Wolfy

博客园 首页 新随笔 订阅 管理

[C#基础]说说lock到底锁谁？

写在前面

最近一个月一直在弄文件传输组件，其中用到多线程的技术，但有的地方确实需要只能有一个线程来操作，如何才能保证只有一个线程呢？首先想到的就是锁的概念，最近在我们项目组中听的最多的也是锁谁，如何锁？看到有同事使用lock(this)，也有lock(private static object)，那就有点困惑了，lock到底锁谁才是最合适的呢？

lock

首先先上官方Msdn的说法

> lock 关键字可确保当一个线程位于代码的临界区时，另一个线程不会进入该临界区。 如果其他线程尝试进入锁定的代码，则它将一直等待（即被阻止），直到该对象被释放。
> lock 关键字在块的开始处调用 Enter，而在块的结尾处调用 Exit。
> ThreadInterruptedException 引发，如果 Interrupt 中断等待输入 lock 语句的线程。
> 通常，应避免锁定 public 类型，否则实例将超出代码的控制范围。
>
> 常见的结构 lock (this)、lock (typeof (MyType)) 和 lock ("myLock") 违反此准则：
> 如果实例可以被公共访问，将出现 lock (this) 问题。
> 如果 MyType 可以被公共访问，将出现 lock (typeof (MyType)) 问题。
> 由于进程中使用同一字符串的任何其他代码都将共享同一个锁，所以出现 lock("myLock") 问题。
> 最佳做法是定义 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。
> 在 lock 语句的正文不能使用 等待 关键字。

Enter指的是Monitor.Enter(获取指定对象上的排他锁。)，Exit指的是Monitor.Exit(释放指定对象上的排他锁。)

有上面msdn的解释及Exit方法，可以这样猜测"直到该对象被释放"，"该对象"应该是指锁的对象，对象释放了或者对象改变了，其他的线程才可以进入代码临界区（是不是可以这样来理解？）。

在多线程中，每个线程都有自己的资源，但是代码区是共享的，即每个线程都可以执行相同的函数。这可能带来的问题就是几个线程同时执行一个函数，导致数据的混乱，产生不可预料的结果，因此我们必须避免这种情况的发生。

打个比方，有这样一个情景，很多公司所在的大厦的厕所的蹲位都是小单间型的，也就是一次只能进去一个人，那么为了避免每次进去一个人，那怎么做呢？不就是一个人进去之后顺手把门锁上么？这样你在里面干啥事，外边的人也只能等待你解放完了，才能进入。而蹲位的资源（蹲位，手纸等）是共享的。

最常使用的lock是如下格式的代码段：

```
23 0
private static object objlock = new object();
lock (objlock)
{
    //要执行的代码逻辑
}
```

请先登录

为什么锁的对象是私有的呢？还是以厕所为例子吧，私有就好比，这把锁只有你能访问到，而且最好这把锁不会因为外力而有所改变，别人访问不到，这样才能保证你进去了，别人就进不去了，如果是公有的，就好比你蹲位小单间的锁不是安装在里面而是安装在外边的，别人想不想进就不是你所能控制的了，这样也不安全。

lock(this)

通过字面的意思就是锁的当前实例对象。那是否对其他实例对象产生影响？那下面看一个例子：

```
namespace Wolfy.LockDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Test t = new Test();
            Test t2 = new Test();
            Thread[] threads = new Thread[10];
            for (int i = 0; i < threads.Length; i++)
            {
                //通过循环创建10个线程。
                threads[i] = new Thread(() =>
                {
                    t2.Print();
                });
                //为每个线程设置一个名字
                threads[i].Name = "thread" + i;

            }
            //开启创建的十个线程
            for (int i = 0; i < threads.Length; i++)
            {
                threads[i].Start();
            }

            Console.Read();
        }
    }
    class Test
    {
        public void Print()
        {
            lock (this)
            {
                for (int i = 0; i < 5; i++)
                {
                    Console.WriteLine("\t" + Thread.CurrentThread.Name.ToString()
+ "\t" + i.ToString() + " ");
                }
            }
        }
    }
}
```

