云南大学数学与统计学实验教学中心

实验报告

课程名称：操作系统实验	**学期**：2015~2016 学年上学期	**成绩**：
指导教师： 彭程	**学生姓名**：金洋	**学号**：20131910023
实验名称：进程调度模拟		
实验编号：四	**实验日期**： 11 月 20 日	**实验学时**： 3
学院： 数学与统计学院	**专业**： 信息与计算科学	**年级**： 2013 级

一、实验目的

理解操作系统中进程调度的概念和调度算法
学习并掌握 Windows 进程控制以及进程间通信的基本知识；

二、实验内容

1、实现进程调度程序（scheduleprocess）,负责各系统的运行
2、实现创建进程命令
3、实现撤销进程命令
4、实现察看进程状态命令
5、实现时间片轮转调度算法

三、实验环境

平台：Visual C++ 6.0

语言：C 语言

四、算法介绍

1.实现进程调度程序（scheduleprocess）,负责各系统的运行；

2.实现创建进程命令：

格式 creat<name><time> name：进程名 time：计划运行时间，用户通过本命令发送创建进程请求，将进程信息提交各系统。系统创建进程为其分配一个唯一的进程标识 PID，并将状态置为 READY，然后放入就绪队列中；

3.实现撤销进程命令

格式：remove<name>输入撤销命令后，系统会删除待撤销的进程在缓冲区中的内容，如果输入有误，程序会有出错提示；

4.实现察看进程状态命令

格式：current 打印出当前运行进程和就绪队列中各进程的信息。状态信息应该包括：进程 PID、进程名、进程状态（ready、run、wait）

5.实现时间片轮转调度算法

处理机总是优先调度运行就绪队列中的第一个进程，当时间片结束后就 把该进程放在就绪队列的尾部。在系统的实现以及运行中，不必考虑 Windows 操作系统本身的进程调度。假设所有的作业均由 scheduleprocess 调 度执行，而且进程在分配给它的时间片内总是不间断地运行。

程序的改进：

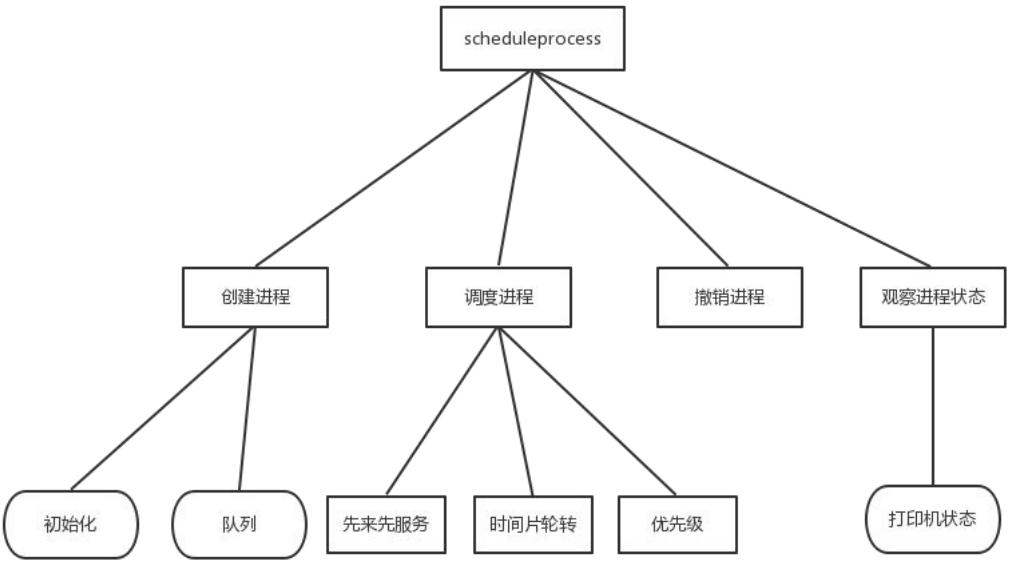
程序中采用的是轮转调度策略，可以对策略进行改进，实现多级反馈的轮转调度策略。每个进程有其动态的优先级，在用完分配的时间片后，可以被优先级更高的进程抢占运行。就绪队列中的进程等待时间越长，其优先级越高。每个进程都具有以下两种优先级：

