**实验五 进程调度**

# 姓 名 蒋文强 学 号 21013017 成绩

实验时间2023.11.30 指导教师(签名)

**（诚信声明：本实验报告内容，均由本人亲自上机完成。 签名：\_\_\_\_\_\_\_\_\_\_\_\_\_）**

一．实验目的

1.深入了解进程调度的策略和机制

2.掌握各种调度算法的基本原理和评价指标

3.实现几种常用的调度算法

二．实验工具与设备

装有 Linux 操作系统的计算机。

三．实验内容

1. 用 C 语言实现 FCFS、SPN 和 RR 算法
2. 计算每种算法下的平均周转时间、平均带权周转时间
3. **计算方法：**

进程周转时间 ＝ 进程完成时间 － 进程提交时间

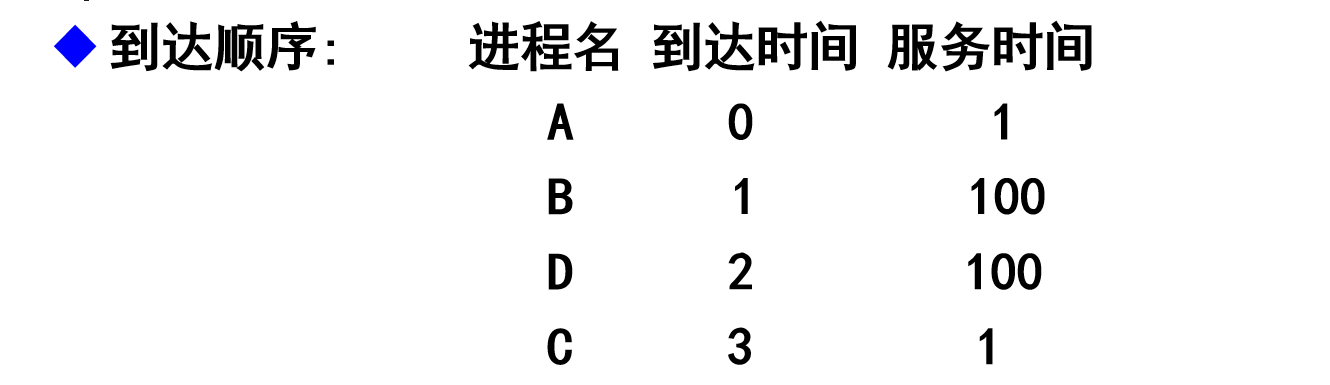
平均周转时间 ＝ 所有进程的周转时间之和/进程数

带权周转时间 ＝ 进程周转时间/进程运行时间

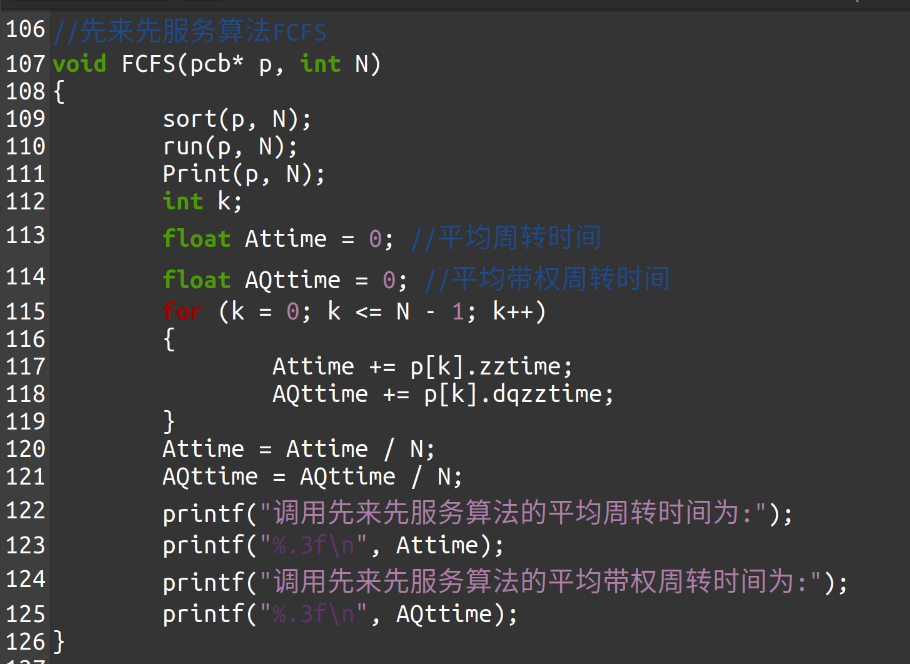
平均带权周转时间 ＝ 所有进程的带权周转时间之和/进程数

实验代码及结果见后，在实验程序中编写并显示每个算法的平均周转时间、平均带权周转时间。