如果在不加锁的情况下输出如下：

```
thread1 0
thread1 1
thread1 2
thread1 3
thread1 4
thread2 0
thread3 0
thread3 1
thread2 1
thread2 2
thread2 3
thread2 4
thread3 2
thread3 3
thread3 4
thread4 0
thread4 1
thread4 2
thread4 3
thread4 4
thread5 0
thread5 1
thread5 2
thread5 3
thread5 4
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread6 0
thread6 1
```

从上面的输出结果也可以看出，线程出现了争抢的现象，而这并不是我们想要的结果，我们想要的是，每次只有一个线程去执行Print方法。那我们就尝试一下lock(this)

```
1     class Test
2     {
3         public void Print()
4         {
5             lock (this)
6             {
7                 for (int i = 0; i < 5; i++)
8                 {
9                     Console.WriteLine("\t" + Thread.CurrentThread.Name.ToString()
+ "\t" + i.ToString() + " ");
10                }
11            }
12        }
13    }
```

输出结果

```
thread1 0
thread1 1
thread1 2
thread1 3
thread1 4
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread2 0
thread2 1
thread2 2
thread2 3
thread2 4
thread4 0
thread4 1
thread4 2
thread4 3
thread4 4
thread3 0
thread3 1
thread3 2
thread3 3
thread3 4
```

从输出结果，觉得大功告成了，可是现在情况又来了，在项目中的其他的地方，有同事也这样写了

2013年8月 (11)

2013年7月 (12)

积分与排名

积分 - 357196

排名 - 354

最新评论

1. Re:[Asp.net]常见word，excel，ppt，pdf在线预览方案，有图有真相，总有一款适合你！

@wanghawk链接： 密码：gupg...

--wolfy

2. Re:[Asp.net]常见word，excel，ppt，pdf在线预览方案，有图有真相，总有一款适合你！

@wolfy谢谢，急需这个，请发个demo到邮箱，万分感谢！wanghui9559@hotmail.com或者xlfdlihailong@126.com...

--wanghawk

3. Re:[EF]vs15+ef6+mysql这个问题，你遇到过么？

经过测试，还是必须安装Mysql Connector Net。否则无法执行选择表的步骤。

--h1nson

4. Re:[实战]MVC5+EF6+MySql企业网盘实战(27)——应用列表

亲爱的Wolfy你好，GITHUB上主件少啊，没法用了，请填充文件。。

是开源中国的码云。

--阿笨虎

5. Re:asp.net中通过单例保存全局参数

@422159763嗯，缓存，application 都用了。...

--wolfy

6. Re:asp.net中通过单例保存全局参

这样的代码，又创建了一个Test对象，而且他也知道使用多线程执行耗时的工作，那么就会出现类似下面的代码。

```
namespace Wolfy.LockDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Test t = new Test();
            Test t2 = new Test();
            t2.Age = 20;
            Thread[] threads = new Thread[10];
            for (int i = 0; i < threads.Length; i++)
            {
                //通过循环创建10个线程。
                threads[i] = new Thread(() =>
                {
                    t.Print();
                    t2.Print();
                });
                //为每个线程设置一个名字
                threads[i].Name = "thread" + i;

            }


            //开启创建的十个线程
            for (int i = 0; i < threads.Length; i++)
            {
                threads[i].Start();
            }

            Console.Read();
        }
    }
    class Test
    {
        public int Age { get; set; }
        public void Print()
        {
            lock (this)
            {
                for (int i = 0; i < 5; i++)
                {
                    Console.WriteLine("\t" + Thread.CurrentThread.Name.ToString()
+ "\t" + i.ToString() + " ");
                }
            }
        }
    }
}
```

这里为Test加了一个Age属性，为了区别当前创建的对象不是同一个对象。

输出的结果为

```
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread1 0
thread1 1
thread1 2
thread1 3
thread1 4
thread1 0
thread1 1
thread1 2
thread2 0
thread2 1
thread1 3
thread2 2
thread2 3
thread2 4
```