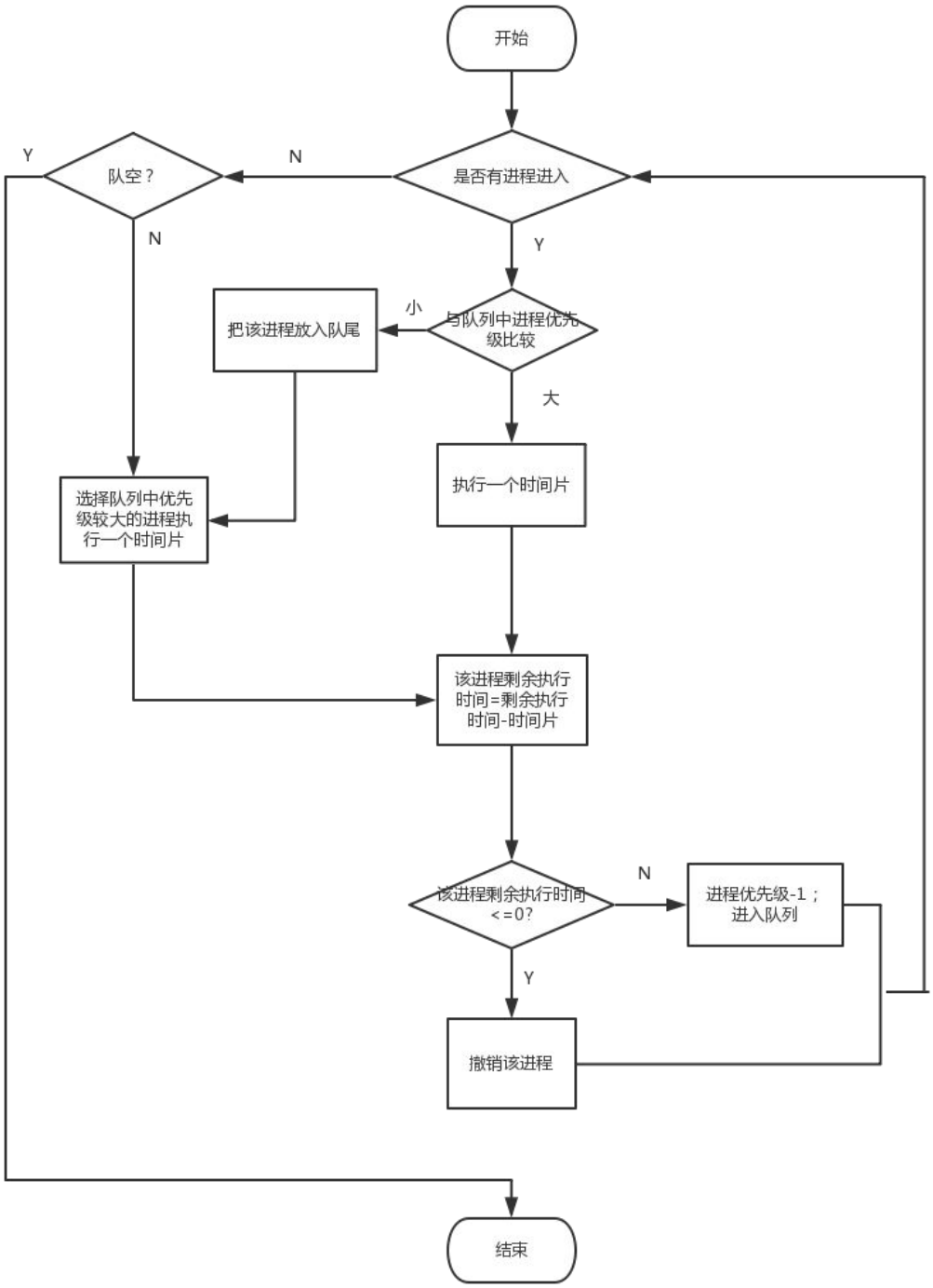
初始优先级（initial priority）:在进程创建时指定，保持不变，直到进程结束。

