

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(2): Thread类的应用

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(1): BeginInvoke和EndInvoke方法

一、 Thread类的基本用法

通过System.Threading.Thread类可以开始新的线程，并在线程堆栈中运行静态或实例方法。可以通过Thread类的的构造方法传递一个无参数，并且不返回值(返回void)的委托(ThreadStart)，这个委托的定义如下:

[ComVisibleAttribute(true)]

public delegate void ThreadStart()

我们可以通过如下的方法来建立并运行一个线程。

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

namespace MyThread
{
    class Program
    {
        public static void myStaticThreadMethod()
        {
            Console.WriteLine("myStaticThreadMethod");
        }
        static void Main(string[] args)
        {
            Thread thread1 = new Thread(myStaticThreadMethod);
            thread1.Start(); // 只要使用Start方法，线程才会运行
        }
    }
}
```

除了运行静态的方法，还可以在线程中运行实例方法，代码如下:

```
using System;
using System.Collections.Generic;
```

2008年7月						
<	一	二	三	四	五	>
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

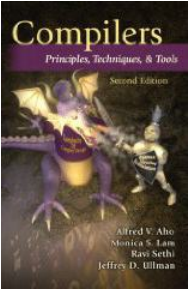
引用 - 29

公告

我的其他Blog

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

正在读的书



```
using System.Linq;
using System.Text;
using System.Threading;

namespace MyThread
{
    class Program
    {
        public void myThreadMethod()
        {
            Console.WriteLine("myThreadMethod");
        }
        static void Main(string[] args)
        {
            Thread thread2 = new Thread(new Program().myThreadMethod);
            thread2.Start();
        }
    }
}
```

如果读者的方法很简单，或出去某种目的，也可以通过匿名委托或Lambda表达式来为Thread的构造方法赋值，代码如下：

```
Thread thread3 = new Thread(delegate() { Console.WriteLine("匿名委托"); });
thread3.Start();

Thread thread4 = new Thread(() => { Console.WriteLine("Lambda表达式"); });
thread4.Start();
```

其中Lambda表达式前面的()表示没有参数。

为了区分不同的线程，还可以为Thread类的Name属性赋值，代码如下：

```
Thread thread5 = new Thread(() => { Console.WriteLine(Thread.CurrentThread.Name); });
thread5.Name = "我的Lamdba";
thread5.Start();
```

如果将上面thread1至thread5放到一起执行，由于系统对线程的调度不同，输出的结果是不定的，如图1是一种可能的输出结果。

我的最新闪存

今天真热

与我联系

发短消息

搜索

常用链接

我的随笔

我的空间

我的短信

我的评论

更多链接

留言簿

给我留言

查看留言

我参加的小组

创业交流

设计模式

AJAX

.NET 3.x

写书译书小组

博客园精华集出版小组

沈阳.NET俱乐部

《编译原理》

我参与的团队

北京.NET俱乐部(0/0)

沈阳.NET俱乐部(0/0)

CLR基础研究团队(0/0)

我的标签

C#(4)

SQL Server(4)

数据库(3)

web(2)

.net(2)

c++(2)

CTE(2)

公共表表达式(2)

SQL Server 2005(1)

递归(1)

更多

随笔分类(169)

获奖作品(2) (rss)

原创(47) (rss)

.net高级技术(7) (rss)

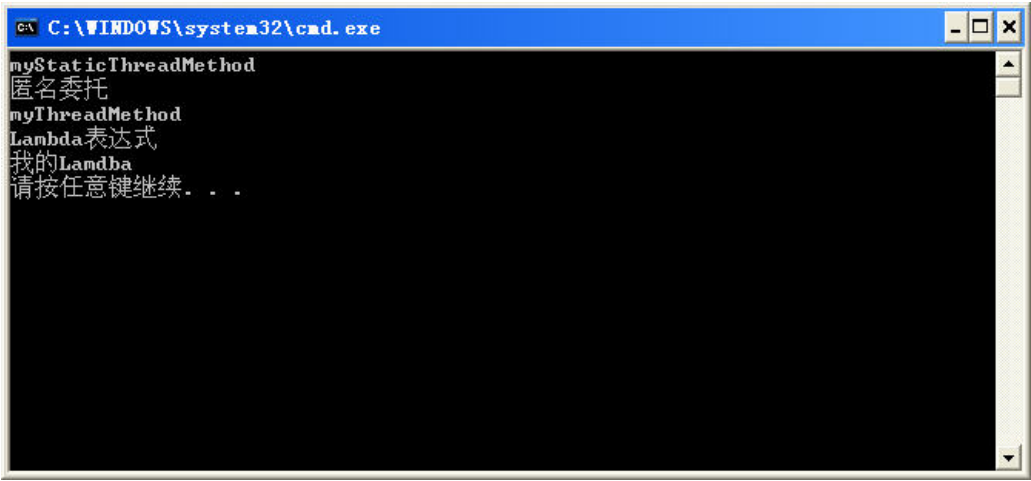


图1

二、定义一个线程类

我们可以将Thread类封装在一个MyThread类中，以使任何从MyThread继承的类都具有多线程能力。MyThread类的代码如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
namespace MyThread
{
    abstract class MyThread
    {
        Thread thread = null;

        abstract public void run();

        public void start()
        {
            if (thread == null)
                thread = new Thread(run);
            thread.Start();
        }
    }
}
```

