14장. 스레드 동기화 기법2

01. 실행순서에 있어서의 동기화

스레드의 실행순서를 동기화 한다는 것은 “메모리에 접근하는 스레드의 실행순서를 동기화 하는 것” 이다. 즉, 실행 순서의 동기화는 메모리 접근 동기화를 포함하는 개념이다. 다만 포커스가 실행 순서에 맞춰져 있기 때문에, “실행 순서의 동기화”로 표현 되는 것이다.

**이벤트(Event) 기반 동기화**

Windows 개발자 들은 스레드의 실행순서를 동기화 한다고 한다면, 가장 먼저 Event 기반 동기화를 생각한다.

|  |
| --- |
| HANDLE CreateEvent(  LPSECURITY\_ATTRIBUTES lpEventAttributes,  BOOL bManualReset,  BOOL bInitialState,  LPCTSTR lpName  ); |

- lpEventAttributes : 보안속성 지정, 핸들을 자식 프로세스에 상속하고자 할 경우 NULL 지정

- bManualResrt : TRUE일 경우 수동 리셋(Manual-Reset), FALSE를 전달하면 자동 리셋(Auto-Reset)모드 오브젝트 생성

- bInitialState : 이벤트의 초기화 상태를 결정, TREU면 Signaled 상태, FALSE면 Non-Signaled 상태로 생성

- lpName : 이름을 줄 경우 하용하는 인자. NULL을 전달하면 이름 없는 오브젝트

생성은 위 함수인 CreateEvent 함수를 사용하며 이벤트 오브젝트를 소멸시킬 때에는 다른 커널 오브젝트와 마찬가지로 CloseHandle 함수를 사용한다.

**이벤트 오브젝트의 가장 큰 특징**은 수동 리셋 모드와 자동 리셋 모드를 선택하여 생성할 수 있다는 것이다.

만약 이벤트 오브젝트가 Signaled 상태가 되어 블로킹 상태에 있던 스레드가 빠져 나온다면, 빠져 나온 이후

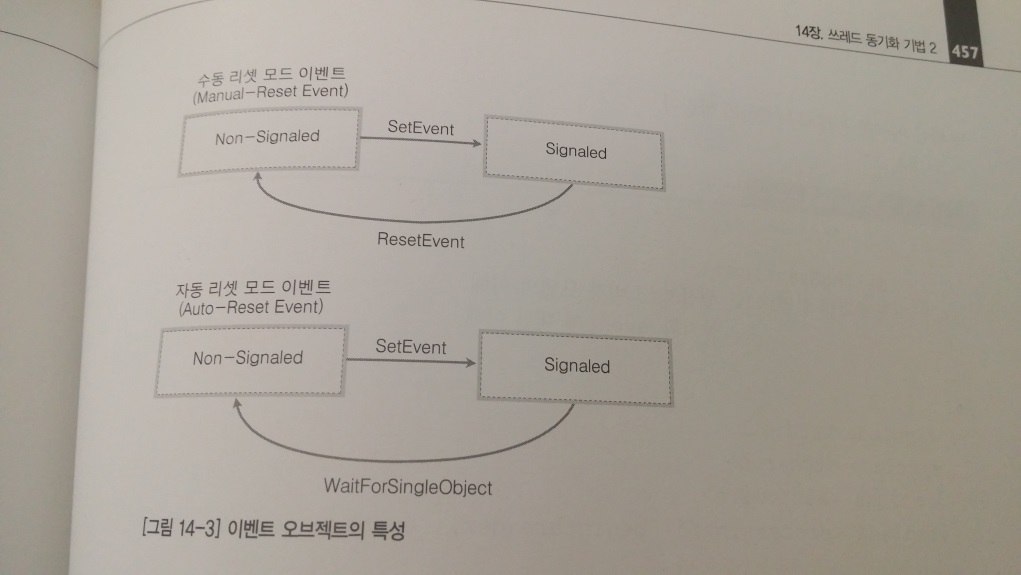
(WaitForSingledObject 함수 호출 완료 이후) 이벤트 오브젝트는 어떠한 상태에 놓이겠는가 Signaled? Non-Signaled?

