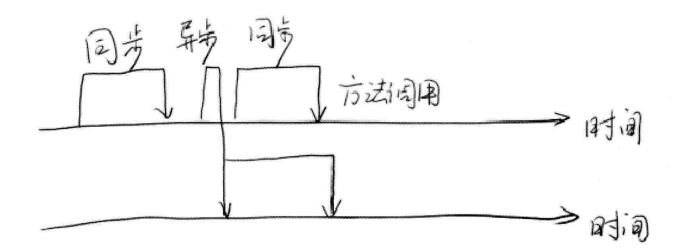
### 一 并发基本概念

#### 同步与异步

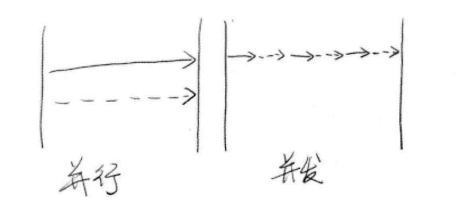
同步与异步用来形容一次方法调用。同步调用必须等方法调用返回以后，才能继续调用。异步调用更像是一个消息传递，一旦开始方法便立即返回，调用者可以继续完成后面相关的调用。此时异步方法就在另一个线程中真实的执行。



如上图所示是同步调用与异步调用的区别，同步调用只能是在调用完成以后才能进行后续的工作，然而异步调用则是只要触发了异步调用开始之后就可以继续完成后面的工作。此时会有一个并发的线程与主线程一起在工作。

形象点说就是，形象的说就是做饭的时候，你是一直在等着饭煮熟，再洗菜，还是在煮饭的时候，去洗菜的问题。无非就是我顺序的去执行，还是同时执行。顺序的去执行就类似于同步。同时去执行就是异步。效率大家可想而知。

#### 并发与并行



如上图所示并发与并行实际概念上是有差异的。并发强调的是时间片的轮换，交替执行。然而在时间片轮换过程中对于使用程序的用户来说错觉上就感觉是同时运行的。并行则不一样是真实的同时运行。由于他们的结果是一样的所以我们混淆的把并发与并行统一归类为并发。实际上如果是单核CPU的话，并做不到真正的并行。因为只有一个处理器，只能靠CPU的时间片的切换来并发执行，以给用户一种真正意义上的并行。

行象点说就像你爬山的时候，爬一会又看一会儿风景。然而如果你做缆车的话，就可以用缆车的上行来代替你上山的过程，你就可以一直看风景。你爬山就像并发执行，边爬边看。你坐缆车上山就是并行执行的。这个效率，自然明白。

#### 临界区

表示一种公共资源或者共享数据，可以被多个线程使用。但每一次只能有一个线程使用它。一旦临界区被占用，其他线程要想使用这个资源就必须等待。在并行程序中，临界区是保护的对象，比如一个会议室不能同时提供给两个开发团队开会一样，只能轮流开会，这就是临界资源的概念。

#### 1.4阻塞与非阻塞

阻塞与非阻塞通常形容的是多线程之间的相互影响。比如一个线程占用了临界区资源，那么其他需要这个资源的线程就必须在临界区等待。等待会导致线程挂起，这种情况就是阻塞。此时如果占用资源的线程不释放资源，那么其他的线程就不能执行。

非阻塞意思恰恰相反，它强调的是没有一个线程能够阻碍其他线程的执行，所以线程都会尝试不断的向前执行。

#### 1.5死锁，饥饿与活锁

饥饿是指某一个或多个线程因为种种原因无法获得所需资源。导致一直无法执行的情况。比如某个线程的优先级比较低，或者说是某个线程一直占着核心资源不放的情况。但是相比于死锁来说，活锁是在未来某一段时间内可以解决的。

活锁就是指线程碰撞，主动让出资源给别人，导致两个线程不断的碰撞。形象的说就好比我们出电梯门的时候总有那么一种情况两个人互相让，两个人同时让右结果撞，又同时让左结果又碰上了，这个时候人的话两到三次以后就解决了。但是计算机的线程就没那么幸运了，可能会不断的尝试造成短暂的活锁，可是可以解决的。

死锁，相比于饥饿与活锁，死锁的危害性似乎会是更严重的。我们说就像爱情里面的三角恋。你爱我，我爱她，她爱你。造成了你，我，她三个线程互相等待，形成了环。当等待形成了环的时候我们就说这个程序已经死锁了，因为没发玩了。要解决这个问题唯一的办法就是其中一个线程主动让出资源。

#### 1.6 并发的级别

并发的级别大致分类可以分为阻塞，无饥饿，无障碍，无锁，无等待几种。

##### 1.6.1 阻塞

一个线程是阻塞的，那么在其他线程释放资源之前，当前线程是无法继续执行的。当我们使用synchronized，或可重入锁的时候，在执行后面的代码的时候，都会试图得到临界区资源。如果得不到，线程就会挂起等待，直到占有了所需资源为止。

##### 1.6.2 无饥饿

如果线程之间是有优先级的，那么线程调度的时候总是倾向于高优先级的线程，也就是说对于同一个资源的分配是不共平的。对于非共平锁来说因为是有优先级的，导致线程之间会有饥饿产生，对于非公平锁来说，线程之前是没有优先级的。先来先执行，所以线程直接就不会产饥饿的情况。

##### 1.6.3 无障碍

就是多个线程可以同时对共享资源进行修改，然而如果多个线程把数据改坏了的话，就做线程资源的回滚，相比重量级锁来说，这是一种乐观的操作，类似于我们框架的hibernate中的version操作。所有线程一起改，在这个线程开始改的时候通过设置共享资源的版本，等线程改完之后，再拿着修改前的共享资源版本去对比，如果是同一个版本。那么就允许修改，如果不是同一个版本，那么就不允许修改，做线程与共享资源的回滚操作。

##### 1.6.4 无锁

也是一种无障碍的并发执行模式，多个线程一起竞争修改资源，不断的循环去尝试，总有一个线程可以从中胜出。不至于全军覆灭。至于临界区竞争失败的资源，总是尝试不成功则会一直处于饥饿状态。



典型的例子就是我们的比较设值操作。

##### 1.6.5 无等待

无等待是在无锁的基础上更胜了一步，也就是无锁是只要其中一个线程能在有限的步数里面完成就可以了。但是无等待的话是所有的线程都在有限的步数里面完成。