选择**进程样例**如下：



**FCFS算法：**根据进程提交或进程变为就绪状态的先后次序，分派CPU。当前进程占用 CPU，直到执行完或阻塞，才让出CPU。在进程唤醒后（如 I/O 完成），并不立即恢复执行，通常等到当前进程出让 CPU。这种算法比较有利于长进程，而不利于短进程。



**步骤如下：**

1. 排序：所有进程按照它们进入系统的顺序进行排序。这是先来先服务算法的基础，即首先服务的总是最先到达的进程。
2. 执行：按照排序后的顺序执行所有的进程。如果一个进程正在执行，有其他进程到达，那么新的进程会打断正在执行的进程，并接替它执行。
3. 计算和输出：在所有进程执行完成后，计算它们的平均周转时间和平均带权周转时间，并打印出来。

**运行结果：**



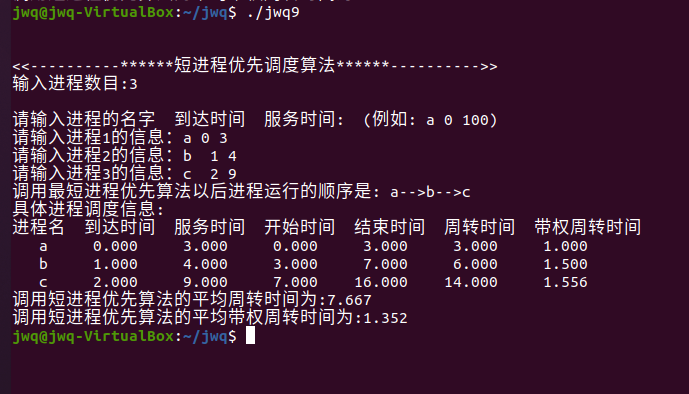
**SPN算法：**非剥夺策略的短进程优先调度算法。对预计执行时间短的进程优先分派处理机。缺点是对长进程非常不利，可能长时间得不到执行，且未能依据进程的紧迫程度来划分执行的优先级，以及难以准确估计进程的执行时间，从而影响调度性能。