在输出的结果中已经出现了线程抢占执行的情况了，而不是一个线程执行完另一个线程在执行。

lock(private obj)

那么我们现在使用一个全局的私有的对象试一试。

```
1 namespace Wolfy.LockDemo
2 {
3     class Program
4     {
5         private static object objLock = new object();
6         static void Main(string[] args)
7         {
8             Test t = new Test();
9             Test t2 = new Test();
10            t2.Age = 20;
11            Thread[] threads = new Thread[10];
12            for (int i = 0; i < threads.Length; i++)
13            {
14                //通过循环创建10个线程。
15                threads[i] = new Thread(() =>
16                {
17                    lock (objLock)
18                    {
19                        t.Print();
20                        t2.Print();
21                    }
22                });
23                //为每个线程设置一个名字
24                threads[i].Name = "thread" + i;
25
26            }
27
28
29            //开启创建的十个线程
30            for (int i = 0; i < threads.Length; i++)
31            {
32                threads[i].Start();
33            }
34
35            Console.Read();
36        }
37    }
38    class Test
39    {
40        public int Age { get; set; }
41        public void Print()
42        {
43            for (int i = 0; i < 5; i++)
44            {
45                Console.WriteLine("\t" + Thread.CurrentThread.Name.ToString() +
"\t" + i.ToString() + " ");
46            }
```

15)

10. [UML]UML系列——用例图Use Case(5485)

评论排行榜

1. [Asp.net]常见word，excel，ppt，pdf在线预览方案，有图有真相，总有一款适合你！(141)

2. 耗时两月，NHibernate系列出炉(103)

3. 如何高效的利用博客园？(91)

4. 在北京这两年(88)

5. 百度云，360云盘能否做网站文件服务器的遐想(64)

6. FlexPaper+SWFTool+操作类=在线预览PDF(63)

7. 看过《大湿教我写.net通用权限框架(1)之菜单导航篇》之后发生的事(61)

8. [Asp.Net]最近一个项目的总结(61)

9. [UML]UML系列——时序图（顺序图）sequence diagram(39)

10. [Asp.net]常见数据导入Excel，Excel数据导入数据库解决方案，总有一款适合你！(36)

推荐排行榜

1. [Asp.net]常见word，excel，ppt，pdf在线预览方案，有图有真相，总有一款适合你！(141)

2. 耗时两月，NHibernate系列出炉(136)

3. 各种排序算法汇总(87)

4. 如何高效的利用博客园？(76)

5. [Asp.net]常见数据导入Excel，Excel数据导入数据库解决方案，总有一款适合你！(54)

6. 在北京这两年(41)

7. [UML]UML系列——时序图（顺序图）sequence diagram(36)

8. [c#基础]关于const和readonly常见的笔试题剖析(30)

9. [c#基础]关于try...catch最常见的

```
47          }
48      }
49  }
```

输出的结果

```
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread0 0
thread0 1
thread0 2
thread0 3
thread0 4
thread1 0
thread1 1
thread1 2
thread1 3
thread1 4
thread1 0
thread1 1
thread1 2
thread1 3
thread1 4
thread2 0
thread2 1
thread2 2
thread2 3
thread2 4
```

从输出的结果也可以看出，有序的，每次进来一个线程执行。

那通过上面的比较可以有这样的一个结论，lock的结果好不好，还是关键看锁的谁，如果外边能对这个谁进行修改，lock就失去了作用。所以一般情况下，使用静态的并且是只读的对象。

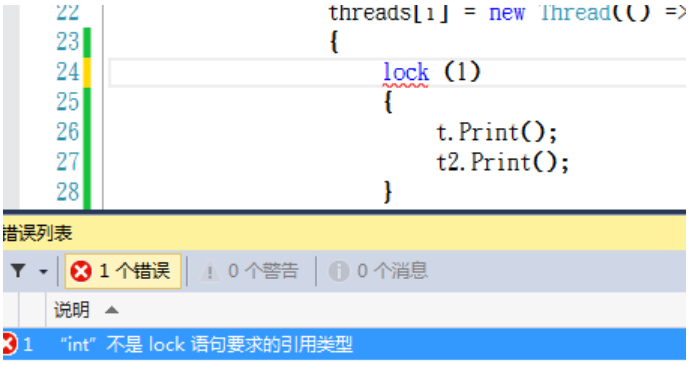
也就有了类似下面的代码

```
1  private static readonly object objLock = new object();
```

