石家庄铁道大学

操作系统实验

2017 年 春 季学期

学 院： 信息科学与技术学院

专 业： 软件工程

实 验 名 称： 进程模拟调度算法

学 生 姓 名： 祖浩然

学 号： 20142903

指 导 教 师： 陈娜

完 成 日 期： 2017年 4月25日

**进程模拟调度算法实验 实验报告**

课程名称 操作系统 信息科学与技术 学院 信1405-2班

实验者姓名 祖浩然 实验日期 2017年4月

评分 教师签名

**一、实验目的**

进程调度是处理机管理的核心内容。本实验要求用高级语言编写模拟进程调度程序，以便加深理解有关进程控制快、进程队列等概念，并体会和了解优先数算法和时间片轮转算法的具体实施办法。

1. **实验要求**

1.设计进程控制块 PCB 的结构，通常应包括如下信息：

进程名、进程优先数（或轮转时间片数）、进程已占用的 CPU 时间、进程到完成还需要的时间、进程的状态、当前队列指针等。

2.编写两种调度算法程序：

优先数调度算法程序循环轮转调度算法程序

3.按要求输出结果。

**三、实验过程**

**1．准备**

A．查阅相关资料；

B．初步编写程序；

C．准备测试数据；

**2．上机调试**

**3．主要流程和源代码**

开始

优先级，时间片算法

就绪队列为空

就绪队列为空

指针指向就绪队列第一个任务

指针便利就绪队列寻找优先级最高任务

Int i=0;i<round;i++

运行指针指向优先级最高任务，

并从就绪队列删除

指针运算，cputime+=1,needtiem-=1,if(needtime==0)break;

