xugang

从2007年开始，记录一个DotNET程序员的成长

C#多线程学习(四) 多线程的自动管理(线程池)

在多线程的程序中，经常会出现两种情况：

一种情况：　应用程序中，线程把大部分的时间花费在等待状态，等待某个事件发生，然后才能给予响应

　　这一般使用ThreadPool（线程池）来解决；

另一种情况：线程平时都处于休眠状态，只是周期性地被唤醒

　　这一般使用Timer（定时器）来解决；

ThreadPool类提供一个由系统维护的线程池（可以看作一个线程的容器），该容器需要 Windows 2000 以上系统支持，因为其中某些方法调用了只有高版本的Windows才有的API函数。

将线程安放在线程池里，需使用ThreadPool.QueueUserWorkItem()方法，该方法的原型如下：

```
//将一个线程放进线程池，该线程的Start()方法将调用WaitCallback代理对象代表的函数
public static bool QueueUserWorkItem(WaitCallback);

//重载的方法如下，参数object将传递给WaitCallback所代表的方法
public static bool QueueUserWorkItem(WaitCallback, object);
```

注意：

ThreadPool类是一个静态类，你不能也不必要生成它的对象。而且一旦使用该方法在线程池中添加了一个项目，那么该项目将是无法取消的。

在这里你无需自己建立线程，只需把你要做的工作写成函数，然后作为参数传递给ThreadPool.QueueUserWorkItem()方法就行了，传递的方法就是依靠WaitCallback代理对象，而线程的建立、管理、运行等工作都是由系统自动完成的，你无须考虑那些复杂的细节问题。

ThreadPool 的用法：

首先程序创建了一个ManualResetEvent对象，该对象就像一个信号灯，可以利用它的信号来通知其它线程。本例中，当线程池中所有线程工作都完成以后，ManualResetEvent对象将被设置为有信号，从而通知主线程继续运行。

ManualResetEvent对象有几个重要的方法：

初始化该对象时，用户可以指定其默认的状态（有信号/无信号）；

在初始化以后，该对象将保持原来的状态不变，直到它的Reset()或者Set()方法被调用：

Reset()方法：将其设置为无信号状态；

Set()方法：将其设置为有信号状态。

WaitOne()方法：使当前线程挂起，直到ManualResetEvent对象处于有信号状态，此时该线程将被激活。然后，程序将向线程池中添加工作项，这些以函数形式提供的工作项被系统用来初始化自动建立的线程。当所有的线程都运行完了以后，ManualResetEvent.Set()方法被调用，因为调用了ManualResetEvent.WaitOne()方法而处在等待状态的主线程将接收到这个信号，于是它接着往下执行，完成后边的工作。

ThreadPool 的用法示例：

```
using System;
using System.Collections;
using System.Threading;

namespace ThreadExample
{
```

```
    //这是用来保存信息的数据结构，将作为参数被传递
    public class SomeState
    {
    public int Cookie;
    public SomeState(int iCookie)
    {
       Cookie = iCookie;
    }
    }

    public class Alpha
    {
public Hashtable HashCount;
public ManualResetEvent eventX;
public static int iCount = 0;
public static int iMaxCount = 0;

      public Alpha(int MaxCount)
{
      HashCount = new Hashtable(MaxCount);
      iMaxCount = MaxCount;
}

//线程池里的线程将调用Beta()方法
public void Beta(Object state)
{
   //输出当前线程的hash编码值和Cookie的值
```

```
        Console.WriteLine(" {0} {1} :", Thread.CurrentThread.GetHashCode(),((SomeState)state).Cooki
e);
      Console.WriteLine("HashCount.Count=={0}, Thread.CurrentThread.GetHashCode()=={1}", Hash
Count.Count, Thread.CurrentThread.GetHashCode());
      lock (HashCount)
      {
    //如果当前的Hash表中没有当前线程的Hash值，则添加之
    if (!HashCount.ContainsKey(Thread.CurrentThread.GetHashCode()))
          HashCount.Add (Thread.CurrentThread.GetHashCode(), 0);
       HashCount[Thread.CurrentThread.GetHashCode()] =
           ((int)HashCount[Thread.CurrentThread.GetHashCode()])+1;
     }
         int iX = 2000;
         Thread.Sleep(iX);
         //Interlocked.Increment()操作是一个原子操作，具体请看下面说明
         Interlocked.Increment(ref iCount);

         if (iCount == iMaxCount)
         {
        Console.WriteLine();
    Console.WriteLine("Setting eventX ");
    eventX.Set();
        }
  }
  }

        public class SimplePool
```

```
    {
        public static int Main(string[] args)
        {
            Console.WriteLine("Thread Pool Sample:");
            bool W2K = false;
            int MaxCount = 10;//允许线程池中运行最多10个线程
            //新建ManualResetEvent对象并且初始化为无信号状态
            ManualResetEvent eventX = new ManualResetEvent(false);
            Console.WriteLine("Queuing {0} items to Thread Pool", MaxCount);
            Alpha oAlpha = new Alpha(MaxCount);
            //创建工作项
            //注意初始化oAlpha对象的eventX属性
            oAlpha.eventX = eventX;
            Console.WriteLine("Queue to Thread Pool 0");
            try
            {
                //将工作项装入线程池
                //这里要用到Windows 2000以上版本才有的API，所以可能出现NotSupportException异常
                ThreadPool.QueueUserWorkItem(new WaitCallback(oAlpha.Beta), new SomeState(0));
                W2K = true;
            }
            catch (NotSupportedException)
            {
                Console.WriteLine("These API's may fail when called on a non-Windows 2000 system."
);
                W2K = false;
            }
```

```
            if (W2K)//如果当前系统支持ThreadPool的方法.
            {
                for (int iItem=1;iItem < MaxCount;iItem++)
                {
                    //插入队列元素
                    Console.WriteLine("Queue to Thread Pool {0}", iItem);
                    ThreadPool.QueueUserWorkItem(new WaitCallback(oAlpha.Beta), new SomeState(iIt
em));
                }
                Console.WriteLine("Waiting for Thread Pool to drain");
                //等待事件的完成，即线程调用ManualResetEvent.Set()方法
                eventX.WaitOne(Timeout.Infinite,true);
                //WaitOne()方法使调用它的线程等待直到eventX.Set()方法被调用
                Console.WriteLine("Thread Pool has been drained (Event fired)");
                Console.WriteLine();
                Console.WriteLine("Load across threads");
                foreach(object o in oAlpha.HashCount.Keys)
                    Console.WriteLine("{0} {1}", o, oAlpha.HashCount[o]);
            }
            Console.ReadLine();
            return 0;
        }
    }
  }


}
```

