实验报告四 处理器调度

一、 实验目的

- 理解进程调度的过程。
- 掌握各种进程调度算法的实现方法
- 通过实验比较各种进程调度算法的优劣。

二、 相关知识

- 指针、结构体。
- 操作系统相关内存交换知识。
- 进程调度算法是系统管理进程调度，提高系统性能的重要手段。通过本次实验理解进程调度的机制，在模拟实现先来先服务 FCFS、轮转 RR（q=1）、最短进程优先 SPN、最短剩余时间 SRT、最高响应比优先 HRRN 算法的基础上，比较各种进程调度算法的效率和优劣，从而了解系统进程调度的实现过程。

三、 实验内容

随机给出一个进程调度实例，如：

进程	到达时间	服务时间
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

模拟进程调度，给出按照算法先来先服务 FCFS、轮转 RR（q=1）、最短进程优先 SPN、最短剩余时间 SRT、最高响应比优先 HRRN 进行调度各进程的完成时间、周转时间、响应比的值。

四、 实验环境

- PC + Linux Red Hat 操作系统 + GCC
- 或 Windows xp + VC
- 或 任意 OS + Java

五、 实验中遇到的主要问题及其解决方法

无特别大问题。

六、 源代码和流程图

✧ **初始化**

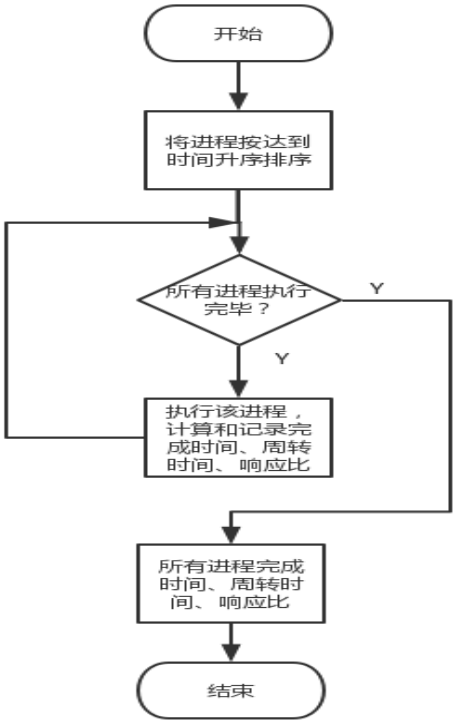
```
cjx@ubuntu:~/Desktop/Lab4$ gcc tools.c FCFS.c RR.c SPN.c SRT.c HRRN.c Main.c -o
main
cjx@ubuntu:~/Desktop/Lab4$ ./main
进程：A
        到达时间：0
        服务时间：3
进程：B
        到达时间：2
        服务时间：6
进程：C
        到达时间：4
        服务时间：4
进程：D
        到达时间：6
        服务时间：5
进程：E
        到达时间：8
        服务时间：2
```

✧ **FCFS 算法：**

思路：1. 将所有进程按到达时间从小到大排序

2. 依次执行所有进程

流程图：



实验结果：

```
------ FCFS -------
进程 A:
        完成时间：3
        周转时间：3
        响应比： 1.00
进程 B:
        完成时间：9
        周转时间：7
        响应比： 1.17
进程 C:
        完成时间：13
        周转时间：9
        响应比： 2.25
进程 D:
        完成时间：18
        周转时间：12
        响应比： 2.40
进程 E:
        完成时间：20
        周转时间：12
        响应比： 6.00
```

