.Net主线程扑捉子线程中的异常

首先看一段C#代码：运行后发现主线程通过try{}catch{}是不能扑捉子线程中的抛出来的异常。
代码

```
 class Program
   {
      static void Main(string[] args)
      {
         try
         {
            System.Threading.Thread thread = new System.Threading.Thread(new Program().run);
            thread.Start();
         }
         catch (Exception ex)
         {
            Console.WriteLine(ex.Message);
         }
         Thread.Sleep(1000);
      }
      public void run()
      {
         throw new Exception();
      }
   }
```

为什么呢?

首先需要了解异常的实现机制：异常的实现机制是严重依赖与线程的栈的。每个线程都有一个栈，线程启动后会在栈上安装一些异常处理帧，并形成一个链表的结构，在异常发生时通过该链表可以进行栈回滚，如果你自己没有安装的话，可能会直接跳到链表尾部，那可能是CRT提供的一个默认处理帧，弹出一个对话框提示某某地址内存错误等，然后确认调试，取消关闭。

所以说，线程之间是不可能发生异常处理的交换关系的。

但是在实际的程序设计中，会牵涉到扑捉子线程的异常，那么该怎样解决这个问题呢?
代码

```
   class Program
   {
      private delegate void ExceptionHandler(Exception ex);
      private static ExceptionHandler exception;
      private static void ProcessException(Exception ex)
      {
         Console.WriteLine(ex.Message);
      }
      static void Main(string[] args)
      {
         exception = new ExceptionHandler(ProcessException);
         System.Threading.Thread thread = new System.Threading.Thread(new Program().run);
         thread.Start();
```

```
            Thread.Sleep(1000);
        }
        public void run()
        {
            try
            {
                throw new Exception();
            }
            catch (Exception ex)
            {
                if (exception != null)
                {
                    exception(ex);
                }
            }
        }
    }
```

上面使用委托的方式，间接的解决了：把子线程中的异常信息交个主线程的一个方法去执行。(通过委托方式)