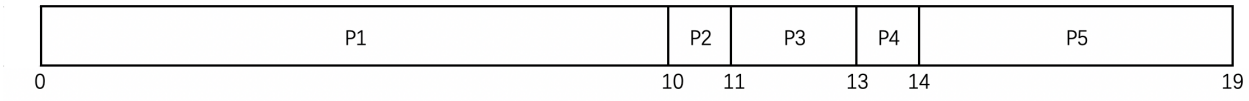
操作系统第四次作业（201700130011 菁英班 刘建东）

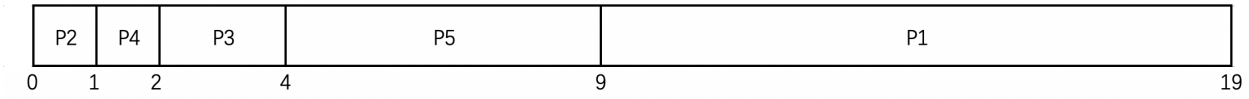
5.4

> 题目：考虑下列进程集，进程占用的CPU区间长度以毫秒来计算：
>
> | 进程 | 区间时间 | 优先级 |
> | --- | --- | --- |
> | P_1 | 10 | 3 |
> | P_2 | 1 | 1 |
> | P_3 | 2 | 3 |
> | P_4 | 1 | 4 |
> | P_5 | 5 | 2 |
>
> 假设在时刻 0 以进程 P_1, P_2, P_3, P_4, P_5 的顺序到达。
>
> a. 画出 4 个 Gantt 图分别演示用 FCFS、SJF、非抢占优先级（数字小代表优先级高）和RR（时间片=1）算法调度时进程的执行过程。

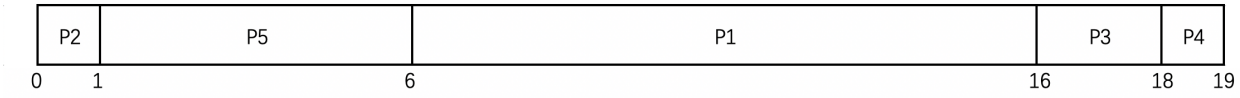
FCFS：先到先执行。



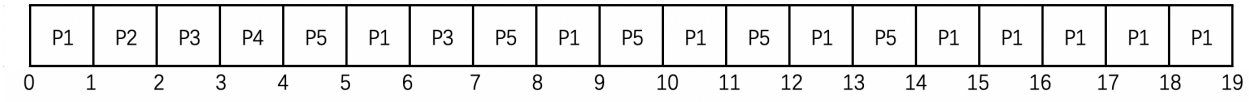
SJF：（非抢占式）每次在等待队列中挑选CPU占用时间最短的进程执行。



非抢占式优先级：每次从等待队列中选出优先级最高的进程执行。



RR（时间片=1）：每个进程执行完时间片之后回到队尾，重新分配时间片。



> b. 在a里每个进程在每种调度算法下的周转时间是多少?

	FCFS	SJF	非抢占式优先级	RR
P1	10	19	16	19
P2	11	1	1	2
P3	13	4	18	7
P4	14	2	19	4
P5	19	9	6	14

> c. 在a里每个进程在每种调度算法下的等待时间是多少?

等待时间 = 周转时间 - CPU使用时间

	FCFS	SJF	非抢占式优先级	RR
P1	0	9	6	9
P2	10	0	0	1
P3	11	2	16	5
P4	13	1	18	3
P5	14	4	1	9

> d. 在a里哪一种调度算法对于所有进程的平均等待时间最小?

FCFS：$\frac{0+10+11+13+14}{4}=12$

SJF：$\frac{9+0+2+1+4}{4}=4$

非抢占式优先级调度：$\frac{6+0+16+18+1}{4}=10.25$

RR：$\frac{9+1+5+3+9}{4}=6.75$

因此，SJF调度算法对于所有进程的平均等待时间最小。

5.5

> 题目：下面哪些算法会引起饥饿
>
> a. 先来先服务（FCFS）
>
> b. 最短工作优先调度（SJF）
>
> c. 轮换法调度（RR）
>
> d. 优先级调度（Priority）

最短工作优先调度和优先级调度算法会引起饥饿。因为存在一个进程执行时间过长或优先级过低，导致其始终无法被分配CPU，引发饥饿现象。

5.10

> 题目：阐述以下调度算法在区分短进程方面的区别
>
> a. FCFS
>
> b. RR
>
> c. Multilevel feedback queues

- FCFS：由于任何在长任务之后到达的短进程都会有很长的等待时间，因此该算法可以通过等待时间来区分短进程。
- RR：该算法分配相同的时间片给所有的进程，因此短进程可以更快完成，周转时间更短。
- 多级反馈队列：该算法与RR算法相类似，没有本质区别，因此也能通过判断任务的周转时间来区分短进程，周转时间短的为短进程。

