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(1)：BeginInvoke和EndInvoke方法

本文为原创，如需转载，请注明作者和出处，谢谢！

开发语言：**C#3.0**

IDE：**Visual Studio 2008**

本系列教程主要包括如下内容：

1. BeginInvoke和EndInvoke方法
2. Thread类
3. 线程池
4. 线程同步基础
5. 死锁
6. 线程同步的7种方法
7. 如何在线程中访问GUI组件

一、线程概述

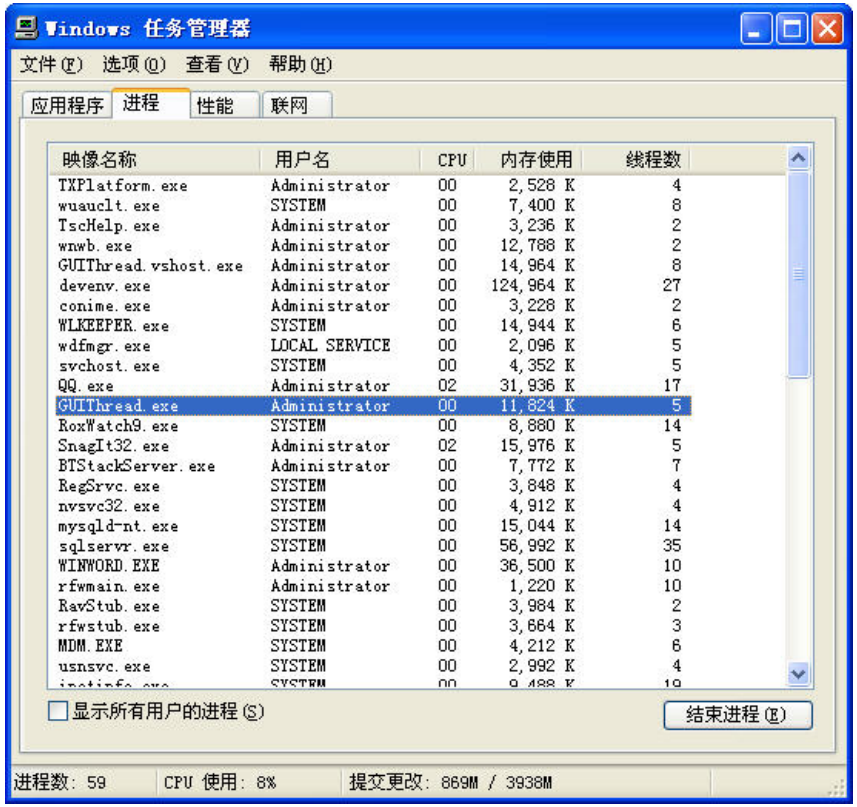
在操作系统中一个进程至少要包含一个线程，然后，在某些时候需要在同一个进程中同时执行多项任务，或是为了提供程序的性能，将要执行的任务分解成多个子任务执行。这就需要在同一个进程中开启多个线程。我们使用C#编写一个应用程序（控制台或桌面程序都可以），然后运行这个程序，并打开windows任务管理器，这时我们就会看到这个应用程序中所含有的线程数，如下图所示。

< 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



如果任务管理器没有“线程数”列，可以【查看】>【选择列】来显示“线程计数”列。从上图可以看出，几乎所有的进程都拥有两个以上的线程。从而可以看出，线程是提供应用程序性能的重要手段之一，尤其在多核CPU的机器上尤为明显。

二、用委托**(Delegate)**的**BeginInvoke**和**EndInvoke**方法操作线程

在C#中使用线程的方法很多，使用委托的BeginInvoke和EndInvoke方法就是其中之一。BeginInvoke方法可以使用线程异步地执行委托所指向的方法。然后通过EndInvoke方法获得方法的返回值（EndInvoke方法的返回值就是被调用方法的返回值），或是确定方法已经被成功调用。我们可以通过四种方法从EndInvoke方法来获得返回值。

