浙 江 工 商 大 学

计算机与信息工程学院

**《操作系统实验》**

大作业报告



实验名称：多级反馈队列调度

学 号：1712190115

姓 名：张羽丰

班 级：计科1701

指导教师：任午令

日 期：2019/5/26

### 一、实验目的

目的：在实验操作的课程中，在实验3中，我有了不同的想法，因为在进程调度的算法中，老师给的实验步骤用了链表的方法，但是我认为除了链表还有其他的方法，因此我选择了实验二作为我操作系统实验期末大作业的，用不是链表的方法来写多级反馈队列。

最后预期达到的效果：每次编译过后都能输出答案，并且答案与用链表的答案相同。在用C++写完之后能用C语言来代替，并且能正确编译成功，且输出一样的答案。

### 二、实验原理

算法介绍：

1. 进程在进入待调度的队列等待时，首先进入优先级最高的Q1等待。
2. 首先调度优先级高的队列中的进程。若高优先级中队列已没有调度的进程，则调度次优先级队列中的进程。
3. 对于同一个队列中的各个进程，按照FCFS分配时间片调度。
4. 在最后一个队列QN中的各个进程，按照时间片轮转分配时间片调度
5. 在低优先级的队列中的进程在运行时，又有新到达的作业，此时须立即把正在运行的进程放回当前队列的队尾，然后把处理机分给高优先级进程。换而言之，任何时刻，只有当第1~i-1队列全部为空时，才会去执行第i队列的进程。

算法描述：

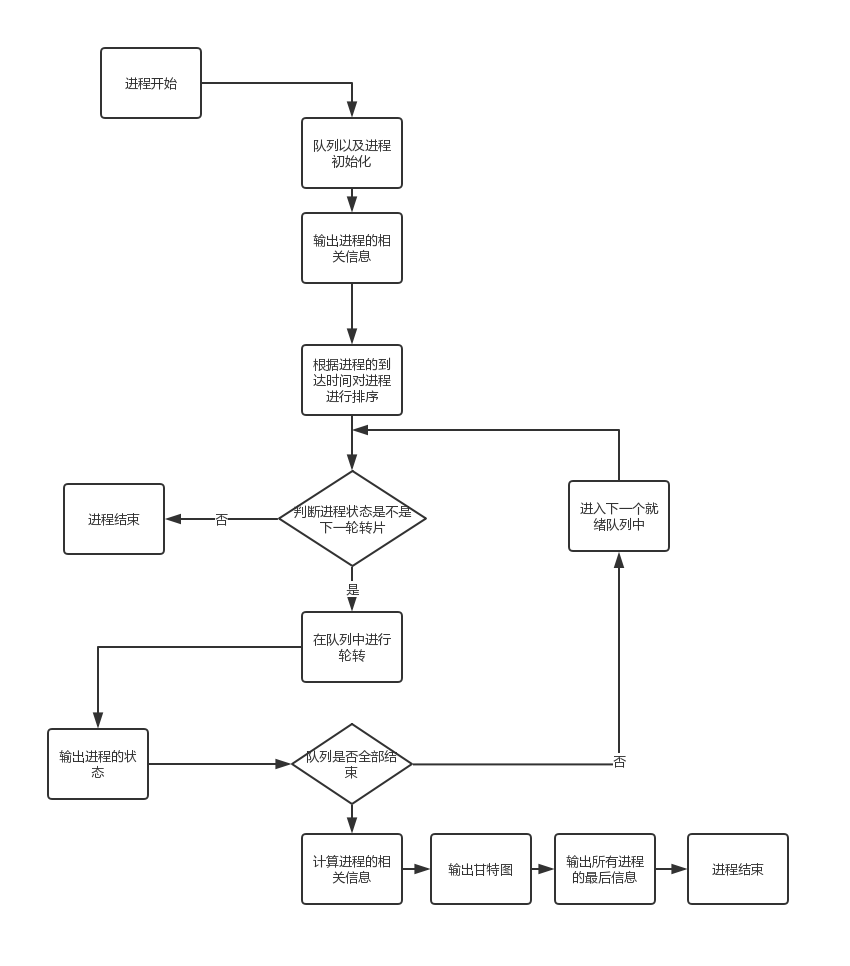
采用三级调度策略，所有进程先按到达顺序进入一级队列，按照时间片为2轮转一次，一个时间片内未完成的进程被依次移入二队列尾部。当一级队列中没有进程时，开始调度二级队列，按照时间片4轮转一次，，一个时间片内未完成的进程被依次移入三队列尾部。当一级队列和二级队列中都没有进程时，开始调度三级队列，三级队列按照FCFS调度。队列之间按抢占式优先级调度，即只有高级队列中无进程时才能执行低级队列中的进程，且一级队列中的新来进程可抢占二级队列或者三级队列中的进程。