当前优先级（current priority）:由 scheduleprocess 调度更新，用以调度进程运行。Scheduleproccess 总是选择当前优先级最高的那个进程来运行。进程当前运行优先级的更新主要取决于以下两种情况：

一个进程在就绪对列中等待了若干个时间片（如 5 个），则将它的当前优先级加 1；若当前运行的进程时间片到，则中止其运行（抢占式多任务），将其放入就绪队列中，它的当前优先级也恢复为初始优先级；

结构图和优先级调度流程图：




开始
是否有进程进入
N
队空？
Y
N
Y
小
与队列中进程优先级比较
把该进程放入队尾
大
执行一个时间片
选择队列中优先级较大的进程执行一个时间片
该进程剩余执行时间=剩余执行时间-时间片
该进程剩余执行时间<=0?
N
进程优先级-1；进入队列
Y
撤销该进程
结束

五、调试过程

1、程序代码

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#define UNREADY 0 //未就绪
#define READY 1 //就绪
#define RUN 2 //运行
#define WAIT 3 //阻塞
#define LEN sizeof(struct PCB)
#define TIMER 5//每个时间片是 5 个单位时间

/*定义进程结构体*/
struct PCB
{
   int pID;//PID
   char name[10];
   int statue;//进程状态
   int needTime;//所需执行时间(以 1 为单位，下同)
   int waitTime;//等待时间
   int initialPriority;//初始优先级
   int currentPriority;//当前优先级
   struct PCB *next;
};

struct PCB *HEAD,*TAIL;
int PRONUM=0,WATCH=0;//总进程数
char *STATUE[]={"UNREADY","READY","RUN","WAIT"};


void printAuthor()
{
   printf("这是金洋(Sno:20131910023)的进程调度模拟实验.\n");
}
```

```
/*进程信息输入*/
void create()
{
   int num,i;

   printf("请输入你想建立的进程个数:");
   scanf("%d",&num);


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

      //工作指针
      struct PCB *P;
      P=(struct PCB*) malloc(LEN);
      PRONUM++;//更新总进程数
      printf("\n 请输入第%d 个进程信息:\n",PRONUM);
      P->pID=PRONUM;
      printf("请输入进程名:");
      scanf("%s",&P->name);
      P->statue=UNREADY;//初始时都是未就绪状态
      printf("请输入进程所需执行时间:");
      scanf("%d",&P->needTime);
      P->waitTime=0;
      printf("请输入进程优先级:");
      scanf("%d",&P->initialPriority);
      P->currentPriority=P->initialPriority;//初始时，当前优先级=
初始优先级
      P->next=NULL;

      /*将新进程放到队列尾部*/
      TAIL->next=P;
      TAIL=P;

   }
}
```

```
/*显示已创建的进程*/
void current()
{

   int i;
   struct PCB *P;
   P=HEAD->next;
   if (P==NULL)
   {
      printf("就绪队列无进程,无法显示进程信息.");
      return;
   }

   for (i=0;i<PRONUM;i++)
   {
      printf("PID:%d\n",P->pID);
      printf("进程名:%s\n",P->name);
      printf("进程状态:%s\n",STATUE[P->statue]);
      printf("\n");
      P=P->next;
   }
}
```

/*1、一个进程在就绪对列中等待了若干个时间单位（如 15 个），则将它的当前优先级加 1。

2、若当前运行的进程时间片到，则中止其运行（抢占式多任务），将其放入就绪队列中，它的当前优先级也恢复为初始优先级*/

/*优先级调度进程方式*/

```
void priModel()
{
   struct PCB *P,*PRun;//P 为工作指针，PRun 为指向优先级最高的进程



   while (PRONUM>0)
   {
      P=PRun=HEAD;
      if (P==NULL)
      {
         printf("就绪队列无进程,无法调度.");
         return;
      }


