



链滴

# 【并发编程】怎么样让 Java 里的线程安全 停止工作 (interrupt())

作者: [moonce](#)

原文链接: <https://ld246.com/article/1541573383336>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

# 怎么样让Java里的线程安全停止工作

作者:Moonce

博客:望舒阁

## 方法

- sleep()
- suspend()和resume()
- yield()
- stop()
- interrupt() - 推荐使用

## 实现

suspend()和resume() 但是这两个方法因为有缺点(操作不当独占\不同步)而已经被弃用;

yield()方法的作用是放弃当前的CPU资源, 将它让给其他任务去占用CPU执行时间。但放弃的时间不定, 有可能刚刚放弃, 马上又获得CPU时间片。

虽然stop()可以停止一个线程, 但是这个方法是不安全的, 已经被弃用。

## interrupt()实现

调用一个线程的interrupt() 方法中断一个线程, 并不是强行关闭这个线程, 只是跟这个线程打个招呼将线程的中断标志位置为true, 线程是否中断, 由线程本身决定。

isInterrupted() 判定当前线程是否处于中断状态。

static方法interrupted() 判定当前线程是否处于中断状态, 同时中断标志位改为false。

方法里如果抛出InterruptedException, 线程的中断标志位会被复位成false, 如果确实是需要中断线, 要求我们自己在catch语句块里再次调用interrupt()。

```
package com.moonce.thread;

import java.util.concurrent.ExecutionException;

/**
 * @author moonce
 *
 * 类说明: 守护线程的使用和守护线程中的finally语句块
 */
public class InterruptThread {

    private static class UseThread extends Thread {
        @Override
        public void run() {
            try {
                while (!isInterrupted()) {
                    System.out.println(Thread.currentThread().getName())
                }
            }
        }
    }
}
```

```

        + " I am extends Thread.");
    }
    System.out.println(Thread.currentThread().getName()
        + " interrupt flag is " + isInterrupted());
} finally {
    System.out.println(Thread.currentThread().getName()
        + " finally");
}
}
}

public static void main(String[] args) throws InterruptedException,
    ExecutionException {
    UseThread useThread = new UseThread();
//    useThread.setDaemon(true); //随主线程一起结束
    useThread.start();
    Thread.sleep(5);
    useThread.interrupt();//协商终端
}
}

```

## sleep()实现

在Java多线程中，可以使用sleep()方法在指定毫秒数内让当前正在执行的线程休眠。

下面这段代码，使得主函数的main线程休眠了1000ms。

```

public static void main(String[] args) {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
}

```