异步 I/O

*此示例与2015年11月的Xbox One XDK或更高版本兼容*

# 描述

此样本演示了使用重叠 I/O 执行异步文件操作的三种主要模式。三个主要模式为

1. 等待与操作相关的事件
2. 定期查询操作状态
3. 在操作完成时使用回调函数

# 使用样本

运行样本，将循环访问每个模式。可以逐步执行调试器中的代码，以准确了解所发生的情况。代码大量记录了正在发生的事件和原因。

# 实施说明

演示的所有代码都包含在 OverlappedSample 类中，定义在 OverlappedSample.h 中。该类的实现分为四个源文件。

1. OverlappedSample.cpp
   1. 共享管理对每种模式使用的内部数据结构。有些示例是未决重叠结构的打开文件句柄、线程管理和缓存管理。
2. WaitOverlapped.cpp - 包含两个主要功能
   1. EventTypeThreadProc - 基于等待模式创建异步请求的线程过程。每次不断更新 c\_maxRequestsInFlight 的未决请求。达到这个数字时，将停止创建新请求，直到有些未决请求完成为止。
   2. WaitForEventOverlappedToFinish - 执行实际等待请求完成的步骤。将同时等待至少一个未决请求完成，然后清除该请求。
      1. 可从代码中的任何其他位置和任何其他线程调用此函数。仅能从 EventTypeThreadProc 调用，以便于在样本中实现。
3. QueryOverlapped.cpp - 包含两个主要功能
   1. QueryTypeThreadProc - 基于查询模式创建异步请求的线程过程。每次不断更新 c\_maxRequestsInFlight 的未决请求，将停止创建新请求，直到有些未决请求完成。
   2. QueryForOverlappedFinish - 扫描所有未决请求以查看已完成请求，并在完成后进行处理。
      1. 可从代码中的任何其他位置、任何其他线程以任何频率调用此函数。仅能从频率快速的 QueryTypeThreadProc 调用，以便于在样本中实现。
4. AlertableOverlapped.cpp - 包含三个主要功能
   1. AlertableTypeThreadProc - 基于可报警模式创建异步请求的线程过程。每次不断更新 c\_maxRequestsInFlight 的未决请求，将停止创建新请求并在可报警状态下进入休眠模式，直到有些未决请求完成。
   2. FileIOCompletionRoutine - 用作 OS 和 OverlappedSample 类实例之间的钩连点。这是请求完成时作为回调传递给 OS 的函数。
      1. 可在任意时间点和已进入可报警状态的任意线程上调用此函数。
   3. AlertableCompletionRoutine - 请求完成时由 FileIOCompletionRoutine 调用。将通过未决列表，将已完成的请求与未决请求进行匹配。
      1. 此函数未添加任何线程安全性，因为只有 AlertableTypeThreadProc 线程进入可报警状态。这意味着只有该线程才会收到回调。
      2. 根据您的实现，可能需要将线程安全性添加到回调例程中。

等待和查询模式非常相似。这两种模式都使用内部事件在请求完成时发出信号。还可以等待可配置的时间。主要区别在于等待模式可使用 WaitForMultipleObjects，这意味着可同时等待所有未决请求。带 GetOverlappedResult 的查询模式一次只能查看一个请求。

由于这两种模式都使用手动重置事件来发送信号，因此可以混合这两种模式。例如，主文件加载系统可对所有未决请求使用等待模式，而另一个线程可使用查询模式进行关键请求。如果混合了两种模式，由于手动重置事件，应使用某种形式的线程安全性，可同时返回完成。

强烈建议不要使用自动重置事件。很容易陷入 WaitFor###Object 返回完成的问题。但是，事件随后将切换为无信号，以致于下一次调用 GetOverlappedResult 失败，因为事件不再发出信号。

# 更新历史记录

2016年10月初次发布