三、直接使用**EndInvoke**方法来获得返回值

当使用BeginInvoke异步调用方法时，如果方法未执行完，EndInvoke方法就会一直阻塞，直到被调用的方法执行完毕。如下面的代码所示：

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

namespace MyThread
{
    class Program
```

```
    {
        private static int newTask(int ms)
        {
            Console.WriteLine("任务开始");
            Thread.Sleep(ms);
            Random random = new Random();
            int n = random.Next(10000);
            Console.WriteLine("任务完成");
            return n;
        }

        private delegate int NewTaskDelegate(int ms);


        static void Main(string[] args)
        {
            NewTaskDelegate task = newTask;
            IAsyncResult asyncResult = task.BeginInvoke(2000, null, null);

            // EndInvoke方法将被阻塞2秒
            int result = task.EndInvoke(asyncResult);
            Console.WriteLine(result);
        }
    }
}
```

在运行上面的程序后，由于newTask方法通过Sleep延迟了2秒，因此，程序直到2秒后才输出最终结果（一个随机整数）。如果不调用EndInvoke方法，程序会立即退出，这是由于使用BeginInvoke创建的线程都是后台线程，这种线程一但所有的前台线程都退出后（其中主线程就是一个前台线程），不管后台线程是否执行完毕，都会结束线程，并退出程序。关于前台和后台线程的详细内容，将在后面的部分讲解。

读者可以使用上面的程序做以下实验。首先在Main方法的开始部分加入如下代码：

```
Thread.Sleep(10000);
```

以使Main方法延迟10秒钟再执行下面的代码，然后按Ctrl+F5运行程序，并打开企业管理器，观察当前程序的线程数，假设线程数是4，在10秒后，线程数会增至5，这是因为调用BeginInvoke方法时会建立一个线程来异步执行newTask方法，因此，线程会增加一个。

四、使用**IAsyncResult asyncResult**属性来判断异步调用是否完成

虽然上面的方法可以很好地实现异步调用，但是当调用EndInvoke方法获得调用结果时，整个程序就象死了一样，这样做用户的感觉并不会太好，因此，我们可以使用asyncResult来判断异步调用是否完成，并显示一些提示信息。这样做可以增加用户体验。代码如下：

```
static void Main(string[] args)
{
    NewTaskDelegate task = newTask;
    IAsyncResult asyncResult = task.BeginInvoke(2000, null, null);

    while (!asyncResult.IsCompleted)
    {
        Console.Write("*");
        Thread.Sleep(100);
```

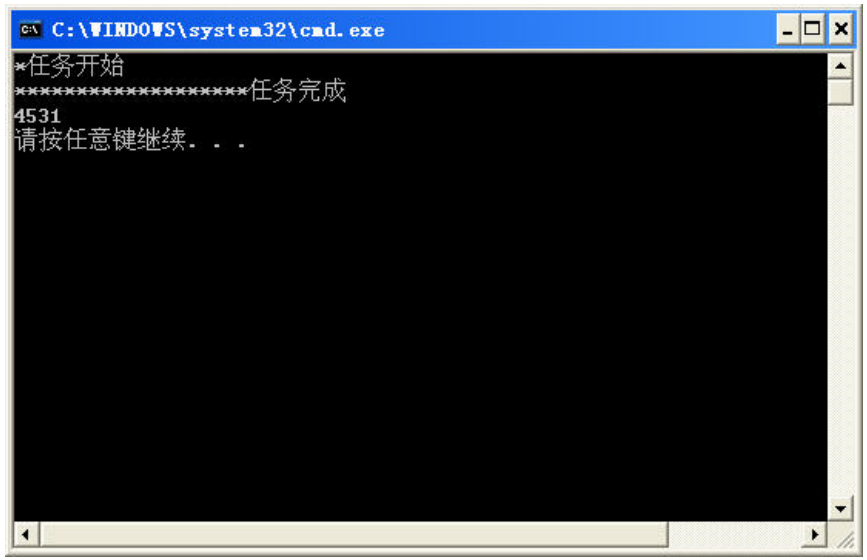
我的标签
C#(4)
SQL Server(4)
数据库(3)
web(2)
.net(2)
c++(2)
CTE(2)
公共表表达式(2)
SQL Server 2005(1)
递归(1)
更多

随笔分类(169)
获奖作品(2) (rss)
原创(47) (rss)
.net高级技术(7) (rss)
.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)

```
    }
    // 由于异步调用已经完成，因此， EndInvoke会立刻返回结果
    int result = task.EndInvoke(asyncResult);
    Console.WriteLine(result);
}
```

上面代码的执行结果如下图所示。



由于是异步，所以“*”可能会在“任务开始”前输出，如上图所示。

五、使用**WaitOne**方法等待异步方法执行完成

使用WaitOne方法是另外一种判断异步调用是否完成的方法。代码如下：