여전히 Signaled 상태라면 **수동 리셋 모드(auto-Reset) 이벤트 오브젝트** 라는 의미이고 Non-Signaled 상태로 자동 변경 되었다면, **자동 리셋 모드(Auto-Reset 모드)이벤트 오브젝트** 라는 의미이다. (CreateEvent 함수의 두번째 전달 인자를 통해 결정)

다음 특성을 이용해서 스레드의 실행 순서를 동기화 하게 된다.

(1) 이벤트 커널 오브젝트는 프로그래머의 요청에 의해서 Signaled 상태가 된다.

(2) Non-Signaled 상태의 이벤트 오브젝트 때문에 WaitForSignaledObject 함수 호출이 블로킹 되었다면, Signaled 상태가 되는 순간 블로킹된 함수를 빠져 나오게 된다. 그리고 이 때 자동 리셋 모드 이벤트 오브젝트라면, non-Signaled 상태로의 변경은 자동으로 이뤄진다.



위 그림의 윗 부분은 수동 리셋 모드 이벤트 이기 때문에 Signaled 상태에서 Non-Signaled 상태로의 변경을 위해 ResetEvent 함수를 호출해야 함을 설명하고 있다. 아랫 부분은 자동 리셋 모드의 이벤트 이기 때문에 WaitForSingledObject 함수를 반환하면서 자동으로 Non-Signaled 상태로 돌아감을 설명한다.

하지만 수동 리셋 모드라도 WaitForSingleObject 함수 호출이 필요 없는 것은 아니다. 블로킹된 WaitForSingledObject 함수 호출을 빠져 나올 경우 자동으로 Non-Signaled 상태로변경되지 않으므로, 추가로 ResetEvent 함수를 호출해서 변경해 줘야 함을 설명하는 것.

|  |
| --- |
| BOOL ResetEvent (  HANDLE hEvent  ); |

- hEvent : 이벤트 오브젝트의 핸들을 인자로 전달. 오브젝트는 Non-Signaled 상태가 된다.

|  |
| --- |
| BOOL SetEvent (  HANDLE hEvent  ); |

- hEvent : 이벤트 오브젝트의 핸들을 인자로 전달. 오브젝트는 Signaled 상태가 된다.

[예제 14-1 StringEvent.cpp]

**수동 리셋(Manual-Reset)모드 이벤트(Event)의 활용예**

[예제 14-1 StringEvent.cpp] 의 경우 수동 리셋 모드를 사용하던, 자동 리셋 모드를 사용하던 개인의 취향에 맞게 결정할 수 있는 문제 였다. 하지만 수동 리셋 모드 이벤트가 아주 유용하게 사용되는 상황이 존재한다.

[예제 14-2 StringEvent2.cpp]의 결론은 수동 리셋 모드 이벤트는 **둘 이상의 스레드를 동시에 깨워서 실행해야 할 때 아주 좋**

**은 도두가 될 수 있다는 것**이다.

**실행결과**

이 예제의 문제점은 putts 함수가 서로 다른 스레드에서 동시에 호출되었을 경우의 문제이다. 일부 문자열을 출력한 후 타임 슬라이스가 종료되어 다음 스레드로 변경되어 다음 스레드에서도 문자열 출력이 일어 난다면 문제가 생길 수 있음을 보여 준다.

02 이벤트(Event) 더하기 뮤텍스(Mutex)

**이벤트와 뮤텍스 오브젝트 적용 예제**

이벤트와 뮤텍스를 동시에 사용하는 상황은 실무 개발환경에서 자주 접하는 상황이다.

결과 적으로는 [예제 14-2 StringEvent2.cpp]에서 발생하는 문제는 발생하지 않게 해결하기 위해서 putts 관련 함수를 뮤텍스로 감싸 스레드 내에서 출력이 완료 되기 전까지 다른 스레드에서 실행되지 못하도록 하였다.

[예제 14-3 StringEvent3.cpp]

**실행 결과**

03. 타이머(Timer) 기반 동기화

타이머(정확히는 Waitable Timer) 동기화 오브젝트는 **정해진 시간이 지나면 자동으로 Signaled 상태가 되는 특성**을 지닌다.

타이머를 기본으로 스레드를 동기화 한다는 것은 임계 영역 문제 해결을 위한 동기화와는 그 관점이 다르다. 여기서 말하는 동기화는 스레드의 실행 시간 및 실행 주기를 결정하겠다는 의미이다.

