//READY——就需队列头指针

//TAIL——就需队列尾指针

//FINISH——完成队列头指针

/\*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块、

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。\*/

#include<iostream>

#include<stdlib.h>

#include<string>

using namespace std;

typedef struct node

{

char name[10];//进程的名字

int prio;//进程的优先级

int round;//分配给CPU的时间片

int cputime;//CPU执行时间

int needtime;//进程执行所需要的时间

char state;//进程的状态，W就绪，R执行，F完成

int count;//记录执行的次数

struct node \*next;

}PCB;

PCB \*ready = NULL, \*run = NULL, \*finish = NULL;//定义三个队列

int number;//输入数字

void getfirst();//从就绪队列取得第一个节点

void output();//输出队列信息

void insertPrio(PCB \*in);//创建优先级队列，优先数越小，优先级越高，然后插入进程

void insertReady(PCB \*in);//将进程插入到就绪队列

void insertFinish(PCB \*in);//将进程插入到完成队列

void PrioCreate();//优先级输入函数

void Priority();//按照优先级调度

void TimeCreate();//时间片输入函数

void RoundRun();//按照时间片转轮调度

int main()

{

char choose;

cout << "输入用户所想要创建的进程数目：" << endl;

cin >> number;

cout << "输入用户所选用的进程的调度算法（P/R）：" << endl;

cin >> choose;

switch (choose)

{

case 'P'://用户输入优先级调度大写的字母

case 'p'://用户输入优先级调度小写的字母

PrioCreate();//优先级输入函数

Priority();//按照优先级调度

break;

case 'R'://用户输入时间片转轮调度大写的字母

case 'r'://用户输入时间片转轮调度大写的字母

TimeCreate();//时间片输入函数

RoundRun();//按照时间片转轮调度

break;

default:break;//输入错误，直接停止

}

output();//输出队列信息

return 0;

}

void getfirst()//从就绪队列取得第一个节点

{

run = ready;

if (ready != NULL)

{

run->state = 'R';//开始执行

ready = ready->next;//等待的向后移

run->next = NULL;//一次执行一个故执行的下一个为空

}

}

void output()//输出队列信息

{

PCB \*q;

q = ready;//当进程在就绪队列时

cout << "进程名" << " " << "优先级" << " " << "轮数" << " " << "CPU时间" << " " << "需要时间" << " " << "进程状态" << " " << "计数器" << endl;

while (q != NULL)

{

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

q = q->next;

}

q = finish;//当进程已经完成

while (q != NULL)

{

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

q = q->next;

}

q = run;//当进程正在执行时

while (q != NULL)

{

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

q = q->next;

}

}

void insertPrio(PCB \*in)//创建优先级队列

{

PCB \*f, \*n;//f->first 第一个,n->next

f = n = ready;

if (ready == NULL)//队列为空。则为第一个元素

{

in->next = ready;

ready = in;

}

else //队列不为空，则找到合适的位置插入

{

if (in->prio >= f->prio)//优先级比第一个元素的还大插入到队头

{

in->next = ready;

ready = in;

}

else//比第一个优先级小

{

while (f->next != NULL)//如果还有别的元素往下找

{

n = f;

f = f->next;

}

if (f->next == NULL)//就一个元素

{

in->next = f->next;

f->next = in;//插入到队尾

}

else//还有元素，插入到队中

{

n = in;

in->next = f;

}

}

}

}

void insertReady(PCB \*in)//将进程插入到就绪队列的尾部

{

PCB \*f;

f = ready;

if (ready == NULL)//就绪队列为空

{

in->next = ready;//输入的进入就绪队列

ready = in;

}

else

{

while (f->next != NULL)

{

f = f->next;//一直向后移动

}

in->next = f->next;

f->next = in;

}

}

void insertFinish(PCB \*in)//将进程插入到完成队列尾部

{

PCB \*f;

f = finish;

if (finish == NULL)//完成队列为空

{

in->next = finish;

finish = in;

}

else

{

while (f->next != NULL)

{

f = f->next;

}

in->next = f->next;

f->next = in;

}

}

void PrioCreate()//优先级输入函数

{

PCB \*temp; int i;

cout << "输入进程名和进程完成所需要的时间：" << endl;

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

{

if ((temp = (PCB\*)malloc(sizeof(PCB))) == NULL)

{

perror("malloc");

exit(1);

}

cin >> temp->name;

getchar();

cin >> temp->needtime;

temp->cputime = 0;

temp->state = 'W';

temp->prio = 100 - temp->needtime;

temp->round = 0;

temp->count = 0;

insertPrio(temp);//将其插入到就绪队列中

}

}

void Priority()//按照优先级调度,每次执行一个时间片

{

int flag = 1;

getfirst();

while (run != NULL)//就绪队列不为空

{

output();

while (flag)

{

run->prio -= 5;//优先级减少5级

run->cputime++;//时间片加1

run->needtime--;//进程需要的时间减1

if (run->needtime == 0)//已经完成

{

run->state = 'F';

run->count++;//进程执行次数加1

insertFinish(run);//将进程调到完成队列

flag = 0;

}

else//否则还是在就绪队列

{

run->state = 'W';

run->count++;//进程执行次数加1

insertReady(run);//进程带入到就绪队列

flag = 0;

}

}

flag = 1;

getfirst();//继续取就绪队列的第一个进程进入执行队列

}

}

void TimeCreate()//时间片输入函数

{

PCB \*temp; int i;

cout << "输入进程名和进程完成所需要的时间：" << endl;

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

{

if ((temp = (PCB\*)malloc(sizeof(PCB))) == NULL)

{

//cout<<"分配内存空间失败！"<<endl;

perror("malloc");

exit(1);

}

cin >> temp->name;

getchar();

cin >> temp->needtime;

temp->cputime = 0;

temp->state = 'W';

temp->prio = 0;

temp->round = 2;//每个进程所分配的时间片为2

temp->count = 0;

insertReady(temp);//调入到就绪队列

}

}

void RoundRun()//按照时间片转轮调度

{

int flag = 1;

getfirst();

while (run != NULL)//就绪队列不为空

{

output();

while (flag)

{

run->count++;//进程执行次数加1

run->cputime++;//时间片加1

run->needtime--;//进程需要的时间减1

if (run->needtime == 0)//已经完成

{

run->state = 'F';

insertFinish(run);//将进程调到完成队列

flag = 0;

}

else if (run->count == run->round)//时间片用完

{

run->state = 'W';

run->count = 0;//计数器清0

insertReady(run);//进程带入到就绪队列

flag = 0;

}

}

flag = 1;

getfirst();//继续取就绪队列的第一个进程进入执行队列

}

}