

<2018年4月>

日	一	二	三	四	五	六
25	26	27	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
29	30	1	2	3	4	5



昵称：只会一点java
园龄：4年5个月
粉丝：52
关注：6
+加关注

搜索

找找看

谷歌搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签
- 更多链接

我的标签

- jdk源码剖析 (21)
- 疑难杂症 (11)
- kafka (6)
- java (6)
- spring ioc (4)
- 高级java必会系列 (3)
- mysql (2)
- pom (2)
- jdk新特性 (2)
- 核心算法 (2)
- 更多

随笔档案(61)

- 2018年3月 (2)
- 2018年1月 (3)
- 2017年12月 (4)
- 2017年11月 (7)
- 2017年10月 (5)
- 2017年9月 (5)
- 2017年8月 (2)
- 2017年7月 (5)
- 2017年6月 (2)
- 2017年5月 (4)
- 2017年4月 (7)
- 2017年3月 (2)

从Thread.start()方法看Thread源码，多次start一个线程会怎么样

Posted on 2017-09-29 18:10 只会一点java 阅读(2231) 评论(0) 编辑 收藏

这篇文章作为[Thread类源码剖析](#)的补充，从一个侧面来看Thread源码。也解答了面试高频问题：“多次start一个线程会怎么样？”

答案是：java.lang.IllegalThreadStateException 线程状态非法异常 继承关系是：--->extends IllegalArgumentException--->extends RuntimeException一个运行时异常，下面我们从源码来透彻分析一下start()时做了什么。

```
1  /**
2      * Causes this thread to begin execution; the Java Virtual Machine
3      * calls the run method of this thread.
4      * 

线程被执行，JVM调用run方法
5      * The result is that two threads are running concurrently: the
6      * current thread (which returns from the call to the
7      * start method) and the other thread (which executes its
8      * run method).
9      * 

10     * It is never legal to start a thread more than once.多次调用start方法启动一个线程是非法的
11     * In particular, a thread may not be restarted once it has completed
12     * execution.
13     *
14     * @exception  IllegalThreadStateException  if the thread was already已经启动的线程再次start，异常
15     *
16     *          started.
17     * @see      #run()
18     * @see      #stop()
19     */
20     public synchronized void start() {
21         /**
22          * This method is not invoked for the main method thread or "system"
23          * group threads created/set up by the VM. Any new functionality added
24          * to this method in the future may have to also be added to the VM.
25          *
26          * A zero status value corresponds to state "NEW".
27          */
28         if (threadStatus != 0)//状态校验 0：NEW 新建状态
29             throw new IllegalThreadStateException();
30
31         /* Notify the group that this thread is about to be started
32          * so that it can be added to the group's list of threads
33          * and the group's unstarted count can be decremented. */
34         group.add(this);//添加进线程组
35
36         boolean started = false;
37         try {
38             start0();//调用native方法执行线程run方法
39             started = true;
40         } finally {
41             try {
42                 if (!started) {
43                     group.threadStartFailed(this);//启动失败，从线程组中移除当前线程。
44                 }
45             } catch (Throwable ignore) {
46                 /* do nothing. If start0 threw a Throwable then
47                  it will be passed up the call stack */
48             }
49         }
50
51         private native void start0();


```

greop.add(this)，把当前线程添加进线程组，源码如下：

```
1  /**
2      * Adds the specified thread to this thread group.
3      *
4      * 

Note: This method is called from both library code
5      * and the Virtual Machine. It is called from VM to add


```

2017年1月 (1)
2016年12月 (1)
2016年11月 (3)
2016年10月 (1)
2016年9月 (2)
2016年8月 (1)
2016年7月 (2)
2016年6月 (1)
2016年5月 (1)

积分与排名

积分 - 75655
排名 - 4666

最新评论

1. Re:kafka原理和实践 (二)
spring-kafka简单实践
不是大神，能帮到你很开心！
--只会一点java

2. Re:kafka原理和实践 (二)
spring-kafka简单实践
@只会一点java谢谢大神~搞定了~...
--奥巴马说我长得丑

3. Re:kafka原理和实践 (二)
spring-kafka简单实践
@奥巴马说我长得丑
org.springframework.kafka.listener.config.ContainerProperties看一下这个类的构造，注意MessageListener是自动.....
--只会一点java

