비동기 I/O

*이 샘플은 Xbox One XDK 2015 년 11 월 이상 버전과 호환됩니다.*

# 설명

이 샘플에서는 겹쳐진 I/O를 사용하여 비동기 파일 작업을 수행하는 3가지 주요 패턴을 보여줍니다. 3가지 주요 패턴은 다음과 같습니다

1. 작업과 관련된 이벤트 대기 중
2. 주기적으로 작업 상태 쿼리
3. 작업이 완료되면 콜백 함수가 사용됨

# 샘플 사용하기

샘플을 실행하면 각 패턴을 반복합니다. 디버거의 코드를 단계별로 실행하여 정확히 어떤 일이 일어나는지 확인할 수 있습니다. 이 코드는 현재 일어나고 있는 일과 이유에 대해 자세히 기록되어 있습니다.

# 구현 정보

데모 용 코드는 모두 OverlappedSample 클래스에 포함되어 있으며 정의는 OverlappedSample.h에 있습니다. 클래스의 구현은 4개의 소스 파일로 나뉩니다.

1. OverlappedSample.cpp
   1. 각 패턴에 사용되는 내부 데이터 구조의 공유 관리. 대기 중인 겹침 구조에 대한 열린 파일 핸들, 스레드 관리 및 캐시 관리 등이 일부 예입니다.
2. WaitOverlapped.cpp - 두 가지 주요 기능을 포함합니다.
   1. EventTypeThreadProc - 대기 패턴을 기반으로 비동기 요청을 생성하는 스레드 프로시저입니다. 한 번에 최대 c\_maxRequestsInFlight 보류 중인 요청을 유지합니다. 이 숫자에 도달하면 일부 보류 중인 요청이 완료될 때까지 새 요청 만들기가 중지됩니다.
   2. WaitForEventOverlappedToFinish - 요청이 완료될 때까지 실제 대기를 수행하는 프로시저입니다. 보류 중인 요청 중 적어도 하나가 완료될 때까지 동시에 대기하고 그 요청을 처리합니다.
      1. 이 함수는 코드 및 다른 스레드의 다른 위치에서 호출할 수 있습니다. 이 샘플에서는 구현하기 쉽도록 EventTypeThreadProc에서만 호출됩니다.
3. QueryOverlapped.cpp - 두 가지 주요 기능을 포함합니다.
   1. QueryTypeThreadProc - 쿼리 패턴을 기반으로 비동기 요청을 생성하는 스레드 프로시저입니다. 한 번에 최대 c\_maxRequestsInFlight 보류 중인 요청을 유지하므로 일부 보류가 완료될 때까지 새 요청 만들기가 중지됩니다.
   2. QueryForOverlappedFinish - 보류 중인 모든 요청을 검사하여 완료된 요청을 확인하고 완료되면 처리합니다.
      1. 이 함수는 코드의 다른 위치, 다른 스레드 및 임의의 빈도에서 호출될 수 있습니다. 샘플에서 구현하기 쉽도록 빠른 빈도로 QueryTypeThreadProc에서만 호출됩니다.
4. AlertableOverlapped.cpp - 3가지 주요 기능을 포함합니다.
   1. AlertableTypeThreadProc - 경고 가능한 패턴을 기반으로 비동기 요청을 생성하는 스레드 프로시저입니다. 한 번에 최대 c\_maxRequestsInFlight 보류 중인 요청을 유지하고, 보류 중인 요청이 완료될 때까지 새로운 요청 생성을 중지하고 경고 가능한 상태로 절전 모드가 됩니다.
   2. FileIOCompletionRoutine - OS와 OverlappedSample 클래스 인스턴스 간의 후크 역할을 합니다. 이것은 요청이 완료되었을 때의 콜백으로서 OS에 전달되는 함수입니다.
      1. 이 함수는 어느 시점에서나 경고 상태로 들어간 모든 스레드에서 호출될 수 있습니다.
   3. AlertableCompletionRoutine - 요청이 완료되면 FileIOCompletionRoutine에 의해 호출됩니다. 보류 중인 목록을 살펴보고 완료된 요청을 보류 중인 요청과 일치시킵니다.
      1. AlertableTypeThreadProc 스레드만 경고 가능 상태가 되기 때문에 스레드 안전성이 이 함수에 추가되지 않았습니다. 즉, 해당 스레드만 콜백을 수신합니다.
      2. 구현에 따라 스레드 안전성을 콜백 루틴에 추가해야 할 수도 있습니다.

Wait 및 Query 패턴은 매우 유사합니다. 둘 다 요청이 완료되면 신호를 보내는 내부 이벤트를 사용합니다. 또한 구성 가능한 시간 동안 기다릴 수도 있습니다. 주요 차이점은 Wait 패턴이 WaitForMultipleObjects를 사용할 수 있다는 것입니다. 즉, 대기중인 모든 요청을 동시에 기다릴 수 있습니다. GetOverlappedResult를 사용하는 쿼리 패턴은 한 번에 하나의 요청만 검사할 수 있습니다.

둘 다 신호를 위해 수동 리셋 이벤트를 사용하기 때문에 패턴을 혼합할 수 있습니다. 예를 들어, 메인 파일 로딩 시스템은 보류 중인 모든 요청에 대해 대기 패턴을 사용할 수 있는 반면 다른 스레드는 중요한 요청에 대해 쿼리 패턴을 사용할 수 있습니다. 패턴이 혼합되면 수동 리셋 이벤트로 인해 스레드 안전의 일부 형식을 사용해야 하며 동시에 완료할 수 있습니다.

자동 리셋 이벤트 사용시에는 사용하지 않는 것이 좋습니다. WaitFor### Object가 완료되면 문제가 발생하기 쉽습니다. 그러나 이 이벤트는 더 이상 신호가 보내지지 않으므로 GetOverlappedResult에 대한 다음 호출이 실패하도록 신호를 보내지 않는 것으로 전환됩니다.

# 업데이트 기록

초기 출시 2016년 10월