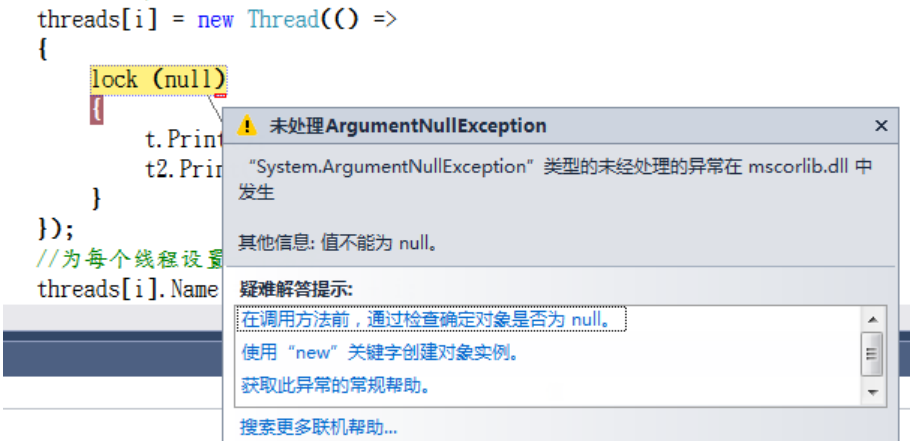
你可能会说，不对啊，你下面的代码跟上面的代码不一样啊，为什么就得出这样的结论？难道就不能把Object放在test类中么，放在test类中的话，在new Test()的时候，其实放在Test中也是可以的，只要保证objLock在外部是无法修改的就可以。

上面说的最多的是lock对象，那么它能不能lock值类型？

答案是否定的，如



当然lock(null)也是不行的，如图



虽然编译可以通过，但是运行就会出错。

lock(string)

string也是应用类型，从语法上来说是没有错的。

但是锁定字符串尤其危险，因为字符串被公共语言运行库 (CLR)"暂留"。 这意味着整个程序中任何给定字符串都只有一个实例，就是这同一个对象表示了所有运行的应用程序域的所有线程中的该文本。因此，只要在应用程序进程中的任何位置处具有相同内容的字符串上放置了锁，就将锁定应用程序中该字符串的所有实例。通常，最好避免锁定 public 类型或锁定不受应用程序控制的对象实例。例如，如果该实例可以被公开访问，则 lock(this) 可能会有问题，因为不受控制的代码也可能会锁定该对象。这可能导致死锁，即两个或更多个线程等待释放同一对象。出于同样的原因，锁定公共数据类型（相比于对象）也可能导致问题。而且lock(this)只对当前对象有效，如果多个对象之间就达不到同步的效果。lock(typeof(Class))与锁定字符串一样，范围太广了。

总结

关于lock的介绍就到这里，有下面几点需要注意的地方

1、lock的是引用类型的对象，string类型除外。

2、lock推荐的做法是使用静态的、只读的、私有的对象。

3、保证lock的对象在外部无法修改才有意义，如果lock的对象在外部改变了，对其他线程就会畅通无阻，失去了lock的意义。

参考文章

http://www.cnblogs.com/jintianhu/archive/2010/11/19/1881494.html

博客地址：http://www.cnblogs.com/wolf-sun/
博客版权:如果文中有不妥或者错误的地方还望高手的你指出，以免误人子弟。如果觉得本文对你有所帮助不如【推荐】一下！如果你有更好的建议，不如留言一起讨论，共同进步！ 再次感谢您耐心的读完本篇文章。

分类: [C#]

