#include<iostream>

#include<string>

using namespace std;

#include<windows.h>

typedef struct node

{

char name[10];//进程的名字

int prio; //进程的优先级

int round; //分配CPU的时间片

int cputime; //CPU的执行时间

int needtime; //进程执行所需要的时间

int count; //记录执行的次数

char state; // 进程的状态，W——就绪态，R——执行态，F——完成态

struct node \*next; //链表指针

}PCB;

PCB \*ready,\*run,\*finish;//定义三个队列，就绪队列，执行队列和完成队列

PCB \*tail;

int N;

void firstin(void); //取得第一个就绪队列节点

void print1(char a); //输出队列信息

void print2(char chose,PCB \*p); //输出队列信息

void print(char chose);

void insert\_prio(PCB \*q);//\*创建优先级队列，规定优先数越小，优先级越高\*

void prior\_init(char chose);

void priority(char chose);

void insert\_rr(PCB \*q);

void roundrun\_init(char chose);

void roundrun(char chose);//时间片轮转调度

void main()//主函数

{

char chose=' ';

while((chose!='q')&&(chose!='Q'))

{

fflush(stdin);

cout<<"选择进程优先级算法请输入P，选择循环轮转算法请输入R，退出请输入Q\n";

cout<<"请输入你的选择：";

cin>>chose;

if((chose!='q')&&(chose!='Q'))

{

system("cls");

if((chose=='P')||(chose=='p'))

{

prior\_init(chose);

priority(chose);

system("cls");

}

else if((chose=='r')||(chose=='R'))

{

roundrun\_init(chose);

roundrun(chose);

system("cls");

}

}

}

cout<<"谢谢使用！\n";

}

void firstin(void) //取得第一个就绪队列节点

{

run=ready;

if(ready!=NULL)

{

ready=ready->next;

run->state='R';

run->next=NULL;

}

else

{

run=NULL;

}

}

void print1(char a) //输出队列信息

{

if(toupper(a)=='P')

{

cout<<"name cputime needtime priority state \n";

}

else

{

cout<<"name cputime needtime count round state \n";

}

}

void print2(char chose,PCB \*p)//输出队列信息

{

if(toupper(chose)=='P')

{

cout<<p->name<<""<<p->cputime<<""<<p->needtime<<""<<p->prio<<""<<p->state<<endl;

}

else

{

cout<<p->name<<""<<p->cputime<<""<<p->needtime<<""<<p->count<<""<<p->round<<""<<p->state<<endl;

}

}

void print(char chose) //输出所有信息

{

PCB \*p;

print1(chose);

if(run!=NULL)

{

print2(chose,run);

}

p=ready;

while(p!=NULL)

{

print2(chose,p);

p=p->next;

}

p=finish;

while(p!=NULL)

{

print2(chose,p);

p=p->next;

}

}

void insert\_prio(PCB \*s)//创建优先级队列，规定优先数越小，优先级越低

{

PCB \*p,\*r;

p=ready;

r=p;

if(s->prio>ready->prio)//如果队列为空，则为第一个元素

{

s->next=ready;

ready=s;

}

else//查到合适的位置进行插入

{

while(p)

{

if(p->prio>=s->prio)//比第一个还要大，则插入到队头

{

r=p;

p=p->next;

}

else

break;

} s->next=p;

r->next=s;

}

}

void prior\_init(char chose)//\*优先级输入函数\*/

{

PCB \*p;

int i,time;

char na[10];

ready=NULL;

finish=NULL;

run=NULL;

cout<<"输入进程的个数 N:\n";

cin>>N;

for(i=0;i<N;i++)

{

p=(PCB\*)malloc(sizeof(PCB));

cout<<"输入第"<<i+1<<"个进程名\n";

cin>>na;

cout<<"完成进程需要的时间片数\n";

cin>>time;

strcpy(p->name,na);

p->cputime=0;//CPU的执行时间

p->needtime=time;//进程执行所需要的时间

p->state='W';

p->prio=50-time; //设置其优先级，需要的时间越多，优先级越低

if(ready==NULL)

{

ready=p;

ready->next=NULL;

}

else

{

insert\_prio(p);

}

cout<<"当前就绪队列的进程的信息\n";

print(chose);

}

cout<<i+1<<"个进程已按优先级从高到低进到就绪队列中\n"<<N;

cout<<"按回车键开始模拟优先级算法.....\n";

fflush(stdin);

getchar();

firstin();

}

void priority(char chose)//按照优先级调度，每次执行一个时间片

{

int i=1;

while(run!=NULL)//当就绪队列不为空时，则调度进程如执行队列执行

{

run->cputime++;//CPU时间片加一

run->needtime--;//进程执行完成的剩余时间减一

run->prio-=3;//进程的优先级,优先级减去三

if(run->needtime==0)//进程执行完成的剩余时间减一

{

run->next=finish;

finish=run;

run->state='F';

run->prio=0;

run=NULL;

firstin();

}

else//将进程状态置为W，入就绪队列

{

if((ready!=NULL)&&(run->prio<ready->prio))

{

run->state='W';

insert\_prio(run);

run=NULL;

firstin();//继续取就绪队列队头进程进入执行队列

}

}

print(chose);

}

getchar();

}

void insert\_rr(PCB \*q)

{

tail->next=q;

tail=q;

q->next=NULL;

}

void roundrun\_init(char chose)//时间片输入函数\*/

{

PCB \*p;

int i,time;

char na[10];

ready=NULL;

finish=NULL;

run=NULL;

cout<<"\t\t循环轮转算法模拟全过程\n\n";

cout<<"输入进程 的个数 N:\n";

cin>>N;

for(i=0;i<N;i++)

{

p=(PCB\*)malloc(sizeof(PCB));

cout<<"输入第"<<i+1<<"个进程名\n";

cin>>na;

cout<<"完成进程需要的时间片数\n";

cin>>time;

strcpy(p->name,na);

p->cputime=0;

p->needtime=time;

p->count=0;

p->state='W';

p->round=2;

if(ready!=NULL)

{

insert\_rr(p);

}

else

{

p->next=ready;

ready=p;

tail=p;

}

cout<<"当前就绪队列的进程的信息\n";

print(chose);

}

cout<<i+1<<"个进程已按FIFO进到就绪队列中\n"<<N;

cout<<"按回车键开始模循环轮转算法.....\n";

fflush(stdin);

getchar();

run=ready;

ready=ready->next;

run->state='R';

}

void roundrun(char chose)//时间片轮转调度算法

{

int i=1;

while(run!=NULL)

{

run->cputime++;

run->needtime--;

run->count++;

if(run->needtime==0)//进程执行完毕

{

run->next=finish;

finish=run;

run->state='F';

run->prio=0;

run=NULL;

if(ready!=NULL)

{

firstin();

}

}

else

{

if(run->count==run->round)//时间片用完

{

run->count=0; //计数器清零，为下次做准备

if(ready!=NULL)

{

run->state='W';

insert\_rr(run);

firstin();

}

}

}

print(chose);

}

getchar();

}