### 三、实验设计

#### 1. 实验方案

实验环境描述：

我用的是window10下的一个Ubuntu子系统，版本是：Linux DESKTOP-11O19UK 4.4.0-17134-Microsoft #706-Microsoft Mon Apr 01 18:13:00 PST 2019 x86\_64 x86\_64 x86\_64 GNU/Linux。编译器我用的是Ubuntu下的文本编辑器vim，在vim中写完之后用gcc对其进行编译。



插入框图

图 1. 多级反馈队列调度流程图

#### 2. 数据结构和算法描述

在本实验中，我定义了pcb的结构体用来从存储进程的所有信息，queue来定义轮转的时间片，用mapp[][]存储甘特图。用了冒泡排序对所有进程进行排序，排序的关键词是到达时间。

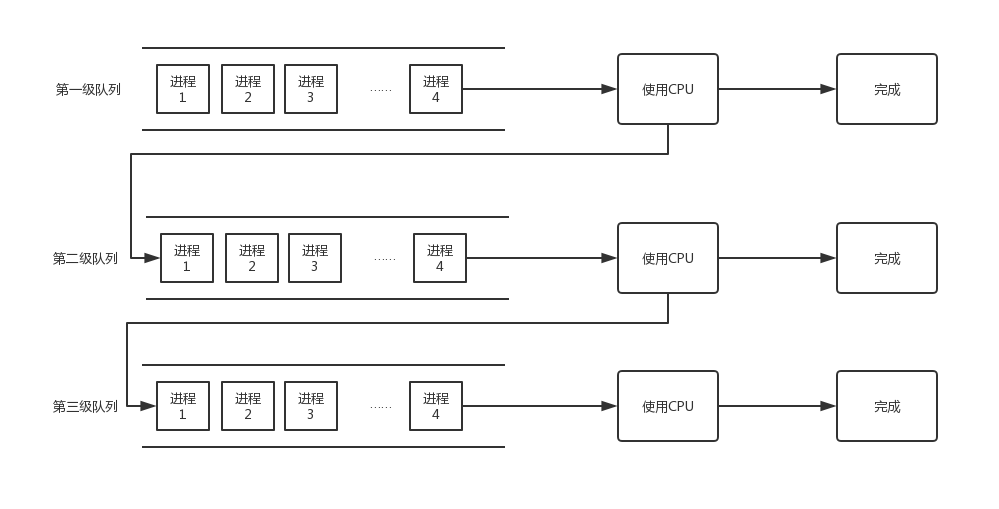
插入流程图图

图 2. 多级优先队列调度算法流程图

#### 3. 实验测试数据

特定实验测试数据（调试用）

表1.特定试验测试数据­-数据

|  |  |  |
| --- | --- | --- |
| 进程名称 | 进程到达时间 | 进程服务时间 |
| P1 | 5 | 6 |
| P2 | 1 | 3 |
| P3 | 4 | 5 |
| P4 | 4 | 5 |
| P5 | 2 | 9 |

任意/随机测试数据（测试用）

表2. 任意/随机测试数据-数据

|  |  |  |
| --- | --- | --- |
| 进程名称 | 进程到达时间 | 进程服务时间 |
| P1 | 7 | 0 |
| P2 | 9 | 6 |
| P3 | 3 | 3 |
| P4 | 7 | 4 |
| P5 | 4 | 6 |
| P6 | 7 | 9 |
| P7 | 0 | 4 |