源代码：

FCFS.c

✧ **RR 算法：**

思路：1. 将所有到达的加入到队列中

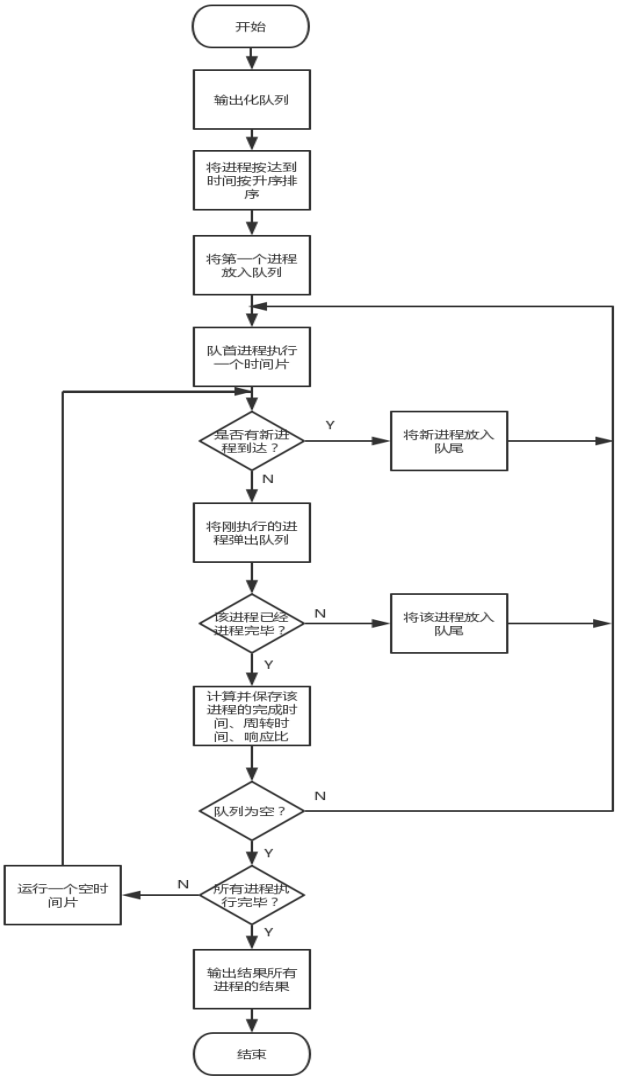
2. 队首进程执行一个时间片

3. 将新到达的进程加到队尾

4. 将执行的进程弹出队首，若该进程还没执行完毕则将其加到队尾

5. 重复步骤 2-4 直到所有进程执行完毕

流程图：



执行结果：

```
------ RR -------
进程 A:
        完成时间：4
        周转时间：4
        响应比： 1.33
进程 B:
        完成时间：18
        周转时间：16
        响应比： 2.67
进程 C:
        完成时间：17
        周转时间：13
        响应比： 3.25
进程 D:
        完成时间：20
        周转时间：14
        响应比： 2.80
进程 E:
        完成时间：15
        周转时间：7
        响应比： 3.50
```

