#### 典型回答

当一个进程申请资源时，如果这时没有可用资源，那么这个进程就进入等待状态。有时，如果所申请的资源被其他等待进程占用，那么该等待进程有可能再也无法改变状态，这种情况称为死锁。

如果在一个系统中以下四个条件同时成立，那么就能引起死锁：

* 互斥：至少有一个资源必须处于非共享模式，即一次只有一个进程可使用。如果另一个进程申请该资源，那么申请进程应等到该资源释放为止。
* 占用并等待：一个进程应占有至少一个资源，并等待另一个资源，而该资源为其他进程所占有。
* 非抢占：资源不能被抢占，即资源只能被进程在完成任务后自愿释放。
* 循环等待：有一组等待进程 {P0，P1，· · · Pn}，P0 等待的资源为 P1 占有，P1 等待的资源为 P2 占有，Pn-1 等待的资源为 Pn 占有，Pn 等待的资源为 P0 占有。

我们强调所有四个条件必须同时成立才会出现死锁。循环等待条件意味着占有并等待条件，这样说四个条件其实并不是完全独立的。然鹅，后面我们会看到分开考虑这些条件还是有用的。

死锁处理方法一般来说有三种：

1. 通过协议来预防或避免死锁，确保系统不会进入死锁状态
2. 可以允许系统进入死锁状态，然后检测它，并加以恢复
3. 忽视这个问题，认为死锁不可能在系统内发生

第三种解决方案为大多数系统所采用，包括 Linux 和 Windows。

#### 知识延伸

下面着重讲解死锁的处理方法。

##### 死锁预防

发生死锁有四个必要条件，只要确保至少一个必要条件不成立，就能预防死锁。

* 互斥

互斥条件必须成立。可共享资源不要求互斥访问，因此不会参与死锁。只读文件是一个很好的共享资源例子。然鹅，通常不能通过否定互斥条件来预防死锁，因为有的资源本身就是非共享的。例如，一个互斥锁不能同时被多个进程所共享。

* 持有并等待

为了确保持有并等待条件不会出现在系统中，应保证：当每个进程申请一个资源时，它不能占有其他资源。一种可以采用的协议是，每个进程在执行前申请并获得所有资源。另外一种协议允许进程仅在没有资源时才可申请资源。一个进程可申请一些资源并使用它们。然而，在它申请更多其他资源之前，它应释放现已分配的所有资源。

这两种协议有两个主要缺点。第一，资源利用率可能比较低，因为许多资源可能已分配，但是很长时间没有被使用。第二，可能发生饥饿。

* 无抢占

为了确保这一条件不成立，可以采用如下协议：如果一个进程持有资源并申请另一个不能立即分配的资源，那么它现在持有的资源都可被抢占。换句话说，这些资源都被隐式释放了。

这个协议通常用于状态可以保存和恢复的资源，如 CPU 寄存器和内存。它一般不适用于其他资源，如互斥锁和信号量。

* 循环等待

确保这个条件不成立的一个方法是：对所有资源类型进行完全排序，而且要求每个进程按递增顺序来申请资源。

**死锁避免**

在前面讨论死锁预防算法中，通过限制如何申请资源来预防死锁。这种限制确保，至少有一个死锁的必要条件不会发生。然而，通过这种方法预防死锁有副作用：设备使用率低和系统吞吐量低。

避免死锁的另一种方法需要额外信息，即如何申请资源。每个进程都应声明可能需要的每种类型资源的最大数量。鉴于这个先验信息，有可能构造一个算法，以便确保系统不会进入死锁状态。死锁避免算法动态检查资源分配状态，以便确保循环等待条件不能成立。资源分配状态包括可用的资源、已分配的资源及进程的最大需求。

下面讨论两个死锁避免算法：

资源分配图算法：

如果有一个资源分配系统，它的每种资源类型只有一个实例。在资源分配图中，如果没有环存在，那么资源的分配就使得系统处于安全状态，如果有环存在，那么分配会导致系统处于非安全状态。

银行家算法：

对于每种资源类型有多个实例的资源分配系统，资源分配图算法就不适用了。

当一个新的进程进入系统时，它应声明可能需要的每种资源类型实例的最大数量，这一数量不能超过系统资源的总和。当用户申请一组资源时，系统应确定这些资源的分配是否仍然会使系统处于安全状态。如果会，就分配资源；否则，进程应等待，直到某个进程释放足够多资源为止。

**死锁检测与恢复**

如果所有资源类型只有单个实例，我们可以定义这样一个死锁检测算法，该算法使用了资源分配图的一个变形，称为等待图。当每种资源类型有多个实例时，可以使用类似银行家算法。

当检测算法确定已有死锁时，打破死锁有两个选择。一个是，简单地中止一个或多个进程来打破循环等待。另一个是，从一个或多个死锁进程那里抢占一个或多个资源。

**忽视死锁**

当发生死锁时，会导致系统性能下降，因为资源被不能运行的进程占有，而越来越多的进程会因申请资源而进入死锁。最后整个系统会停止工作，且需要人工重新启动。

虽然这看起来似乎不是一个解决死锁问题的可行方法，但是它却为大多数操作系统所采用。代价是一个重要的考虑因素。忽略死锁的可能性要比其他方法更便宜。对于许多系统，死锁很少发生（如一年一次），因为，与使用频繁的并且开销昂贵的死锁预防、死锁避免和死锁检测与恢复相比，这种方法更为便宜。