高并发程序设计（2）—— 线程操作基础篇

作者：wanglei0622
原文链接：https://ld246.com/article/1488438707246
来源网站：链滴

<h2>新建线程</h2>

Thread：继承Thread，重载run()方法，调用start(),或Thread t1 = new Thread() ..run(){...} } 匿名内部类来新建
Runnable：JAVA单继承，继承本身是一种宝贵资源，推荐用接口来实现，此外Thread类有一个常重要的构造方法，形参就是Runnable,默认的Thread.run()就是直接调用的Runnable.run()方法。此使用Runnable实例告诉Thread需要做什么更合理。

<h2>终止线程</h2>

Thread.stop()方法：直接终止线程，并立即释放这个线程持有的锁，如果这些锁恰好是为了维持象一致性的话，就会发生，数据写到一半，被强行终止，然后别的线程读到错误的数据。

<h2>中断线程</h2>

Thread.interrupt():并不会使线程立即退出，只是设置了一个中断标志，用isInterrupted()判断否被终端，或者interrupted()判断，后者同时会清除中断标志状态。
线程的退出，需要在线程开始的代码里，加入中断标志判断代码，如果是设置了中断标志，就使程结束。
wait(),sleep(),join()：会抛出中断异常并清除中断标志，这个异常是一般异常，不同于运行时异，必须捕获处理。而且为了保证线程等待位置的后续代码正确执行，应在catch中捕获异常，再次设置中断，然后在线程开始的代码里判断标志位，结束线程。

<h2>等待(wait)和通知(notify)</h2>

这两个方法是JDK为支持多线程协作提供的，这两个方法是属于Object类的，而不是Thread，任对象都可以调用。
Object.wait():线程中当一个对象实例调用了wait()，当前线程就会在wait()这个位置等待，直到他线程调用了这个对象的notify()方法。

每当一个线程调用对象的object.wait()方法，它就会进入object的等待队列，这个队列里可能有个线程在等待，object.notify()只能唤起随机的一个线程。
object.notifyAll()可以唤醒队列里所有的线程。
wait()方法不能随便调用，必须包含在对应的synchronzied语句中，无论wait还是notify都要先得目标对象的同步锁，wait()方法执行后会立即释放同步锁。

调用object.wait()方法的线程被唤醒时，还需等线程拿到object的对象锁时才能继续行。

<h2>挂起(suspend)和继续执行(resume)</h2>

这两个方法是属于Thread。
Thread.suspend():使线程暂停的同时，并不释放任何资源，任何想要访问被他占用的锁时，都被牵连，而且线程还是RUNNABLE状态，严重影响我们对当前系统状态的判断。

<h2>等待线程结束(join)和谦让(yield)</h2>

Thread.join()：t1.join()时阻塞当前线程，直到t1实例执行结束。

本质是当t1线程未结束时，循环调用t1对象的wait(0)方法，当t1执行结束时，wait(0)方法最后会用notifyAll()，因此要注意不要在Thread对象上使用wait或notify方法，可能会影响其他代。

Thread.yield()：当觉得一个线程不重要，怕占用太多资源时，调用，但是并不代表调用后就不

执行了，cpu资源会重新争抢，是否不会被分配到也不一定。
