## 死锁

死锁的必要条件：

1. 互斥条件，资源要么分配给进程，要么是可用的

2. 占有和等待条件，进程、线程获取资源后还能继续获取资源

3. 不可抢占，已经分配的资源不能被强制性的抢占

4. 环路等待条件，一定有多个进程、线程形成环路，互相等待

1-3必要条件一定满足，在Java程序中，引发死锁大多是因为加锁的顺序不对，满足了第4个必要条件。

发生死锁的场景：

1. 对多个资源加锁顺序不一致导致死锁（明显）：

例如线程1先后对A/B资源加锁，而线程2先后对B/A资源加锁，当对某个资源的锁定时间较长或线程并发执行时很容易产生死锁，互相等待资源，形成环路等待。可以人为控制加锁的顺序，防止死锁。《Java并发编程实战》P172，的例子用了identityHashCode方式，解决了X向Y、Y向X同时发起转账可能引发的死锁。

2. 协作对象死锁（隐式）：

本质也是对象加锁顺序不同，例如在对象A的同步方法中调用对象B的同步方法。可以通过减少锁的粒度，将同步加在数据修改而不是方法上。

3. 资源死锁

上面两个场景都是锁定了资源本身，还有一种情况，不是锁定了资源，而是资源不足。例如线程1需要获取两个数据源的连接，获取了D1但D2资源不足，线程2反之。

另一种资源死锁是，例如线程池资源不足，导致后续请求都不能执行。饥饿死锁。

避免死锁：

打破任意一个必要条件，即可避免死锁。

Java程序中可以使用带有超时时间的获取锁的方法。tryLock(long, unit)。

诊断死锁：

使用jstack dump Java进程堆栈，检查是否有两个线程形成环路等待。