#### 4. 程序预期运行效果

因为在每一次关键的步骤之后我都有将进程的相关信息打印出来。所以在进入轮转前，所有进程会根据进程的到达时间进行排序

表3.特定试验测试数据-第一个轮转之后进程的状态

|  |  |
| --- | --- |
| 进程名称 | 进程状态 |
| P2 | Q2 |
| P5 | Q2 |
| P4 | Q2 |
| P3 | Q2 |
| P1 | Q2 |

根据“表1.特定试验测试数据­-数据”，我们不难看出进程名称的顺序就是进程到达时间先后的顺序，“表4. 任意/随机测试数据-第一个轮转之后进程的状态”也是如此。

表4. 任意/随机测试数据-第一个轮转之后进程的状态

|  |  |
| --- | --- |
| 进程名称 | 进程状态 |
| P1 | Q2 |
| P3 | Q2 |
| P5 | Q2 |
| P4 | END |
| P2 | Q2 |
| P6 | Q2 |
| P7 | Q2 |

在每次时间片轮转之后，我都会记录每个进程的状态，

表5.特定试验测试数据-第二个轮转之后进程的状态

|  |  |
| --- | --- |
| 进程名称 | 进程状态 |
| P2 | END |
| P5 | RR |
| P4 | END |
| P3 | END |
| P1 | END |

表6. 任意/随机测试数据-在第二个轮转之后进程的状态

|  |  |
| --- | --- |
| 进程名称 | 进程状态 |
| P7 | END |
| P3 | END |
| P5 | END |
| P1 | END |
| P4 | END |
| P6 | RR |
| P2 | END |

“表5.特定试验测试数据-第二个轮转之后进程的状态”和“表3.特定试验测试数据-第一个轮转之后进程的状态”经过手写运算无误，“表4. 任意/随机测试数据-第一个轮转之后进程的状态”和“表6. 任意/随机测试数据-在第二个轮转之后进程的状态”经过手写运算无误。

再次是输出的甘特图

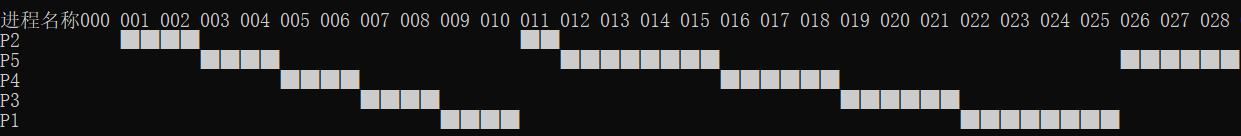


图 3.特定试验测试数据-甘特图

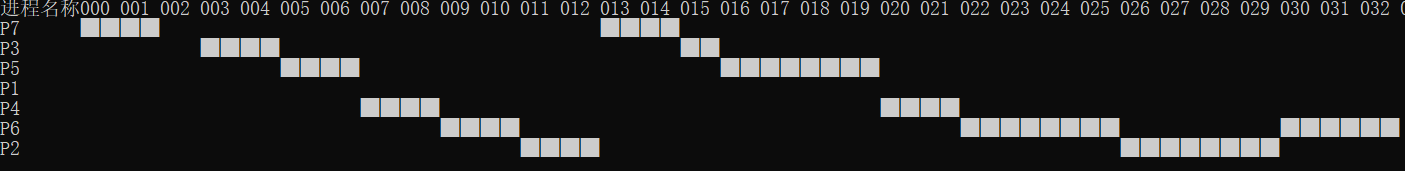


图 4. 任意/随机测试数据-甘特图

两个甘特图中，每个小方块代表0.5个时间单位，经过手写运算之后发现结果无误。

在最后，运行完所有的函数之后输出进程的所有信息，包括进程到达时间、进程服务时间、进程结束时间、进程等待时间、进程周转时间、进程带权周转时间，在服务时间为0时将进程等待时间、进程周转时间、进程带权周转时间设置为0。

表7.特定试验测试数据-进程的所有信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 进程名称 | 进程到达时间 | 进程服务时间 | 进程结束时间 | 进程响应时间 | 进程等待时间 | 进程周转时间 | 带权周转时间 |
| P2 | 1 | 3 | 12 | 0 | 8 | 11 | 3.67 |
| P5 | 2 | 9 | 29 | 0 | 18 | 27 | 3 |
| P4 | 4 | 5 | 19 | 0 | 10 | 15 | 3 |
| P3 | 4 | 5 | 22 | 2 | 13 | 18 | 3.6 |
| P1 | 5 | 6 | 26 | 0 | 15 | 21 | 3.5 |