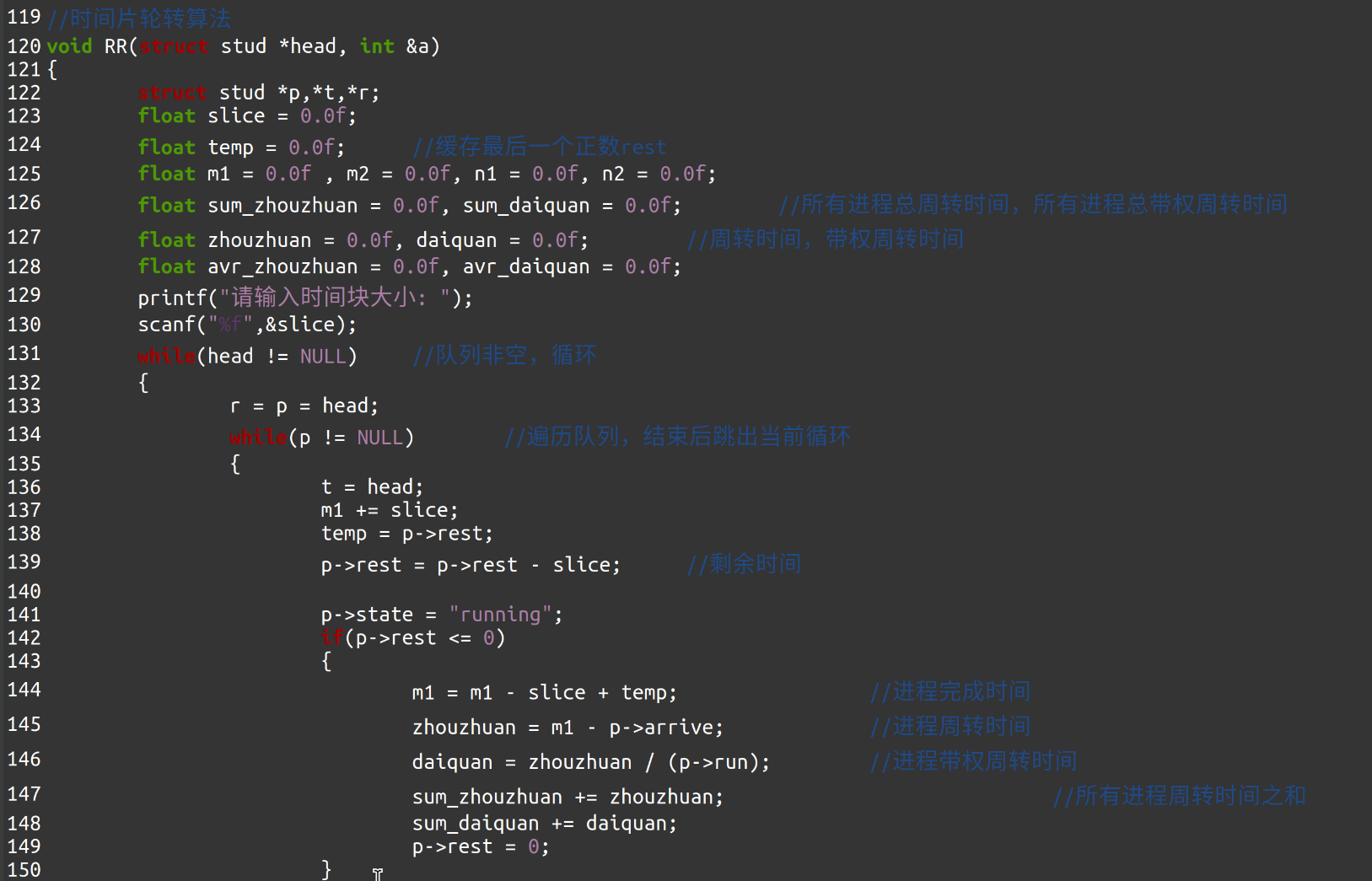
**步骤如下：**

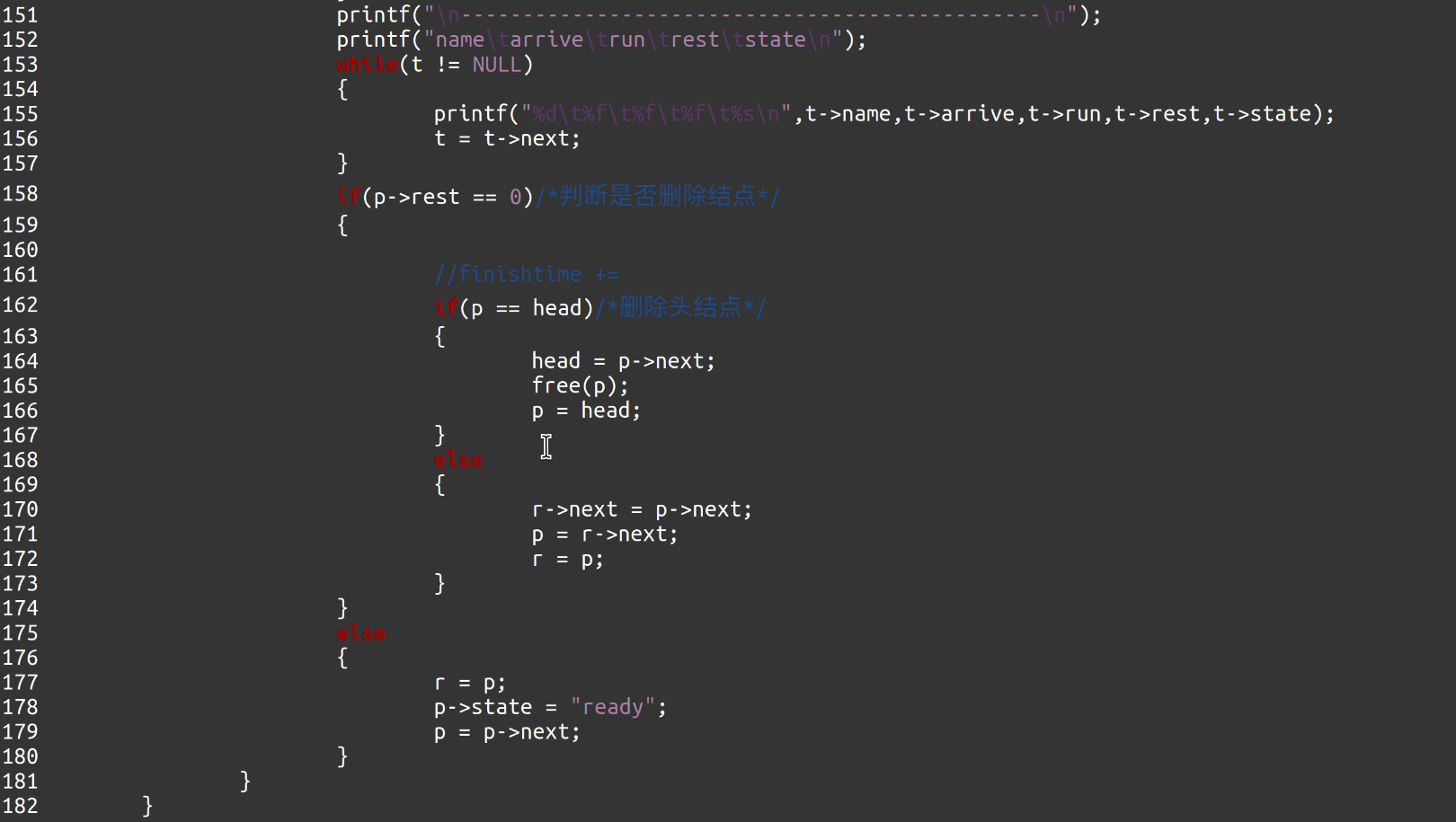
1. 排序：所有的进程按照它们的预计执行时间进行排序。这是短进程优先算法的基础，即优先执行的总是预计执行时间最短的进程。
2. 执行：按照排序后的顺序执行所有的进程。如果一个进程正在执行，有其他进程到达，那么新的进程会打断正在执行的进程，并接替它执行。
3. 计算和输出：在所有进程执行完成后，计算它们的平均周转时间和平均带权周转时间，并打印出来。

**运行结果：**



**RR算法：**时间片轮转的调度算法。让就绪进程按时间片轮流使用 CPU 的调度方式，将系统中所有的就绪进程排成一个队列，每次调度时将 CPU 分派给队首进程，让其执行一个时间片。在一个时间片结束时，发生中断，调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程。