可以用下面的代码来使用MyThread类。

```
class NewThread : MyThread
{
    override public void run()
```

- [.net入门技术\(1\)](#) (rss)
- [.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)

```
        {
            Console.WriteLine("使用MyThread建立并运行线程");
        }
    }

    static void Main(string[] args)
    {

        NewThread nt = new NewThread();
        nt.start();
    }
}
```

我们还可以利用MyThread来为线程传递任意复杂的参数。详细内容见下节。

三、 为线程传递参数

Thread类有一个带参数的委托类型的重载形式。这个委托的定义如下：

```
[ComVisibleAttribute(false)]

public delegate void ParameterizedThreadStart(Object obj)
```

这个Thread类的构造方法的定义如下：

```
public Thread(ParameterizedThreadStart start);
```

下面的代码使用了这个带参数的委托向线程传递一个字符串参数：

```
public static void myStaticParamThreadMethod(Object obj)
{
    Console.WriteLine(obj);
}

static void Main(string[] args)
{
    Thread thread = new Thread(myStaticParamThreadMethod);
    thread.Start("通过委托的参数传值");
}
```

要注意的是，如果使用的是不带参数的委托，不能使用带参数的Start方法运行线程，否则系统会抛出异常。但使用带参数的委托，可以使用thread.Start()来运行线程，这时所传递的参数值为null。

也可以定义一个类来传递参数值，如下面的代码如下：

```
class MyData
{
    private String d1;
    private int d2;
    public MyData(String d1, int d2)
    {
        this.d1 = d1;
```

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

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En

```
        this.d2 = d2;
    }
    public void threadMethod()
    {
        Console.WriteLine(d1);
        Console.WriteLine(d2);
    }
}

MyData myData = new MyData("abcd", 1234);
Thread thread = new Thread(myData.threadMethod);
thread.Start();
```

如果使用在第二节定义的MyThread类，传递参数会显示更简单，代码如下：

```
class NewThread : MyThread
{
    private String p1;
    private int p2;
    public NewThread(String p1, int p2)
    {
        this.p1 = p1;
        this.p2 = p2;
    }

    override public void run()
    {
        Console.WriteLine(p1);
        Console.WriteLine(p2);
    }
}

NewThread newThread = new NewThread("hello world", 4321);
newThread.start();
```

四、 前台和后台线程

使用Thread建立的线程默认情况下是前台线程，在进程中，只要有一个前台线程未退出，进程就不会终止。主线程就是一个前台线程。而后台线程不管线程是否结束，只要所有的前台线程都退出（包括正常退出和异常退出）后，进程就会自动终止。一般后台线程用于处理时间较短的任务，如在一个Web服务器中可以利用后台线程来处理客户端发过来的请求信息。而前台线程一般用于处理需要长时间等待的任务，如在Web服务器中的监听客户端请求的程序，或是定时对某些系统资源进行扫描的程序。下面的代码演示了前台和后台线程的区别。

```
public static void myStaticThreadMethod()
{
    Thread.Sleep(3000);
}

Thread thread = new Thread(myStaticThreadMethod);
// thread.IsBackground = true;
thread.Start();
```

dInvoke方法(5551)

3. C# 线程系列讲座(2): Thread类的应用(4358)
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)
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)

如果运行上面的代码，程序会等待3秒后退出，如果将注释去掉，将thread设成后台线程，则程序会立即退出。

要注意的是，必须在调用Start方法之前设置线程的类型，否则一旦线程运行，将无法改变其类型。

通过BeginXXX方法运行的线程都是后台线程。

五、 判断多个线程是否都结束的两种方法

确定所有线程是否都完成了工作的方法有很多，如可以采用类似于对象计数器的方法，所谓对象计数器，就是一个对象被引用一次，这个计数器就加1，销毁引用就减1，如果引用数为0，则垃圾搜集器就会对这些引用数为0的对象进行回收。

