++)
  {
   if(front[i]!=rear[i]) // 该级队列中有进程
   {
    ff=1;
    index=i;
```

```
     break;
    }
   }
   if(ff==1)
   {
    p = MQ[index][front[index]++];
    cout<<now<<" "<<p.name<<"开始执行"<<endl;
   }
   outputMQ(num,front,rear);
  }
  now++;
  t++;
  p.RR_time++;
  Sleep(500);
 }
 output(mfq);
}

int main()
{
 int n;
 init();

 while(1)
 {
  menu();
  scanf("%d",&n);
  switch(n)
  {
   case 1:
    input();
    break;
   case 2:
    display();
    break;
   case 3:
    FCFS();
    break;
   case 4:
    SJF();
    break;
   case 5:
    HRRN();
    break;
   case 6:
    RR();
    break;
   case 7:
    MFQ();
    break;
   case 8:
    outsum();
    break;
   case 0:
    exit(1);
    break;
   default :
    printf("请输入有效选项!\n");
  }
 }
```

```
}
}
```