      /*寻找优先级最高的进程*/
      while (P->next!=NULL)
      {

         P=P->next;
         /*寻找优先级最高的进程*/
         if (P->currentPriority > PRun->currentPriority)
            PRun=P;
         else
         {
            /* 一个进程在就绪对列中等待了 15 个单位时间，则将它的当前优先级加 1 */
            /*实际上应该在时间片运转之后加，此处为了方便和节约时间，在寻找优先级最高进程的过程中提前加了优先级，效果是一样的*/

            P->waitTime+=TIMER;//一个优先级最高的进程被调度了，意味着其他进程都要等待 TIMER 个时间
            if (P->waitTime%15==0)
               P->currentPriority++;
         }
      }
```

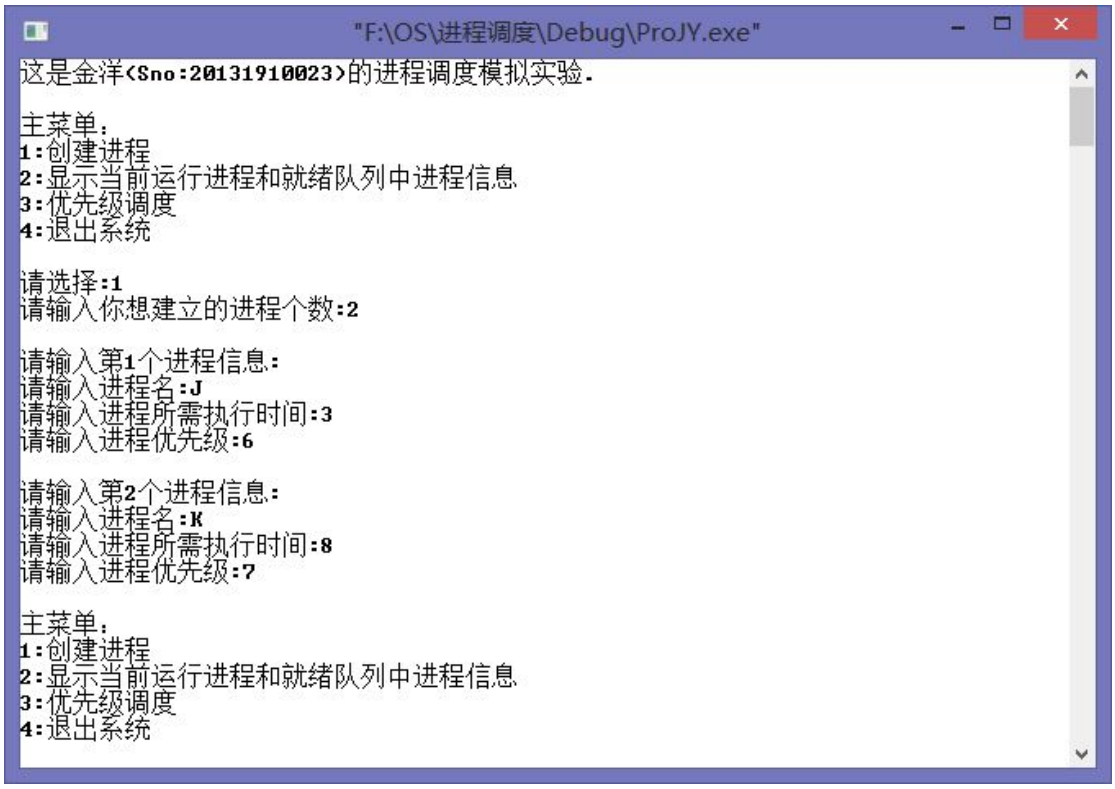
```
/*PRun 所指的进程被调度运行*/
int i;
PRun->statue=RUN;/*先将状态改变*/
for (i=0;i<TIMER;i++)
{
   Sleep(1000);
   printf("\n");
   printf("第%d 单位时间:\n",++WATCH);