源代码：

RR.c

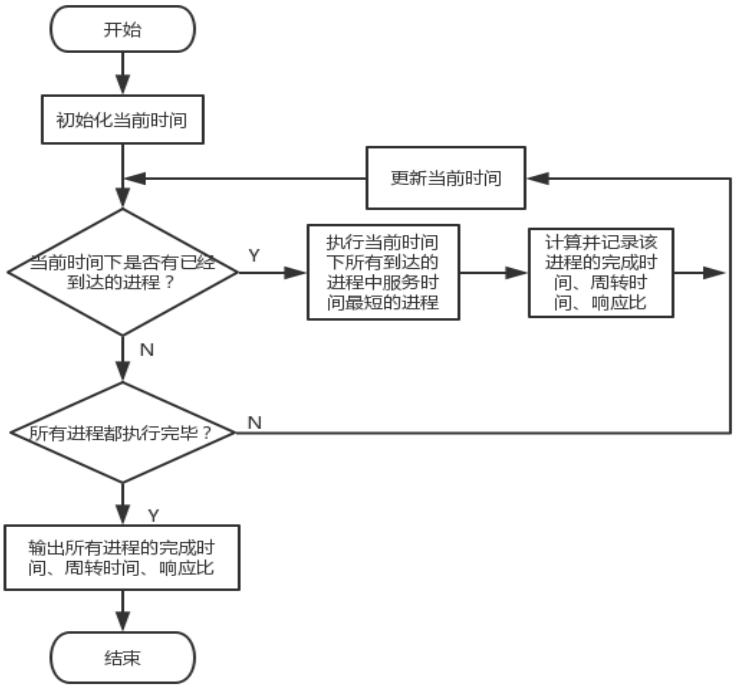
✧ **SPN 算法：**

思路：1. 找出所有到达进程中服务时间最短的进程

2. 将该进程执行完毕

3. 重复步骤 1-2 直到所有进程执行完毕

流程图：



运行结果：

```
------ SPN -------
进程 A:
        完成时间：3
        周转时间：3
        响应比： 1.00
进程 B:
        完成时间：9
        周转时间：7
        响应比： 1.17
进程 C:
        完成时间：15
        周转时间：11
        响应比： 2.75
进程 D:
        完成时间：20
        周转时间：14
        响应比： 2.80
进程 E:
        完成时间：11
        周转时间：3
        响应比： 1.50
```

源代码：

SPN.c

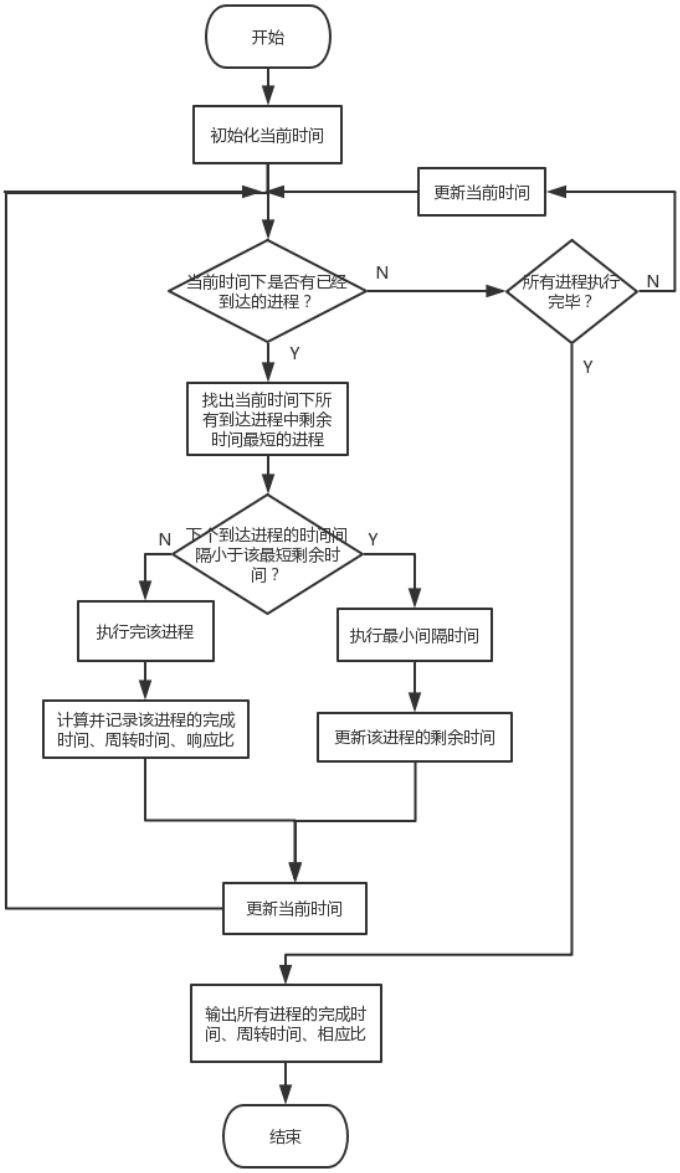
✧ **SRT 算法：**

思路：1. 找出所有到达进程中剩余时间最短的进程

2. 若有即将到来的时间且间隔小于最短剩余时间，则最短剩余时间的进程只执行间隔时间，否则执行完所有剩余时间

3. 重复步骤 1-2 直到所有进程执行完毕

流程图：



运行结果：

```
------ SRT -------
进程 A:
        完成时间：3
        周转时间：3
        响应比： 1.00
进程 B:
        完成时间：15
        周转时间：13
        响应比： 2.17
进程 C:
        完成时间：8
        周转时间：4
        响应比： 1.00
进程 D:
        完成时间：20
        周转时间：14
        响应比： 2.80
进程 E:
        完成时间：10
        周转时间：2
        响应比： 1.00
```

