Chapter 5 Homework

陈文迪 519021910071

作业中的引用内容均已标出

5.4 考虑下列的进程集合，CPU执行已经给出，单位是毫秒。

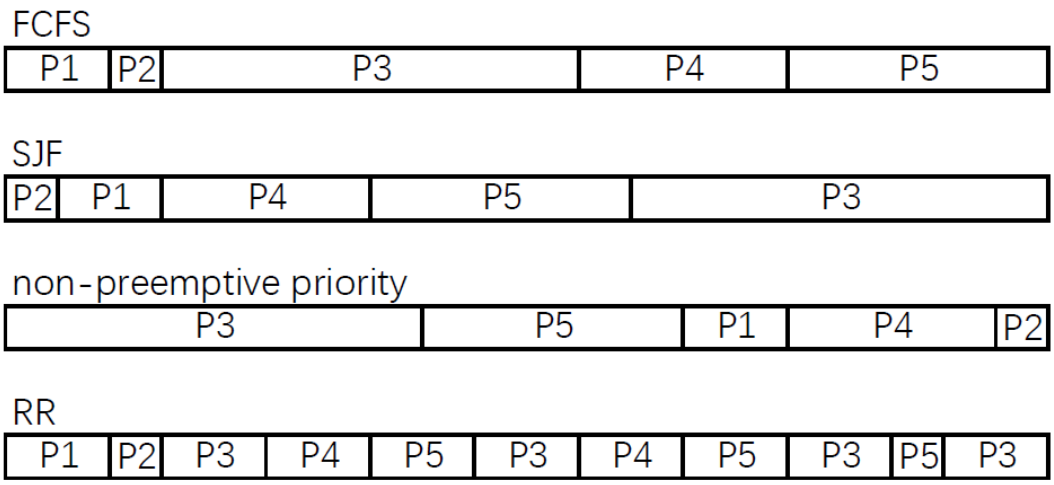
Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

进程到达的顺序为P_1，P_2，P_3，P_4，P_5，到达时间均为0。回答以下问题：

1. 绘制4幅Gantt图，分别展示在以下调度算法下进程执行的情况：FCFS、SJF、非抢占优先级调度和RR（时间片长度为2）。
2. 在第1题中每个算法中各进程的周转时间为多少?
3. 在第1题中每个算法中各进程的等待时间为多少?
4. 哪种算法实现了最小化平均等待时间?

问题解答：

1. Gantt图如下。



2. 各进程的周转时间如下表所示。

	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

3. 各进程的等待时间如下表所示。

	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	9
P_5	15	7	8	13
Average	6.2	4.6	11	7.2

4. 从理论上可以证明，SJF在平均等待时间上是最优的。上表的计算也验证了这一点。

5.5 下列进程使用抢占的RR算法进行调度。

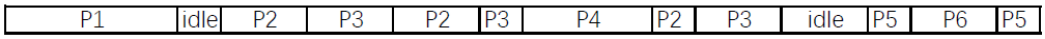
Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

每个进程都分配有一个数字优先级，数字越大表示相对优先级越高。除了上方列出的进程外，系统还具有一个空闲任务（该任务不占用CPU资源，被标识为P_{idle}）。该任务的优先级为0，并且在系统没有其他可用进程要运行时进行调度。一个时间片的长度是10个单位。如果一个进程被更高优先级的进程所抢占，那么被抢占的进程会位于队列的最后。

1. 使用Gantt图展示调度结果。
2. 每个进程的周转时间是多少?
3. 每个事件的等待时间是多少?
4. CPU利用率为多少?

问题解答：

1. Gantt图如下。



2.

3.

Process	Turnaround Time	Waiting Time
P_1	20	0
P_2	55	30
P_3	60	35
P_4	15	0
P_5	20	10
P_6	10	0

4. $\frac{120-5-10}{120} \times 100\% = 87.5\%$

5.10 考虑如下的进程调度问题。

传统的UNIX调度程序在优先级数字和优先级之间建立反比关系：数字越大，优先级越低。调度程序使用以下式子每秒重新计算一次进程优先级：

$$Priority = \frac{\text{recent CPU usage}}{2} + base$$

其中 base = 60，recent CPU usage 表示自从上次重新计算优先级以来进程使用CPU的时长。

假设 P_1、P_2和P_3的 recent CPU usage 分别是40、18和10。重新计算优先级后，这三个进程的新优先级将是什么？据此分析，传统的UNIX调度程序会提高还是降低CPU密集型进程的相对优先级？

问题解答：

依据公式，我们可以得到 P_1、P_2和P_3的新优先级分别为80、69和65。对于CPU密集型进程，其使用CPU的时长较长，recent CPU usage 的值较大，相比于其他进程会得到更大的优先级数字，因此相对优先级降低。

5.18 下列进程使用抢占的基于优先级的RR算法进行调度。

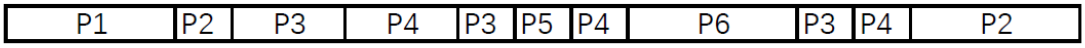
Process	Priority	Burst	Arrival
P_1	8	15	0
P_2	3	20	0
P_3	4	20	20
P_4	4	20	25
P_5	5	5	45
P_6	5	15	55

每个进程都分配有一个数字优先级，数字越大表示相对优先级越高。调度程序会选择优先级最高的进程执行。对于具有相同优先级的一系列进程，会使用时间片为10个单位的RR调度程序进行调度。如果一个进程被一个更高优先级的进程抢占，被抢占的进程会被放置到队列的尾端。

1. 使用Gantt图展示进程调度的结果。
2. 每个进程的周转时间是多少?
3. 每个进程的等待时间是多少?

问题解答：

1. Gantt图如下。



2.

3.

Process	Turnaround Time	Waiting Time
P_1	15	0
P_2	95	75
P_3	55	35
P_4	55	35
P_5	5	0
P_6	15	0

5.20 下列哪些调度算法可能产生饥饿?

1. First-come, first-served（FCFS）
2. Shortest job first（SJF）
3. Round robin（RR）
4. Priority

问题解答：

1. 不会产生饥饿。因为FCFS是一个FIFO队列，任何进程都不会被无穷阻塞。
2. 有可能。SJF是优先级调度的特例，若有一个进程的CPU执行长度较长，就有可能被CPU执行较短的进程无穷阻塞。因此，我们需要采用老化等方案予以解决。
3. 若是单纯的RR算法不会产生饥饿。RR算法保证了只要在循环队列中的进程，一定会有被执行到的机会。但若是RR算法与优先级算法结合，或是进程的数量过多以至于无法全部添加到循环队列中，则也有可能产生饥饿。
4. 有可能。优先级低的进程有可能被优先级高的进程无穷阻塞。我们可以采用老化的方法予以解决。