```
static void Main(string[] args)
{
    NewTaskDelegate task = newTask;
    IAsyncResult asyncResult = task.BeginInvoke(2000, null, null);

    while (!asyncResult.AsyncWaitHandle.WaitOne(100, false))
    {
        Console.Write("*");
    }

    int result = task.EndInvoke(asyncResult);
    Console.WriteLine(result);
}
```

WaitOne的第一个参数表示要等待的毫秒数，在指定时间之内，WaitOne方法将一直等待，直到异步调用完成，并发出通知，WaitOne方法才返回true。当等待指定时间之后，异步调用仍未完成，WaitOne方法返回false，如果指定时间为0，表示不等待，如果为-1，表示永远等待，直到异步调用完成。

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
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

六、使用回调方式返回结果

上面介绍的几种方法实际上只相当于一种方法。这些方法虽然可以成功返回结果，也可以给用户一些提示，但在这个过程中，整个程序就象死了一样（如果读者在GUI程序中使用这些方法就会非常明显），要想在调用的过程中，程序仍然可以正常做其它的工作，就必须使用异步调用的方式。下面我们使用GUI程序来编写一个例子，代码如下：

```
private delegate int MyMethod();
private int method()
{
    Thread.Sleep(10000);
    return 100;
}
private void MethodCompleted(IAsyncResult asyncResult)
{
    if (asyncResult == null) return;
    textBox1.Text = (asyncResult.AsyncState as
    MyMethod).EndInvoke(asyncResult).ToString();
}

private void button1_Click(object sender, EventArgs e)
{

    MyMethod my = method;
    IAsyncResult asyncResult = my.BeginInvoke(MethodCompleted, my);
}
```

要注意的是，这里使用了BeginInvoke方法的最后两个参数（如果被调用的方法含有参数的话，这些参数将作为BeginInvoke的前面一部分参数，如果没有参数，BeginInvoke就只有两个参数了）。第一个参数是回调方法委托类型，这个委托只有一个参数，就是IAsyncResult,如MethodCompleted方法所示。当method方法执行完后，系统会自动调用MethodCompleted方法。BeginInvoke的第二个参数需要向MethodCompleted方法中传递一些值，一般可以传递被调用方法的委托，如上面代码中的my。这个值可以使用IAsyncResult.AsyncState属性获得。

由于上面的代码通过异步的方式访问的form上的一个textbox，因此，需要按ctrl+f5运行程序（不能直接按F5运行程序，否则无法在其他线程中访问这个textbox，关于如果在其他线程中访问GUI组件，并在后面的部分详细介绍）。并在form上放一些其他的可视控件，然在点击button1后，其它的控件仍然可以使用，就象什么事都没有发生过一样，在10秒后，在textbox1中将输出100。

七、其他组件的**BeginXXX**和**EndXXX**方法

在其他的.net组件中也有类似BeginInvoke和EndInvoke的方法，如System.Net.HttpWebRequest类的BeginGetResponse和EndGetResponse方法，下面是使用这两个方法的一个例子：

```
private void requestCompleted(IAsyncResult asyncResult)
{
    if (asyncResult == null) return;
    System.Net.HttpWebRequest hwr = asyncResult.AsyncState as System.Net.HttpWebRequest;
    System.Net.HttpWebResponse response =
(System.Net.HttpWebResponse)hwr.EndGetResponse(asyncResult);
    System.IO.StreamReader sr = new
System.IO.StreamReader(response.GetResponseStream());
    textBox1.Text = sr.ReadToEnd();
}
```