if

指针运算，cputime+=1,needtiem-=1,prio-=1

Needtime==0

Needtime==0

插入就绪队列队尾

插入完成队列

插入就绪队列队尾

插入完成队列

结束

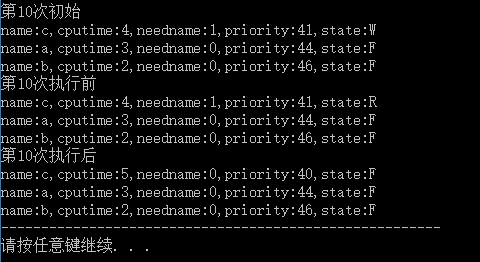
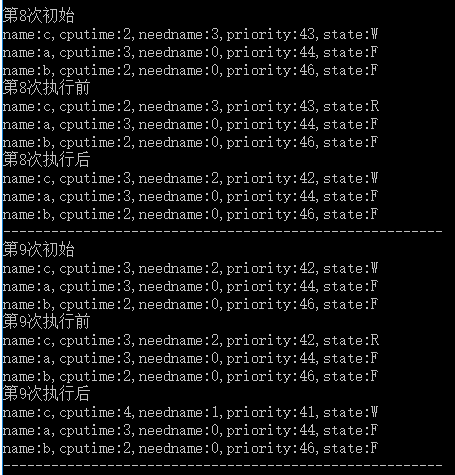
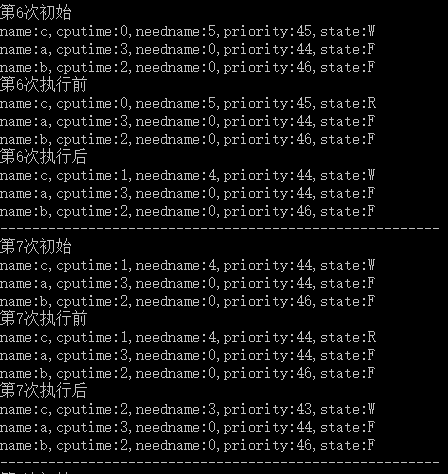
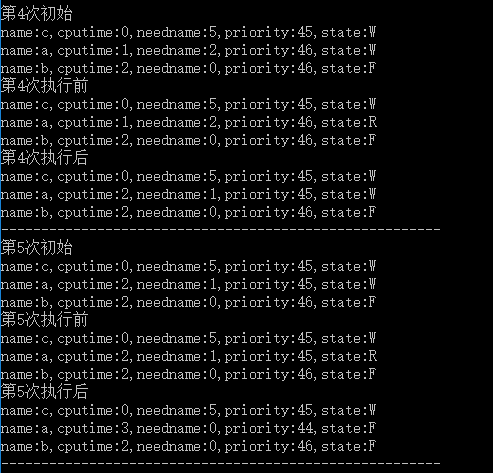
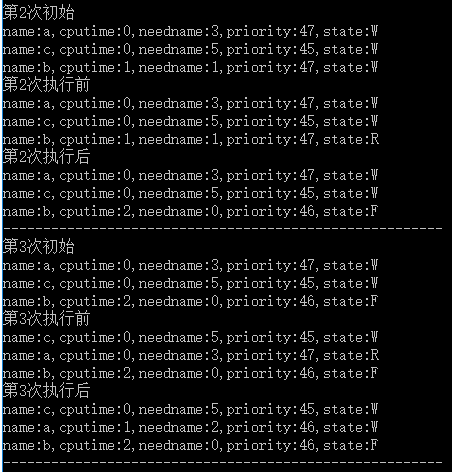
|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  typedef struct PCB  {  string name="";//进程标识符  int prio\_round=0;//进程优先数/进程轮转时间片数  int cputime=0;//进程累计占用cpu的  int needtime=0;  string state="";  PCB \*next=NULL;  }PCB;  void initlist(PCB\* &L)//第一个节点为空  {  L = new PCB;  L->next = NULL;  }  PCB readyHpfInput()  {  PCB pcb;  cin >> pcb.name;  cin >> pcb.needtime;  pcb.prio\_round = 50 - pcb.needtime;  pcb.state = "W";  return pcb;  }  PCB readyTimeInput()  {  PCB pcb;  cin >> pcb.name;  cin >> pcb.needtime;  pcb.prio\_round = 2;  pcb.state = "W";  return pcb;  }  //查找ready队列尾指针  void tailfind(PCB\* &ready, PCB\* &tail)  {  PCB\* s;  s = ready;  while (true)  {  if (s->next == NULL)  {  break;  }  s = s->next;  }  tail = s;  }  void readyadd(PCB\* &ready, PCB\* &tail,PCB pcb)//ready队列添加  {  PCB\* s;  s = new PCB;  \*s = pcb;  tail->next = s;  tail = tail->next;  }  PCB checkHpfpcb(PCB\* &ready,PCB\* &run ,PCB\* &tail)//查找优先级最高的任务,输入就绪队列头指针,run为要执行的任务  {  PCB\* s;  s = ready->next;  int prio=0;  int xh = 0,xz=0;//记录循环次数，第几个被选中  while (true)  {  if (s->prio\_round >= prio)  {  prio = s->prio\_round;  run = s;  xz = xh;  }  if (s->next==NULL)  {  break;  }  s = s->next;  xh++;  }  //删除选中任务  s = ready;  for (int i = 0; i < xz; i++)//指针到选中的前一个  {  s = s->next;  }  s->next = s->next->next;  tailfind(ready, tail);  //修改状态  run->state = "R";  return \*run;  }  PCB checkTimepcb(PCB\* &ready, PCB\* &run, PCB\* &tail)//查找队头任务,输入就绪队列头指针,run为要执行的任务  {  PCB\* s;  s = ready->next;//s取值第一个数  int prio = 0;  run = s;//run指针取值s  //删除选中任务  s = ready;  s->next = s->next->next;  tailfind(ready, tail);  //修改状态  run->state = "R";  return \*run;  }  void intoready(PCB\* &run,PCB\* &tail)//执行队列任务进入就绪队列队尾  {  run->state = "W";  tail->next = run;  //cout << tail->name << endl;  tail = tail->next;  //cout << tail->name << endl;  run = NULL;  }  void intofinsh(PCB\* &run ,PCB\* &finish)//执行队列任务进入完成队列  {  run->state = "F";  run->next=finish->next;  finish->next = run;  run = NULL;  }  void runHpfpcb(PCB\* &run ,PCB\* &tail ,PCB\* &finish)//任务在执行队列进行的计算。如果需要时间等于0，放入完成队列，否则放入就绪的队列队尾  {  run->cputime += 1;  run->needtime -= 1;  run->prio\_round -= 1;  run->next = NULL;  if (run->needtime <= 0)//如果需要时间小于等于0.放入完成队列  {  intofinsh(run,finish);  }  else//放入就绪队列队尾  {  intoready(run,tail);  }  }  void runTimepcb(PCB\* &run, PCB\* &tail, PCB\* &finish)//任务在执行队列进行的计算。如果需要时间等于0，放入完成队列，否则放入就绪的队列队尾  {  for (int i = 0; i < run->prio\_round; i++)  {  run->cputime += 1;  run->needtime -= 1;  if (run->needtime == 0)  {  break;  }  }  run->next = NULL;  if (run->needtime <= 0)//如果需要时间小于等于0.放入完成队列  {  intofinsh(run, finish);  }  else//放入就绪队列队尾  {  intoready(run, tail);  }  }  void pout(PCB\* &run)//输出run指针  {  PCB\* s;  if (run ==NULL)  {  }  else if(run->name=="")  {  }  else  {  s = run;  cout << "name:" << s->name << ",cputime:" << s->cputime << ",needname:" << s->needtime << ",priority:" << s->prio\_round << ",state:" << s->state << endl;  }  }  void listout(PCB\* &head)//输出链表  {  PCB\* s;  s = head;  while (true)  {  if (s->next==NULL)  {  break;  }  else if (s->next->name == "")  {  break;  }  else  {  s = s->next;  cout << "name:" << s->name << ",cputime:" << s->cputime << ",needname:" << s->needtime << ",priority:" << s->prio\_round << ",state:" << s->state << endl;  }  }  }  void allout(PCB\* &ready, PCB\* &run, PCB\* &finish)  {  listout(ready);  pout(run);  listout(finish);  }  void main()  {  PCB \*run;//运行进程指针  PCB \*ready;//就绪队列头指针  PCB \*tail;//就绪队列尾指针  PCB \*finish;//完成队列头指针  PCB pcb;  PCB zxrw;//进入执行队列的任务  int xz,num;//选择算法，输入数量  initlist(ready);  initlist(run);  initlist(finish);  tail = ready;  cout << "选择算法(输入1或2)：1.优先权算法，2.时间片轮转"<<endl;  cin >> xz;  cout << "输入个数" << endl;  cin >> num;  cout << "依次输入名称和需要时间" << endl;  if (xz == 1)  {  for (int i = 0; i < num; i++)  {  pcb = readyHpfInput();  readyadd(ready, tail, pcb);//就绪队列添加任务  }  int cs = 0;  while (true)  {  cs++;  if (ready->next == NULL&&run == NULL)  {  break;  }  //循环  cout << "第" << cs << "次初始" << endl;  allout(ready, run, finish);  checkHpfpcb(ready, run, tail);//查找优先级最高的任务  cout << "第" << cs << "次执行前" << endl;  allout(ready, run, finish);  runHpfpcb(run, tail, finish);//对进入执行队列的任务进行操作  cout << "第" << cs << "次执行后" << endl;  allout(ready, run, finish);  cout << "-------------------------------------------------------" << endl;  }  }  else if(xz==2)  {  for (int i = 0; i < num; i++)  {  pcb = readyTimeInput();  readyadd(ready, tail, pcb);//就绪队列添加任务  }  int cs = 0;  while (true)  {  cs++;  if (ready->next == NULL&&run == NULL)  {  break;  }  //循环  cout << "第" << cs << "次初始" << endl;  allout(ready, run, finish);  checkTimepcb(ready, run, tail);//查找优先级最高的任务  cout << "第" << cs << "次执行前" << endl;  allout(ready, run, finish);  runTimepcb(run, tail, finish);//对进入执行队列的任务进行操作  cout << "第" << cs << "次执行后" << endl;  allout(ready, run, finish);  cout << "-------------------------------------------------------" << endl;  }  }  else  {  cout << "输入错误";  }  } |

