# **[Java线程的5种状态及切换](https://www.cnblogs.com/hejing-swust/p/8038263.html)**

<https://www.cnblogs.com/hejing-swust/p/8038263.html>

java中的线程的生命周期大体可分为5种状态。

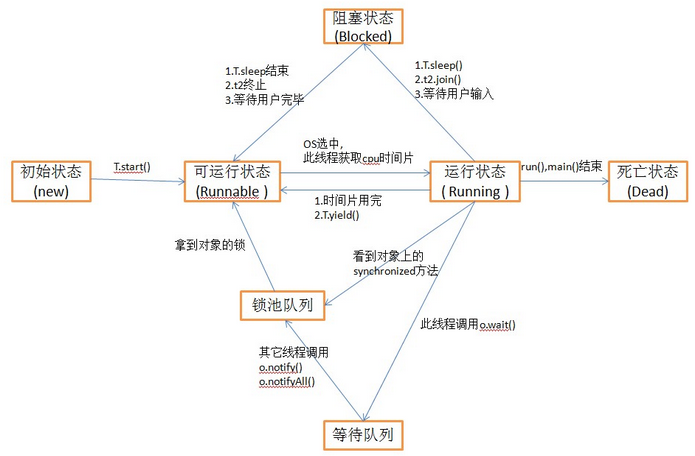
1. **新建(NEW)**：新创建了一个线程对象。

2. **可运行(RUNNABLE)**：线程对象创建后，其他线程(比如main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，等待被线程调度选中，获取cpu 的使用权 。

3. **运行(RUNNING)**：可运行状态(runnable)的线程获得了cpu 时间片（timeslice） ，执行程序代码。  
4. **阻塞(BLOCKED)**：阻塞状态是指线程因为某种原因放弃了cpu 使用权，也即让出了cpu timeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有机会再次获得cpu timeslice 转到运行(running)状态。阻塞的情况分三种：

(一). 等待阻塞：运行(running)的线程执行o.wait()方法，JVM会把该线程放入等待队列(waitting queue)中。  
(二). 同步阻塞：运行(running)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会把该线程放入锁池(lock pool)中。  
(三). 其他阻塞：运行(running)的线程执行Thread.sleep(long ms)或t.join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。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，线程重新转入可运行(runnable)状态。

5. **死亡(DEAD)**：线程run()、main() 方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。



**二.初始状态**

1. 实现Runnable接口和继承Thread可以得到一个线程类，new一个实例出来，线程就进入了初始状态

**三.可运行状态**

1. 可运行状态只是说你资格运行，调度程序没有挑选到你，你就永远是可运行状态。
2. 调用线程的start()方法，此线程进入可运行状态。
3. 当前线程sleep()方法结束，其他线程join()结束，等待用户输入完毕，某个线程拿到对象锁，这些线程也将进入可运行状态。
4. 当前线程时间片用完了，调用当前线程的yield()方法，当前线程进入可运行状态。
5. 锁池里的线程拿到对象锁后，进入可运行状态。

**四.运行状态**

1. 线程调度程序从可运行池中选择一个线程作为当前线程时线程所处的状态。这也是线程进入运行状态的唯一一种方式。

**五.死亡状态**

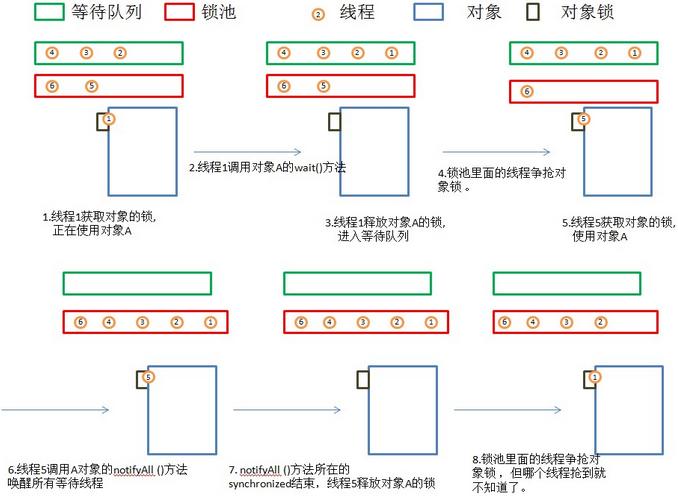
1. 当线程的run()方法完成时，或者主线程的main()方法完成时，我们就认为它死去。这个线程对象也许是活的，但是，它已经不是一个单独执行的线程。线程一旦死亡，就不能复生。
2. 在一个死去的线程上调用start()方法，会抛出java.lang.IllegalThreadStateException异常。

**六.阻塞状态**

1. 当前线程T调用Thread.sleep()方法，当前线程进入阻塞状态。
2. 运行在当前线程里的其它线程t2调用join()方法，当前线程进入阻塞状态。
3. 等待用户输入的时候，当前线程进入阻塞状态。

**七.等待队列(本是Object里的方法，但影响了线程)**

1. 调用obj的wait(), notify()方法前，必须获得obj锁，也就是必须写在synchronized(obj) 代码段内。
2. 与等待队列相关的步骤和图



**八.锁池状态**

1. 当前线程想调用对象A的同步方法时，发现对象A的锁被别的线程占有，此时当前线程进入锁池状态。简言之，锁池里面放的都是想争夺对象锁的线程。
2. 当一个线程1被另外一个线程2唤醒时，1线程进入锁池状态，去争夺对象锁。
3. 锁池是在同步的环境下才有的概念，一个对象对应一个锁池。

**九.几个方法的比较**

1. Thread.sleep(long millis)，一定是当前线程调用此方法，当前线程进入阻塞，但不释放对象锁，millis后线程自动苏醒进入可运行状态。作用：给其它线程执行机会的最佳方式。
2. Thread.yield()，一定是当前线程调用此方法，当前线程放弃获取的cpu时间片，由运行状态变会可运行状态，让OS再次选择线程。作用：让相同优先级的线程轮流执行，但并不保证一定会轮流执行。实际中无法保证yield()达到让步目的，因为让步的线程还有可能被线程调度程序再次选中。Thread.yield()不会导致阻塞。
3. t.join()/t.join(long millis)，当前线程里调用其它线程1的join方法，当前线程阻塞，但不释放对象锁，直到线程1执行完毕或者millis时间到，当前线程进入可运行状态。
4. obj.wait()，当前线程调用对象的wait()方法，当前线程释放对象锁，进入等待队列。依靠notify()/notifyAll()唤醒或者wait(long timeout)timeout时间到自动唤醒。
5. obj.notify()唤醒在此对象监视器上等待的单个线程，选择是任意性的。notifyAll()唤醒在此对象监视器上等待的所有线程。

来自：http://blog.csdn.net/pange1991/article/details/53860651

路漫漫其修远兮，吾将上下而求索！