멀티스레드 II

(10주차)

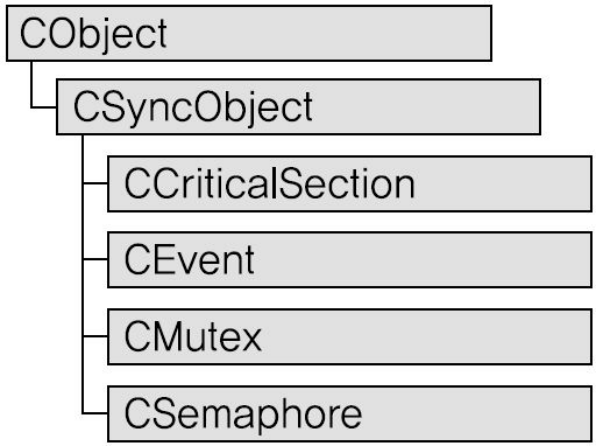
학습개요

- 학습 목표
  - 임계영역, 뮤텍스, 이베트, 세마포어와 같은 스레드 동기화 기법을 이해하고 활용한다.

- 학습 내용
  - 스레드 동기화
  - 임계 영역
  - 뮤텍스
  - 이벤트
  - 세마포어
  - 실습

스레드 동기화

- MFC 클래스 계층도



스레드 동기화

- 클래스 요약
  - CSyncObject
    - 스레드 동기화 클래스를 위한 공통의 인터페이스 제공
  - CCriticalSection, CEvent, CMutex, CSemaphore
    - 윈도우 운영체제에서 제공하는 스레드 동기화 객체(임계 영역, 이벤트, 뮤텍스, 세마포어)를 편리하고 일관되게 사용할 수 있게 해줌

스레드 동기화

- 스레드 동기화가 필요한 상황
  - 둘 이상의 스레드가 공유 자원에 접근하는 경우
  - 한 스레드가 작업을 완료한 후, 기다리고 있는 다른 스레드에 알려주는 경우
- 스레드 동기화 원리

임계 영역

- 용도
  - 공유 자원에 접근하는 다수의 스레드가 있을 때 오직 하나의 스레드만 접근할 수 있게 함
- 장점
  - 속도가 빠름
- 단점
  - 서로 다른 프로세스에 속한 스레드 간 동기화에는 원칙적으로 사용할 수 없음

임계 영역

- 사용 예

```
// 헤더 파일
#include <afxmt.h>
…
// 전역 변수로 선언
CCriticalSection g_cs;
…
// 스레드 1
g_cs.Lock();
공유 자원 접근;
g_cs.Unlock();
…
// 스레드 2
g_cs.Lock();
공유 자원 접근;
g_cs.Unlock();
```

실습

뮤텍스

- 용도
  - 공유 자원에 접근하는 다수의 스레드가 있을 때 오직 하나의 스레드만 접근할 수 있게 함
  - 임계 영역과 기능이 동일
- 장점
  - 서로 다른 프로세스에 속한 스레드 간 동기화에 사용할 수 있음
- 단점
  - 임계 영역보다 속도가 느림

뮤텍스

- 뮤텍스 생성

```
CMutex::CMutex(
① BOOL bInitiallyOwn = FALSE,
② LPCTSTR lpszName = NULL,
③ LPSECURITY_ATTRIBUTES lpsaAttribute = NULL
);
```

  - bInitiallyOwn: TRUE면 뮤텍스를 생성한 스레드가 소유자가 됨
  - lpszName: 뮤텍스에 이름을 부여함. NULL을 사용하면 이름 없는 (Anonymous) 뮤텍스가 됨
  - lpsaAttribute: 보안 설명자와 핸들 상속 정보

실습

이벤트

- 이벤트
  - 신호(Signaled)와 비신호(Nonsignaled) 두 개의 상태를 가진 동기화 객체
- 용도
  - 한 스레드가 작업을 완료한 후, 기다리고 있는 다른 스레드에 알려줄 때 주로 사용

이벤트

- 이벤트 사용 절차
  ① 이벤트를 비신호 상태로 생성
  ② 한 스레드가 작업을 진행하고, 나머지 스레드는 이벤트에 대해 Lock() 함수를 호출함으로써 이벤트가 신호 상태가 될 때까지 대기함(Sleep)
  ③ 스레드가 작업을 완료한 후 이벤트를 신호 상태로 바꿈
  ④ 기다리고 있던 스레드 중 하나 혹은 전부가 깨어남(Wakeup)