   PRun->needTime--;
   if (PRun->needTime>=0)
   /*PRun 所指的进程还未运行完*/
   {
      printf("PID:%d\n",PRun->pID);
      printf("进程名:%s\n",PRun->name);
      printf("进程状态:%s\n",STATUE[PRun->statue]);
      printf("剩余执行时间:%d\n",PRun->needTime);
      printf("当前优先级:%d\n",PRun->currentPriority);
   }
   else
      printf("无进程运行\n");
}
```

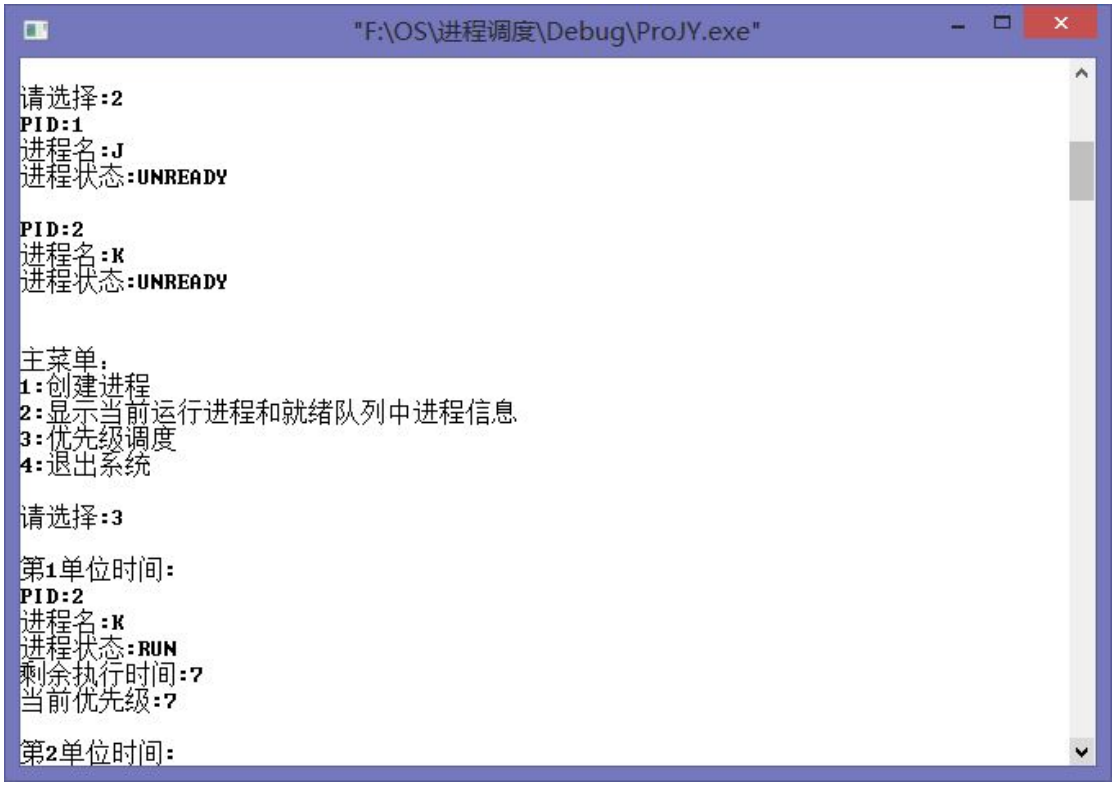
```
