死锁与饥饿

**死锁的定义：**是指两个或两个以上的进程（或线程）在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

对死锁的理解比喻：三个鸡蛋和三份面粉做成一块蛋糕。现在总共有三个鸡蛋和三份面粉，甲现在有其中的三个鸡蛋，乙现在有其中的三份面粉，两个都需要对方的东西，但都不肯放手，这就产生了死锁。

死锁发生的四个条件  
1、互斥条件：线程对资源的访问是排他性的，如果一个线程对占用了某资源，那么其他线程必须处于等待状态，直到资源被释放。  
2、请求和保持条件：线程T1至少已经保持了一个资源R1占用,但又提出对另一个资源R2请求，而此时，资源R2被其他线程T2占用，于是该线程T1也必须等待，但又对自己保持的资源R1不释放。  
3、不剥夺条件：线程已获得的资源，在未使用完之前，不能被其他线程剥夺，只能在使用完以后由自己释放。  
4、环路等待条件：在死锁发生时，必然存在一个“进程-资源环形链”，即：{p0,p1,p2,...pn},进程p0（或线程）等待p1占用的资源，p1等待p2占用的资源，pn等待p0占用的资源。（最直观的理解是，p0等待p1占用的资源，而p1而在等待p0占用的资源，于是两个进程就相互等待）

饥饿的定义：是指如果线程T1占用了资源R，线程T2又请求封锁R，于是T2等待。T3也请求资源R，当T1释放了R上的封锁后，系统首先批准了T3的请求，T2仍然等待。然后T4又请求封锁R，当T3释放了R上的封锁之后，系统又批准了T4的请求......，T2可能永远等待。

对饥饿的理解比喻：在“首堵”北京的某一天，天气阴沉，空气中充斥着雾霾和地沟油的味道，某个苦逼的临时工交警正在处理塞车，有两条道A和B上都堵满了车辆，其中A道堵的时间最长，B相对相对堵的时间较短，这时，前面道路已疏通，交警按照最佳分配原则，示意B道上车辆先过，B道路上过了一辆又一辆，A道上排队时间最长的确没法通过，只能等B道上没有车辆通过的时候再等交警发指令让A道依次通过，这也就是ReentrantLock显示锁里提供的不公平锁机制（当然了，ReentrantLock也提供了公平锁的机制，由用户根据具体的使用场景而决定到底使用哪种锁策略），不公平锁能够提高吞吐量但不可避免的会造成某些线程的饥饿。