멀티스레드 I

(9주차)

학습개요

- 학습 목표
  - 스레드의 개념과 동작 원리를 이해한다.
  - MFC 스레드의 두 종류인 작업자 스레드와 UI 스레드 구현 방법을 익힌다.

- 학습 내용
  - 멀티스레드 기초
  - MFC 스레드
  - 실습

개요

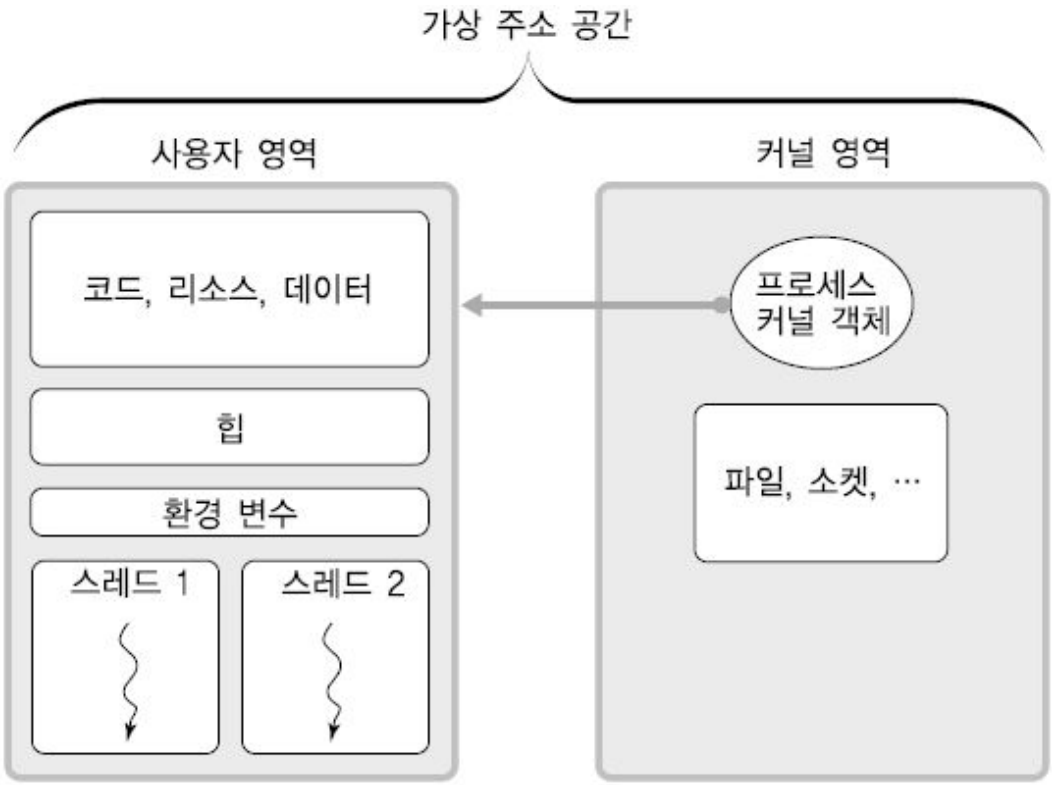
- 멀티태스킹과 멀티스레딩
  - 멀티태스킹
    - 운영체제가 여러 개의 프로세스를 동시에 실행한다.
  - 멀티스레딩
    - 응용 프로그램이 여러 개의 스레드를 동시에 실행한다.
- 멀티스레딩의 중요성
  - 응용 프로그램이 스레드 생성과 파괴를 직접 관리
  - 스레드 사용 여부에 따라 성능 차이가 생김

프로세스와 스레드

- 프로세스
  - 실행 중인 프로그램
- 프로세스 구성 요소
  - 가상 주소 공간
    - 32비트 윈도우의 경우 4GB(사용자 영역 2GB+커널 영역 2GB)
    - 64비트 윈도우의 경우 16TB(사용자 영역 8TB+커널 영역 8TB)
  - 실행 파일과 DLL
    - 코드, 리소스, 데이터(전역 변수, 정적 변수)
  - 힙 / 환경 변수 / 하나 이상의 스레드
  - 프로세스 커널 객체
  - 운영체제가 프로세스를 위해 할당한 각종 자원
    - 파일, 소켓, ...