/*PRun 所指的进程是否运行完*/
if (PRun->needTime<=0)
{
   /*运行完时，删除该节点来模拟撤销进程*/
   P=HEAD;
   while (P->next!=PRun)
      P=P->next;
   if (PRun==TAIL)
   {
      TAIL=P;
```

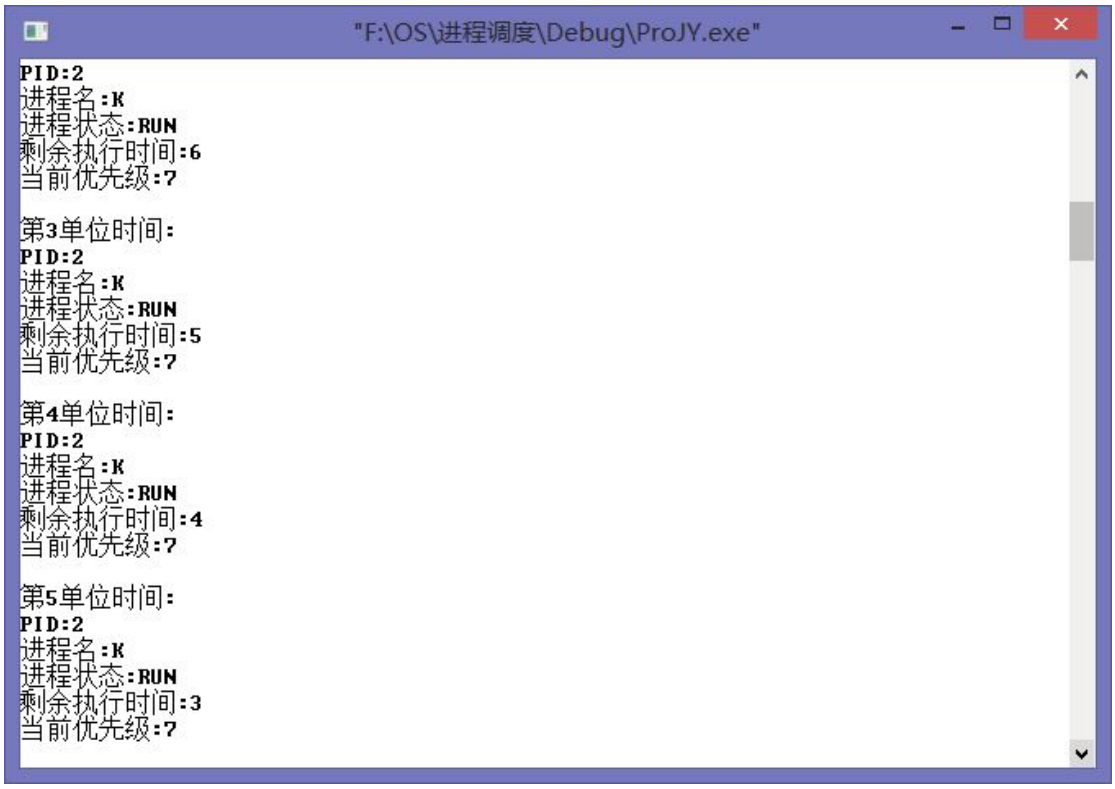
```
            TAIL->next=NULL;
        }
        else
        {
            P->next=PRun->next;
            PRun->next=NULL;
        }
        free(PRun);
        PRONUM--;
    }