标签: lock, lock(this)

wolfy
关注 - 12
粉丝 - 994
荣誉：推荐博客
+加关注

« 上一篇：Linq之Lambda进阶
» 下一篇：Linq之隐式类型、自动属性、初始化器、匿名类

posted @ 2015-01-08 22:28 wolfy 阅读(3066) 评论(29) 编辑 收藏

评论列表

#1楼 2015-01-09 00:04 cisabc

最近读码正好用的上这些知识。。。谢谢！！

支持(1) 反对(0)

#2楼 2015-01-09 00:23 杨恒连

解释的太好了，点赞 (来源:合仔茶端)

支持(0) 反对(0)

#3楼 2015-01-09 08:26 KuBiCoder

好文章，顶

支持(1) 反对(0)

#4楼 2015-01-09 08:45 囧月言炎

```
private static object obj;
lock(obj){}
```

支持(0) 反对(0)

#5楼[楼主] 2015-01-09 09:27 wolfy

@ 我和小菜
@KuBiCoder
@cisabc
谢谢支持

支持(0) 反对(0)

#6楼[楼主] 2015-01-09 09:28 wolfy

@ 狄云
这个obj是否得初始化？

支持(0) 反对(0)

#7楼 2015-01-09 09:59 爱你不打烊

一直对lock模糊不清的，这次可以好好学习一下了。。。。

支持(0) 反对(0)

#8楼 2015-01-09 10:18 Aulan

this 表示的就是当前实例，当你再new一个的时候，锁定的就不再是同一个对象了。 不能锁定值类型的原因是，当这个值类型传递到另一个线程的时候，会创建一个副本，锁定的也不再是同一个对象了。 锁定字符串带来的问题是，字符串在CLR中会暂存在 内存中，如果有两个变量被分配了相同的字符串内容，那么这两个引用会指向同一块内存，实际锁定也就是同一个对象，这就会导致整个应用程序的阻塞。所以锁定字符串是非常危险的行为。

支持(2) 反对(0)

#9楼 2015-01-09 10:20 二进制小男人

楼主，你的是什么字体？

支持(0) 反对(0)

#10楼[楼主] 2015-01-09 10:26 wolfy

@ 爱你不打烊
谢谢支持

支持(0) 反对(0)

#11楼[楼主] 2015-01-09 10:26 wolfy

@ Aulan
嗯 这个解释也非常清楚

支持(0) 反对(0)

#12楼[楼主] 2015-01-09 10:26 wolfy

@ 二进制小男人
默认的字体啊

支持(0) 反对(0)

#13楼 2015-01-09 10:32 joyment

其实关键在于锁的位置,
如果锁在 Test 类里面, 不管它是 private static 还是 this, 都会导致线程抢占.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace LockTest
{
    public class Program
    {
        static void Main(string[] args)
        {
            Test t1 = new Test();
            Test t2 = new Test();
            t1.Age = "boy";
            t2.Age = "girl";

            int threadlength =10;
            Thread[] threads = new Thread[threadlength];
            for (int i = 0; i < threadlength; i++)
            {
                threads[i] = new Thread(() => {
                    t1.Print();
                    t2.Print();
                });
                threads[i].Name = "Thread" + i;
            }

            for (int i = 0; i < threadlength; i++)
            {
                threads[i].Start();
            }

            Console.Read();
        }
    }

    public class Test
    {
        private static object locker = new Object();
        public string Age { get; set; }
        public void Print()
        {
            lock (locker)
            {
                for(int i =0;i<5;i++)
                {
                    Console.WriteLine("{0} : {1}, {2}", Thread.CurrentThread.Name, Age,
                }
            }
        }
    }
}
```

执行结果:
Thread6 : girl, 1
Thread6 : girl, 2
Thread6 : girl, 3
Thread6 : girl, 4
Thread8 : boy, 0
Thread8 : boy, 1
Thread8 : boy, 2
Thread8 : boy, 3
Thread8 : boy, 4
Thread4 : girl, 0
Thread4 : girl, 1
Thread4 : girl, 2
Thread4 : girl, 3
Thread4 : girl, 4
Thread9 : girl, 0
Thread9 : girl, 1
Thread9 : girl, 2
Thread9 : girl, 3
Thread9 : girl, 4
Thread8 : girl, 0
Thread8 : girl, 1
Thread8 : girl, 2
Thread8 : girl, 3
Thread8 : girl, 4

支持(0) 反对(0)

#14楼 2015-01-09 10:40 bidaas

这是线程基础知识了，不懂这个千万别碰线程，不然要出大事。

支持(0) 反对(0)

#15楼 2015-01-09 10:40 6572789

这个也是我需要理解的

支持(0) 反对(0)

#16楼 2015-01-09 10:52 SuperEVO

@ joyment
Test内部private static 是不会线程抢占的。多线程情况下锁的顺序本身就不是能确定的事，所以结果顺序不能说明什么。你可以把lock内部改成：