方法一：线程计数器

线程也可以采用计数器的方法，即为所有需要监视的线程设一个线程计数器，每开始一个线程，在线程的执行方法中为这个计数器加1，如果某个线程结束（在线程执行方法的最后为这个计数器减1），为这个计数器减1。然后再开始一个线程，按着一定的时间间隔来监视这个计数器，如是标个计数器为0，说明所有的线程都结束了。当然，也可以不用这个监视线程，而在每一个工作线程的最后（在为计数器减1的代码的后面）来监视这个计数器，也就是说，每一个工作线程在退出之前，还要负责检测这个计数器。使用这种方法不要忘了同步这个计数器变量啊，否则会产生意想不到的后果。

方法二：使用Thread.join方法

join方法只有在线程结束时才继续执行下面的语句。可以对每一个线程调用它的join方法，但要注意，这个调用要在另一个线程里，而不要在主线程，否则程序会被阻塞的。

个人感觉这种方法比较好。

线程计数器方法演示：

```
class ThreadCounter : MyThread
{
    private static int count = 0;
    private int ms;
    private static void increment()
    {
        lock (typeof(ThreadCounter)) // 必须同步计数器
        {
            count++;
        }
    }
    private static void decrease()
    {
        lock (typeof(ThreadCounter))
        {
            count--;
        }
    }
    private static int getCount()
    {
        lock (typeof(ThreadCounter))
        {
            return count;
        }
    }
    public ThreadCounter(int ms)
    {
```

```
        this.ms = ms;
    }
    override public void run()
    {
        increment();
        Thread.Sleep(ms);
        Console.WriteLine(ms.ToString()+"毫秒任务结束");
        decrease();
        if (getCount() == 0)
            Console.WriteLine("所有任务结束");
    }
}

ThreadCounter counter1 = new ThreadCounter(3000);
ThreadCounter counter2 = new ThreadCounter(5000);
ThreadCounter counter3 = new ThreadCounter(7000);

counter1.start();
counter2.start();
counter3.start();
```

上面的代码虽然在大多数的时候可以正常工作，但却存在一个隐患，就是如果某个线程，假设是counter1，在运行后，由于某些原因，其他的线程并未运行，在这种情况下，在counter1运行完后，仍然可以显示出“所有任务结束”的提示信息，但是counter2和counter3还并未运行。为了消除这个隐患，可以将increment方法从run中移除，将其放到ThreadCounter的构造方法中，在这时，increment方法中的lock也可以去掉了。代码如下：

```
public ThreadCounter(int ms)
{
    this.ms = ms;
    increment();
}
```

