

第三章 实验二

模拟进程优先级调度

1. 按照优先级出列

- 动态地输入进程，按照输入次序建立队列
- 按照优先级的次序将队列中的进程逐个出队

例如：

```
新建的进程控制表为：
key priority message
1 20 process1
2 50 process2
3 30 process3
4 10 process4
5 40 process5
0 0 0
```

输入

```
The table is:
1,20, process1
2,50, process2
3,30, process3
4,10, process4
5,40, process5

第1次出队的进程为：
key=2,priority=50,message= process2

第2次出队的进程为：
key=5,priority=40,message= process5

第3次出队的进程为：
key=3,priority=30,message= process3

第4次出队的进程为：
key=1,priority=20,message= process1

第5次出队的进程为：
key=4,priority=10,message= process4
Press any key to continue
```

输出结果

- 使用 Microsoft Visual Studio C++ 6.0 或 CodeBlocks 编程：程序 3_2_outPRIOR.cpp。完善如下程序代码：

```
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#define NULL 0
typedef struct table
{
    int key; /*进程 ID 号*/
    int priority; /*优先数值*/
    char message[10]; /*进程说明信息*/
    struct table *next;
}node;
node *creat(void) /*定义函数，输入 ID 号和优先级，按照输入次序建立进程链表*/
{
    填补程序
}
void print (node *head) /*输出链表*/
{
    填补程序
}
node *processdo(node *head) /*模拟当前最大优先数进程出队的过程*/
{
    填补程序
}
void main()
{
    填补程序：模拟创建进程控制块队列，并按照优先级逐个出队过程
}
```

2. 按照优先级入列

- 动态地输入进程，按照优先级建立优先级队列
- 按照前后次序将队列中的进程逐个出队

例如：

```
新建的进程控制表为：
key priority message
1 20 process1
2 50 process2
3 30 process3
4 10 process4
5 40 process5
0 0 0
```

输入

```
The table is:
2,50, process2
5,40, process5
3,30, process3
1,20, process1
4,10, process4
第1个出队进程为:
key=2,priority=50,message= process2
第2个出队进程为:
key=5,priority=40,message= process5
第3个出队进程为:
key=3,priority=30,message= process3
第4个出队进程为:
key=1,priority=20,message= process1
第5个出队进程为:
key=4,priority=10,message= process4
Press any key to continue
```

输出结果

- 使用 Microsoft Visual Studio C++ 6.0 或 CodeBlocks 编程：程序 3_3_inPRIOR.cpp。完善如下程序代码：

```
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#define NULL 0
typedef struct table
{
    int key;          /*进程 ID 号*/
    int priority;     /*优先数值*/
    char message[10]; /*进程说明信息*/
    struct table *next;
}node;
node *insert(node *head,node *news)
{
    { 填补程序 } /*将新进程插入到已经排序的队列中，对进程表按优先数从大到小排序*/
}
node *creat(void) /*定义函数，调用 insert 函数建立排好序的进程链表*/
{
    { 填补程序 }
}
void print (node *head) /*输出链表*/
{
    { 填补程序 }
}
void rank_out(node *head) /*模拟按优先数大小进程分级出队的过程*/
{
    { 填补程序 }
}
void main()
{
    { 填补程序：模拟进程控制块队列按照优先级逐个出队过程 }
}
```