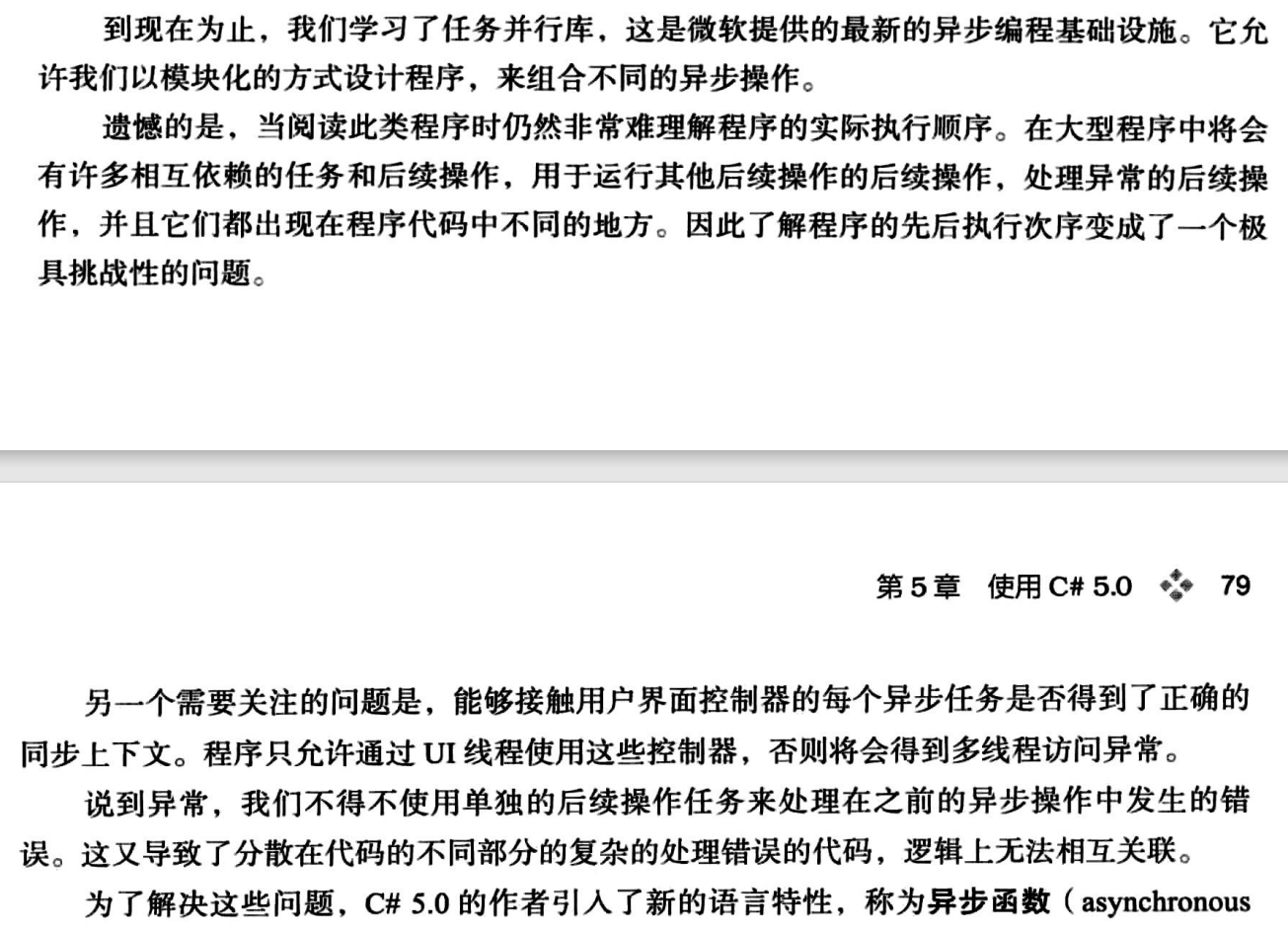
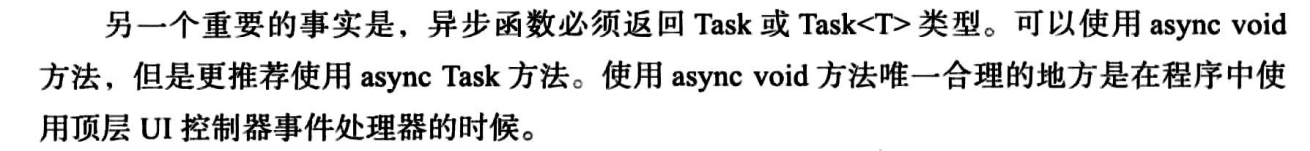
为什么需要异步函数：



