

XDK 中协同程序示例

# 描述

本示例演示了 XDK 中协同程序的使用。该代码实现了协程技术规范中的部分功能，包括：

* 基本 future 类型
* 基本 awaitable 类型
* 重载 co\_await 运算符以使用现有类型

其中还展示了如何使用 C / WinRT 基本头中定义的 awaitable 类型进行异步编程。

# 使用样本

协同程序是一个编译器扩展，需要使用 /await 编译器开关。使用 Visual Studio 时，可以在项目属性的“其他选项”(Additional options) 部分添加此项。协同程序至少需要使用 Visual Studio 2015 Update 3 或 Visual Studio 2017。本示例为 Visual Studio 2017（15.5 更新）或更高版本构建，使用这些版本效果最佳。

# 使用样本

本示例中提供了以下操作。

|  |  |
| --- | --- |
| 操作 | 游戏手柄控制 |
| 启动帐户选择器 | A | |
| 查询主机上的可用用户 | X | |
| 启动虚拟键盘 | Y | |
| 退出示例 | 左侧扳机键 + 右侧扳机键 + 右缓冲键 | |

各操作演示了使用 awaitable 类型和协程以处理程序流的不同机制。

## 使用重载 co\_await 运算符处理帐户选择器

一元 **co\_await** 运算符用于请求协程挂起。执行返回到调用该协程的函数。此运算符只能用于特定类型，即 **awaitable** 类型。Awaitable 类型必须包含以下公共成员函数。

* **await\_ready** - 调用此函数以确定在使用 co\_await 运算符挂起协程时是否需要挂起。如操作完成的速度超过预期，则此函数可返回 **true** 以发信号通知无需挂起。否则，函数可返回 **false** 以挂起协程。
* **await\_suspend** - 协程挂起时调用此函数。如果 **await\_ready** 返回 **true**，则不会调用此函数。这可用于处理任何必要的操作，例如在协程挂起前将信息存储在 awaitable 对象中。
* **await\_resume** - 协程恢复时，此函数用于确定 co\_await 运算符返回的值。

用于打开帐户选择器的 API 会返回一个 **IAsyncOperation<AccountPickerResult^>^** 对象，但这不是 awaitable 类型。客户端代码提供了全局**运算符 co\_await** 重载，以允许 co\_await 运算符用于此类对象。它提供了一个替代对象 **accountpicker\_await\_adapter**，这是 awaitable 对象，用于处理挂起逻辑。**await\_ready** 函数会检查 **IAsyncOperation** 的状态：如果已完成，则无需挂起协程，它可以运行完成。**await\_suspend** 函数使用 IAsyncOperation 的**已完成**事件创建新上下文，以在帐户选择器关闭后继续协程。在本示例中，**accountpicker\_await\_adapter** 的实现基于在 XDK 随附的 C / WinRT 投影头中定义的 **winrt::accountpicker\_await\_adapter** 结构体。

## 处理自定义 future 类型的虚拟键盘

协程签名外观与任何其他子例程相同。它有一个返回类型、一个名称和一个参数列表。唯一的区别是，返回类型必须是 **future** 类型。可使用 **std::future** 类型，且大多数用途适用。本示例实现了基本 future 类型 **awaitable\_future**，可作为其他实现的参考。与 awaitable 类型不同，future 类型不需要特定公共接口。但是，future 类型必须在其范围内提供名称为 **promise\_type** 的结构体。Promise\_type 必须提供以下公共接口。

* **get\_return\_object** - 此函数在调用协程时调用，但要在协程主体开始执行之前。它用于处理 future 对象的构造，如果协程挂起，则该对象返回给调用者。
* **（可选）get\_return\_object\_on\_allocation\_failure** - 提供此函数可方便您处理调用协同时上下文无可用内存的情况。
* **initial\_suspend** - 在协同程序的主体开始执行前调用，以确定协程是否应立即挂起并将执行返回给调用者。如果此函数返回 **true**，将开始执行协程。如果返回 **false**，则协程将在开始执行前挂起，但要在创建 future 对象后。
* **final\_suspend** - 执行运行到协程末尾或遇到 **co\_return** 语句时调用。如果此函数返回 **true**，则协程上下文将立即释放，promise\_type 使得其包含的任何数据无效。如果此函数返回 **false**，则调用者仍可查询 promise\_type 保存的数据，但调用者也将负责清除协程上下文。
* **RETURN\_VALUE** 或 **return\_void** - promise 类型必须实现这些函数中的*一个且唯一一个*。如果协程使用了 **co\_return** 语句将值返回给调用者，那么该值传递至 **RETURN\_VALUE** 以存储在 promise\_type 中以供调用者查询。如果没有 co\_return 语句或 co\_return 语句没有值，则必须实现 **return\_void**。不论是哪种情况，都会在协程完成执行时 **final\_suspend** 之前调用。
* **（可选）operator new** - 如果这是在 promise\_type 中提供，那么它将用于为协程上下文分配内存而不是 **::operator new**。对于跟踪内存使用情况或提供优化的分配器，这非常有用。
* **（可选）运算符删除** - 如果提供了重载 operator new，则必须在释放协程上下文后提供以释放内存。

与 **accountpicker\_await\_adapter** 不同，**keyboard\_await\_adapter** 由客户端代码提供，异步 TCUI 函数完成后不会自动恢复协程。相反，需要调用者在适当时恢复协程。

## 在同一个函数范围内用多线程查询用户集合

查询 User::Users 可能需要很长时间，且如果它在呈现线程上访问，则会导致挂结。C / WinRT 投影头包含许多 awaitable 类型，即使在不使用 C / WinRT 投影的游戏中，也可用于在单个函数内无缝处理线程。使用带有 co\_await 运算符的 **resume\_background** 对象实例，调用协程的函数立即恢复执行，且会创建一个新线程以立即继续执行协程。这会将访问 User::Users 卸载到协程主体其余部分的相同范围内的后台线程上。有关其他有用的 awaitable 类型源代码，请参阅 XDK 随附的 C / WinRT 投影头中的 base.h。

# 更新历史记录

2018 年 2 月 - 首次发布

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。