操作系统实验-进程调度　20141121204 3邓依伊

- 源程序

```
//
//  main.c
//  jincheng
//
//  Created by Yiyi Deng on 10/24/16.
//  Copyright © 2016 邓依伊. All rights reserved.
//
//  编写并调试一个模拟的进程调度程序，采用“最高优先数优先”调度算法对多个并发进程进行调度。
//  要求：采用动态优先数。即进程的优先数在创建进程时可以给定一个初始值，并且可以按一定原则修改优先数：在进程获得一次CPU后就将其优先数减少1。

#include <stdio.h>
#include <stdlib.h>

#define getpch(type) (type*)malloc(sizeof(type))


struct pcb { /* 定义进程控制块PCB */
    char state;
    int id;
    int  prio;
    int  needtime;
    int  runtime;
    struct pcb * next;
}*ready=NULL, *p; // 就绪队列的头进程，当前运行进程

typedef struct pcb PCB;

void insert();

void input() /* 建立进程控制块函数 */
{
    int i,num;

    printf("\n number of processes:");
    scanf("%d",&num);
    for(i=0;i<num;i++)
    {
        printf("\n No.%d:",i);
        p=getpch(PCB);
```

```
        p->id = i;
        printf("\n priority:");
        scanf("%d",&p->prio);
        printf(" required time:");
        scanf("%d",&p->needtime);
        p->runtime=0;
        p->state='W';
        p->next=NULL;
        insert();
    }
}

// 优先数算法的插入
void insert() // 创建一个进程后&一个进程运行1个单位时间后，要将这个进程插入就绪队列
{
    PCB *in;
    in=ready;

    if (ready==NULL) // 此时就绪队列为空
        ready=p;
    else
    {
        if (in->prio<=p->prio) // 就绪队列第一个进程的优先级<=当前进程
        {
            p->next=in; // 直接把当前进程插到队头
            ready=p;
        }
        else
        {
            while (in->next!=NULL)
            {
                if (in->next->prio>=p->prio)
                    in=in->next; // 继续往后找
                else break;
            }
            // 插到in后面（此时这个in，要么是最后一个，此时所有就绪进程优先级都比p大；要么是第一个出现的优先级比p小的进程的前面一个）
            p->next=in->next;
            in->next=p;
        }
    }
}

void disp(PCB * pr) /* 建立进程显示函数 */
{
    printf("\n id\tstate\tpriority\tneedtime\truntime \n");
```

```
    printf(" %d \t\t",pr->id);
    printf(" %c \t\t",pr->state);
    printf(" %d \t\t",pr->prio);
    printf(" %d \t\t",pr->needtime);
    printf(" %d \t\t",pr->runtime);
    printf("\n");
}

void check() /* 建立进程查看函数 */
{
    PCB* pr;
    printf("\n **** Current Process:%d",p->id); /*显示当前运行进程*/
    disp(p);
    pr=ready;
    printf("\n **** Current ready Queue:\n"); /*显示就绪队列状态*/
    while(pr!=NULL)
    {
        disp(pr);
        pr=pr->next;
    }
}

void destroy() /*建立进程撤消函数(进程运行结束,撤消进程)*/
{
    printf("\n Process [%d] has finished.\n",p->id);
    free(p);
}

void running() /* 建立进程就绪函数 */
{
    (p->runtime)++;
    (p->prio)--;
    if(p->runtime==p->needtime)
        destroy();
    else
    {
        p->state='W';
        insert();
    }
}

int main()
{
    int h=0;
    char ch;
    input();
```

```
    ch=getchar();
    while(ready!=NULL)
    {
        h++;
        printf("\n The execute number:%d \n",h);
        p=ready;
        ready=p->next;
        //p->next=NULL;
        p->state='R';
        check();
        running();
        printf("\n Press any key to continue");
        ch=getchar();
    }
    printf("\n\n Finished.\n");
    ch=getchar();
    return 0;
}
```

- 运行结果

输入4个进程，优先级和所需时间如下：

```
number of processes:4

No.0:
priority:5
required time:4

No.1:
priority:7
required time:4

No.2:
priority:3
required time:3

No.3:
priority:4
required time:3
```

首先按优先级初始化就绪队列，再把队首移出，运行。

```
The execute number:1

**** Current Process:1
id state   priority    needtime    runtime
1       R       7       4       0

**** Current ready Queue:

id state   priority    needtime    runtime
0       W       5       4       0

id state   priority    needtime    runtime
3       W       4       3       0

id state   priority    needtime    runtime
2       W       3       3       0

Press any key to continue
```

一个进程每运行一个CPU时间，其运行时间加1，其优先级减少1；同时插入就绪队列；再从就绪队列中摘出队首来运行。

对于每一个CPU时间的运行，输出一次运行进程和就绪队列的情况，如下图所示。运行的进程状态标注为‘R’，就绪进程状态标注为‘W’。

```
The execute number:2

**** Current Process:1
id state   priority    needtime    runtime
1       R       6       4       1

**** Current ready Queue:

id state   priority    needtime    runtime
0       W       5       4       0

id state   priority    needtime    runtime
3       W       4       3       0

id state   priority    needtime    runtime
2       W       3       3       0

Press any key to continue
```

```
The execute number:3

**** Current Process:1
id state   priority    needtime    runtime
1       R       5       4       2

**** Current ready Queue:

id state   priority    needtime    runtime
0       W       5       4       0

id state   priority    needtime    runtime
3       W       4       3       0

id state   priority    needtime    runtime
2       W       3       3       0

Press any key to continue
```

直到进程[1]的优先级降到5， 再运行一次后降至4，插入就绪队列，之后进程[0]开始运行。

```
The execute number:4

**** Current Process:0
id state   priority    needtime    runtime
0       R       5       4       0

**** Current ready Queue:

id state   priority    needtime    runtime
3       W       4       3       0

id state   priority    needtime    runtime
1       W       4       4       3

id state   priority    needtime    runtime
2       W       3       3       0

Press any key to continue
```

如此反复下去，直到当前运行进程运行后其运行时间runtime与所需时间needtime相等，此进程运行完毕。

```
The execute number:7

**** Current Process:1
id state   priority    needtime    runtime
1       R       4       4       3

**** Current ready Queue:

id state   priority    needtime    runtime
2       W       3       3       0

id state   priority    needtime    runtime
0       W       3       4       2

id state   priority    needtime    runtime
3       W       3       3       1

Process [1] has finished.

Press any key to continue
```

直到最后一个进程运行完毕：

```
The execute number:14

**** Current Process:2
id state   priority    needtime    runtime
2       R       1       3       2

**** Current ready Queue:

Process [2] has finished.

Press any key to continue


Finished.
```