程序中应该引起注意的地方：

SomeState类是一个保存信息的数据结构，它在程序中作为参数被传递给每一个线程，因为你需要把一些有用的信息封装起来提供给线程，而这种方式是非常有效的。

程序出现的InterLocked类也是专为多线程程序而存在的，它提供了一些有用的原子操作。

原子操作：就是在多线程程序中，如果这个线程调用这个操作修改一个变量，那么其他线程就不能修改这个变量了，这跟lock关键字在本质上是一样的。

程序的输出结果：

我们应该彻底地分析上面的程序，把握住线程池的本质，理解它存在的意义是什么，这样才能得心应手地使用它。

posted on 2008-03-23 17:06 钢钢 阅读(2750) 评论(1) 编辑 收藏 网摘 所属分类: C#

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

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
你的大脑可以多线程工作吗?
C++内存管理与优化公开课
日程管理常见句型
你想交流项目管理经验、学习项目管理知识吗
Facebook如何管理150亿张照片
学习BEC需要具备怎样的基础?

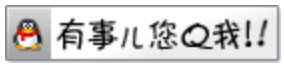
导航
博客园
首页
新随笔
联系
订阅XML
管理

统计

随笔 - 166
文章 - 27
评论 - 579
引用 - 68

公告

穷则独善其身
达则兼济天下



新闻

Delphi 2010初体验 2小时前
谷歌经济学家：搜索关键词表
明美经济正复苏 2小时前
Facebook应吸取谷歌经验避免
重蹈雅虎覆辙 3小时前
唐骏传授成功秘笈：创业要有
自己的杀手锏” 4小时前

在线词典

查词

我的最新闪存

链接

与我联系

发短消息

Nicky
Q.yuhen
Scott Guthrie
怪怪
蝈蝈俊
黄忠成
李天平
孟岩
木子
萧寒
张子阳

Tools

Convert vbnet to C# (rss)
DotNet API (rss)
My Diagram
Text Translation (rss)
WorldLingo (rss)

Websites

AspAlliance (rss)
C# Corner (rss)
C#开源资源大全 (rss)
C#语言参考视频 (rss)
cnBeta (rss)
CodePlex (rss)
codeusing.com (rss)
connectionstrings.com (rss)
CSDN DotNet
dotNET_程序小作坊 (rss)
DotNetTools (rss)
DotNet开源社区
http://csharp-source.net (rss)
ItPub论坛 (rss)
MSDN中国 (rss)
MSProject 开源技术 (rss)
planet-source-code.com (rss)
Sawin (rss)
开源中国
懒人图库 (rss)

浪曦视频在线 (rss)
中国协议分析网 (rss)

积分与排名

积分 - 216335
排名 - 224

最新评论 XML

1. Re:程序员心理小测试：你是否患上抑郁症？
天啊,我得了13分,我是不是应该去曙光医院看看啊!
--purplesu

2. Re:Donald Knuth 简介
虽然学编程一年了,但对这行业总是迷茫,怎样才算会编程也不知道
--Greenhan

3. Re:一个C#的加锁解锁示例
没看懂什么意思呀
--王维

4. Re:将GridView导入到Excel和word(完全可实现)
我的加了还是倒不出来你呢
--ASP.NET爱好

5. Re:程序员心理小测试：你是否患上抑郁症？
看来，我也得了抑郁怔拉----该怎么办呢----真的好烦啊-------
--伤心雨

阅读排行榜

1. C#多线程学习(一) 多线程的相关概念(9310)
2. IIS日志分析方法及工具(9230)
3. Javascript 刷新框架及页面的方法总集(9020)
4. (新)中华人民共和国劳动合同法(8263)

5. C#多线程学习(二) 如何操纵一个线程(7237)

评论排行榜

1. 论不使用ObjectDataSource之数据绑定控件纯代码实现是否更优化?(52)
2. 程序员心理小测试：你是否患上抑郁症？(51)
3. 电视剧《奋斗》能叫奋斗吗?45)
4. (新)中华人民共和国劳动合同法(42)
5. Donald Knuth 简介(21)