**输入时间块大小：**

**从用户那里接收时间块的大小，这个大小决定了每个进程在CPU上执行的时间片长度。**

**遍历进程队列：**

**针对进程队列中的每一个进程，按照时间块大小分配执行时间。**

**更新剩余时间：**

**每个进程的剩余时间减去分配的时间块大小。**

**判断并更新状态：**

**如果进程的剩余时间小于或等于0，说明这个进程已经完成。**

**记录下这个进程的周转时间和带权周转时间，并将总周转时间和总带权周转时间相应地增加。**

**同时，从进程队列中删除这个进程。**

**打印进程状态：**

**在每次循环中，打印当前所有进程的名称、到达时间、运行时间、剩余时间和状态。**

**循环进行：**

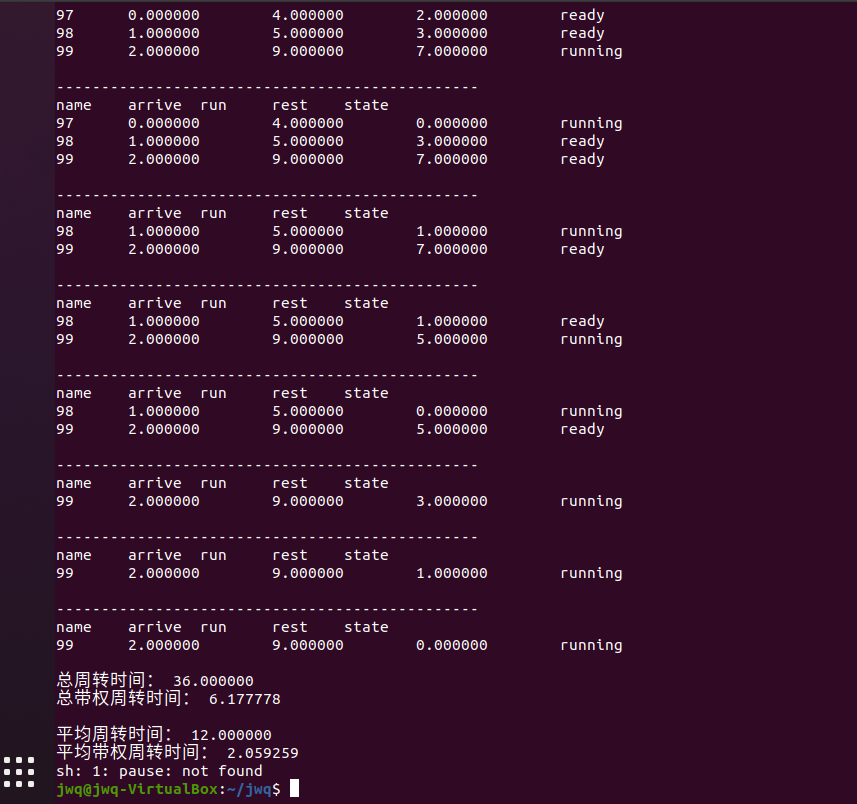
**重复步骤2至步骤5，直到所有的进程都已经执行完毕。**

**输出结果：**

**打印总周转时间和总带权周转时间，以及平均周转时间和平均带权周转时间。**

**重新输出一份当前进程的状态。**

**运行结果：（截取部分轮转中的结果展示）**



# 四．思考题

1. 在实际的进程调度中，除了按调度算法选择下一个执行的进程外，还应该处理哪些工作？

答：除了按调度算法选择下一个执行的进程外，还需要处理以下工作：

记录系统中所有进程的执行情况：

进程控制块（PCB）是记录进程执行情况的关键数据结构。它包含了进程的状态、寄存器的值、程序计数器（PC）、进程标识符（PID）等信息。每个进程都有一个对应的PCB。

进行进程上下文切换：

当操作系统决定切换到另一个进程时，需要进行进程上下文切换。这包括保存当前执行进程的上下文信息到其对应的PCB中，然后加载下一个进程的上下文信息，以确保进程之间的平滑切换。

1. 几种进程调度算法有何区别与联系？

FCFS (First-Come-First-Serve) 算法：

优点：实现简单，公平，按照提交顺序执行，适用于长时间占用CPU的任务。

缺点：不考虑任务执行时间，可能导致"先来先服务"的长作业等待时间过长，这可能导致平均等待时间增加。

不利于短作业：因为长作业可能会长时间占用CPU，短作业需要等待。

不利于I/O繁忙的作业：如果一个作业在等待I/O完成时不释放CPU，其他进程可能需要等待很长时间。

RR (Round Robin) 算法：

优点：公平，每个进程都有机会执行，并且响应时间较短。

缺点：对于长时间运行的任务，平均周转时间可能较长。

适用于I/O密集型程序：因为每个任务都有时间片，CPU不会被长时间占用，允许其他任务在等待I/O时执行。

SPN (Shortest Process Next) 算法：

优点：最小化平均周转时间和带权周转时间，对短作业有利。

缺点：对长作业不利，可能会等待很长时间。

难以准确估计执行时间：因为需要提前知道每个进程的执行时间，这在实际中通常是难以准确估计的。

联系：

RR 和 SPN 算法都涉及到进程的优先级，尽管优先级的定义和计算方式有所不同。RR 通过时间片轮转实现公平性，而 SPN 通过考虑预计执行时间实现优先级。

RR 和 FCFS 算法都可以实现非抢占式调度，即正在执行的进程不会被其他进程中断。这对于某些应用场景和系统设计有一定的影响。

1. RR 调度算法中，如果将时间定量增长为一个任意大的数目，那么会产生什么影响？

答：产生的影响：

1. 如果时间片长度过短，导致调度程序剥夺处理器的次数增加，进程的上下文切换的次数增加，系统的开销也增加。
2. 如果时间片长度过长，长到能使就绪队列中所需要执行时间最长的进程执行完毕，轮转法就变成了FCFS算法，有FCFS的缺点，不利于短作业。