计算机科学与工程学院实验报告（首页）

|  |  |
| --- | --- |
| **课程名称** | **操作系统** |

1. 实验目的

1、 理解有关进程控制块、进程队列的概念。   
2、 掌握进程优先权调度算法和时间片轮转调度算法的处理逻辑。

1. 实验要求
2. 设计进程控制块PCB的结构，分别适用于优先权调度算法和时间片轮转调度算法。

2、 建立进程就绪队列。

3、 编制两种进程调度算法：优先权调度算法和时间片轮转调度算法。

1. 实验内容

1.时间片轮转调度算法（round robin）

a.该算法采取了非常公平的方式，即让就绪队列上的每个进程每次仅运行一个时间片。如果就绪队列上有N个进程，则每个进程每次大约都可获得1/N的处理机时间。

b时间片的大小对于系统性能有很大的影响。若选择很小的时间片，将有利于短作业，但意味着会频繁地执行进程调度和进程上下文的切换，这无疑会增加系统的开销。反之，若时间片选择得太长，且为使每个进程都能在一个时间片内完成，RR算法便退化为FCFS算法，无法满足短作业和交互式用户的需求。

c.进程的切换时机体现出RR算法的特点。若一个进程在时间片还没结束时就已完成，此时立即激活调度程序，将它从执行队列中删除。若一个进程在时间片结束时还未运行完毕，则调度程序将把它送往就绪队列的末尾，等待下一次执行。用C语言编程模拟调度程序时，将时间片，程序运行时间量化为整数。此时代码

2.优先权调度算法

a.在时间片算法中，无法对进程的紧急程度加以区分。而优先级算法正好可以解决这一问题。

b.进程优先级的确定同样重要。进程优先级可以分为静态优先级和动态优先级。静态优先级是在进程创建初期就被确定的值，此后不再更改。动态优先级指进程在创建时被赋予一个初值，此后其值会所进程的推进或等待时间的增加而改变。

