真的有外星人吗？

假如这个世界上只剩下你一个人，当你正坐在屋子里的时候，这时突然响起了敲门声，那么会是谁呢？

C#线程系列讲座(4)：同步与死锁

本文为原创，如需转载，请注明作者和出处，谢谢！

上一篇：C#线程系列讲座(3)：线程池和文件下载服务器

虽然线程可以在一定程度上提高程序运行的效率，但也会产生一些副作用。让我们先看看如下的代码：

```
class Increment
{
    private int n = 0;
    private int max;
    public Increment(int max)
    {
        this.max = max;
    }
    public int result
    {
        get
        {
            return n;
        }
        set
        {
            n = value;
        }
    }
    public void Inc()
    {
        for (int i = 0; i < max; i++)
        {
            n++;
        }
    }
}
class Program
{
    public static void Main()
    {
        Increment inc = new Increment(10000);
        Thread[] threads = new Thread[30];
        for (int i = 0; i < threads.Length; i++)
        {
            threads[i] = new Thread(inc.Inc);
```


< 2008年7月 >

日	一	二	三	四	五	六
29	30	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	31	1	2
3	4	5	6	7	8	9

导航

博客园
首页
新随笔
联系
订阅XML
管理

统计

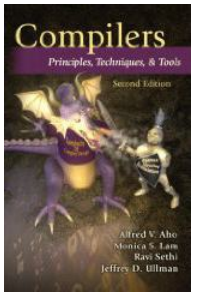
随笔 - 85
文章 - 0
评论 - 650
引用 - 29

公告

我的其他Blog

http://nokiaguy.blogjava.net
http://blog.csdn.net/nokiaguy

正在读的书



```
                threads[i].Start();
            }
            for (int i = 0; i < threads.Length; i++)
            {
                threads[i].Join();  // 等待30个线程都执行完
            }
            Console.WriteLine(inc.result);  //输出n的值
        }

    }
```

上面的程序的基本功能是使用Increment的Inc方法为n递增max，所不同的是，将在Main方法中启动30个线程同时执行Inc方法。在本例中max的值是10000（通过Increment的构造方法传入）。读者可以运行一下这个程序，正常的结果应该是300000，但通常不会得到这个结果，一般获得的结果都比300000小。其中的原因就是Inc方法中的n++上，虽然从表面上看，n++只是一条简单的自增语言，但从底层分析，n++的IL代码如下：

ldsfld // 获得n的初始值，并压到方法栈中

ldc.i4.1 // 将1压到方法栈中

add // 从方法栈中弹出最顶端的两个值，相加，然后将结果保存在方法栈中

stfld // 从当前方法栈中弹出一个值，并更新类字段n

对于上面每一条IL语句是线程安全的，但是n++这条C#语句需要上面的四步才能完成，因此，n++这条语句并不是线程安全的。只要在执行stfld指令之前的任何一步由于其他线程获得CPU而中断，那么就会出现所谓的“脏”数据。

假设n的初始值为0， 在thread1在执行完ldc.i4.1后被thread2中断（add指令并未执行），这时thread2获得的n的初始值仍然是0，假设thread2顺利执行完，那么这时n的值已经是1了，当thread2执行完后，thread1继续执行add指令，并且thread1也顺利执行完，这时，在thread1中的执行结果n仍然是1。因此，这也就出现了调用两次n++，n仍然为1的情况。要解决这个问题的方法也很容易想到，就是让上述四条IL语句要不都不执行，要执行就都执行完，这有点事务处理的意思。

在C#中解决这个问题的技术叫同步。同步的本质就是为某一个代码块加锁，使其成为一个整体，共同进退。最简单的是使用lock为代码块加锁。这个语句在前几讲已经多次使用过了。lock语句可以锁定任何的对象，如果锁定的是类成员，直接使用lock(obj)的形式即可，如果锁定的是静态成员，可以把锁主在object类型上，代码如下：

```
lock(typeof(StaticClass))
{
  ... ...
}
```