如何创建一个异步函数：

在返回类型前、修饰符后添加async关键字，并且异步函数必须返回类型必须是Task<T>或者Task类型或者void类型



使用async方法的内部可以使用await操作符，该操作符可以和TPL的任务一起工作，并获取任务的异步操作的结果。

当在异步函数中遇到await关键字时，会发生以下几个步骤：

1.暂停当前方法的执行：遇到await关键字时，异步函数会暂停当前方法的执行，并将控制权返回给调用者。这意味着异步方法会暂时停止执行，并允许其他任务继续执行。

2.等待异步操作的完成：await关键字后面通常是一个返回Task或Task<T>的异步操作。异步函数会等待这个异步操作的完成。在等待期间，异步方法会释放当前线程，并允许其他任务继续执行。

3.恢复执行：一旦异步操作完成，异步方法会从之前暂停的地方恢复执行，并继续执行await之后的代码。这意味着异步方法会继续执行剩余的逻辑。

[【C# TAP 异步编程】二 、await运算符已经可等待类型Awaitable - 小林野夫 - 博客园 (cnblogs.com)](https://www.cnblogs.com/cdaniu/p/15703901.html)

Ps：

1.await操作符只能在async标记的函数内使用，并且每个async函数内至少要有一个await操作符，否则会出现编译错误。

2.调用async方法有显著的性能消耗，async方法比一般方法慢40~50倍。

使用await操作符获取异步任务结果：

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

从结果可知，代表TPL和异步函数的两个方法在结果上是等价的，

书上的解释：

文本

描述已自动生成

异步函数也可以使用lambda表达式：

电脑萤幕的截图

描述已自动生成

对连续的异步任务使用await操作符：

图形用户界面, 文本

描述已自动生成

文本

描述已自动生成

文本

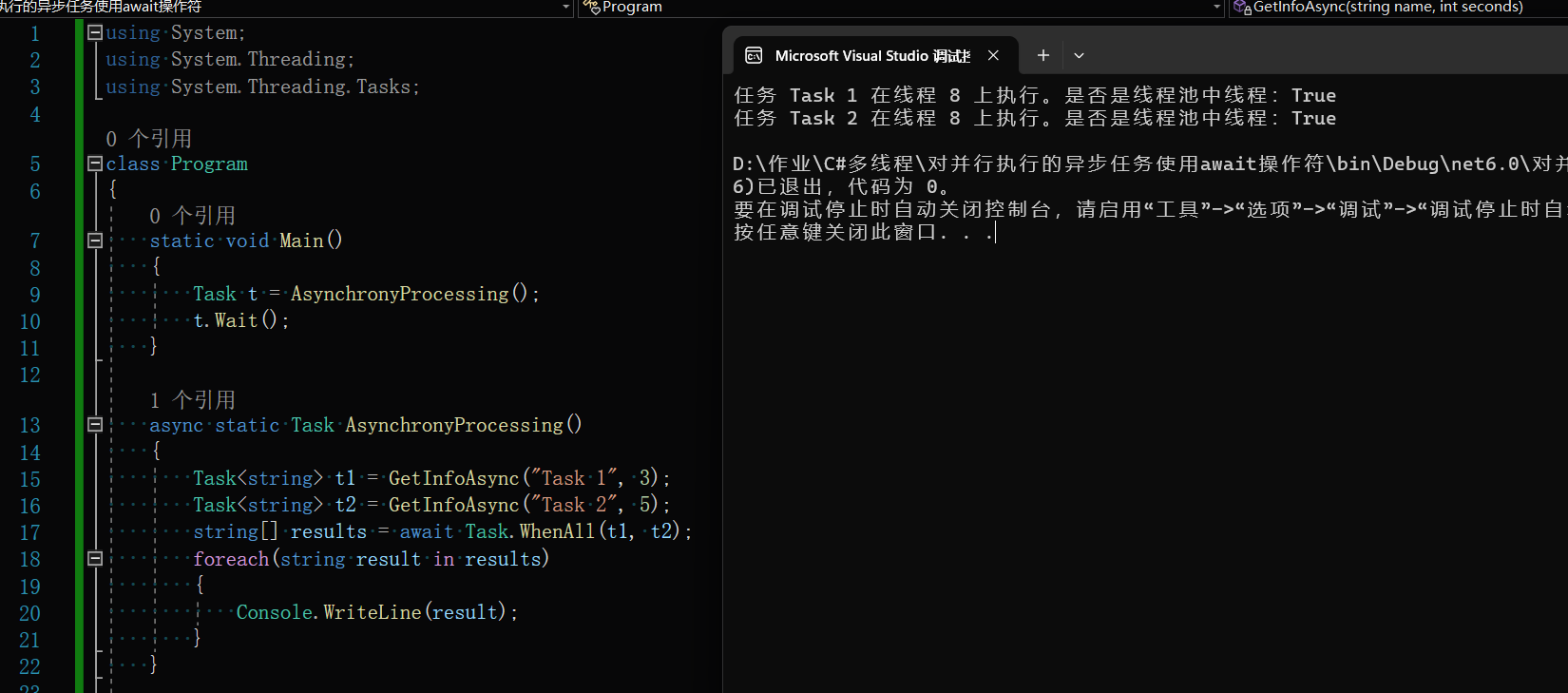
描述已自动生成

这是书上的解释：

图形用户界面, 文本, 信件

描述已自动生成

对并行执行的异步任务使用await操作符：



电视萤幕的截图

描述已自动生成

这两种情况下执行的结果也不一样，

电脑萤幕截图

描述已自动生成

电脑的屏幕截图

描述已自动生成

解释：

文本