c.用C语言模拟调度程序时，可用run->prio -= 3; /\*优先级减去三,若设为0则优先级不变\*/ 这条语句控制静态动态优先级的切换。

3.程序流程图   
图示

描述已自动生成

4.程序代码及注释

---省略---

1. 说明

本实验应在教师指导下进行，尽量不要擅自进行，否则可能会破坏原有系统。

实验报告应包含以下内容：

1、 优先权调度算法和时间片轮转调度算法原理。

2、 程序流程图。

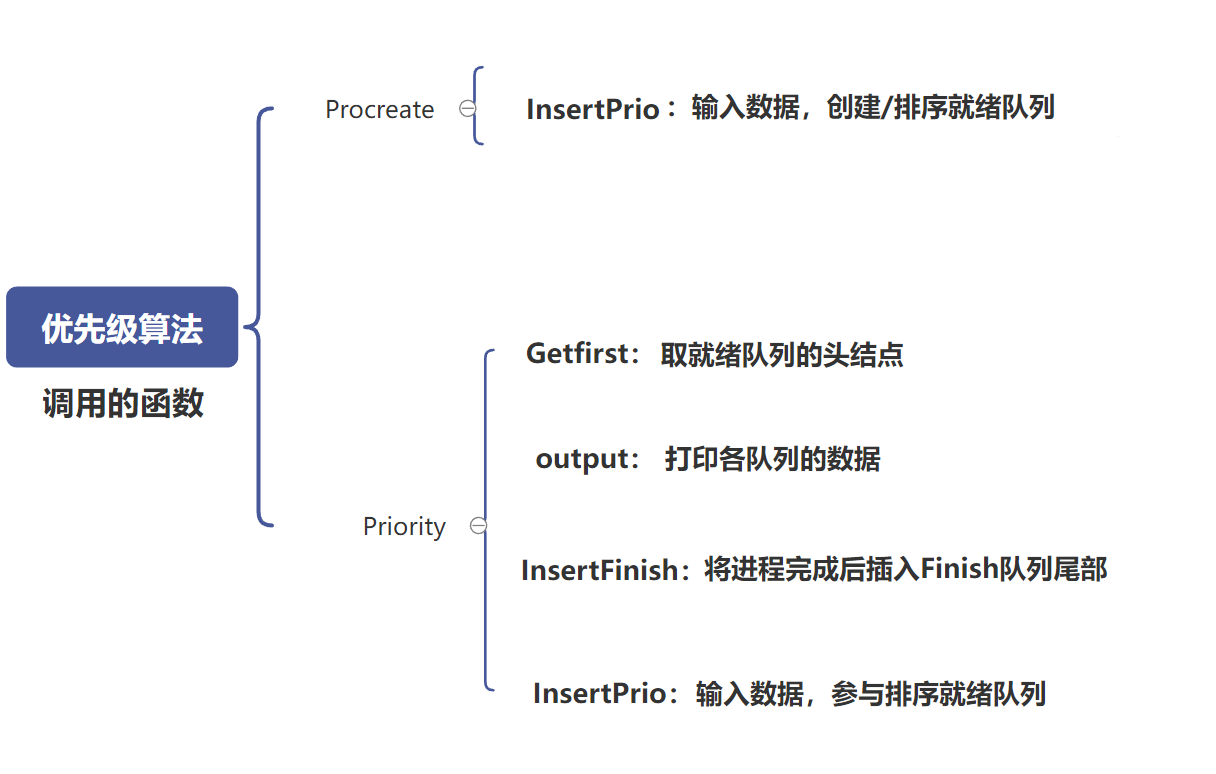
3、 程序及注释。

4、 运行结果以及结论。

1. 运行结果及实验结果、实验体会

#### 1.优先级算法

读了几遍代码后，我在源代码的注释，代码段中发现有错误之处，如下，我列出了优先级算法调用的所有函数。



图表 1 优先级算法的函数调用

在逐句运行了几次代码后，我得出一个结论，这个代码的进程调度机制是以“**所有进程同时到达**”来展开的，因为代码是先把每个进程都排序后再打印出结果，再进行运行。所以这个代码也没有“进程到达时间”这一说。由此我得出：

**（1）InsertPrio函数声明旁的注释错误。**

这一句函数声明的源代码是：

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

注释中强调：**优先数越小，优先级越高。**

但在PrioCreate()函数中，关于优先级数变量的定义如下：

tmp->prio = 50 - tmp->needtime; /\*设置其优先级，需要的时间越多，优先级越低\*/

由此可见，prio优先级值定义是**50-所需时间**。

但在InsertPrio（）函数中，含有以下代码段：

if (in->prio > fst->prio) /\*比第一个还要大(大于等于)，则插入到队头\*/

{

in->next = ready;

ready = in;

}

很明显，算法是让优先级数值最大的排在队头，也就是说**优先级数prio值最高的，优先级最高**。

这与代码开头，InsertPrio函数声明的注释含义是相反的。

根据函数的定义，是注释出现了错误。

**（2）InsertPrio函数代码段中，链表排序错误。**

InsertPrio函数代码片段如下：

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

{ if (ready == NULL){。。。} /\*如果队列为空，则为第一个元素\*/

else /\*查到合适的位置进行插入\*/

{if (in->prio > fst->prio) /\*比第一个还要大(大于等于)，则插入到队头\*/

{。。。}

else

{while (fst->next != NULL) /\*移动指针查找第一个别它小的元素的位置进行插入\*/

{ nxt = fst;

* 出现严重错误！

fst = fst->next; }

if (fst->next == NULL) /\*已经搜索到队尾，则其优先级数最小，将其插入到队尾即可\*/

{ in->next = fst->next;

fst->next = in; }

else /\*插入到队列中\*/

这里代码变成了摆设，根本无法运行。

{ nxt = in;

