

Java 浅析Thread.join()


概要

本文分为三部分对 Thread.join() 进行分析：


- 1. join() 的示例和作用
- 2. join() 源码分析
- 3. 对网上其他分析 join() 的文章提出疑问

1. join() 的示例和作用


1.1 示例

// 主线程

```
public class Parent extends Thread {  
    public void run() {  
        Child child = new Child();  
        child .start();  
        child .join();  
        // ...  
    }  
}
```

// 子线程

```
public class Child extends Thread {  
    public void run() {  
        // ...  
    }  
}
```



上面代码中有两个类：Parent（主线程类），Child（子线程类）。

在 Parent.run() 中，通过 new Child() 新建 child 子线程（此时 child 处于 NEW 状态），然后调用 child.start()（child 转换为 RUNNABLE 状态），再调用 child.join()。

在 Parent 调用 child.join() 后，child 子线程正常运行，Parant 主线程会等待 child 子线程结束后再继续运行。

1.2 join() 的作用

让主线程等待子线程结束之后才能继续运行。

我们来看看在 Java 7 Concurrency Cookbook 中相关的描述（很清楚地说明了 join() 的作用）：

Waiting for the finalization of a thread

In some situations, we will have to wait for the finalization of a thread. For example, we may have a program that will begin initializing the resources it needs before proceeding with the rest of the execution. We can run the initialization tasks as threads and wait for its finalization before continuing with the rest of the program. For this purpose, we can use the join() method of the Thread class. **When we call this method using a thread object, it suspends the execution of the calling thread until the object called finishes its execution.**

公告

欢迎转载和评论，转载请注明原文出处。

昵称：[北斗玄机](#)
园龄：[5年2个月](#)
粉丝：[0](#)
关注：[13](#)
[+加关注](#)

< 2018年2月 >						
日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	1	2	3
4	5	6	7	8	9	10

搜索

找找看

谷歌搜索

我的标签

- Java(15)
- MyBatis(4)
- MySQL(4)
- JVM(3)
- 设计模式(3)
- Spring(3)
- SQL(2)
- Sublime Text 2(2)
- Java Collections Framework(2)
- Design Pattern(2)
- [更多](#)

随笔分类

- Algorithm
- Digest(1)
- Git(1)
- Java(14)
- MyBatis(4)
- Redis(1)
- Spring(3)
- SQL(6)
- Sublime(2)
- TCS(1)
- Tomcat(1)
- Typography(1)
- 前端(2)
- 设计模式(3)

随笔档案

- 2018年1月 (3)
- 2017年12月 (20)
- 2017年11月 (1)
- 2016年10月 (1)
- 2016年9月 (1)
- 2016年7月 (1)
- 2016年4月 (1)
- 2016年3月 (4)
- 2015年12月 (1)
- 2015年11月 (1)
- 2015年10月 (2)
- 2015年8月 (2)
- 2015年7月 (1)
- 2015年6月 (1)

2. join() 源码分析

以下是 JDK 8 中 join() 的源码：

```
1 public final void join() throws InterruptedException {
2     join(0);
3 }
4
5 public final synchronized void join(long millis)
6 throws InterruptedException {
7     long base = System.currentTimeMillis();
8     long now = 0;
9
10    if (millis < 0) {
11        throw new IllegalArgumentException("timeout value is negative");
12    }
13
14    if (millis == 0) {
15        while (isAlive()) {
16            wait(0);
17        }
18    } else {
19        while (isAlive()) {
20            long delay = millis - now;
21            if (delay <= 0) {
22                break;
23            }
24            wait(delay);
25            now = System.currentTimeMillis() - base;
26        }
27    }
28 }
29
30 public final synchronized void join(long millis, int nanos)
31 throws InterruptedException {
32
33     if (millis < 0) {
34         throw new IllegalArgumentException("timeout value is negative");
35     }
36
37     if (nanos < 0 || nanos > 999999) {
38         throw new IllegalArgumentException(
39             "nanosecond timeout value out of range");
40     }
41
42     if (nanos >= 500000 || (nanos != 0 && millis == 0)) {
43         millis++;
44     }
45
46     join(millis);
47 }
```

我们可以看到 join() 一共有三个版本：

```
1 public final void join();
2
3 public final synchronized void join(long millis);
4
5 public final synchronized void join(long millis, int nanos);
```

其中

- join() 和 join(long millis, int nanos) 最后都调用了 join(long millis)。
- 带参数的 join() 都为 synchronized method。
- join() 调用了 join(0)，从源码可以看到 join(0) 不断检查线程（join() 所属的线程实例，非调用线程）是否是 Active。

