## 第14章 并发和异步:

大多数应用程序需要处理一次发生的多件事情（并发）。在本章中，我们从基本的先决条件开始，即线程和任务的基础，然后详细描述异步的原理和C#的异步函数。

在第21章中，我们将更详细地回顾多线程，在第22章中，将介绍并行编程的相关主题。

### 简介

下面是最通常的并发场景:

*编写响应式用户界面*

在Windows Presentation Foundation（WPF）、移动和Windows窗体应用程序中，必须与运行用户界面的代码同时运行耗时的任务，以保持响应能力。

*允许同时处理请求*

在服务器上，客户端请求可以并发到达，因此必须并行处理以保持可伸缩性。如果您使用ASP.NET核心或Web API，运行时会自动为您执行此操作。但是，您仍然需要了解共享状态（例如，使用静态变量进行缓存的效果）。

*并行编程*

如果工作负载在内核之间分配，执行密集计算的代码可以在多核/多处理器计算机上更快地执行（第22章专门介绍这一点）。

*推测性执行*

在多核计算机上，有时可以通过预测可能需要做的事情，然后提前完成来提高性能。LINQPad使用这种技术来加快新查询的创建速度。一种变体是并行运行许多不同的算法，这些算法都能解决同一任务。无论哪一个先完成，都会“获胜”——当你无法提前知道哪种算法执行得最快时，这是有效的。

程序可以同时执行代码的通用机制称为多线程。CLR和操作系统都支持多线程，这是并发中的一个基本概念。因此，了解线程的基本知识，特别是线程对共享状态的影响是至关重要的。

### 线程(Threading)

线程是一个可以独立于其他线程进行的执行路径

每个线程都在一个操作系统进程中运行，该进程提供了一个程序运行的独立环境。对于单线程程序，只有一个线程在进程的隔离环境中运行，因此线程可以独占访问它。对于多线程程序，多个线程在单个进程中运行，共享相同的执行环境（尤其是内存）。这在一定程度上就是多线程之所以有用的原因：例如，一个线程可以在后台提取数据，而另一个线程在数据到达时显示数据。此数据称为共享状态。

#### 创建线程(Creating a Thread)

客户端程序（Console、WPF、UWP或Windows窗体）在操作系统自动创建的单个线程（“主”线程）中启动。在这里，除非你做了其他事情，通过创建更多的线程（直接或间接）, 它作为一个单线程应用程序度过了它的生命。

您可以通过实例化thread对象并调用其start方法来创建和启动新线程。Thread最简单的构造函数接受ThreadStart委托：一个无参数的方法，指示应该从哪里开始执行。以下是一个示例：

// NB: All samples in this chapter assume the following namespace imports:

using System;

using System.Threading;

Thread t = new Thread (WriteY); // Kick off a new thread

t.Start(); // running WriteY()

// Simultaneously, do something on the main thread.

for (int i = 0; i < 1000; i++) Console.Write ("x");

void WriteY()

{

for (int i = 0; i < 1000; i++) Console.Write ("y");

}

// Typical Output:

xxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyy

yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

yyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

...

主线程创建一个新线程t，在该线程上运行一个重复打印字符y的方法。同时，主线程重复打印字符x，如图14-1所示。在单核计算机上，操作系统必须为每个线程分配“时间片”（在Windows中通常为20ms）来模拟并发性，从而产生重复的x和y块。在多核或多处理器计算机上，这两个线程可以真正并行执行（受计算机上其他活动进程的竞争），尽管在本例中，由于控制台处理并发请求的机制的微妙性，您仍然会得到重复的x或y块。

