Thread类可以创建和控制线程，Thread类的构造函数重载为接受ThreadStart和ParameterizedThreadStart类型的委托参数。下面我们用一个例子来解释怎样用Thread类来创建一个简单的线程

[IMG_256](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

static void Main(string[] args)

{

#region Thread无参数举例

Thread th = new Thread(ThreadChild);

th.Start();

Console.WriteLine("Main Thread Start!");

#endregion

}

static void ThreadChild()

{

Console.WriteLine("Child Thread Start!");

}

[IMG_257](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

输出结果

IMG_258

　　程序运行的结果不能保证哪个先输出，因为线程是由操作系统调度，每次哪个线程在前面可以不同

**给线程传递数据**

　　上面的例子演示了怎样用Thread类来创建一个不带传参的线程，下面我门来创建一个带传入参数的线程。给线程传递参数，有两种方式，一种是使用带ParameterizedThreadStart委托参数的Thread的构造函数，另外一种是定义一个自定义类。首先我们使用ParameterizedThreadStart委托来创建有传入参数的类。使用ParameterizedThreadStart，线程的入口（线程调用的方法）必须有一个Object类型的参数，使用Object我们首先想到的就是类型不安全。而且在执行线程的时候多半有装箱拆箱操作。管它的，我们先用这种方式来创建一个带传入参数的线程！！

　　废话再多，还是没有直接上代码来得实在，看代码！！

[IMG_259](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

static void Main(string[] args)

{

#region 使用parameterizedThreadStart委托执行带参数的委托

Thread th2 = new Thread(Thread\_param);

th2.Start(20);

#endregion

}

static void Thread\_param(object msg)

{

int message = (int)msg;

Console.WriteLine("Result:{0}",message);

}

[IMG_260](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

运行结果

IMG_261

上面创建的线程是类型不安全的，那用什么样的方式执行带传入参数的线程的方法是类型安全的呢，答案就是创建一个自定义类，在类中定义一个作为传入参数的字段，将线程的主方法定义为一个类的实例方法。然而使用这种方法就可以使用泛型来解决使用ParameterizedThreadStart的类型不安全

看招！！！！

[IMG_262](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

class Program

{

static void Main(string[] args)

{

#region 使用自定义类实现带参数的线程

MyThread<string> mythread = new MyThread<string>("Thread\_child");

Thread th3 = new Thread(mythread.ThreadChild);

th3.Start();

#endregion

}

}

class MyThread<T>

{

private T data;

public MyThread(T data)

{

this.data = data;

}

public void ThreadChild()

{

Console.WriteLine("Child Thread Start! Result:{0}",data);

}

}

[IMG_263](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

运行结果：

IMG_264

后台线程

　　Thread类默认创建的是前台线程，所以我们前面创建的线程全部都是前台线程。只要有一个前台线程在运行，应用程序的进程就在运行。如果有多个前台线程在运行，而Main()方法（主线程）结束了，应用程序的进程就仍然是激活的，直到所有前台线程完成其任务为止。

　　那后台线程呢？显然和前台线程相反。当主线程结束后，应用程序的进程就终止了，在所有前台线程结束后，后台线程就会被终止。

　　在编码的时候我们可以设置Thread类的IsBackground的属性来确定该线程是前台线程还是后台线程。当IsBackground设置为False的时候，为前台线程，设置为Ture的时候为后台线程，下面我们举例来说明前台线程和后台线程的区别。首先我们创建一个前台线程。

[IMG_265](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

static void Main(string[] args)

{

Thread th\_pre = new Thread(Thread\_pre)

{Name="Thread\_pre",IsBackground=flase};;

th\_pre.Start();

Console.WriteLine("主线程执行完成！");

}

static void Thread\_pre()

{

Console.WriteLine("子线程开始执行！");

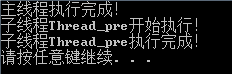
Thread.Sleep(3000);

Console.WriteLine("子线程执行完成！");

}

[IMG_266](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

运行结果



　　从上面的运行结果可以看到，当主线程执行完成后，应用程序终止前就会子线程执行完成。

　　下面我们来看看后台线程，看代码！！

[IMG_268](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

　　　　　static void Main(string[] args)

{

　　　　　　　Thread th\_back = new Thread(Thread\_back)

　　　　　　　{ Name="Thread\_back",IsBackground=true };

th\_back.Start();

Console.WriteLine("主线程执行完成！");

}

static void Thread\_back()

{

Console.WriteLine("子线程开始执行！");

Thread.Sleep(3000);

Console.WriteLine("子线程执行完成！");

}

[IMG_269](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

运行结果

IMG_270

　　从运行结果可以看出，当主线程结束后，进程就终止了，后台线程也被终止，所以没有后台线程结束的输出信息。

**控制线程**

　　我们使用Thread创建线程后，我们需要对线程进行控制。

　　1、  使用Start()方法使线程处于Running状态，线程开始执行。

　　2、  使用Join（）方法使线程处于WaitSleepJoin状态，在继续执行标准的 COM 和 SendMessage 消息泵处理期间，阻塞调用线程，直到某个线程终止或经过了指定时   间为止。

　　3、  使用Sleep()方法，也会使线程处于WaitSleepJoin状态，在经历Sleep()方法定义的时间段后，线程就会被再次唤醒。、

　　4、  使用Abort()方法，会使线程处于ResetAbort()状态，线程在接到这个命令的时候，会抛出一个ThradAbordException类型的异常。

各位看官，看代码

[IMG_271](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

using System;using System.Text;using System.Threading;namespace ConsoleThreadContral

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("mainThread Start!");

Thread th = new Thread(newThread);

th.Start();//将当前实例的状态更改为 ThreadState.Running。

Console.WriteLine("newThread State:{0}",th.ThreadState);

th.Join(100);//在继续执行标准的 COM 和 SendMessage 消息泵处理期间，阻塞调用线程，直到某个线程终止或经过了指定时间为止。

Console.WriteLine("newThread State:{0}", th.ThreadState);

th.Abort();//在调用此方法的线程上引发 ThreadAbortException，以开始终止此线程的过程。 调用此方法通常会终止线程。

Console.WriteLine("newThread State:{0}", th.ThreadState);

}

static void newThread()

{

Console.WriteLine("newThread Start!");

Thread.Sleep(10000);

Console.WriteLine("newThread Complete!");

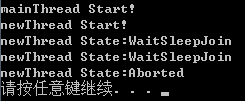
}

}

}

[IMG_272](http://www.cnblogs.com/lvcy/archive/2012/06/16/javascript:void(0);" \o "复制代码)

运行结果



http://www.cnblogs.com/lvcy/archive/2012/06/16/2551539.html