프로세스와 스레드

- 프로세스 구성 요소

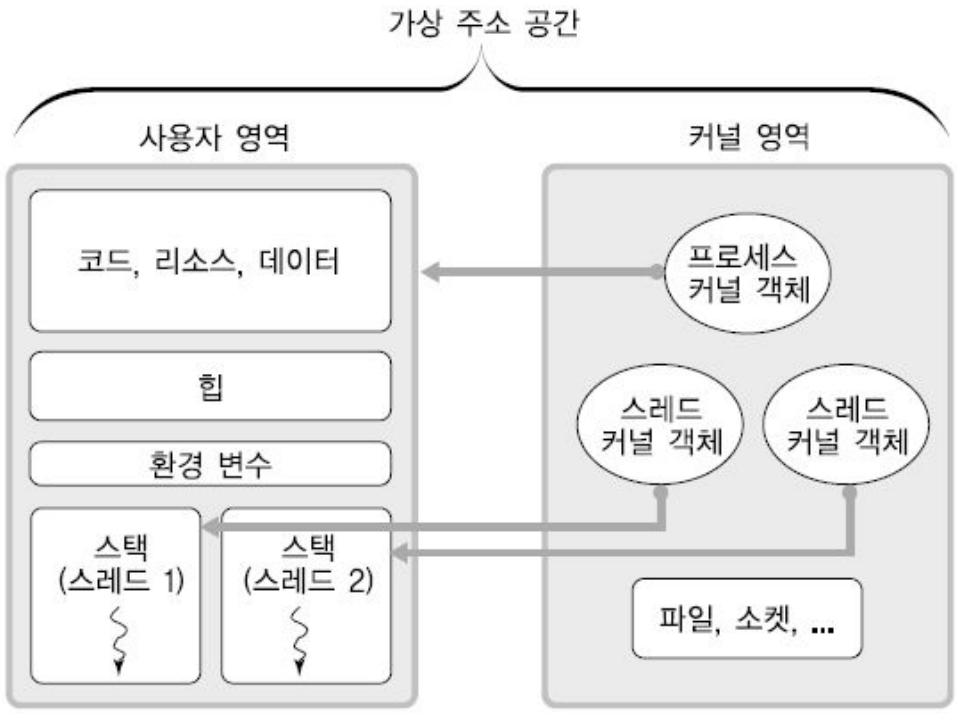


프로세스와 스레드

- 스레드
  - 프로세스의 가상 주소 공간에 존재하는 실행 흐름
  - 운영체제는 각 스레드에 CPU 시간을 나누어 할당함으로써 여러 개의 스레드가 동시에 실행되는 효과를 냄
- 스레드 구성 요소
  - 스택
    - 함수 인자 전달과 지역 변수 저장을 위한 공간
  - 스레드 지역 저장소(TLS, Thread Local Storage)
    - 스레드별 고유 데이터를 저장하기 위한 공간
  - 스레드 커널 객체

프로세스와 스레드

- 프로세스와 스레드 구성 요소



CPU 스케줄링

- CPU 스케줄링
  - 운영체제가 한정된 CPU 시간을 여러 개의 프로세스(전통적인 유닉스 운영체제) 혹은 스레드(윈도우 운영체제)에 분배하는 정책
- 윈도우의 CPU 스케줄링
  - 우선순위(Priority)에 기반한 CPU 스케줄링 기법을 사용
    - 우선순위가 높은 스레드에 CPU 시간을 우선 할당
- 스레드의 우선순위 결정 요소
  - 우선순위 클래스(Priority Class)
  - 우선순위 레벨(Priority Level)