in->next = fst; }

}

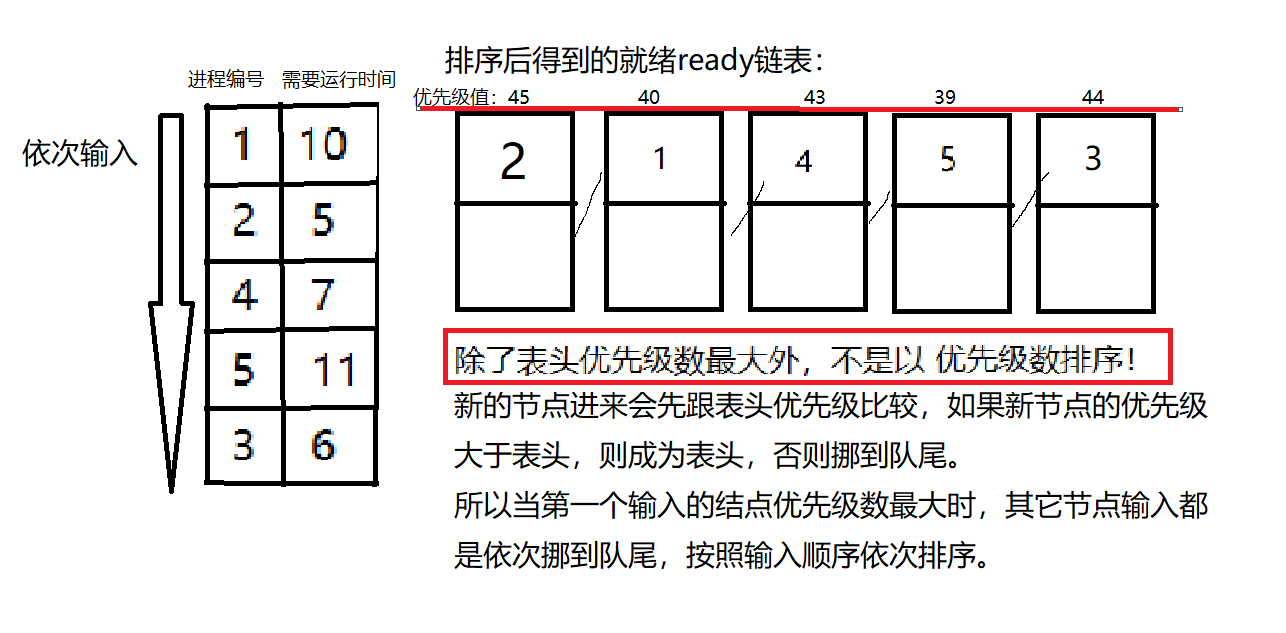
}

}

除了这三个条件语句外，我省略了其它语句，其中**红色框内出现了最主要的错误让优先级算法运算错误：源代码的while循环是一直查找到尾部的**，因为while循环停止的条件是遍历到链表的尾部，**直到尾部才会跳出循环，所以else部分插入队列的代码段完全属于是摆设作用。**