表8. 任意/随机测试数据-进程的所有信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 进程名称 | 进程到达时间 | 进程服务时间 | 进程结束时间 | 进程响应时间 | 进程等待时间 | 进程周转时间 | 进程带权周转时间 |
| P7 | 0 | 4 | 15 | 0 | 11 | 15 | 3.75 |
| P3 | 3 | 3 | 16 | 0 | 10 | 13 | 4.3 |
| P5 | 4 | 6 | 20 | 0 | 10 | 16 | 2.67 |
| P1 | 7 | 0 | 7 |  |  |  |  |
| P4 | 7 | 4 | 22 | 0 | 11 | 15 | 3.75 |
| P6 | 7 | 9 | 33 | 2 | 17 | 26 | 2.89 |
| P2 | 9 | 6 | 30 | 0 | 15 | 21 | 3.5 |

经过手算之后发现无误。

### 四、实验过程

#### 1．调试过程

在编写代码过程中，编译器错误提示或者程序运行逻辑错误截图。

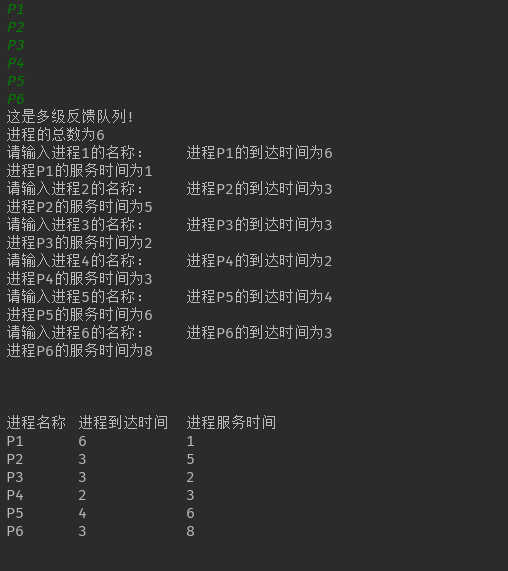


图 5.错误代码截图

分析出现上述错误的原因，说明你如何解决上述错误。

在上面的图片中，我只有在输入全部才会有输出，但是我一开始的想法时便输入进程的名称边输出进程的到达时间以及服务时间。后来我去网上查找了相关资料，找到了一个用于解决这个问题的方法，就是在输出“请输入进程的名称”之后机上一个fflush(stdout);fflush用于清空缓冲流，恰巧printf默认是缓冲输出的，所以用了这个之后就会立刻输出所有在缓冲区的内容。