    /*进程尚未运行完*/
    else

    {
        PRun->statue=WAIT;//状态设为阻塞
        PRun->waitTime=0;//等待时间清零
        PRun->currentPriority=PRun->initialPriority;//优先级设为初始优先级
    }

  }
}




void main()
{
  printAuthor();
  int inputOfUser;

  HEAD=(struct PCB*) malloc(LEN);//开辟一个带有头结点链表单元
  HEAD->next=NULL;
  HEAD->currentPriority=-99999;
  TAIL=HEAD;
```

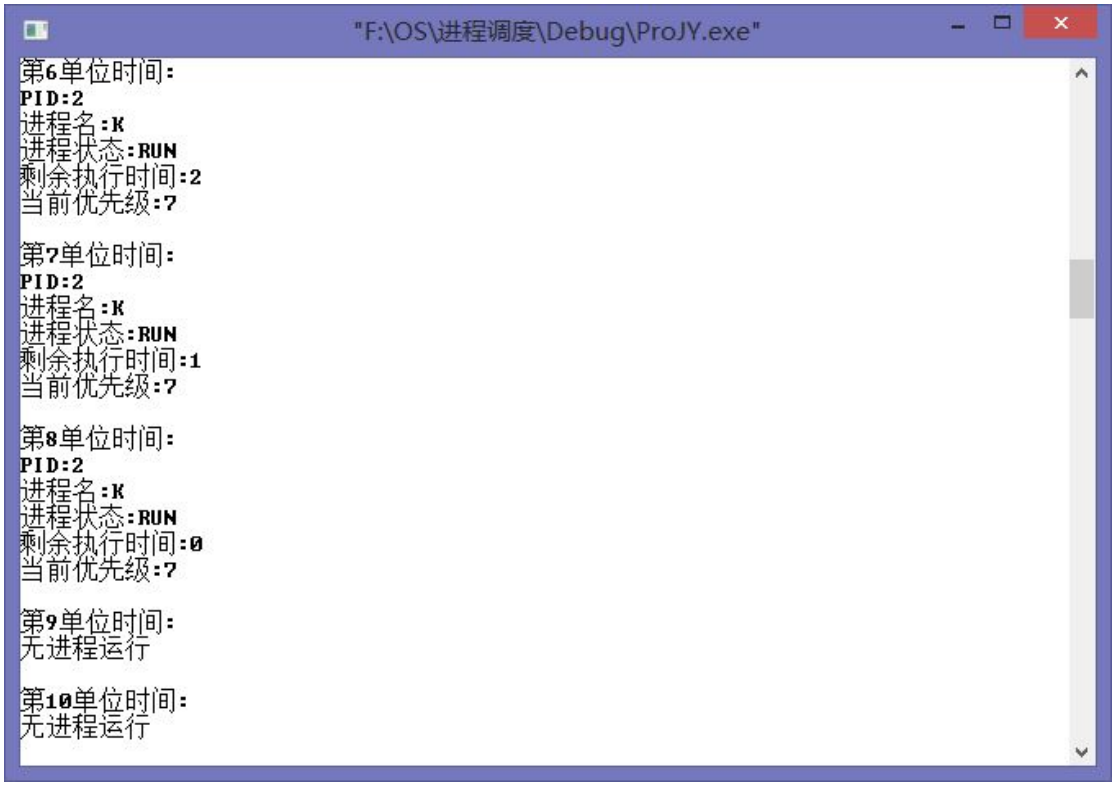
```
    while (1)
    {
        printf("\n 主菜单: \n");
        printf("1:创建进程\n");
        printf("2:显示当前运行进程和就绪队列中进程信息\n");
        printf("3:优先级调度\n");
        printf("4:退出系统\n");
        printf("\n 请选择:");