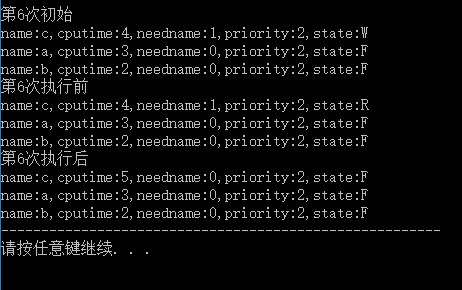
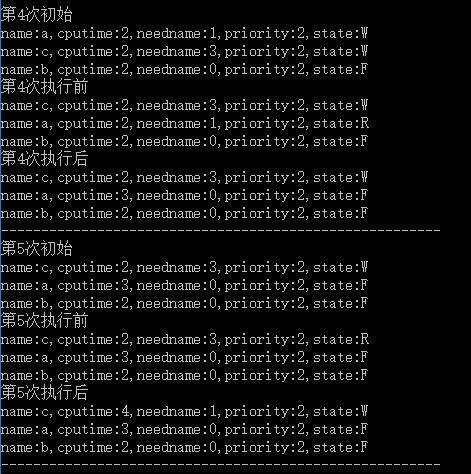
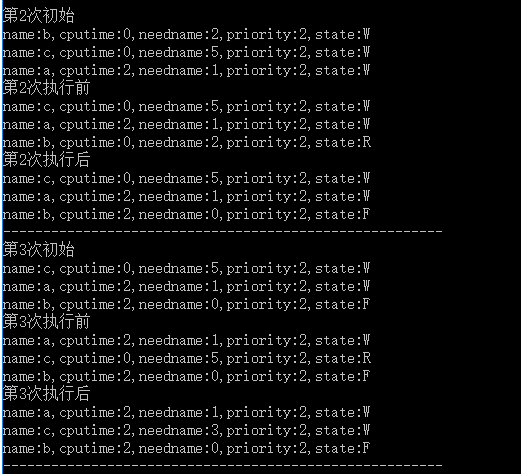
**4．遇到的主要问题和解决方法**

A．在编写优先级算法时，运行指针插入就绪队列队尾，出现了无限循环；解决方法，将运行指针的next域赋值为空。

B．时间片轮转发，出现了needtime=-1 的情况；将needtime-=round 改写为for(int i=0;i<rount ;i++){needtime-=1,if(needtime==0){break;}}。

四、实验结果





**五、实验总结**

通过本次实验，我学到了优先级调度算法和时间片轮转法的基本的用法，了解了调度算法的运行过程，掌握了调度算法的使用方法，提高了对调度算法的掌握程度。