对于Increment类，我们可以锁定n++，也可以锁定Inc方法，如锁定n++的Increment类的代码如下：

```
    class Increment
    {
        private int n = 0;
        private int max;
        private Object lockN = new Object();
        public Increment(int max)
        {
            this.max = max;
```

```
        }
        public int result
        {
            get
            {
                return n;
            }
            set
            {
                n = value;
            }
        }

        private void IncN()
        {
            lock (lockN)
            {
                n++;
            }
        }
        public void Inc()
        {
            for (int i = 0; i < max; i++)
            {
                  IncN();
            }
        }
    }
```

也可以直接将如下的代码放到for循环中取代调用IncN方法，

```
lock (lockN)
{
    n++;
}
```

或者直接将Inc方法锁住，代码如下：

```
        public void Inc()
        {
            lock (lockN)
            {
                for (int i = 0; i < max; i++)
                {
                    n++;
                }
            }
        }
```

但笔者并不建议直接将Inc锁住，因为这样就和单线程没什么区别了，虽然可以避免出现读脏数据的情况，但却牺牲的效率。

.net新特性(2) (rss)
ajax(4) (rss)
algorithm(14) (rss)
C#(13) (rss)
C/C++(9) (rss)
database(11) (rss)
delphi(2) (rss)
IE（6至8）(1) (rss)
javascript(5) (rss)
linux(1) (rss)
MSIL(2) (rss)
mysql(4) (rss)
open source(3) (rss)
SQL Server(11) (rss)
VBA(1) (rss)
web(8) (rss)
WPF(1) (rss)
wxWidgets(1) (rss)
安全(2) (rss)
进程、线程、并发(5) (rss)
设计模式(1) (rss)
网络营销(1) (rss)
宇宙探秘(1) (rss)
杂七杂八(9) (rss)

随笔档案(85)
2009年8月 (1)
2009年7月 (2)
2009年6月 (3)
2009年3月 (3)
2009年2月 (6)
2009年1月 (5)
2008年12月 (3)
2008年10月 (4)
2008年9月 (3)
2008年8月 (3)
2008年7月 (6)
2008年6月 (9)
2008年5月 (36)
2008年4月 (1)

相册
images
其他
微软认证

Blogs
anytao
leo zhang（职业规划师）
刘江（图灵主编）

开源

从本例分析得知，产生问题的原因就是因为n++不是原子操作。而在.net framework中提供了一个Interlocked类，可以使n++变成原子操作。Interlocked有一些方法，可以保证对变量的操作是原子的，如Increment方法保证n++的操作是原子的，Decrement方法保证n--的操作是原子的，Exchange方法保证为变量赋值的操作是原子的。因此，可以使用Increment方法来替换n++，代码如下：

```
public void Inc()
{
    for (int i = 0; i < max; i++)
    {
        Interlocked.Increment(ref n);
    }
}
```

任何事物都具有两面性，同步技术也不例外，在某些情况下，可以由于两个线程互相锁定某些对象而造成死锁（就是两个线程互相等待对方释放对象）。这就象有两个学生晚上在复习功课，他们都希望学习能超过对方，而且他们都很累了，但是谁也不肯先休息，是都在盯着对方屋里的灯，期望着对方休息后。自己才休息。但却谁也不肯先关灯，所以他们就只有这样耗到天亮了。当然，解决这个问题的方法有两个，第一个就是其中一个学生或两个学生根本就不关心对方是否先睡觉，自己学累了就直接关灯了。当然，另外一个方法就有点暴力了，就是到点就直接断电，那谁也别学了（这也相当于线程中断，不过不到万不得以时最好别用这招）。

