啊

线程池

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。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。**

**当一个线程开始一个活动（此活动必须完成后，其他线程才能开始）时，它调用**[**Reset**](https://msdn.microsoft.com/zh-cn/library/system.threading.eventwaithandle.reset(v=vs.90).aspx)**以将ManualResetEvent 置于非终止状态。此线程可被视为控制 ManualResetEvent。调用 ManualResetEvent 上的**[**WaitOne**](https://msdn.microsoft.com/zh-cn/library/system.threading.waithandle.waitone(v=vs.90).aspx)**的线程将阻止，并等待信号。当控制线程完成活动时，它调用**

[**Set**](https://msdn.microsoft.com/zh-cn/library/system.threading.eventwaithandle.set(v=vs.90).aspx)**以发出等待线程可以继续进行的信号。并释放所有等待线程。**