```
lock (locker)
            {
                Thread.Sleep(1000);
                Console.WriteLine("{0} : {1}, T:{2}", Thread.CurrentThread.Name, Age, Da
            }
```

你会发现同一时间只会运行1个输出。

支持(0) 反对(0)

#17楼 2015-01-09 11:46 joyment

@ SuperEVO
你是对的，我理解错了.

支持(0) 反对(0)

#18楼[楼主] 2015-01-09 11:49 wolfy

@ SuperEVO
@joyment

不错，支持一个

支持(0) 反对(0)

#19楼[楼主] 2015-01-09 11:49 wolfy

@ 6572789
@bidaas
多谢支持

支持(0) 反对(0)

#20楼 2015-01-09 17:14 andy-gao

lock(this)时候，当类为建立一个私有构造函数。应该就会出现问题吧？

支持(0) 反对(0)

#21楼 2015-01-09 17:45 Ant

重点Mark

支持(0) 反对(0)

#22楼 2015-01-09 19:10 囧月言炎

@ wolfy
private static object obj= new object();
lock(obj){}

支持(0) 反对(0)

#23楼[楼主] 2015-01-09 19:32 wolfy

@ andy-gao
你说的这个有点类似单例模式中的使用，单例中用的也是lock静态私有的object对象

支持(0) 反对(0)

#24楼[楼主] 2015-01-09 19:32 wolfy

@ Ant
多谢支持

支持(0) 反对(0)

#25楼 2015-01-27 11:22 越过

请问LZ，Lock(this)如果在多个实例对象下无法保证资源读取安全，那它具体可以用在什么情况下。再请教一下：有这么一种情况，抽奖时，有时会出现多人同时进行抽奖，那用Lock是应该的，可以保证每次只有一个线程，一个实例可以操作这部分资源，防止奖品数据出错，用静态私有的object对象。你觉得需要在里面套一个lock(this)吗，因为我觉得，如果同一个用户不停的点击抽奖按钮，既不是新线程，又不是新实例，但是上一次点击操作还没执行完，有点，会不会出现问题？

支持(0) 反对(0)

#26楼[楼主] 2015-01-27 13:05 wolfy

@ 越过
使用中应保证lock的对象是私有的，外部无法修改，就能达到互斥的结果。lock(this) 如果实例能在外部修改，你lock（this）就失去意义了。没必要在加层锁了。

支持(0) 反对(0)

#27楼 2015-01-27 14:41 越过

@ wolfy
我理解你说的，用来锁的对象不应该被外部修改到，但我的意思是，像抽奖这种情况，用户点了一次，又点一次，重复点了几次，像这种情况，应该是每点一次都会实例一个新的对象，所以加Lock(this)也就没什么意义，只要一个私有的Lock对象就可以了，是这样理解吧。

支持(0) 反对(0)

#28楼[楼主] 2015-01-27 20:15 wolfy

@ 越过

是的，在目前项目中，也出现你描述的类似的情况，lock私有静态对象，是可以的，没发现什么情况。

支持(0) 反对(0)

#29楼 2015-04-13 12:47 Lumia1020

@ SuperEVO
多线程情况下锁的顺序本身就不是能确定的事，所以结果顺序不能说明什么？ 这句话还是有点不明白，求大神指教？

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。