运行上面的程序后，将显示如图2的结果。

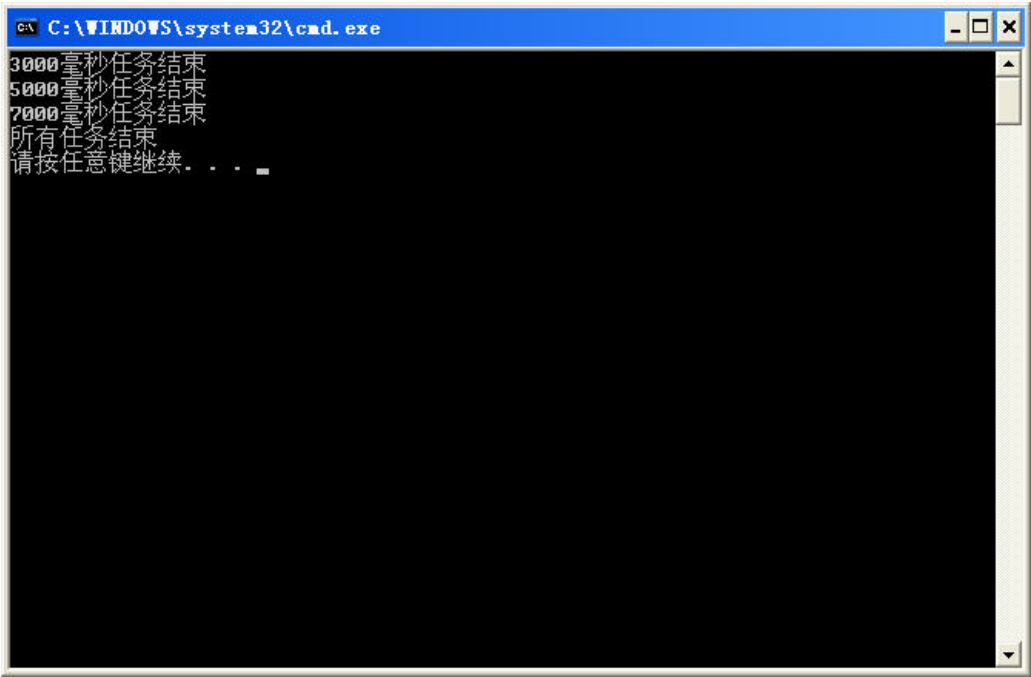


图2

使用**Thread.join**方法演示

```
private static void threadMethod(Object obj)
{
    Thread.Sleep(Int32.Parse(obj.ToString()));
    Console.WriteLine(obj + "毫秒任务结束");
}

private static void joinAllThread(object obj)
{
    Thread[] threads = obj as Thread[];
    foreach (Thread t in threads)
        t.Join();
    Console.WriteLine("所有的线程结束");
}

static void Main(string[] args)
{
    Thread thread1 = new Thread(threadMethod);
    Thread thread2 = new Thread(threadMethod);
    Thread thread3 = new Thread(threadMethod);

    thread1.Start(3000);
    thread2.Start(5000);
    thread3.Start(7000);
}
```



```
Thread joinThread = new Thread(joinAllThread);
joinThread.Start(new Thread[] { thread1, thread2, thread3 });

}
```

在运行上面的代码后，将会得到和图2同样的运行结果。上述两种方法都没有线程数的限制，当然，仍然会受到操作系统和硬件资源的限制。

下一篇：[C#线程系列讲座\(3\)：线程池和文件下载服务器](#)

《银河系列原创教程》发布

0

0

(请您对文章做出评价)

posted on 2008-07-16 23:32 银河使者 阅读(4358) 评论(10) 编辑 收藏 网摘 所属分类: 进程、线程、并发, C#, 原创

评论

#1楼 2008-07-17 08:40 GuoYong.Che

真不错，学习了 [回复](#) [引用](#) [查看](#)

#2楼 2008-07-17 09:33 Seattle

最近在研究线程，会一直关注 [回复](#) [引用](#) [查看](#)

#3楼 2008-07-17 10:23 xiao_p

受教育了

多个线程，一般是弄个异步代理，然后写个回调方法 来通知主线程，线程结束！

[回复](#) [引用](#) [查看](#)

#4楼[楼主] 2008-07-17 10:34 银河使者

确定多个线程是否完成的方法很多，但基本可以分为主动式和被动式。主动式就是由每个线程在结束时来通知主线程或其他监视线程。如本文的第一种方法就属性主动式，当然可以将write(...)改成回调函数形式了。

被动方式是由其他的监视线程对多个线程进行监控，本文的第二种方法就属性这种方式。 [回复](#) [引用](#) [查看](#)

#5楼 2008-07-17 10:46 程序一生[未注册用户]

学习了 [回复](#) [引用](#)

#6楼 2008-07-17 11:56 清炒白菜

不错，用Join来等待所有线程完成工作的办法很神奇～

[回复](#) [引用](#) [查看](#)

#7楼 2008-07-17 15:28 daodao

join方法是啥左右？能解释一下不？ [回复](#) [引用](#) [查看](#)

#8楼 2008-07-17 15:30 daodao

join方法是啥作用？能解释一下不？ [回复](#) [引用](#) [查看](#)

#9楼[楼主] **2008-07-17 15:53** [银河使者](#)

当调用一个线程的join方法时，如果这个线程未执行完，程序将被阻塞。 [回复](#) [引用](#) [查看](#)

#10楼 **2008-10-22 12:02** [zw11](#)[未注册用户]

非常感谢楼主，因为你的文章，让我对多线程有了进一步的了解 [回复](#) [引用](#)

[免费学习asp.net课程](#)
本市人员可享受50-100%政府补贴 合格颁发国家职业资格和微软双认证
[www.zili.cn](#)

[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](#)

[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](#)
权威:华浦ISEP国际软件工程师 免费:就业讲座帮您找到好工作
[www.isen.com.cn](#)
[免费.NET报表软件--博计](#)
做ASP.NET报表，不用ReportViewer控件 全面兼容
VB.NET, C#.NET, Delphi.NET
[www.bonzerreport.com](#)
[IP*Works! TFTP Component](#)
Components for VB, .NET, ActiveX, Java, Delphi,

Embedded, & more!
www.nsoftware.com
Explorer controls c#
Explorer shell controls for .NET Written in 100% C#.
Free Trial!
www.iam-software.com/developer/



China-pub 计算机图书网上专卖店! 6.5万品种 2-8折!
China-Pub 计算机绝版图书按需印刷服务
链接: 切换模板
导航: 网站首页 个人主页 社区 新闻 博问 闪存 网摘 招聘 找找看 Google搜索

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“杀手铜”
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
<http://www.hjbbs.com/thread-55-534042.htm>
07考研数学基础系列
大学英语语法系列讲座
考研英语词汇系列节目
C#设计模式系列
英乐帖索引查询