源代码：

SRT.c

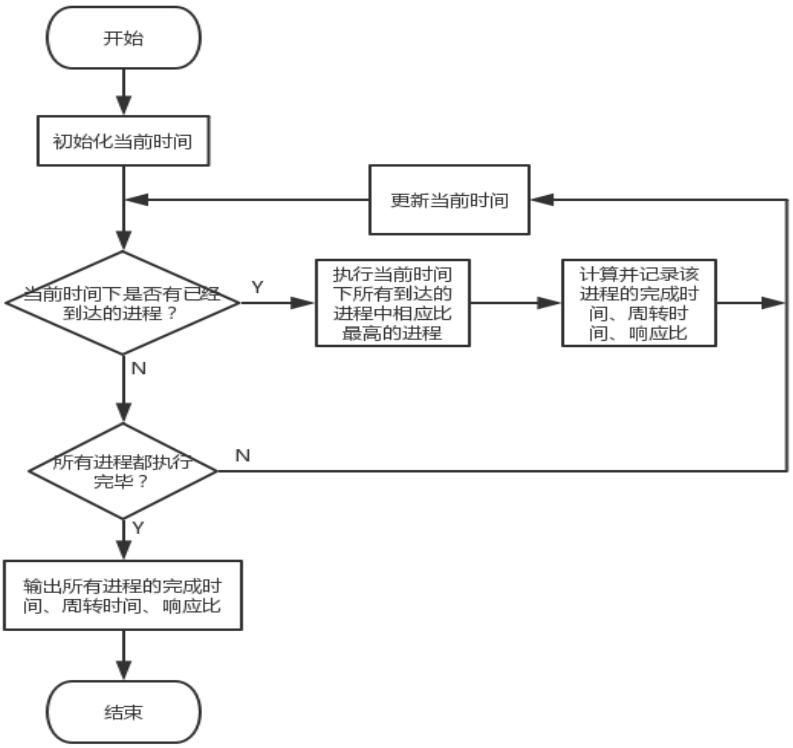
✧ **HRRN 算法：**

思路：1. 找出所有到达进程中响应比最高的进程

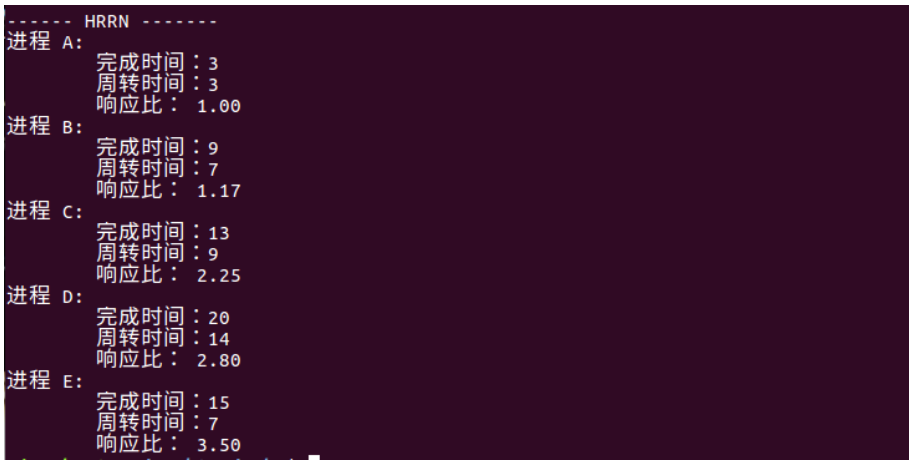
2. 执行该进程

3. 重复步骤 1-2 直到所有进程执行完毕

流程图：



运行结果：



源代码：

HRRN.c

七、 实验总结

- ✧ 通过实现对比 FCFS、RR、SPN、SRT、HRRN 几种不同的进程调度算法，我对进程的调度有了更加深入的理解。简单的算法资源消耗较少，但是整体调度效果往往不是很好，比如 FCFS 算法。要想实现较好的调度效果，往往要以消耗资源为代价。