이벤트

- 종류
  - 자동 리셋(Auto Reset)
    - 이벤트를 신호 상태로 바꾸면, 기다리는 스레드 중 하나만 깨운 후 자동으로 비신호 상태가 됨
  - 수동 리셋(Manual Reset)
    - 이벤트를 신호 상태로 바꾸면, 기다리는 스레드 전부를 깨운 후 계속 신호 상태를 유지함
    - 비신호 상태로 바꾸려면 명시적으로 함수를 호출해야 함

이벤트

- 이벤트 생성

```
CEvent::CEvent(
① BOOL bInitiallyOwn = FALSE,
② BOOL bManualReset = FALSE,
③ LPCTSTR lpszName = NULL,
④ LPSECURITY_ATTRIBUTES lpsaAttribute = NULL
);
```

- bInitiallyOwn: FALSE면 비신호, TRUE면 신호 상태
- bManualReset: FALSE면 자동 리셋, TRUE면 수동 리셋
- lpszName: 이벤트에 이름을 부여함. NULL을 사용하면 이름 없는(Anonymous) 이벤트가 됨
- lpsaAttribute: 보안 설명자와 핸들 상속 정보

이벤트

- 이벤트 상태 변경
  - 이벤트를 신호 상태로 바꿈

```
BOOL CEvent::SetEvent();
```

  - 이벤트를 비신호 상태로 바꿈

```
BOOL CEvent::ResetEvent();
```

실습

세마포어

- 세마포어
  - 자원에 접근할 수 있는 스레드 수를 제어하는 동기화 객체
  - 가용 자원의 개수를 나타내는 리소스 카운트(Resource Count)를 유지함으로써 동시에 실행될 수 있는 스레드 수를 조절 가능

세마포어

- 세마포어 사용 절차
  ① 세마포어를 생성. 이때 리소스 카운트를 가용 자원의 개수로 초기화
  ② 자원을 사용할 스레드는 자신이 필요한 자원의 개수만큼 Lock() 함수를 호출하는데, Lock() 함수가 성공할 때마다 리소스 카운트 값은 1씩 감소.리소스 카운트가 0인 상태에서 Lock() 함수를 호출하면 해당 스레드는 대기함(Sleep)
  ③ 자원 사용을 마친 스레드는 자신이 사용한 자원의 개수만큼 Unlock() 함수를 호출하는데, Unlock() 함수가 성공할 때마다 리소스 카운트 값은 1씩 증가. 대기중인 다른 스레드가 있다면 깨어남(Wakeup)

세마포어

- 세마포어 생성

```
CSemaphore::CSemaphore(
① LONG lInitialCount = 1,
② LONG lMaxCount = 1,
③ LPCTSTR pstrName = NULL,
④ LPSECURITY_ATTRIBUTES lpsaAttributes = NULL
);
```

  - lInitialCount: 리소스 카운트의 초기값
  - lMaxCount: 리소스 카운트의 최대값
  - pstrName: 세마포어에 이름을 부여함. NULL을 사용하면 이름 없는(Anonymous) 세마포어가 됨
  - lpsaAttribute: 보안 설명자와 핸들 상속 정보

실습

학습정리

- 둘 이상의 스레드가 공유 자원에 접근하는 경우, 한 스레드가 작업을 완료한 후, 기다리고 있는 다른 스레드에 알려주는 경우 스레드 동기화가 필요하다.
- CCriticalSection, CEvent, CMutex, CSemaphore 클래스의 객체를 통해 스레드 동기화를 편리하고 일관되게 사용할 수 있게 해준다.
- 임계영역과 뮤텍스는 공유 자원에 접근하는 다수의 스레드가 있을 때 오직 하나의 스레드만 접근할 수 있게 해준다.
- 이벤트는 신호와 비신호 두 개의 상태를 가진 동기화 객체로 한 스레드가 작업을 완료한 후, 기다리고 있는 다른 스레드에 알려줄 때 주로 사용한다.
- 세마포어는 자원에 접근할 수 있는 스레드 수를 제어하는 동기화 객체로 가용 자원의 개수를 나타내는 리소스 카운트를 유지함으로써 동시에 실행될 수 있는 스레드 수를 조절할 수 있다.