

Thread 的 join() 源码解读

作者: [wanglei0622](#)

原文链接: <https://ld246.com/article/1461813593628>

来源网站: [链滴](#)

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

<h2>join()使用场景</h2>

<p>当程序希望各个线程执行完后，将他们的计算结果最终合并运算时，换句话说要等待多个线程将任务执行完后，才能进行合并结果操作，这时候就可以使用join()。</p>

<h2>源码</h2>

```

/** 
 * Waits at most {@code millis} milliseconds for this thread to
 * die. A timeout of {@code 0} means to wait forever.
 *
 * &lt;p&gt; This implementation uses a loop of {@code this.wait} calls
 * conditioned on {@code this.isAlive}. As a thread terminates the
 * {@code this.notifyAll} method is invoked. It is recommended that
 * applications not use {@code wait}, {@code notify}, or
 * {@code notifyAll} on {@code Thread} instances.
 *
 * @param millis
 *        the time to wait in milliseconds
 *
 * @throws IllegalArgumentException
 *        if the value of {@code millis} is negative
 *
 * @throws InterruptedException
 *        if any thread has interrupted the current thread. The
 *        &lt;i&gt;interrupted status&lt;/i&gt; of the current thread is
 *        cleared when this exception is thrown.
 */
public final synchronized void join(long millis)
throws InterruptedException {
    long base = System.currentTimeMillis();
    long now = 0;

    if (millis < 0) {
        throw new IllegalArgumentException("timeout value is negative");
    }

    if (millis == 0) {
        while (isAlive()) {
            wait(0);
        }
    } else {
        while (isAlive()) {
            long delay = millis - now;
            if (delay <= 0) {
                break;
            }
            wait(delay);
            now = System.currentTimeMillis() - base;
        }
    }
}<br /><br /></pre>
```

<p>有同学疑问，调用join后到底是哪个线程阻塞了？其实很好理解，就是当前执行的线程阻塞，而不调用join的线程！</p>

```
<pre class="brush: java ">public static void main(String[] args) throws InterruptedException {  
    final Thread t = new Thread(new Runnable() {  
        public void run() {  
            try {  
                System.out.println("t线程开始"+System.currentTimeMillis());  
                Thread.sleep(2000);  
                System.out.println("t线程结束"+System.currentTimeMillis());  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    });  
});
```

```
Thread t2 = new Thread(new Runnable() {
    public void run() {
        try {
            t.start();
            t.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("t2线程结束"+System.currentTimeMillis());
    }
});
t2.start();</pre>
```

```
<p>&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; Thread.currentThread().sleep(1000);<br />&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; System.out.println(t2.getState());</p>
<pre class="brush: java">    System.out.println("t2线程开始"+System.currentTimeMillis());
```

```
}
```

```
<p><br />将上面的测试代码里的join方法，用join的源码替换就一目了然了如下</p>
<pre class="brush: java">public static void main(String[] args) throws InterruptedException {
    final Thread t = new Thread(new Runnable() {
        public void run() {
            try {
                System.out.println("t线程开始"+System.currentTimeMillis());
                Thread.sleep(2000);
                System.out.println("t线程结束"+System.currentTimeMillis());
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    });
}
```

```
Thread t2 = new Thread(new Runnable() {  
    public void run() {  
        try {
```

```
t.start();
//t.join();
//join源码开始
long millis = 0;
long base = System.currentTimeMillis();
long now = 0;
if (millis < 0) {
    throw new IllegalArgumentException("timeout value is negative");
}

if (millis == 0) {
    while (t.isAlive()) {
        //wait就是对t2线程的操作
        t.wait(0);
    }
} else {
    while (t.isAlive()) {
        long delay = millis - now;
        if (delay <= 0) {
            break;
        }
        t.wait(delay);
        now = System.currentTimeMillis() - base;
    }
}
} catch (InterruptedException e) {
    e.printStackTrace();
}//join源码结束
System.out.println("t2线程结束"+System.currentTimeMillis());
}
});
t2.start();
System.out.println("t2线程开始"+System.currentTimeMillis());
```

}

</pre>

<p> </p>