#### 2. 核心代码截图

**代码截图：**经调试后正确的核心代码截图。

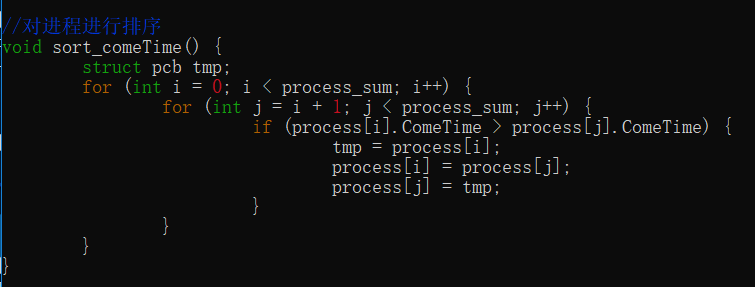


图6. 多级反馈队列调度算法核心代码1

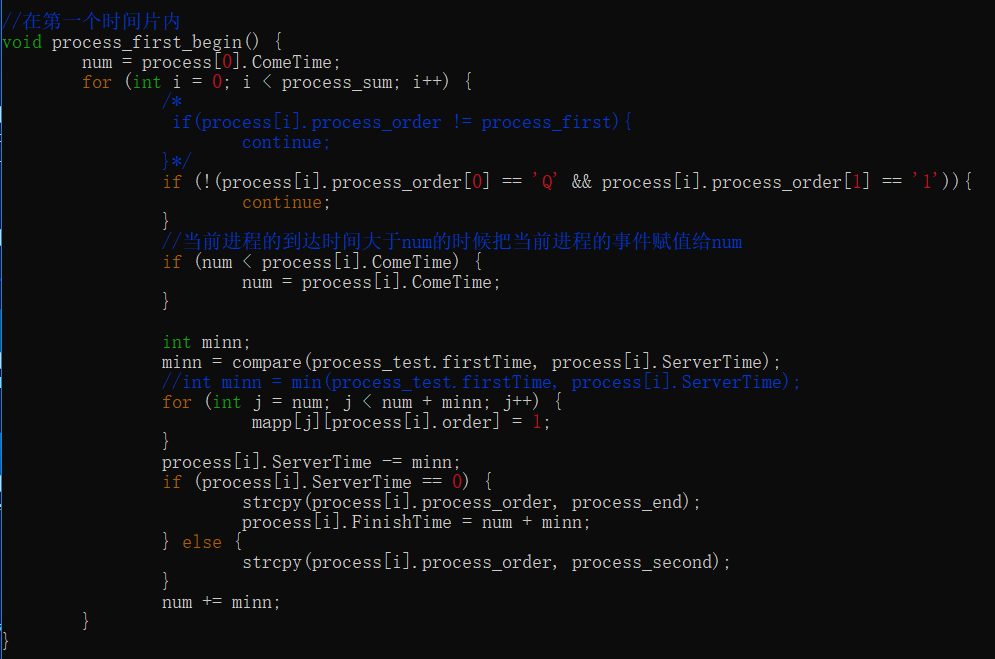


图7. 多级反馈队列调度算法核心代码2

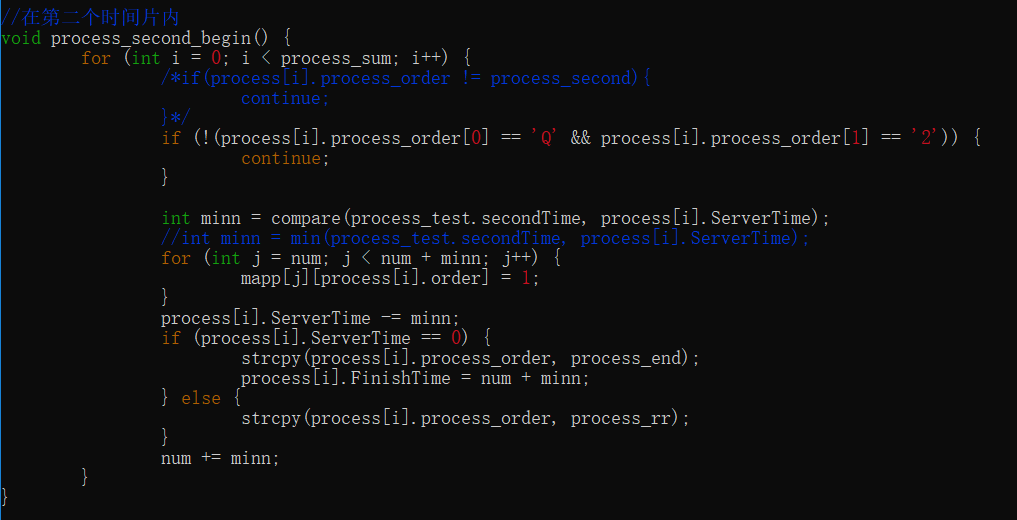


图8. 多级反馈队列调度算法核心代码3

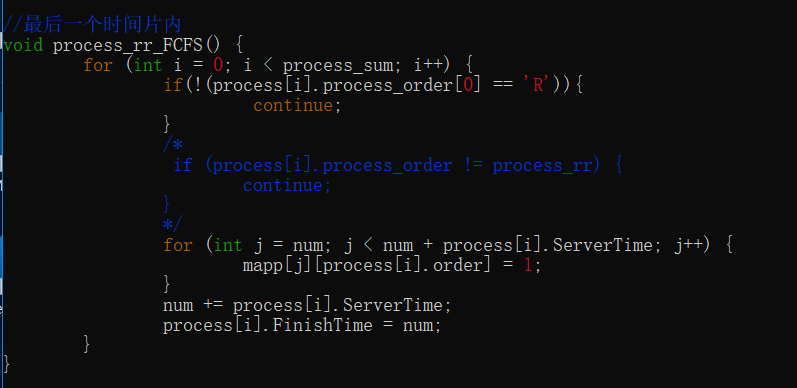


图9. 多级反馈队列调度算法核心代码4

**代码解释：**核心代码1主要是用冒泡排序对所有的进程根据进程的到达时间进行排序。核心代码2是写所有进程在队列一中进行扫描，当扫描到的进程的状态是‘Q1’时进行轮转，轮转之后判断进程的服务时间是否为0，若为0，则进程的状态赋值为‘END’，否则赋值为’Q2’。核心代码3代表所有进程在队列二中进行扫描，当扫描到的进程的状态是‘Q2’时进行轮转，轮转之后判断进程的服务时间是否为0，若为0，则进程的状态赋值为‘RR’，反之赋值为‘END’。核心代码4是所有进程在最后一个轮转片中进行轮转，我用 ‘RR’来代替‘Q3’，因为在最后一个轮转片中是FCFS算法，因此先对所有进程进行挨个扫描，当扫描到的进程的状态时’RR’时进行轮转，轮转完之后直接将该进程的状态设置为‘END’。

#### 3. 程序运行结果

**运行结果：**

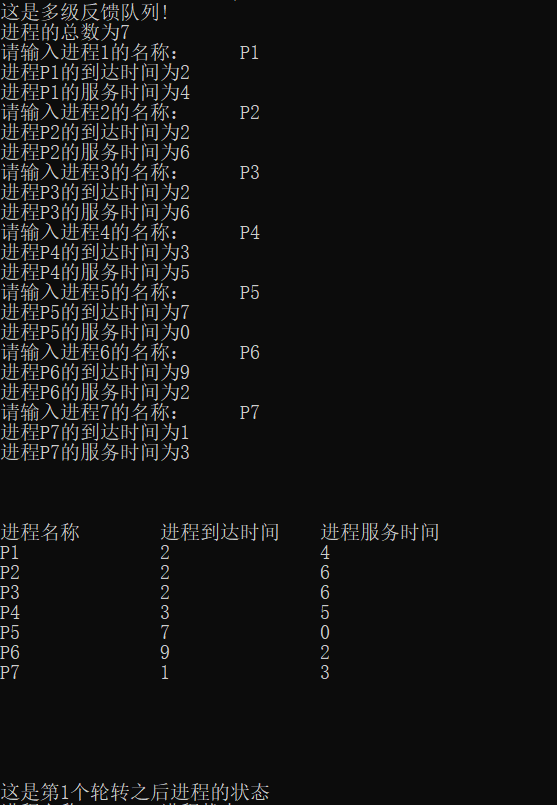