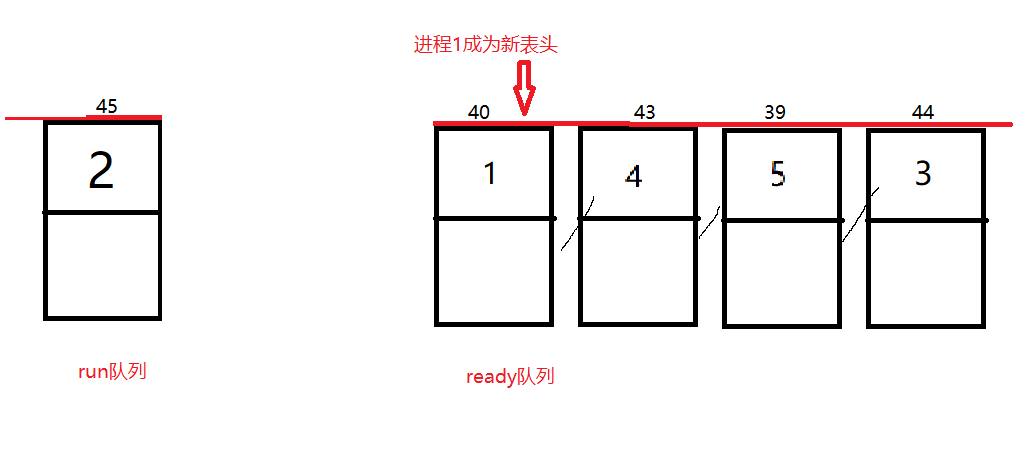
我通过逐句编译理顺了源代码的逻辑：

首先源代码的就绪队列是这样的。



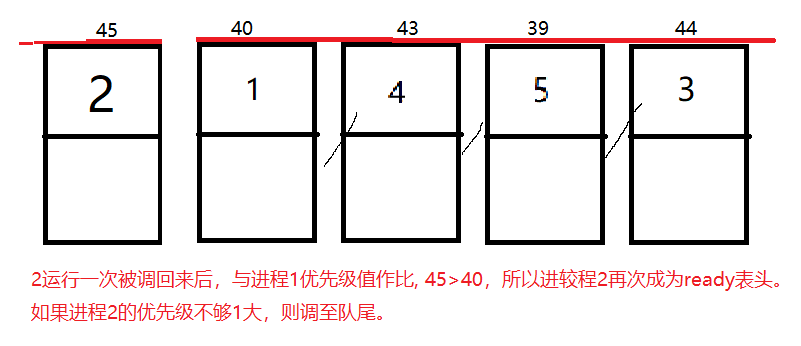
图表 2 源代码就绪队列排列机制

当运行到getfrist函数时，排在ready就绪队列的表头被调到run队列中运行一个时间片。与此同时，原表头的下一个链表节点作为新的ready表头。



图表 3 进程2被调入run队列

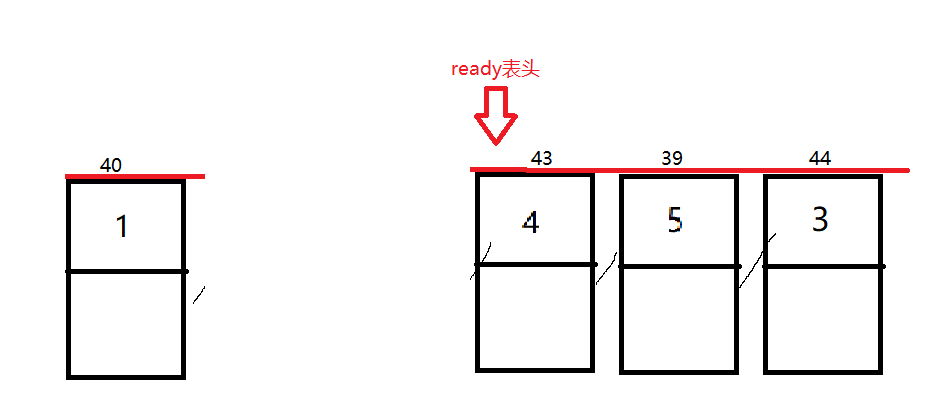
进程2运行一个时间片，所需时间-1，随后再次被调入Insertprio函数进行再次调试。



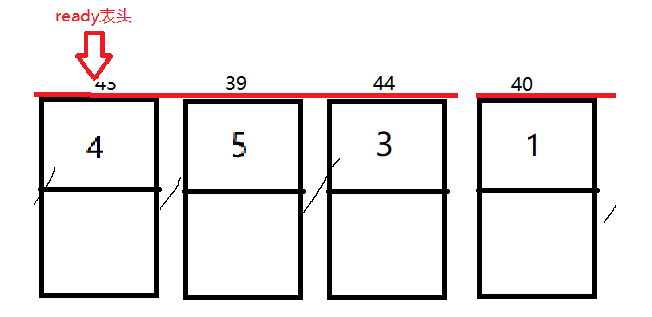
图表 4 进程2再次被调回ready队列，优先级比进程1高，成为表头

以此反复，直到进程2运行完调入finish队列（相当于被杀死）

进程1作为ready表头，被调入run队列继续运行，运行完了一个时间片时间又被调回ready队列开始与表头（进程4）进行优先级对比。

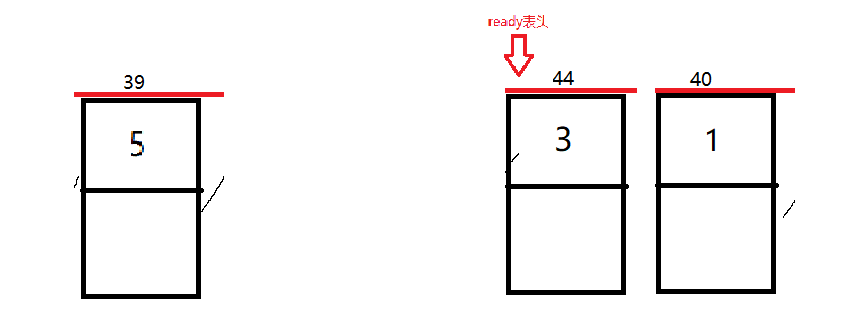


图表 5 进程2运行完，进程1被调走

这时进程1的优先级数小于进程4的，**所以进程1被运行了一次后调回ready队尾。**

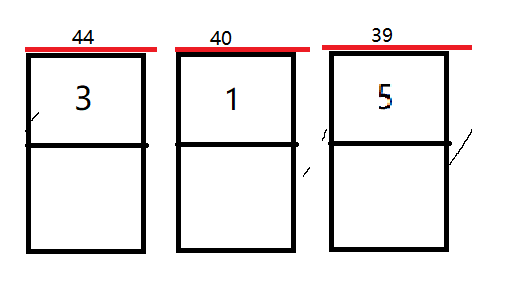
图表 6 进程1被调回，因优先级较表头低被排到队尾

进程4被运行完，轮到进程5被调去run队列执行，进程3变成表头。



图表 7 进程4运行完，进程5被调走

由于进程5的优先级比进程3优先级低，被调回队尾。



图表 8 进程5被调回，因优先级比进程3低被调入队尾

以此反复直到最后的进程5运行完。以上就是这个代码的优先级算法运行逻辑。

**这样ready队列的排序错误会导致错误的如下输出结果，优先级低的反而比优先级高的更快运行完成，更早被杀死。**



图表 9 错误的结果

接下来，我在void InsertPrio(PCB\* in)的函数定义下，把代码进行了修改。



图表 10 改动部分

**附上改动后的完整函数代码**：

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

{

PCB\* fst, \* nxt;

fst = nxt = ready;

if (ready == NULL) /\*如果队列为空，则为第一个元素\*/

{

in->next = ready;

ready = in;