友情链接

[Apache Tomcat](#)
[Doug Lea's Workstation](#)
[Git](#)
[Java SE 8 Documentation](#)
[Martin Fowler](#)
[MySQL 5.6 Reference Manual](#)
[nvie.com](#)
[OpenJDK](#)
[何登成的技术博客](#)

积分与排名

积分 - 1915
排名 - 87467

最新评论

1. Re:Java 浅析Thread.join()
3.a，并不完全是不断去检查子线程状态，同时调用了Object.wait方法。
--阿振_1994
2. Re:Java 浅析Thread.join()
很欣赏博主这种严谨的态度！

技术文章中不应该出现 不严谨的类比和模拟两可的描述。
--Apolo_Du

阅读排行榜

1. Java 浅析Thread.join()(1574)
2. Spring Error : No unique bean of type
[org.apache.ibatis.session.SqlSessionFactory is defined(93)
3. JDK8中JVM对类的初始化探讨(43)
4. Spring 自动注册及自动装配(26)
5. DCL双检锁的失效：现实与初衷的背离(22)

评论排行榜

1. Java 浅析Thread.join()(2)

推荐排行榜

1. Java 浅析Thread.join()(1)

以本文开头的示例为例我们分析一下代码的逻辑：

Parent 调用 child.join()，child.join() 再调用 child.join(0)（此时 Parent 会获得 child 实例作为锁，其他线程可以进入 child.join()，但不可以进入 child.join(0)，因为没有获得锁），child.join(0) 会不断地检查 child 线程是否是 Active。

如果是 Active，则不断地调用 child.wait(0)（此时 Parent 会释放 child 实例锁，其他线程可以竞争锁并进入 child.join(0)）。我们可以得知，Parent 线程在不断地对 child.wait(0) 入栈和出栈。

一旦 child 线程不为 Active（状态为 TERMINATED），child.join(0) 会直接返回到 child.join()，child.join() 会直接返回到 Parent 主线程，Parent 主线程就可以继续运行下去了。

3. 对网上其他分析 join() 的文章提出疑问

网上有很多文章的描述我觉得有歧义，下面挑选一些描述进行分析，也欢迎大家留言一起讨论。

a. 子线程结束之后，**"会唤醒主线程"**，主线程重新获取cpu执行权，继续运行。

"唤醒"令人误解。其实是主线程调用方法不断去检查子线程的状态，这是个主动的动作，**而不是子线程去唤醒主线程。**

b. join() 将几个并行线程的线程**"合并为一个单线程"**执行。

我理解提这个说法的人的意思。但是这样描述只会让读者更难理解。

在调用 join() 方法的程序中，原来的多个线程仍然多个线程，**并没有发生"合并为一个单线程"**。真正发生的是调用 join() 的线程进入 TIMED_WAITING 状态，等待 join() 所属线程运行结束后再继续运行。

一点感想：技术人员写作技术文章时，最好尽量避免使用过于口语化的词汇。因为这种词汇歧义比较大，会让读者感到更加困惑或形成错误的理解。

分类: Java

标签: Java, Thread, Java Concurrency, Java Multithreading

好文要顶

关注我

收藏该文

北斗玄机
关注 - 13
粉丝 - 0

1

0

[+加关注](#)

« 上一篇: MySQL5.6中date和string的转换和比较

» 下一篇: 面向对象设计原则汇总

posted @ 2017-11-28 11:51 北斗玄机 阅读(1574) 评论(2) 编辑 收藏

评论列表

- #1楼 2018-01-26 08:55 Apolo_Du

很欣赏博主这种严谨的态度!

技术文章中不应该出现 不严谨的类比和模拟两可的描述.

支持(0) 反对(0)
- #2楼 2018-01-29 21:34 阿振_1994

3.a，并不完全是不断去检查子线程状态，同时调用了Object.wait方法。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

最新IT新闻:

- [95后回乡见闻：尴尬的伪互联网生活](#)
 - [连收两家智能门铃 亚马逊正不遗余力进入用户家中](#)
 - [360今日正式重组更名登陆A股 开盘涨3.84%](#)
 - [传音成非洲手机老大 “华米OV”却找不到存在感](#)
 - [微软悄然退出消费市场 专职“伺候”企业客户](#)
- » [更多新闻...](#)

最新知识库文章:

- [写给自学者的入门指南](#)
 - [和程序员谈恋爱](#)
 - [学会学习](#)
 - [优秀技术人的管理陷阱](#)
 - [作为一个程序员，数学对你到底有多重要](#)
- » [更多知识库文章...](#)