```
private delegate System.Net.HttpWebResponse RequestDelegate(System.Net.HttpWebRequest request);

private void button1_Click(object sender, EventArgs e)
{
    System.Net.HttpWebRequest request =
    (System.Net.HttpWebRequest)System.Net.WebRequest.Create("http://www.cnblogs.com");
    IAsyncResult asyncResult =request.BeginGetResponse(requestCompleted, request);
}
```

下一篇：C#线程系列讲座(2)：Thread类的应用

《银河系列原创教程》发布

posted on 2008-07-13 13:04 银河使者 阅读(5551) 评论(34) 编辑 收藏 网摘 所属分类: 进程、线程、并发, C#, 原创

评论

#1楼 2008-07-13 13:42 SOSOS's BLog
正想学习线程方面的知识..期望在楼主的文章以学习到...支持!~ 回复 引用 查看

#2楼 2008-07-13 14:15 GuoYong.Che
不错，学习了 回复 引用 查看

#3楼 2008-07-13 17:51 XeonWell
改天再看,打开该文章好几次,cpu都长期在94%以上,不知道楼主用了啥. 虽然很喜欢楼主的文章,但是在这炎热的夏天,cpu要紧. 回复 引用 查看

#4楼[楼主] 2008-07-13 17:56 银河使者
文章没用啥，只有两个图，是不是浏览器的问题啊，有可能中招了。 回复 引用 查看

#5楼 2008-07-13 18:32 Angel Lucifer
三，四的方式实际上还是阻塞的。
在主线程内调用 EndXXX 方法，效果跟调用 Thread.Join 方法一样。
因为楼主计算的结果非常快速，所以结果会立刻返回。

虽然基础，不过仍然欢迎楼主写此类文章，:-) 回复 引用 查看

#6楼[楼主] 2008-07-13 18:40 银河使者
是的，EndXXX方法和join方法类似，因此，本文所述的方法除了回调外，其他的都是阻塞的，只是四、五所用的方法在EndXXX返回前给用户一些动作显示，这样用户就知道程序没有死，还在正常运行。 回复 引用 查看

#7楼[楼主] 2008-07-13 18:40 银河使者
第一篇文章很简单，以后会逐渐增加一些难度。循序见进。 回复 引用 查看

#8楼 2008-07-13 18:55 Angel Lucifer
再次讨论：

BeginInvoke 方法实际上是调用 ThreadPool 中的线程完成操作。

在进行 I/O 操作的时候，虽然看起来跟异步 I/O 的行为很相似，但跟操作系统提供的异步 I/O 还是有区别的。

在 I/O 操作并不频繁密集的时候，BeginInvoke 的 I/O 操作可能比操作系统的异步 I/O 要高效，但一旦密集起来，性能可能会严重下降。

希望楼主能在后续文章内加以区分，:-) 回复 引用 查看

#9楼[楼主] 2008-07-13 19:10 银河使者
如果程序中需要大量的线程，还是使用Thread来处理吧。BeginInvoke方法并不是用来执行大量密集操作的，我个人感觉这个方法只是为了程序中的某个按钮或其他的控件触发某个事件时，使用这个方法来执行一下异步操作。但这个异步操作的时间可能很短（可能只有几秒甚至更短）。如果使用了BeginInvoke来执行这个操作，程序就不会有停顿的感觉。这有点象AJAX中的XMLHttpRequest的异步请求，只是为了使客户端不会有停顿的感觉。以便增加用户体验。

事实上，真正的线程应用是Thread，这才是.net线程的核心。 回复 引用 查看

#10楼[楼主] 2008-07-13 19:14 银河使者
在下面的文章会提一下这方面内容 回复 引用 查看

#11楼 2008-07-13 20:14 Clingingboy
刚在看异步操作这部分,期待下文,偶先看msdn预习下 回复 引用 查看

#12楼 2008-07-13 20:52 Jeffrey Zhao
正确使用异步操作:
http://www.cnblogs.com/JeffreyZhao/archive/2008/02/24/use-async-operation-properly.html 回复 引用 查看

#13楼 2008-07-14 08:23 雅阁布
UP!!! 回复 引用 查看

#14楼 2008-07-14 10:00 DaveLin
支持楼主一下吧，建议大家注意线程问题，真的很重要 回复 引用 查看

#15楼 2008-07-14 11:10 清炒白菜
借个地方问个多线程的问题。
工作中经常要多线程测试一些业务逻辑，在编写测试程序的时候，如何确定所有的线程都完成了"工作"?
曾经试过用WaitAll，但是有数量限制，具体数字忘记了，反正20线程以下肯定没问题，但有时候测试需要创建50+，甚至100+的线程去测试，WaitAll就无能为力了。
目前的办法是测试项目（console模式）最后用一个ReadKey（）来等待～