7.1

> 题目：考虑下述的交通死锁。
>
> a. 阐述下述例子包括了死锁的四个必要条件。
>
> b. 给出一个简单的规则用于在下述系统中避免死锁。

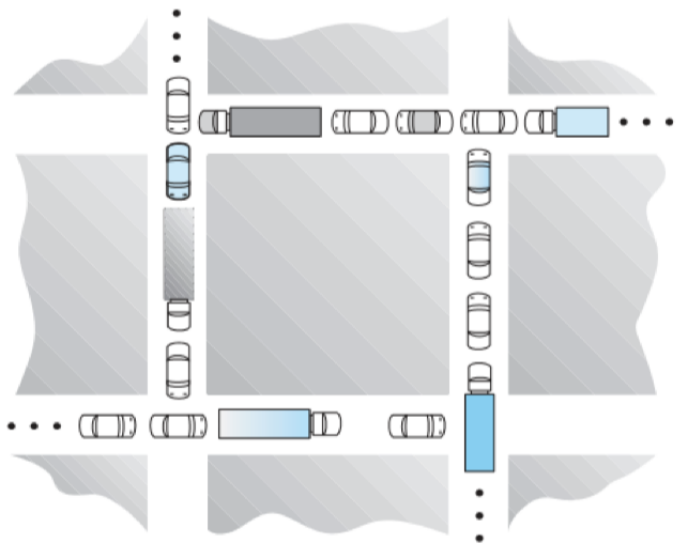


Figure 7.1 Traffic deadlock for Exercise 7.1.

a. 死锁的四个必要条件：

1. 互斥：道路上的每个空间都只有一辆车占用。
2. 占有且等待：一辆车占有着当前的道路位置，并且在等待前进。
3. 无优先级（不可抢占）：一辆车在等待的状态中，不能从道路的当前位置移动到另一个位置。
4. 环路等待：每辆车等待着一辆车向前行驶，其线路构成了一个环路。

b. 避免死锁的规则：

最简单的规则就是保证永远不进入死锁状态，根据该交通图可以发现引发死锁的主要原因是一个十字路口由多个方向的线路共用。因此只要搭建立交桥将所有的十字路口拆分成两个方向，或者规定车辆不准通过十字路口，就不会再发生死锁。

7.5

> 题目：在一个真实的计算机系统中，可用的资源和进程对资源的需求都不会持续数月。资源会损坏或被替换，新进程会产生和结束，新资源也会被带来和添加到系统中。如果用银行家算法控制死锁，下面哪些变化基于什么情况下是安全的（不会导致可能发生的死锁）？
>
> a. 增加可用资源
>
> b. 减少可用资源（资源从系统中永久性移除）
>
> c. 增加一个进程所需资源的上限
>
> d. 减少一个进程所需资源的上限
>
> e. 增加进程数量
>
> d. 减少进程数量

a. 增加可用资源：可以在任何没有错误的情况下安全改变。

b. 减少可用资源：这可能会影响系统并导致死锁。因为系统可能恰好需要这部分减少的资源使某一进程结束，避免死锁发生，但缺少这部分资源后就会导致死锁的出现。

c. 增加一个进程所需资源的上限：这可能会影响系统并导致死锁。因为所需资源数增加，导致原有分配方案不再适用，致使死锁发生。

d. 减少一个进程所需资源的上限：可以在任何没有错误的情况下安全改变。

e. 增加进程数量：如果系统拥有运行该进程所需的所有资源，则不会影响系统安全性。

f. 减少进程数量：可以在任何没有错误的情况下安全改变。

7.8

> 题目：假设哲学家进餐问题中，筷子被摆放在桌子的中央，且任意两根筷子都可以被哲学家使用。假如每次只能请求一根筷子，试描述一个简单规则，用来判定一个特定的请求能否在不会引起死锁的情况下被满足。

引起死锁的原因是每个哲学家都只有一根筷子，因此只要保证不存在同时有两个哲学家都只有一根筷子就可以避免死锁。

规则：哲学家获得第一根筷子当且仅当不存在其他哲学家仅拥有一根筷子。

该规则相当于对筷子加了互斥锁，当一个哲学家拿了两根筷子之后其它哲学家才能继续拿筷子。

7.11

> 题目：考虑下面的一个系统在某一时刻的状态，并使用银行家算法回答下面的问题：
>
> a. Need 矩阵的内容是怎样的?
>
> b. 系统是否处于安全状态?
>
> c. 如果从进程 P_1 发出一个请求 $(0,4,2,0)$，这个请求能否被满足?

a. Need 矩阵内容为 $P_0(0,0,0,0)$、$P_1(0,7,5,0)$、$P_2(1,0,0,2)$、$P_3(0,0,2,0)$、$P_4(0,6,4,2)$

b. 系统处于安全状态，因为可用矩阵为 $(1,5,2,0)$，可以支持 P_0 和 P_3 运行结束。而 P_0 和 P_3 运行结束后，释放的资源可以支持剩下的进程运行结束，因此不存在死锁，系统处于安全状态。

c. 可以被满足，满足 P_1 之后，可用矩阵变为 $(1,1,0,0)$，等待 P_0 运行结束后，可用矩阵变为 $(1,1,1,2)$，可以支持 P_2 运行结束。同理，接下来可以按照 P_1、P_3、P_4 的顺序运行完所有的进程。