4. Re:kafka原理和实践 (二)
spring-kafka简单实践
参数中加Acknowledgment ack，消费完毕ack.acknowledge();
--只会一点java

5. Re:jdk源码剖析二：对象内存布局、synchronized终极原理
哈哈，不用拜师，加粉就可以...
--只会一点java

阅读排行榜

1. JDK8-十大新特性-附demo(15901)
2. PowerDesigner连接mysql逆向生成pdm(14970)
3. eclipse下SVN同步时忽略target文件夹(9794)
4. maven常用插件pom配置(9355)
5. Eclipse Memory Analyzer，内存泄漏插件，安装使用一条龙(8768)

评论排行榜

1. jdk源码剖析二：对象内存布局、synchronized终极原理(12)
2. spring boot容器启动详解(5)
3. kafka原理和实践 (三) spring-kafka生产者源码(5)
4. kafka原理和实践 (二) spring-kafka简单实践(5)
5. 基于Redis的爬虫平台的实现(4)

推荐排行榜

1. spring boot容器启动详解(7)
2. JDK8-废弃永久代 (PermGen) 迎来元空间 (Metaspace) (5)
3. jdk源码剖析二：对象内存布局、synchronized终极原理(5)
4. 终极锁实战：单JVM锁+分布式锁(4)

```
6      * certain system threads to the system thread group.
7      *
8      * @param t
9      *         the Thread to be added
10     *
11     * @throws IllegalStateException
12     *         if the Thread group has been destroyed
13     */
14     void add(Thread t) {
15         synchronized (this) {
16             if (destroyed) {//线程组状态校验
17                 throw new IllegalStateException();
18             }
19             if (threads == null) {
20                 threads = new Thread[4];//初始化长度为4的Thread数组
21             } else if (nthreads == threads.length) {//数组满了就扩容2倍
22                 threads = Arrays.copyOf(threads, nthreads * 2);
23             }
24             threads[nthreads] = t;//新线程t添加进数组
25
26             // This is done last so it doesn't matter in case the
27             // thread is killed
28             nthreads++;//线程数加1
29
30             // The thread is now a fully fledged member of the group, even
31             // though it may, or may not, have been started yet. It will prevent
32             // the group from being destroyed so the unstarted Threads count is
33             // decremented.
34             nUnstartedThreads--;//未启动线程数-1
35         }
36     }
```



启动失败后调用group.threadStartFailed(this)，都是加锁方法，从线程组中移除当前线程，源码如下



```
1 void threadStartFailed(Thread t) {
2     synchronized(this) {
3         remove(t);//移除线程t
4         nUnstartedThreads++;//未启动线程+1
5     }
6 }
7
8 private void remove(Thread t) {
9     synchronized (this) {
10         if (destroyed) {
11             return;
12         }
13         for (int i = 0 ; i < nthreads ; i++) {
14             if (threads[i] == t) {
15                 System.arraycopy(threads, i + 1, threads, i, --nthreads - i);
16                 // Zap dangling reference to the dead thread so that
17                 // the garbage collector will collect it.
18                 threads[nthreads] = null;
19                 break;
20             }
21         }
22     }
23 }
```



----- 掌控自己的生命轨迹，身心自由！ -----

标签: [jdk源码剖析](#)

好文要顶

关注我

收藏该文



只会一点java
关注 - 6
粉丝 - 52

+加关注

« 上一篇：Java中关于WeakReference和WeakHashMap的理解
» 下一篇：Spring IOC (一) 概览

[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](#)。

5. 多线程并发执行任务，取结果归集。终极总结：Future、FutureTask、CompletionService、CompletableFuture(4)



最新IT新闻：

- 星巴克上线支付宝小程序：好友间可互送礼品
 - 雷军：小米硬件综合净利率永远不超5%！
 - Google第N次尝试做通讯，它的社交梦还能成真吗？
 - 滴滴上市提至今年，自动驾驶即将进入战国时代
 - 被“捧杀”的抖音：内容趋同，全民皆敌
- » 更多新闻...



最新知识库文章：

- 如何识别别人的技术能力和水平？
 - 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
- » 更多知识库文章...