结束线程的方式

一、使用退出标志终止线程

```
public class ThreadSafe extends Thread {
    public volatile boolean exit = false;
        public void run() {
        while (!exit){
            //do something
        }
    }
}
```

二、使用interrupt()方法中断当前线程

为什么要区分**进入阻塞状态**和和**非阻塞状态**两种情况了，是因为当阻塞状态时，如果有interrupt()发生，系统除了会抛出InterruptedException异常外，还会调用interrupted()函数，调用时能获取到中断状态是true的状态，调用完之后会复位中断状态为false，所以异常抛出之后通过isInterrupted()是获取不到中断状态是true的状态，从而不能退出循环，因此在线程未进入阻塞的代码段时是可以通过isInterrupted()来判断中断是否发生来控制循环，在进入阻塞状态后要通过捕获异常来退出循环。因此使用interrupt()来退出线程的最好的方式应该是两种情况都要考虑：

```
public class ThreadSafe extends Thread {
    public void run() {
        while (!isInterrupted()){ //非阻塞过程中通过判断中断标志来退出
            try{
                Thread.sleep(5*1000);//阻塞过程捕获中断异常来退出
            }catch(InterruptedException e){
                e.printStackTrace();
                break;//捕获到异常之后，执行break跳出循环。
            }
        }
    }
}
```

使用interrupt()方法来中断线程有两种情况：

- 1.线程处于阻塞状态，

- 如使用了sleep,同步锁的wait,socket中的receiver,accept等方法时，会使线程处于阻塞状态。当调用线程的interrupt()方法时，会抛出InterruptException异常。阻塞中的那个方法抛出这个异常，通过代码捕获该异常，然后break跳出循环状态，

从而让我们有机会结束这个线程的执行。通常很多人认为只要调用interrupt方法线程就会结束，实际上是错的， 一定要先捕获InterruptedException异常之后通过break来跳出循环，才能正常结束run方法。

```
public class ThreadSafe extends Thread {
    public void run() {
        while (true){
            try{
                Thread.sleep(5*1000);//阻塞5妙
            }catch(InterruptedException e){
                e.printStackTrace();
                break;//捕获到异常之后，执行break跳出循环。
            }
        }
    }
}
```

2.线程未处于阻塞状态

使用isInterrupted()判断线程的中断标志来退出循环。当使用interrupt()方法时，中断标志就会置true，和使用自定义的标志来控制循环是一样的道理。

```
public class ThreadSafe extends Thread {
    public void run() {
        while (!isInterrupted()){
            //do something, but no throw InterruptedException
        }
    }
}
```

三、使用stop方法终止线程

程序中可以直接使用thread.stop()来强行终止线程，但是stop方法是很危险的，就象突然关闭计算机电源，而不是按正常程序关机一样，可能会产生不可预料的结果，不安全主要是：thread.stop()调用之后，创建子线程的线程就会抛出ThreadDeatherror的错误，并且会释放子线程所持有的所有锁。一般任何进行加锁的代码块，都是为了保护数据的一致性，如果在调用thread.stop()后导致了该线程所持有的所有锁的突然释放(不可控制)，那么被保护数据就有可能呈现不一致性，其他线程在使用这些被破坏的数据时，有可能导致一些很奇怪的应用程序错误。因此，并不推荐使用stop方法来终止线程。