非同期I/O

*このサンプルは、2015 年 11 月の Xbox One XDK 以降と互換性があります。*

# 概要

このサンプルでは、オーバーラップした I/O を使用して非同期ファイル操作を実行するための 3 つの主要パターンを紹介します。3 つの主なパターンは以下のとおりです。

1. 操作に関連したイベントを待機する
2. 操作のステータスを定期的に照会する
3. 操作完了時にコールバック関数を使用する

# サンプルの使用

サンプルを実行すると、各パターンが繰り返されます。何が起こっているかを正確に追跡するために、デバッガのコードのステップ実行が可能です。何が起こっているか、その理由がコードに詳しく記載されています。

# 実装上の注意

デモ用のコードはすべて OverlappedSample クラスに含まれています。定義は OverlappedSample.h にあります。クラスの実装は 4 つのソース ファイルに分割されています。

1. OverlappedSample.cpp
   1. 各パターンに使用される内部データ構造の共有管理。例としては、オープン ファイル ハンドル、スレッド管理、および保留中のオーバーラップ構造のキャッシュ管理が挙げられます。
2. WaitOverlapped.cpp - 2 つの主要な関数が含まれています。
   1. EventTypeThreadProc - 待機パターンに基づいて非同期要求を作成するスレッド プロシージャ。一度に c\_maxRequestsInFlight 件までの保留中の要求を維持します。この数に達すると、保留中の要求が完了するまで新しい要求の作成を中止します。
   2. WaitForEventOverlappedToFinish - 要求が完了するまでの実際の待機を実行するプロシージャ。保留中の要求の少なくとも 1 つが終了するのを同時に待ってから、その要求をクリーンアップします。
      1. この関数は、コード内の他の場所や他のスレッドから呼び出すことができます。サンプルでの実装を容易にするために、EventTypeThreadProc からのみ呼び出されます。
3. QueryOverlapped.cpp - 2 つの主要な関数が含まれています。
   1. QueryTypeThreadProc - クエリ パターンに基づいて非同期要求を作成するスレッド プロシージャ。一度に c\_maxRequestsInFlight 件までの保留中の要求を維持し、保留中の要求が完了するまで新しい要求の作成を中止します。
   2. QueryForOverlappedFinish - 保留中のすべての要求をスキャンして、どの要求が完了しているかを確認し、完了している場合はそれらを処理します。
      1. この関数は、任意の頻度でコード内の他の場所や他のスレッドから呼び出すことができます。サンプルでの実装を容易にするために、QueryTypeThreadProc からのみ迅速かつ頻繁に呼び出されます。
4. AlertableOverlapped.cpp - 3 つの主要な関数が含まれています。
   1. AlertableTypeThreadProc - 警告パターンに基づいて非同期要求を作成するスレッド プロシージャ。一度に c\_maxRequestsInFlight 件までの保留中の要求を維持し、保留中の要求が完了するまで、新しい要求の作成を中止して警告状態でスリープになります。
   2. FileIOCompletionRoutine - OS と OverlappedSample クラス インスタンスの間のフックとして機能します。これは、要求が終了したときのコールバックとして OS に渡される関数です。
      1. この関数は、警告状態になった任意の時点および任意のスレッドで呼び出すことができます。
   3. AlertableCompletionRoutine - 要求が完了したときに FileIOCompletionRoutine によって呼び出されます。保留中のリストを調べて、完了した要求を保留中の要求と照合します。
      1. AlertableTypeThreadProc スレッドのみが警告状態になるため、この関数にスレッド セーフは追加されていません。これは、そのスレッドだけがコールバックを受け取ることを意味します。
      2. 実装によっては、コールバック ルーチンにスレッド セーフを追加する必要がある可能性があります。

待機パターンとクエリ パターンはよく似ています。どちらも内部イベントを使用して、要求が終了したことを知らせます。どちらも設定可能な時間だけ待機できます。主な違いは、待機パターンは WaitForMultipleObjects を使用できることです。これは、保留中のすべての要求を同時に待機できることを意味します。GetOverlappedResult を使用したクエリ パターンは、一度に 1 つの要求しかチェックできません。

どちらも通知に手動リセット イベントを使用するため、パターンを混在させることが可能です。たとえば、メイン ファイル読み込みシステムではすべての保留中の要求に待機パターンを使用できますが、別のスレッドでは重要な要求にクエリ パターンを使用できます。パターンが混在している場合は、手動リセット イベントのために何らかの形のスレッド セーフを使用する必要があります。両方のパターンが同時に「complete」を返す可能性があります。

自動リセット イベントの使用はお勧めできません。WaitFor###Object が「complete」を返すという問題は非常に頻繁に発生します。ただし、その後イベントは通知されないように切り替わるため、イベントが通知されないことにより、GetOverlappedResultへの次の呼び出しが失敗します。

# 更新履歴

初回リリース 2016 年 10 月