回复 引用 查看

#16楼[楼主] 2008-07-14 12:49 银河使者
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，如是棕个计数器为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。

我在下一讲会详细描述这两种方法的具体实现过程。

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）

哪位读者有更好的方法也可以跟贴！！ 回复 引用 查看

#17楼 2008-07-14 15:41 颜昌钢
文章写的不错..

有个疑问:
以第一种方法来说,主线程应该是阻塞的吧?

呵呵，既然是异步,就不要让主线程阻塞. 回复 引用 查看

#18楼 2008-07-14 16:17 wangxm[未注册用户]
同上疑问，除了回调不会阻塞主线程，前面几种异步委托的有实际意义？ 回复 引用

#19楼[楼主] 2008-07-14 16:57 银河使者
是的，直接使用EndInvoke方法，主线程会被阻塞。其实EndInvoke方法相当于Thread.join方法，这个join方法的最大用处就是可以确保多个线程同时结束，再往下执行，EndInvoke的作用和join类似，如果有多个委托使用了BeginInvoke方法，那么可以用这些委托的EndInvoke方法来确保它们都执行完后，再往下进行。如果没这个需求，就要通过回调来使用EndInvoke方法了 回复 引用 查看

#20楼 2008-07-14 17:49 airwolf2026
问个问题,是不是一些异步操作会用到IOCP功能?MSDN好像没有说哪个XXXBegin...用IOCP吧?另外那些WorkerItem???这样的工作线程和普通thread有啥区别?嘿嘿,要是楼主后续的文章有这些就更好咯. 回复 引用 查看

#21楼 2008-07-15 11:10 KT
好文~ 回复 引用 查看

#22楼 2008-07-17 09:16 随风逝去（叶进）
--引用----------
XeonWell: 改天再看,打开该文章好几次,cpu都长期在94%以上,不知道楼主用了啥. 虽然很喜欢楼主的文章,但是在这炎热的夏天,cpu要紧.

确实是这样的，不过文章也确实是好！ 回复 引用 查看

#23楼[楼主] 2008-07-17 09:39 银河使者
是不是我blog右边加的那个flash时钟的原因啊，我把它去了看看 回复 引用 查看

#24楼 2008-07-17 09:43 Simens
好文章，我顶你个肺！ 回复 引用 查看

#25楼 2008-07-26 14:57 pk的眼泪
顶.学习中... 回复 引用 查看

#26楼 2008-07-31 21:54 兵
```
MyMethod my = method;
IAsyncResult asyncResult = my.BeginInvoke(MethodCompleted, my);
```

这里代码写错了，my.BeginInvoke中的my应该是一个控件的引用才对 回复 引用 查看

#27楼 2008-08-05 10:41 陈思涵
在1.1 或者2.0下能运行吗 回复 引用 查看

#28楼 2008-08-05 10:54 钢钢
正好可以互补一下： C#多线程学习系列
http://www.cnblogs.com/nokiaguy/archive/2008/07/13/1241817.html 回复 引用 查看

#29楼[楼主] 2008-08-05 13:54 银河使者
2.0可以，1.1没测试过，我也没环境了。但现在很少用1.1了，至少也是2.0。 回复 引用 查看

#30楼 2008-08-27 16:25 dr[未注册用户]
委托的Invoke和Control.Invoke有什么联系？ 回复 引用

#31楼 2008-09-29 15:59 追忆似水流年
楼主 关于六、使用回调方式返回结果 我拷贝你的代码执行时,在private void MethodCompleted(IAsyncResult asyncResult)的textBox1.Text = (asyncResult.AsyncState as MyMethod).EndInvoke(asyncResult).ToString()处会提示跨线程作业无效: 存取控制项 'textBox1' 時所使用的线程與建立控制項的线程不同。你这个例子如何改呢? 回复 引用 查看

#32楼 2008-09-30 22:02 eastcowboy
楼主你回调函数写的有问题吧
按你这样写 我运行了下 回调函数根本就没执行
回复 引用 查看

#33楼[楼主] 2008-09-30 22:20 银河使者
不会有问题的，我都是运行成功后，复制过来的代码。 回复 引用 查看

#34楼 2008-11-05 16:39 wgd[未注册用户]
@eastcowboy

可能楼主笔误吧，这问题应该不难找

IAsyncResult asyncResult = my.BeginInvoke(new AsyncCallback(MethodCompleted), my);
回复 引用

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