[interrupt、interrupted 、isInterrupted 区别](http://blog.csdn.net/z69183787/article/details/25076033)

IMG_256 分类：

多线程（52） IMG_257

1、interrupt

interrupt方法用于中断线程。调用该方法的线程的状态为将被置为"中断"状态。

注意：线程中断仅仅是置线程的中断状态位，不会停止线程。需要用户自己去监视线程的状态为并做处理。支持线程中断的方法（也就是线程中断后会抛出interruptedException的方法）就是在监视线程的中断状态，一旦线程的中断状态被置为“中断状态”，就会抛出中断异常。

**一般来说，支持线程中断的方法（也就是线程中断后会抛出InterruptedException的方法），阻塞函数，如：Thread.sleep、Thread.join、Object.wait等在检查到线程的中断状态时，会抛出InterruptedException，同时会清除线程的中断状态**

2、interrupted 和 isInterrupted

首先看一下该方法的实现：

|  |  |
| --- | --- |
| 1 | **public** **static** **boolean** interrupted () { |

|  |  |
| --- | --- |
| 2 | **return** currentThread().isInterrupted(**true**); |

|  |  |
| --- | --- |
| 3 | } |

该方法就是直接调用当前线程的isInterrupted(true)方法。

然后再来看一下 isInterrupted的实现：

|  |  |
| --- | --- |
| 1 | **public** **boolean** isInterrupted () { |

|  |  |
| --- | --- |
| 2 | **return** isInterrupted( **false**); |

|  |  |
| --- | --- |
| 3 | } |

这两个方法有两个主要区别：

**interrupted ：是作用于当前线程，返回线程的上次的中断状态，并清除中断状态。**isInterrupted ：是作用于调用该方法的线程对象所对应的线程。（线程对象对应的线程不一定是当前运行的线程。例如我们可以在A线程中去调用B线程对象的isInterrupted方法。

1. 这两个方法最终都会调用同一个方法，只不过参数一个是true，一个是false；

第二个区别主要体现在调用的方法的参数上，让我们来看一看这个参数是什么含义

先来看一看被调用的方法 isInterrupted(boolean arg)的定义：

|  |  |
| --- | --- |
| 1 | **private** **native** **boolean** isInterrupted( **boolean** ClearInterrupted); |

原来这是一个本地方法，看不到源码。不过没关系，通过参数名我们就能知道，这个参数代表是否要清除状态位。

如果这个参数为true，说明返回线程的状态位后，要清掉原来的状态位（恢复成原来情况）。这个参数为false，就是直接返回线程的状态位。

这两个方法很好区分，只有当前线程才能清除自己的中断位（对应interrupted（）方法）

**interrupted**()

**interrupted**()

测试当前线程是否已经中断。线程的中断状态由该方法清除。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

而调用isInterrupted 只是简单的查询中断状态，不会对状态进行修改。

interrupt（）是用来设置中断状态的。返回true说明中断状态被设置了而不是被清除了。我们调用sleep、wait等此类可中断（throw InterruptedException）方法时，一旦方法抛出InterruptedException，当前调用该方法的线程的中断状态就会被jvm自动清除了，就是说我们调用该线程的isInterrupted 方法时是返回false。如果你想保持中断状态，可以再次调用interrupt方法设置中断状态。这样做的原因是，java的中断并不是真正的中断线程，而只设置标志位（中断位）来通知用户**。如果你捕获到中断异常，说明当前线程已经被中断，不需要继续保持中断位。**

interrupted是静态方法，返回的是当前线程的中断状态。例如，如果当前线程被中断，你调用interrupted方法，第一次会返回true。然后，当前线程的中断状态被方法内部清除了。第二次调用时就会返回false。如果你刚开始一直调用isInterrupted，则会一直返回true，除非中间线程的中断状态被其他操作清除了。