}

else /\*查到合适的位置进行插入\*/

{

if (in->prio > fst->prio) /\*比第一个还要大(大于等于)，则插入到队头\*/

{

in->next = ready;

ready = in;

}

else

{

while (fst!= NULL) {

if (in->prio > fst->prio) /\*插入到队列中\*/

{

nxt->next = in;

in->next = fst;

break;

**改动部分**

}

nxt = fst;

fst = fst->next;

}

if (fst == NULL) /\*已经搜索到队尾，则其优先级数最小，将其插入到队尾即可\*/

{

in->next = fst;

fst = in;

nxt->next = fst;

}

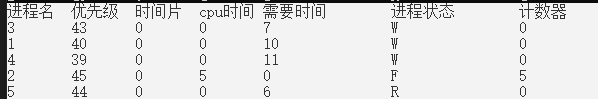
}

}

}

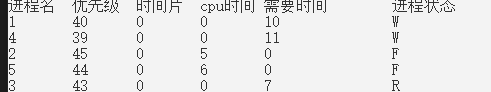
改动后，可以完全按照优先级数大小正确排序，正常运行：

运行结果如下：



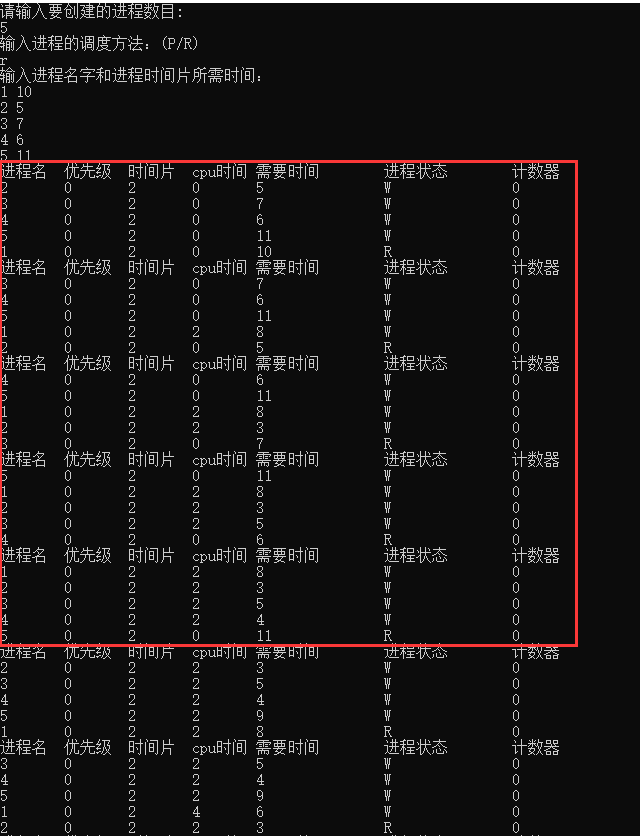
图表 11 优先级最高的完成，第二优先级的开始执行。

W1



图表 12 第二优先级进程运行完成，第三优先级开始执行

## 2.时间片轮转调度算法



从运行结果得知：一个时间片运行两个时间单位，按输入顺序依次运行进行五个进程的轮转。代码没有出现问题。

## 3.实验总结：

这次实验我花了较长时间去读代码，逐句运行，并找到了代码间的错误进行了修正。让代码正确的执行。不知道别的同学有没有发现实验代码里InsertPrio函数的这个错误，我在独自做的过程中是确实有这个错误的，并且在同学的电脑也进行了一次编译，也是一样的输出了错的结果。后来我再经过一段时间的修改编译，把源代码改好了。

也花了时间去做好了这个实验报告，把源代码的算法出错的地方，算法的逻辑希望能更清晰地一一列出来。也把这个错误跟修改过后的代码发给了同学，帮助他们把代码段改正过来。

通过这次的实验，我很清楚了解到优先级算法的原理，时间片算法轮转的原理。学到更多的是有关函数定义的编写，单链表的运用排序，插入增添，还有链表遍历的代码编写。还有大大的提高了我的读代码的能力，学习的专注力，还有推算演算能力、逻辑上表达能力。

总之我认为有心学习的话，认真学操作系统实验认真做的话，一定能在操作系统知识的基础上获得更多。