# Java线程

## 为什么要有线程

早期的操作系统只有进程的概念,但还没有线程出现时，进程是系统调度的基本单位。但后来随着资源调度和任务调度的频繁，因为进程调度任务所需的资源是由进程自己管理的，所以导致切换、销毁和创建进程的代价是及其大的，加上并发处理上进程并不适合相互交换数据，它是重量级的。线程的产生，极大的改善了这种状态，线程不负责管理资源，自己只有少许需要维持自己运行的资源，进程下可以建立多个线程来并发运行任务，线程之间共享进程的资源，现在，线程是才是系统调度的基本单位，切换、销毁和创建线程的开销远小于进程。

## 如何创建线程

Java中，创建线程有两种方法，一种是继承java.lang.**Thread**类，该类实现了Runnable方法。另一种就是实现java.lang.**Runnable**接口。Runnable接口只有一个方法，就是void run()。

## 线程状态

在JVM中，线程有如下几种状态(用枚举Thread.State来表示)：

[**NEW**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#NEW)**（新生）**：至今尚未启动的线程处于这种状态。

[**RUNNABLE**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#RUNNABLE)**（可运行）**：正在 Java 虚拟机中执行的线程处于这种状态。

[**BLOCKED**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#BLOCKED)**（阻塞）**：受阻塞并等待某个监视器锁的线程处于这种状态。

[**WAITING**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#WAITING)**（等待）**：无限期地等待另一个线程来执行某一特定操作的线程处于这种状态。

[**TIMED\_WAITING**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#TIMED_WAITING)**（超时等待）**：等待另一个线程来执行取决于指定等待时间的操作的线程处于这种状态。

[**TERMINATED**](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.State.html#TERMINATED)**（销毁）**：已退出的线程处于这种状态。

任何时刻，虚拟机中的线程只能处于以上之一的某种状态。这是虚拟机的状态，它不一定反应了所有操作系统中线程的状态。

## 状态切换

当我们new一个线程时，它就处于了NEW新生状态。一旦线程调用start()方法，它变成了可运行的RUNNABLE状态了。有以下几种情况会使线程阻塞，比如调用Thread的静态方法sleep(long millis)来主动让当前正在执行的线程休眠（暂停执行），该方法不会使当前线程放弃锁；或者线程调用I/O操作时被阻塞了；或者线程正在试图获得被其他线程正占用的锁；或者线程正在等待某个触发条件。当如下几种情况发生，被阻塞的进程可以变成可运行状态，比如进程sleep过了指定的时间；比如线程等待的I/O操作已经完成；或者该线程等待的锁已经被其他线程释放；或者该线程等待的触发条件已经完成，且另外一个线程已经发出了信号。

## 中断线程

public void **interrupt()：**中断线程，当该方法在一个线程上被调用时，该线程的**中断状态(intertupted status)**将会被置位，这是一个布尔型的标志，存在于每一个线程之中。 如果当前线程没有中断它自己（这在任何情况下都是允许的），则该线程的 [checkAccess](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#checkAccess()) 方法就会被调用，这可能抛出 [SecurityException](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/SecurityException.html)。 如果线程在调用 [Object](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) 类的 [wait()](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html#wait())、[wait(long)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html#wait(long)) 或 [wait(long, int)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html#wait(long, int)) 方法，或者该类的 [join()](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#join())、[join(long)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#join(long))、[join(long, int)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#join(long, int))、[sleep(long)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#sleep(long)) 或 [sleep(long, int)](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/Thread.html#sleep(long, int)) 方法过程中受阻，则其中断状态将被清除，它还将收到一个 [InterruptedException](mk:@MSITStore:E:\编程相关\API\JDK_API_1_6_zh_CN.CHM::/java/lang/InterruptedException.html)。

public static boolean **interrupted()：**静态方法，测试当前线程是否已经中断。调用该方法将清除当前线程的**中断状态**标记。换句话说，如果连续两次调用该方法，则第二次调用将返回 false（在第一次调用已清除了其中断状态之后，且第二次调用检验完中断状态前，当前线程再次中断的情况除外）。如果在中断时不处于活动状态的线程将由此返回 false表示中断被忽略。

public boolean **isInterrupted()**：测试线程是否已经中断。该方法不会修改线程的**中断状态**标记。

**InterruptedException:** 当线程在活动之前或活动期间处于正在等待、休眠或占用状态且该线程被中断时，抛出该异常。有时候，一种方法可能希望测试当前线程是否已被中断，如果已被中断，则立即抛出此异常。