描述已自动生成

处理异步操作中的异常：

第一种方式：try-catch，这种方式针对只有一个异常的情况

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

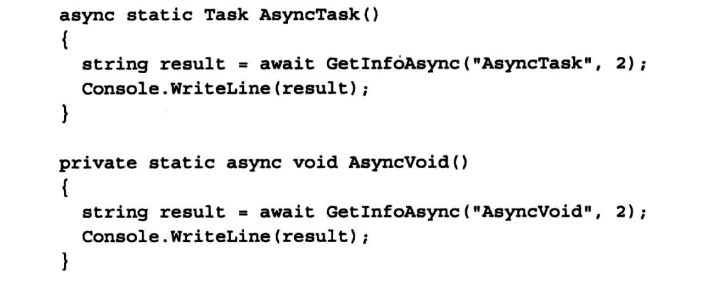
第二种方式：在第一种方式的情况下变为可以获取所有异常

文本, 信件

描述已自动生成

使用AggregateException的Flatten把层级异常放入一个列表中然后可以提取所有的底层异常

使用Async void方法：



为什么说“使用void返回类型的async方法是危险的”：

1．

文本, 信件

描述已自动生成

2.

文本

描述已自动生成

设定一个自定义的awaitable类型：

什么是awaitable类型：

如果有一个表达式t满足下面任意一个条件，则认为是awaitable：

文本, 信件

描述已自动生成

Ps：第一条指dynamic类型

例子：

文本

描述已自动生成

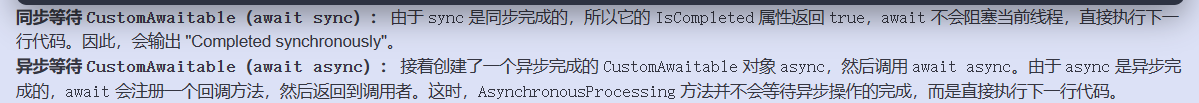
电脑屏幕截图

描述已自动生成

文本

描述已自动生成

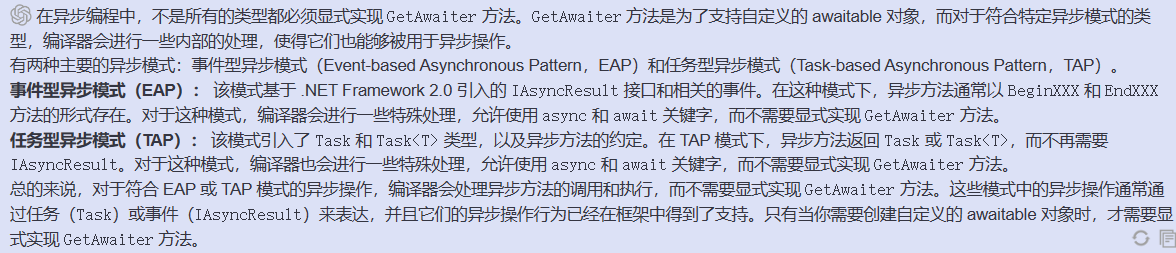
解释：



使用await等待一个异步操作如果被等待对象的IsCompleted属性为false则表示异步操作需要异步执行，如果为true则表示异步操作已经完成不需要进行异步等待，因此await会直接执行后续代码。

因此这段代码的执行逻辑是：

await 调用 async.GetAwaiter() 方法，获得一个 CustomAwaiter 类型的对象，然后调用该对象的 OnCompleted 方法，注册一个回调函数，这个函数的内容是在实现OnCompleted方法时定义的，以在异步操作完成后执行相应的逻辑。



Ps：

OnCompleted 方法所需的委托（Action）通常不需要外部显式地传入。这是因为在异步等待时，await 关键字会隐式地将后续代码包装成一个委托，然后通过 OnCompleted 方法传递给 awaitable 对象的 awaiter。当调用 await 时，编译器会生成异步状态机，将 await 后面的代码封装成一个委托，并传递给 OnCompleted 方法。这个委托的任务是在异步操作完成后继续执行异步方法中 await 后面的代码。