**Unity3D教程：C#多线程**

Posted on 2013年07月22日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b) /被围观 26 次

**什么是进程？**

当一个程序开始运行时，它就是一个进程，进程包括运行中的程序和程序所使用到的内存和系统资源。而一个进程又是由多个线程所组成的。

**什么是线程？**

线程是程序中的一个执行流，每个线程都有自己的专有寄存器(栈指针、程序计数器等)，但代码区是共享的，即不同的线程可以执行同样的函数。

**什么是多线程？**

多线程是指程序中包含多个执行流，即在一个程序中可以同时运行多个不同的线程来执行不同的任务，也就是说允许单个程序创建多个并行执行的线程来完成各自的任务。

**多线程的好处：**

可以提高CPU的利用率。在多线程程序中，一个线程必须等待的时候，CPU可以运行其它的线程而不是等待，这样就大大提高了程序的效率。

**多线程的不利方面：**

线程也是程序，所以线程需要占用内存，线程越多占用内存也越多；

多线程需要协调和管理，所以需要CPU时间跟踪线程；

线程之间对共享资源的访问会相互影响，必须解决竞用共享资源的问题；

线程太多会导致控制太复杂，最终可能造成很多Bug；

接下来将对C#编程中的多线程机制进行探讨。为了省去创建GUI那些繁琐的步骤，更清晰地逼近线程的本质，接下来的所有程序都是控制台程序，程序最后的Console.ReadLine()是为了使程序中途停下来，以便看清楚执行过程中的输出。

任何程序在执行时，至少有一个主线程。

一个直观印象的线程示例：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | <span style="font-size:18px;">**using** System; |
| 02 | **using** System.Threading; |
| 03 |  |
| 04 | **namespace** ThreadTest |
| 05 | { |
| 06 | **class** RunIt |
| 07 | { |
| 08 | [STAThread] |
| 09 | **static** **void** Main(**string**[] args) |
| 10 | { |
| 11 | Thread.CurrentThread.Name="System Thread";*//给当前线程起名为"System Thread"* |
| 12 | Console.WriteLine(Thread.CurrentThread.Name+"'Status:"+Thread.CurrentThread.ThreadState); |
| 13 | Console.ReadLine(); |
| 14 | } |
| 15 | } |
| 16 | } |

输出如下：

System Thread's Status:Running

在这里，我们通过Thread类的静态属性CurrentThread获取了当前执行的线程，对其Name属性赋值“System Thread”，最后还输出了它的当前状态（ThreadState）。

所谓静态属性，就是这个类所有对象所公有的属性，不管你创建了多少个这个类的实例，但是类的静态属性在内存中只有一个。很容易理解CurrentThread为什么是静态的——虽然有多个线程同时存在，但是在某一个时刻，CPU只能执行其中一个。

在程序的头部，我们使用了如下命名空间：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | **using** System; |
| 2 |  |
| 3 | **using** System.Threading; |

在.net framework class library中，所有与多线程机制应用相关的类都是放在System.Threading命名空间中的。如果你想在你的应用程序中使用多线程，就必须包含这个类。

我们通过其中提供的Thread类来创建和控制线程，ThreadPool类用于管理线程池等。

Thread类有几个至关重要的方法，描述如下：

Start()：启动线程；

Sleep(int)：静态方法，暂停当前线程指定的毫秒数；

Abort()：通常使用该方法来终止一个线程；

Suspend()：该方法并不终止未完成的线程，它仅仅挂起线程，以后还可恢复；

Resume()：恢复被Suspend()方法挂起的线程的执行；