图10.程序运行结果1



图11.程序运行结果2

**结果分析：**

手算之后的答案与上述答案无误。

### 五、实验总结

这次实验用了多级反馈队列调度算法，这个算法我们之前没有学过，所以理解有点困难，但是，这个算法涉及到队列，他是队列的升级，时多级队列，因此，我再次不仅学到了新的知识，还对数据结构中队列部分的熟悉与加深，更好的掌握了队列只是。在了解这个算法是什么之后，我对这个算法有了进一步的改良，进行了优化，用数组来写，每进入一个queue时就是进行一遍扫描，扫描所有进程的状态，我取消了原有的链表结构，采用了这种简单的结构。

当然，这次编程中遇到过许多困难，比如无法输出我想要的答案，无法输出我想要的结构，好在最后我都一一解决了。

多级反馈队列调度算法是操作系统中CPU处理机调度算法之一，该算法使得段进程（任务）能迅速完成。UNIX操作系统便采取这种算法。现实中，我们计算机中打开各种程序，就是多级反馈队列调度算法的应用，这次是我们对操作系统的模拟，与实际相联系，增加了趣味性。

这次是我们第一次接触操作系统，对操作系统原理有了一定的了解，为我们将来学习操作系统打下了基础。

### 六、教师评语及成绩

|  |
| --- |
| **成绩：**  **评语：**  **指导老师：**  **日 期：** |