进程调度算法模拟

一、实验分析：

在操作系统中，由于进程总数多于处理机，它们必然竞争处理机。进程调度的功能就是按一定策略、动态地把处理机分配给处于就绪队列中的某一进程并使之执行。根据不同的系统设计目标，可有多种选择某一进程的策略。例如系统开销较少的静态优先数法，适合于分时系统的轮轮法以及 UNIX 采用的动态优先数反馈法等。本实验是采用优先数法进程调度算法来模拟演示进程调度，编程语言为 C 语言。

二、实验设计：

（1）设计一个有 N 个进程共行的进程调度程序。每个进程由一个进程控制块

PCB 表示。进程控制块包括以下信息：进程名，进程优先数，进程需要运行的时间，占用 CPU 的时间以及进程的状态等。

（2）本调度程序用优先数调度算法。

（3）编写程序并调试运行。

三、算法说明：

本程序采用优先数算法对 N 个进程进行调度。每个进程处于 R，就绪 W 和完成 F 三种状态之一，并假定起始状态就是就绪状态 W。

为了方便处理，有如下假设：每个进程的优先数=50-进程完成总耗时。

```
/*进程调度-----优先级算法源程序*/

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

typedef struct node /*创建 PCB*/

{     char name[10];   /*进程标识*/
```

```
    int prio;     /*进程优先数*/
   int cputime; /*进程占用CPU时间*/
    int needtime; /*进程完成所需时间*/
    int count;   /*计数器*/
    char state; /*进程的状态*/
    struct node *next; /*链指针*/

}PCB;
PCB *finish,*ready,*tail,*run;
int N;

firstin()                    /*创建就绪队列对头指针*/
{
    run=ready;
    run->state='R';
    ready=ready->next;
}


void prt(char algo)   /*演示进程调度*/
{
    PCB *p;
    printf("    NAME   CPUTIME    NEEDTIME    PRIORITY    STATUS\n");
    if(run!=NULL)
```

```
    printf("   %-10s%-10d%-10d%-10d %c\n",run->name,
            run->cputime,run->needtime,run->prio,run->state);
    p=ready;
    while(p!=NULL)
    {   printf("   %-10s%-10d%-10d%-10d %c\n",p->name,
          p->cputime,p->needtime,p->prio,p->state);
          p=p->next;
    }
    p=finish;
    while(p!=NULL)
    {   printf("   %-10s%-10d%-10d%-10d %c\n",p->name,
          p->cputime,p->needtime,p->prio,p->state);
          p=p->next;
    }
    getch();
}

insert(PCB *q)
{
    PCB *p1,*s,*r;
    int b;
    s=q;
    p1=ready;
    r=p1;
```

```
    b=1;
    while((p1!=NULL)&&b)
          if(p1->prio>=s->prio)
          {
        r=p1;
        p1=p1->next;
          }
          else
        b=0;
    if(r!=p1)
    {
          r->next=s;
          s->next=p1;
    }
    else
    {
          s->next=p1;
          ready=s;
    }
}

void create(char alg)     /*创建各个进程*/
{
    PCB *p;
```

```
int i,time;
char na[10];
ready=NULL;
finish=NULL;
run=NULL;
  for(i=1;i<=N;i++)
{
      p=malloc(sizeof(PCB));
      printf("Enter NAME of process:\n");
      scanf("%s",na);
      printf("Enter TIME of process(less than 50):\n");
      scanf("%d",&time);
      strcpy(p->name,na);
      p->cputime=0;
      p->needtime=time;
      p->state='w';
      p->prio=50-time;   /*假设优先级与耗时之和为50*/
      if(ready!=NULL)
    insert(p);
      else
      {
    p->next=ready;
    ready=p;
      }
```

```
    }
    clrscr();
    printf("                    DISPLAY OF THE PROGRESS:\n");
    printf("***********************************************\n");
    prt(alg);
    run=ready;
    ready=ready->next;
    run->state='R';
}

priority(char alg)     /*优先级算法调度*/
{
    while(run!=NULL&&run->prio>=0)
    {
          run->cputime=run->cputime+1;
          run->needtime=run->needtime-1;
          run->prio=run->prio-3;
          if(run->needtime==0)
          {
        run->next=finish;
        finish=run;
        run->state='F';
        run=NULL;
        if(ready!=NULL)
```

```
            firstin();
          }
          else
        if((ready!=NULL)&&(run->prio<ready->prio))
        {
            run->state='W';
            insert(run);
            firstin();
        }
          prt(alg);
    }
}

main()
{   char algo;
    clrscr();
loop:       printf("Enter THE TOTAL NUMBER of PCB(less than 10 is better):\n");
    scanf("%d",&N);
    if(N>10)
    {printf("it's too big,and select a small number.\n");
    goto loop;}
    create(algo);
    priority(algo);
}
```