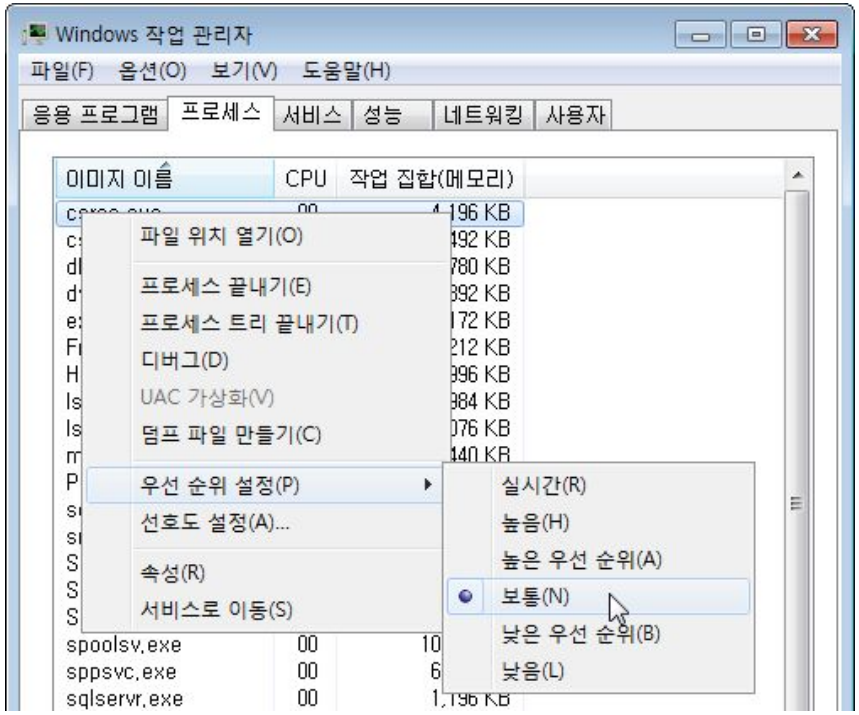
CPU 스케줄링

- 우선순위 클래스
  - 프로세스 속성
  - 같은 프로세스가 생성한 스레드는 모두 동일한 우선순위 클래스를 가짐
- 우선순위 클래스 종류

```
REALTIME_PRIORITY_CLASS(실시간)
HIGH_PRIORITY_CLASS(높음)
ABOVE_NORMAL_PRIORITY_CLASS
(높은 우선 순위; 윈도우 2000/XP 이상)
NORMAL_PRIORITY_CLASS(보통)
 BELOW_NORMAL_PRIORITY_CLASS
(낮은 우선 순위; 윈도우 2000/XP 이상)
IDLE_PRIORITY_CLASS(낮음)
```

CPU 스케줄링

- 우선순위 클래스 종류



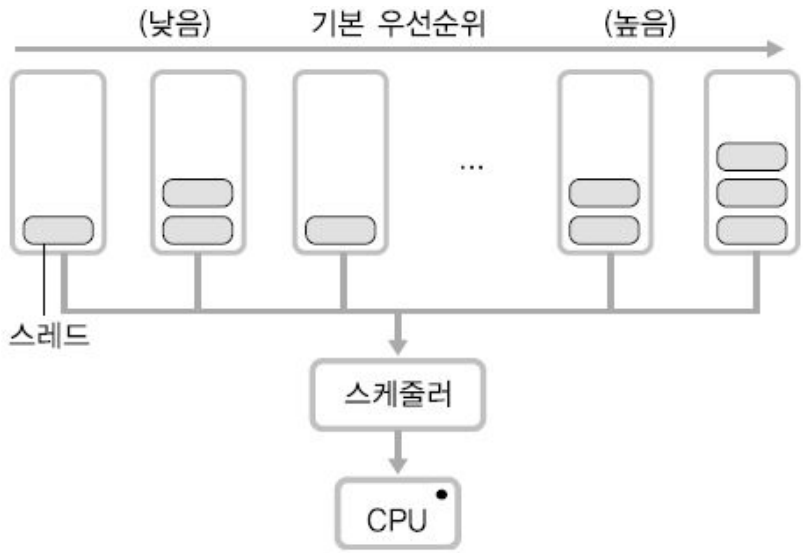
CPU 스케줄링

- 우선순위 레벨
  - 스레드 속성
  - 같은 프로세스에 속한 스레드 간 상대적인 우선순위를 결정
- 우선순위 레벨 종류

```
THREAD_PRIORITY_TIME_CRITICAL
THREAD_PRIORITY_HIGHEST
THREAD_PRIORITY_ABOVE_NORMAL
THREAD_PRIORITY_NORMAL
THREAD_PRIORITY_BELOW_NORMAL
THREAD_PRIORITY_LOWEST
THREAD_PRIORITY_IDLE
```

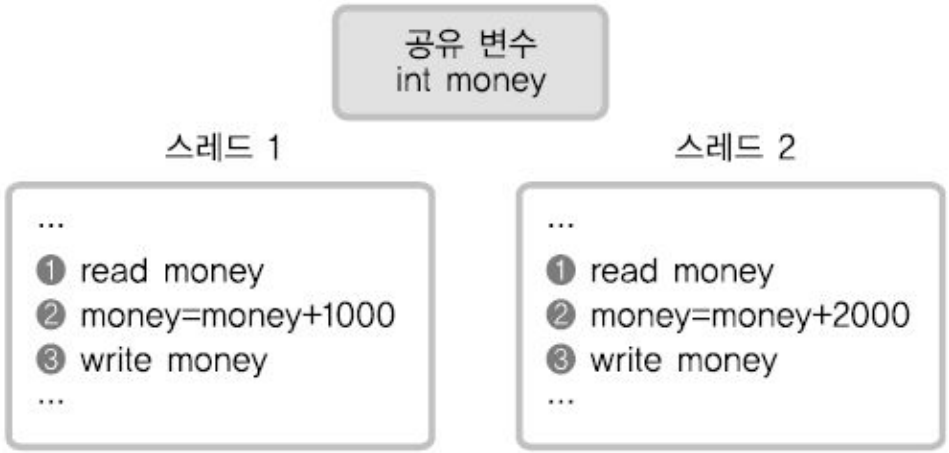
CPU 스케줄링

- 우선순위 클래스 + 우선순위 레벨
  ⇨ 스레드의 기본 우선순위(Base Priority)