        scanf("%d",&inputOfUser);
        switch (inputOfUser)
        {
            case 1: create();     //创建进程
                  break;
            case 2: current();     //显示已创建的进程
                  break;
            case 3: priModel();    //优先级调度进程方式
                  break;
            case 4: exit(0);
        }
    }
}
```

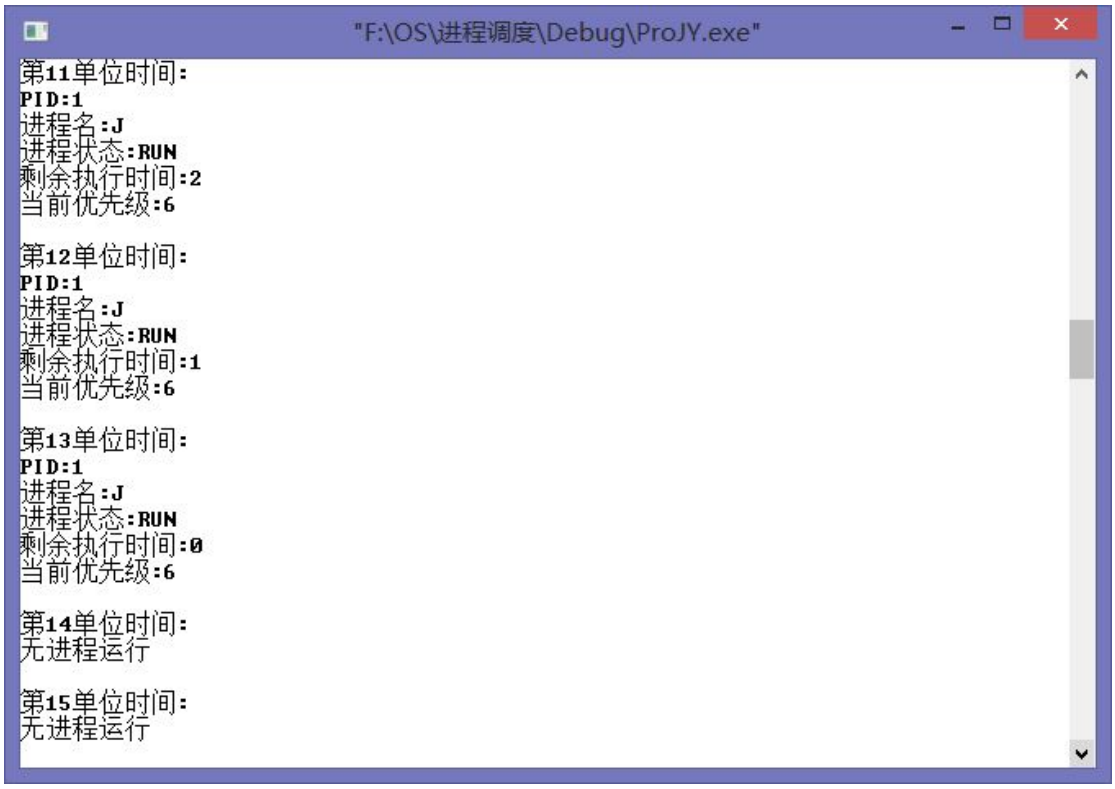
2.运行结果


"F:\OS\进程调度\Debug\ProJY.exe"
这是金洋<Sno:20131910023>的进程调度模拟实验.
主菜单:
1:创建进程
2:显示当前运行进程和就绪队列中进程信息
3:优先级调度
4:退出系统
请选择:1
请输入你想建立的进程个数:2
请输入第1个进程信息:
请输入进程名:J
请输入进程所需执行时间:3
请输入进程优先级:6
请输入第2个进程信息:
请输入进程名:K
请输入进程所需执行时间:8
请输入进程优先级:7
主菜单:
1:创建进程
2:显示当前运行进程和就绪队列中进程信息
3:优先级调度
4:退出系统


"F:\OS\进程调度\Debug\ProJY.exe"
请选择:2
PID:1
进程名:J
进程状态:UNREADY
PID:2
进程名:K
进程状态:UNREADY
主菜单:
1:创建进程
2:显示当前运行进程和就绪队列中进程信息
3:优先级调度
4:退出系统
请选择:3
第1单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:7
当前优先级:7
第2单位时间:


"F:\OS\进程调度\Debug\ProJY.exe"
PID:2
进程名:K
进程状态:RUN
剩余执行时间:6
当前优先级:7
第3单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:5
当前优先级:7
第4单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:4
当前优先级:7
第5单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:3
当前优先级:7


"F:\OS\进程调度\Debug\ProJY.exe"
第6单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:2
当前优先级:7
第7单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:1
当前优先级:7
第8单位时间:
PID:2
进程名:K
进程状态:RUN
剩余执行时间:0
当前优先级:7
第9单位时间:
无进程运行
第10单位时间:
无进程运行



六、总结

1.进程调度算法可以分为:

（1）先来先服务

（2）轮转调度

(3)分级轮转法优先级法

①非抢占方式（优先占有法）
②抢占方式（优先剥夺法）

2.此处使用链式队列来模拟进程调度，还可以用一个调度线程来模拟操作系统的调度进程，用一个线程来模拟一个进程。于是，对应于每一个被模拟的进程，都有相应的线程来模拟，并在调度线程的调度下，实现时间片轮转的调度算法。

3.线程是进程中执行运算的最小单位，是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。

4.通过课堂学习和本实验，学习到了进程调度的一些算法。我们应该把它们结合起来，综合考虑进程的等待时间和进程执行时间，比如使用高响应比优先算法使得进程的调度更加合理高效。当然，目前公认的最优的算法是多级反馈队列调度算法，它是时间片轮转算法和优先级调度算法的综合和发展，通过动态调整进程优先级和时间片大小，不必事先估计进程的执行时间，多级反馈队列可兼顾多方面的系统目标。

七、参考文献

[1]汤小丹，梁红兵，哲凤屏，汤子瀛.计算机操作系统[M](第三版). 西安：西安电子科技大学出版社，2007 年 5 月；
[2]谭浩强著.c 程序设计[M]（第三版）.北京：清华大学出版社.2005.7；

八、教师评语