链滴

JAVA 中线程的 join sleep wait yield 区别

作者：hymn

原文链接：https://ld246.com/article/1594208368479

来源网站：链滴

线程在一定条件下，状态会发生变化。线程一共有以下几种状态：

1. 新建状态(New)：新创建了一个线程对象。

2. 就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池”中，变得可运行，只等待获取CPU的使用权。即在就绪状态的进程除CPU之外，其它运行所需资源都已全部获得。

3. 运行状态(Running)：就绪状态的线程获取了CPU，执行程序代码。

4. 阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进就绪状态，才有机会转到运行状态。

阻塞的情况分三种：

(1)、等待阻塞：运行的线程执行wait()方法，该线程会释放占用的所有资源，JVM会把该线程放入“待池”中。进入这个状态后，是不能自动唤醒的，必须依靠其他线程调用notify()或notifyAll()方法才被唤醒，

(2)、同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程入“锁池”中。

(3)、其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪态。

5. 死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

线程变化的状态转换图如下：

注：拿到对象的锁标记，即为获得了对该对象(临界区)的使用权限。即该线程获得了运行所需的资源进入“就绪状态”，只需获得CPU，就可以运行。因为当调用wait()后，线程会释放掉它所占有的“标志”，所以线程只有在此获取资源才能进入就绪状态。

下面小小的作下解释：

1. 线程的实现有两种方式，一是继承Thread类，二是实现Runnable接口，但不管怎样， 当我们ne了这个对象后，线程就进入了初始状态；

2. 当该对象调用了start()方法，就进入就绪状态；

3. 进入就绪后，当该对象被操作系统选中，获得CPU时间片就会进入运行状态；

4. 进入运行状态后情况就比较复杂了

4.1、run()方法或main()方法结束后，线程就进入终止状态；

4.2、当线程调用了自身的sleep()方法或其他线程的join()方法，进程让出CPU，然后就会进入阻塞状（该状态既停止当前线程，但并不释放所占有的资源即调用sleep ()函数后，线程不会释放它的“锁志”。）。当sleep()结束或join()结束后，该线程进入可运行状态，继续等待OS分配CPU时间片。典地，sleep()被用在等待某个资源就绪的情形：测试发现条件不满足后，让线程阻塞一段时间后重新测，直到条件满足为止。

4.3、线程调用了yield()方法，意思是放弃当前获得的CPU时间片，回到就绪状态，这时与其他进程于同等竞争状态，OS有可能会接着又让这个进程进入运行状态； 调用 yield() 的效果等价于调度程序为该线程已执行了足够的时间片从而需要转到另一个线程。yield()只是使当前线程重新回到可执行状，所以执行yield()的线程有可能在进入到可执行状态后马上又被执行。

4.4、当线程刚进入可运行状态（注意，还没运行），发现将要调用的资源被synchroniza（同步），取不到锁标记，将会立即进入锁池状态，等待获取锁标记（这时的锁池里也许已经有了其他线程在等

获取锁标记，这时它们处于队列状态，既先到先得），一旦线程获得锁标记后，就转入就绪状态，等OS分配CPU时间片；

4.5. suspend() 和 resume()方法：两个方法配套使用，suspend()使得线程进入阻塞状态，并且不会动恢复，必须其对应的resume()被调用，才能使得线程重新进入可执行状态。典型地，suspend()和 resume() 被用在等待另一个线程产生的结果的情形：测试发现结果还没有产生后，让线程阻塞，另一线程产生了结果后，调用 resume()使其恢复。

4.6、wait()和 notify() 方法：当线程调用wait()方法后会进入等待队列（进入这个状态会释放所占有所有资源，与阻塞状态不同），进入这个状态后，是不能自动唤醒的，必须依靠其他线程调用notify()或notifyAll()方法才能被唤醒（由于notify()只是唤醒一个线程，但我们由不能确定具体唤醒的是哪一线程，也许我们需要唤醒的线程不能够被唤醒，因此在实际使用时，一般都用notifyAll()方法，唤醒所线程），线程被唤醒后会进入锁池，等待获取锁标记。

wait() 使得线程进入阻塞状态，它有两种形式：

一种允许指定以毫秒为单位的一段时间作为参数；另一种没有参数。前者当对应的 notify()被调用或超出指定时间时线程重新进入可执行状态即就绪状态，后者则必须对应的 notify()被调用。当调用wait()后，线程会释放掉它所占有的“锁标志”，从而使线程所在对象中的其它synchronized数据可被别线程使用。waite()和notify()因为会对对象的“锁标志”进行操作，所以它们必须在synchronized函或synchronizedblock中进行调用。如果在non-synchronized函数或non-synchronizedblock中进调用，虽然能编译通过，但在运行时会发生IllegalMonitorStateException的异常。

参考