타이머 기반 동기화는 다음과 같은 두 가지 형태로 구분 지을 수 있다.

- 수동 리셋 타이머 : 가장 일반적인 타이머로서, 알람 시계를 생각하면 된다.

- 주기적 타이머 : 수동 리셋 타이머에 주기적인 특성을 가해진 형태로 이해하면 된다. 6시에 알람이 울리고 이후로 30분 마다 한번씩 알람이 울리도록 설정하는 기능을 예로들 수 있다.

**수동 리셋 타이머(Manual-Reset Timer)**

타이머 오브젝트의 생성

|  |
| --- |
| HANDLE CreateWaitableTimer (  LPSECURITY\_ATTRIBUTES lpTimerAttributes,  BOOL bManualReset,  LPCTSTR lpTimername  ); |

- lpTimerAttributes : 보안 속성을 지정, 핸들을 자식 프로세스에게 상속하고자 할 경우 NULL이 아닌 다른 값을 전달’

- bManualReset : 타이머 오브젝트의 리셋 모드를 설정하는 값, FALSE 전달 시 수동 리셋 타이머를 생성

- lpTimerName : 타이머 오브젝트에 이름을 붙여줄 경우에 사용되는 전달인자.

이벤트 오브젝트의 생성과 다른 점은 최초 생성시 Signaled로 생성될지 Non-Signaled로 생성될지 결정하지 않는다는 것인데, 이것은 타이머의 특성 떄문에 그렇다. 타이머의 경우 시간이 지나서 Signaled 상태가 되어야 의미를 지닌다. 즉, 타이머 오브젝트가 생성되지 마자 Signaled 상태가 되어버리면 타이머의 의미가 없어진다. 때문에 타이머 오브젝트는 무조건 Non-Signaled 상태로 생성된다.

타이머 오브젝트를 생성했다면, 다음 함수를 이용해서 알람 시간을 설정할 차례이다.

|  |
| --- |
| BOOL SetWaitableTimer(  HANDLE hTimer,  Const LARGE\_INTEGER \*pDueTime,  LONG lPeriod,  PTIMERAPCROUTINE pfnCompletionRoutine,  LPVOID lpArgToCompletionRoutine,  BOOL fResume  ); |

- hTimer : 알람을 설정할 타이머 오브젝트의 핸들을 인자로 전달한다. 정해진 시간이 되면, 이 인자로 전달된 핸들의 커널 오브젝트는 Signaled상태가 된다.

- pDueTime : 알람이 울리는 시간을 지정하기 위한 매개변수이다. + 값이 전달되면 절대 시간을 의미하게되고, -값이 전달되면 상대시간을 의마하게 된다.

- lPeriod : 타이머가 주기적으로 알람을 울리게 할 때 사용하는 전달인자이다. 주기가 되는 시간 간격을 msec 단위로 전달하게 된다. 0을 전달할 경우 사용하지 않겠다는 의미

- pfnCompletionRoutine, lpArgToCompletionRoutine : 는 완료 루틴 타이머를 생성하는 용도의 함수

- fResume : 전원 관리와 관련 있는 매개변수인데, 기본적으로 FALSE 전달은 원칙으로 한다.

[예제 14-4 ManualResetTimer.cpp

**주기적 타이머(Periodic-Timer)**

주기적 타이머는 일정 시간 간격으로 알람이 울리는 타이머를 의미한다. 앞서 제시한 예제에서는 알람이 10초후에 한번 울리도록 수동 리셋 타이머 오브젝트를 생성하였다. 이번에는 이것을 주기적 타이머로 변경해 본 것이다.

[예제 14-5 Periodic-Timer.cpp]

중간에 타이머를 해제하고자 하는 경우에는 다음 함수를 사용하면 된다. 주의할 것은 CancelWaitable Timer 함수는 가동 중에 있는 타이머를 중지 시키는 기능의 함수이지, 타이머를 소멸시키거나 할당된 자원을 반환하는 종류의 함수가 아니라는 점이다.

|  |
| --- |
| BOOL CancelWaitableTimer (  HANDLE hTimer  ); |

- hTimer : 알람을 해제할 타이머 오브젝트의 핸들을 전달한다. 전달된 핸들의 타이머 오브젝트는 알람이 해제 된다.