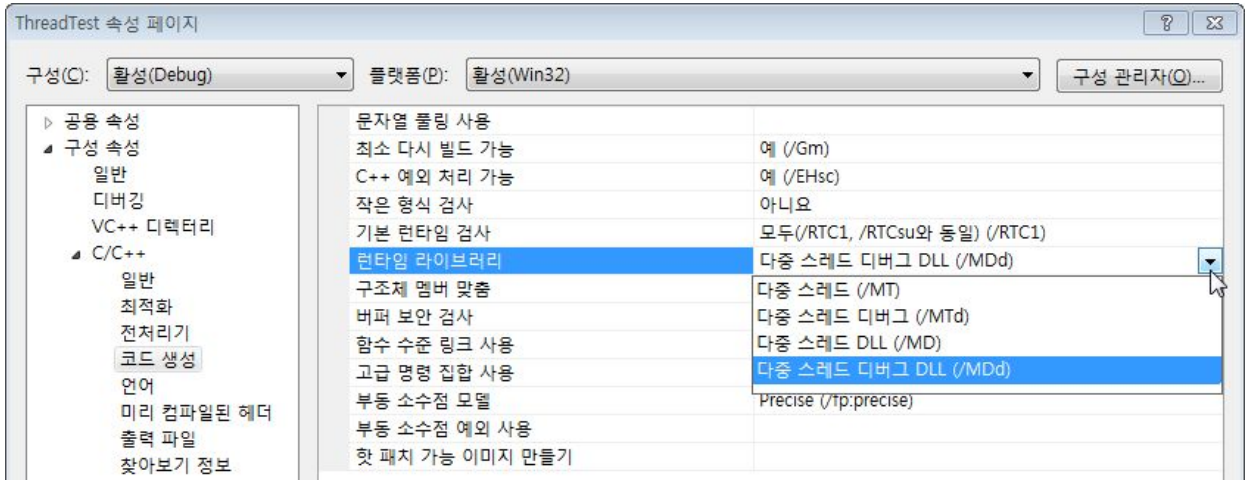
스레드 동기화

- 두 개의 스레드가 데이터를 공유하는 상황



스레드 동기화

- C/C++ 라이브러리 선택



MFC 스레드

- MFC 스레드 종류
  - 작업자 스레드(Worker Thread)
    - 메시지 루프가 없다.
      - 화면에 보이지 않는 백그라운드 작업을 수행할 때 적합
  - 사용자 인터페이스 스레드(User Interface Thread)
    - 메시지 루프가 있다.
      - 윈도우를 만들고 출력을 하거나 사용자의 입력을 받는 등의 작업을 별도의 스레드로 처리할 때 적합

작업자 스레드

- 작업자 스레드 생성

```
CWinThread* AfxBeginThread(
  ① AFX_THREADPROC pfnThreadProc,
  ② LPVOID pParam,
  ③ int nPriority = THREAD_PRIORITY_NORMAL,
  ④ UINT nStackSize = 0,
  ⑤ DWORD dwCreateFlags = 0,
  ⑥ LPSECURITY_ATTRIBUTES lpSecurityAttrs = NULL
);
```

  - CWinThread 타입의 스레드 객체(Thread Object)를 동적으로 생성하고 내부적으로 스레드를 만든 후 스레드 객체의 주소값을 리턴

작업자 스레드

- 작업자 스레드 생성
  - pfnThreadProc: 스레드 실행 시작점이 되는 함수(=제어 함수)의 주소
    - 제어 함수 형태 ⇨ UINT 함수명(LPVOID pParam);
  - pParam: 제어 함수 실행 시 전달할 인자
  - nPriority: 스레드 우선순위 레벨
  - nStackSize: 스레드 스택 크기
  - dwCreateFlags: 스레드 생성을 제어하는 옵션
    - 0 또는 CREATE_SUSPENDED
  - lpSecurityAttrs: 보안 설명자와 핸들 상속 정보

