一、单项选择题

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
C	B	C	D	B	A	D	C	D	A	B	D	A	D	C	B	B	D	C	C	A	A	C	D	C

26	27	28	29	30	31	32	33	34	35	36	37
A	D	C	D	D	A	D	C	B	C	C	A

二、简答题

1、

答：使用用户程序实现一个用户级线程库，使其映射到进程中；

2、

答：子进程需要执行exec()系统调用来拥有自己独立的新代码段。exec()系统调用用于将一个进程的代码段替换为另一个进程的代码段。 exec()系统调用的返回值规定如下：

```
a、如果执行成功，返回值为0。
b、如果执行失败，返回值为-1。
```

3、

答：采用以下标准来选择被杀死的进程： a、优先级：根据进程的优先级来选择。优先级较低的进程更容易被杀死。 b、资源占用：选择占用资源较少的进程作为牺牲品。 c、进程状态：选择处于阻塞状态的进程。 d、进程时长：选择运行时间较短的进程。 e、用户干预：可以根据用户的需求和系统实际情况，由系统管理员或用户手动选择要杀死的进程。

4、

（1）$T_1 = \frac{1+1+0.7+0.7}{4} = 0.85$

$W_1 = \frac{1+2+3.5+7}{4} = 3.375$

（2）$T_2 = \frac{1+0.2+0.2+1.3}{4} = 0.675$

$W_2 = \frac{1+1+2+2.6}{4} = 1.64$

（3）$T_3 = \frac{1+1+0.5+0.8}{4} = 0.825$

$W_3 = \frac{1+2+5+4}{4} = 3$

5、

答：（1）是安全状态，安全序列为：$P_4 \rightarrow P_5 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$

（2）不能，因为资源不足

（3）可以，因为分配资源后仍然可以找到一个安全序列

（4）不可以，因为分配后找不到安全序列

6、

答：（1）总资源（9，3，6）

P_1所需：（2，2，2）

P_2所需：（2，0，2）

P_3所需：（1，0，3）

P_4所需：（4，2，0）

（2）给P_2分配资源而不给P_1分配；因为如果给P_1分配资源，此时找不到安全序列

（3）此时系统中虽然找不到安全序列，但没有新的资源申请，所以系统还未处于死锁状态

7、

答：

步骤	描述
1	P得到满足，此时P，Q，R占有资源数为（2，0，0）
2	Q得到满足，此时P，Q，R占有资源数为（2，4，0）
3	R得到满足，此时P，Q，R占有资源数为（2，4，2）
4	Q无法得到满足，Q进程进入阻塞态
5	R无法得到满足，R进程进入阻塞态
6	P得到满足，此时P进程已得到最大需求量资源，会返还所有占有的资源，唤醒Q进程并给它分配两个资源， 而R进程依旧在阻塞态，因为再分配给R两个资源，系统中就找不到安全序列

8、

答：在分时操作系统中，为了保证不会有进程因为优先级太低而饥饿，可以采用以下策略： a、动态调整优先级：根据进程的执行情况，动态地调整进程的优先级。 b、最高优先级限制：为每个进程设置一个最高优先级限制，即使进程的优先级低于其他进程，但在达到最高优先级限制后，该进程将优先获得CPU资源。 c、进程优先级翻转：在优先级轮转法中，当一个进程执行完毕或主动让出CPU资源时，将其优先级翻转，使其在下次执行时具有较高的优先级。 d、动态调整时间片：根据进程的执行情况和系统负载情况，动态地调整进程的时间片。

9、

答：（1）实际操作系统中应对死锁的可行方法主要包括以下几种： 预防死锁：通过破坏死锁产生的必要条件来预防死锁。 避免死锁：通过动态分配资源和使用锁来避免死锁。

（2）当操作系统发现死锁已经发生时，可以采取以下措施来减小损失：

a、死锁解除：通过重新分配资源或调整进程优先级，解除死锁。例如，可以优先分配高优先级进程所需资源，以使其尽快恢复执行。
b、资源回收：对于不再使用的资源，操作系统可以将其回收，以便重新分配给其他进程。
c、进程恢复：对于因死锁而陷入阻塞的进程，操作系统可以将它们恢复到死锁发生前的状态，以便让它们继续执行。

10、

答：5个；

分析：主线程执行foo()函数，在foo()中，首先调用fork()创建一个子进程，子进程也会调用foo()函数，这样就形成了两个子进程。每个子进程内部，都会创建一个新线程来执行bar()函数。因此，总共会创建4个线程（两个子进程，每个子进程创建两个线程）。bar()函数中，首先递减counter，然后判断counter是否为0。由于counter初始值为1，每个线程执行bar()时，counter都会递减1。当counter为0时，执行fork()创建一个新的子进程，并打印"hello"。由于每个线程执行bar()时都会创建一个新的子进程，所以会打印4个"hello"。最后，主线程执行完毕，再打印一个"hello"，程序结束。