让我们先举一个线程死锁的例子，代码如下：

```
    class Program
    {
        private static Object objA = new Object();
        private static Object objB = new Object();
        public static void LockA()
        {
            lock (objA)
            {
                Thread.Sleep(1000);
                lock (objB)
                {
                }
            }
            Console.WriteLine("LockA");
        }
        public static void LockB()
        {
            lock (objB)
            {
                Thread.Sleep(2000);
                lock (objA)
                {
                }
            }
            Console.WriteLine("LockB");
        }
        public static void Main()
        {
            Thread threadA = new Thread(LockA);
```

微软开源网站

协议

MSN Messenger

积分与排名

积分 - 145137

排名 - 381

最新评论 XML

1. Re:.net framework3.5新特性2：var、初始化、匿名类和扩展方法

我看了文章和大家的说评语，楼主写的也不错，大家说也很好，让我对var又加深对var的了解。

--Dean123

2. Re:全排列算法原理和实现

回8楼和9楼的，在permut函数for循环中，调用完permut之后还要执行一次swap的，要把数组还原成未调用permut之前的状态，保证下一次循环或者上一级循环swap数组元素的正确。愚见愚见。...

--Little_Angel

3. Re:移动的MobileMarket个人终于可以上传软件了

还是即将上线，敬请期待~？~？

--DAP

4. Re:移动的MobileMarket个人终于可以上传软件了

哦，好久没去关注这方面了，去看看

--peterzb

5. Re:想抢先体验Android操作系统的魅力吗？那就使用Android LiveCD吧！

买不起，只有这样体验下了~~~

--J.Motto

阅读排行榜

1. 全排列算法原理和实现(6165)
2. C#线程系列讲座(1)：BeginInvoke和EndInvoke方法(5552)
3. C#线程系列讲座(2)：Thread类的应用(4360)
4. 实现Web程序的自动登录(3762)
5. 用C#2.0实现网络蜘蛛(WebSpider)(3704)

评论排行榜

1. 用VC实现洪水攻击程序(38)
2. C#线程系列讲座(1)：BeginInvoke和EndInvoke方法(34)
3. .net framework3.5新特性1：Lambda表达式(33)

```
            Thread threadB = new Thread(LockB);
            threadA.Start();
            threadB.Start();
        }
    }
```

在上面的代码中，LockA方法会在当前线程中执行完Lock(objA)后延迟1秒，而LockB方法会在执行完lock(objB)后延迟2秒，一般LockA会先执行lock(objB），但这时objB已经被LockB锁住了，而且LockB还在延迟（2秒还没到）。在这时，LockA已经将objA和objB都锁上了，当LockB执行到lock(objA)时，由于objA已经被锁上，因此，LockB就被阻塞了。而LockA在执行到lock(objB）时，由于这时LockA还在延迟，因此，objB也被锁住了。LockA和LockB也就相当于上述的两个学生，互相等待对方关灯，但谁也不肯先关灯，所以就死锁了。如果采用第一种方法非常简单，就是保持被锁定的多个对象的顺序，如将LockB方法的锁定顺序换一下，代码如下：

```
public static void LockB()
{
    lock (objA)
    {
         Thread.Sleep(2000);
          lock (objB)
          {
          }
    }
    Console.WriteLine("LockB");
}
```

或是将LockA方法也改一下，先锁objB，再锁objA。

当然，也可以采用暴力一点的方法，当发现一些线程长时间没反应时，可以使用Abort方法强行中断它们。代码如下：

```
public static void Main()
{
     Thread threadA = new Thread(LockA);
     Thread threadB = new Thread(LockB);
     threadA.Start();
     threadB.Start();

     Thread.Sleep(4000);

     threadA.Abort();
     threadB.Abort();
     Console.WriteLine("线程全部结束");
}
```

在后面的文章中将讲解C#中其他的同步技术。

下一篇：**C#线程系列讲座(5)**：同步技术之**Monitor**

4. .net framework3.5新特性2：var、初始化、匿名类和扩展方法(29)
5. 实现Web程序的自动登录(28)

0 0

(请您对文章做出评价)

posted on 2008-07-25 22:32 银河使者 阅读(2245) 评论(13) 编辑 收藏 网摘 所属分类: 进程、线程、并发, C#, .net高级技术, 原创

.