작업자 스레드

- CWinThread 클래스의 유용한 함수들
  - 스레드 우선순위 레벨 값을 얻음

```
int CWinThread::GetThreadPriority()
```

  - 스레드 우선순위 레벨 값을 변경

```
BOOL CWinThread::SetThreadPriority(int nPriority)
```

  - 스레드 실행을 일시 중지

```
DWORD CWinThread::SuspendThread()
```

  - 일시 중지된 스레드의 실행을 재개

```
DWORD CWinThread::ResumeThread()
```

작업자 스레드

- 작업자 스레드 종료
  - 방법 1: 스레드 제어 함수가 종료 코드를 리턴
    - 0을 리턴하면 일반적으로 정상 종료를 뜻함
  - 방법 2: 스레드 제어 함수 내에서 AfxEndThread() 함수를 호출

작업자 스레드

- 작업자 스레드 종료

```
void AFXAPI AfxEndThread(UINT nExitCode, BOOL bDelete=TRUE);
                              ①            ②
```

  - nExitCode: 스레드 종료 코드
  - bDelete: 스레드 객체를 제거할 것인지를 나타냄
    - FALSE를 사용하면 스레드 객체 재사용 가능

실습

UI 스레드

- UI 스레드 생성 절차
  ① CWinThread 클래스를 상속받아 새로운 클래스를 생성
  ② 클래스 선언부와 구현부에 각각 DECLARE_DYNCREATE, IMPLEMENT_DYNCREATE 매크로를 선언
  ③ CWinThread 클래스가 제공하는 가상 함수 중 일부를 재정의
  CWinThread::InitInstance() 함수는 반드시 재정의. 나머지 함수는 필요에 따라 재정의
  ④ AfxBeginThread() 함수로 새로운 UI 스레드 생성

UI 스레드

- UI 스레드 생성

```
CWinThread* AfxBeginThread(
  ① CRuntimeClass* pThreadClass,
  ② int nPriority = THREAD_PRIORITY_NORMAL,
  ③ UINT nStackSize = 0,
  ④ DWORD dwCreateFlags = 0,
  ⑤ LPSECURITY_ATTRIBUTES lpSecurityAttrs = NULL
);
```

  - CWinThread 타입의 스레드 객체(Thread Object)를 동적으로 생성하고 내부적으로 스레드를 만든 후 스레드 객체의 주소값을 리턴

UI 스레드

- UI 스레드 생성
  - pThreadClass: 클래스 정보를 담고 있는 CRuntimeClass 구조체
    - 인자 전달 형태 ⇨ RUNTIME_CLASS(클래스이름)
  - nPriority: 스레드 우선순위 레벨
  - nStackSize: 스레드 스택 크기
  - dwCreateFlags: 스레드 생성을 제어하는 옵션
    - 0 또는 CREATE_SUSPENDED
  - lpSecurityAttrs: 보안 설명자와 핸들 상속 정보

UI 스레드

- UI 스레드 종료
  - 방법 1: WM_QUIT 메시지를 받아서 메시지 루프가 종료
  - 방법 2: 스레드 제어 함수 내에서 AfxEndThread() 함수를 호출

실습

학습정리

- 프로세스는 가상 주소 공간, 실행 파일과 DLL, 코드, 리소스, 데이터, 힙, 환경 변수, 하나 이상의 스레드, 프로세스 커널 객체, 운영체제가 프로세스를 위해 할당한 각종 자원으로 구성된다.
- 스레드는 프로세스의 가상 주소 공간에 존재하는 실행 흐름으로 운영체제는 각 스레드에 CPU 시간을 나누어 할당함으로써 여러 개의 스레드가 동시에 실행되는 효과를 낸다.
- 스레드는 스택, 스레드 지역 저장소, 스레드 커널 객체로 구성된다.
- 윈도우의 CPU 스케줄링은 우선순위에 기반한 CPU 스케줄링 기법을 사용하며, 우선순위가 높은 스레드에 CPU 시간을 우선 할당한다.
- 스레드의 우선순위 결정 요소로 우선순위 클래스와 우선순위 레벨이 있다.
- MFC 스레드는 작업자 스레드 UI 스레드로 나뉜다.
- 작업자 스레드는 메시지 루프가 없으며, 화면에 보이지 않는 백그라운드 작업을 수행할 때 적합하고, UI 스레드는 메시지 루프가 있으며, 윈도우를 만들고 출력을 하거나 사용자의 입력을 받는 등의 작업을 별도의 스레드로 처리할 때 적합하다. 모든 스레드는 AfxBeginThread()로 생성한다.