评论

#1楼 2008-07-25 23:38 天寒
赞， 回复 引用 查看

#2楼 2008-07-26 01:41 Tony Zhou
private Object lockN = new Object();
lock(lockN)
和
lock(this)
效果上面有什么区别？ 回复 引用 查看

#3楼 2008-07-26 03:28 jiudian[未注册用户]
强呵..... 回复 引用

#4楼 2008-07-26 08:36 雅阁布
不错！！ 回复 引用 查看

#5楼 2008-07-26 08:58 留恋星空
多看看这个,以后的面试题都是小Case 回复 引用 查看

#6楼 2008-07-26 13:48 legio
@Tony Zhou
我觉的，只要this满足typeof(StaticClass)，同时this不是主线程的类，而是辅助线程的类，两者效果就是一样的了，不知道这样理解是否正确 回复 引用 查看

#7楼 2008-07-26 16:17 HedgeHog
非常期待下篇啊.... 回复 引用 查看

#8楼 2008-07-26 16:19 pk的眼泪
继续跟 回复 引用 查看

#9楼[楼主] 2008-07-26 16:20 银河使者
@Tony Zhou
private Object lockN = new Object();
lock(lockN)
和
lock(this)
效果上面有什么区别？

从本质上说，lock(lockN)和lock(this)没有什么区别。都是锁定的对象，所不同的是，一个是锁定的lockN，另一个是锁定的this（当前对象实例）。 如果想省事，而且类中的被锁定的成员（方法、字段等）只有一组互斥的，用lockN和this没什么区别，但如果有多组互斥的，如method1和method2，这两个方法不能被不同线程同时调用，method3和method4不能被不同线程同时调用。
但method1、method2和method3、method4方法没有任何关系，这时，就需要定义两个对象，来分别锁定method1、method2和method3、method4。当然，也可以一个用this，另外一个使用其他的对象变量。 回复 引用 查看

#10楼 2008-07-26 21:15 Leepy
好文，学习了！ 回复 引用 查看

#11楼 2008-07-27 09:57 陈晨

MSDN上建议：

定义 private 对象来锁定，或 private static对象变量来保护所有实例所共有的数据。

回复 引用 查看

#12楼[楼主] 2008-07-27 10:41 银河使者

对，最好是单独使用private对象来锁定共享数据，但有时为了方便，也可直接使用this来锁定。 回复 引用 查看

#13楼 2008-10-16 16:31 Jack Niu

Interlocked有一些方法，可以保证对变量的操作是原子的，如Increment方法保证n++的操作是原子的，Decrement方法保证n--的操作是原子的，Exchange方法保证为变量赋值的操作是原子的。

----受教了,thx 回复 引用 查看

刷新评论列表 刷新页面 返回页首

发表评论

昵称： [登录] [注册]

主页：

邮箱： (仅博主可见)

评论内容： 闪存 个人主页

提交 登录 注册

[使用Ctrl+Enter键快速提交评论]

个人主页上线测试中

今天你闪了吗?

2009博客园纪念T恤

寻找18-28岁待业者

最新**IT**新闻**:**
Delphi 2010初体验
谷歌经济学家：搜索关键词表明美经济正复苏
Facebook应吸取谷歌经验避免重蹈雅虎覆辙
唐骏传授成功秘笈：创业要有自己的"杀手锏"
商业周刊：企业用户不愿甲骨文壮大 称其店大欺客

相关链接**:**
系列教程：C#多线程学习
博客园.NET频道，专业.NET技术门户
ASP.NET MVC 专题，从零开始学.NET技术
系列教程：你必须知道的.NET
博客园.net频道上线啦！
ASP.NET学习之Ajax入门系列
07考研数学基础系列