Windows System Programming
시작을 위한 넓고 얕은 지식

사전진단테스트 : 진단결과 고급 피드백

고급역량	Windows System Programming 시작을 위한 넓고 얕은 지식

Windows:
Windows는 전세계적으로 가장 많이 사용되는 운영체제이다. Windows 상에서 동작하는 S/W를 개발하고자 하는 개발자의 입장에서, Windows의 특징과 그 동작방식을 이해를 하는 것은 매우 중요하다.

System Programming:
System Programming이란, 운영체제와의 상호작용 및 결국은 운영체제를 통해서 H/W와 상호작용을 하는 프로그래밍 분야라고 정의하고 싶다. 즉 User Interface(UI) Programming이 사용자(고객)을 상대하는 프로그래밍 분야라면, System Programming은 운영체제 및 H/W를 상대하는 프로그래밍 분야이다.
즉 UI Programming 분야가 사용의 편리성과 아름다움 등에 더 관심을 가진다면, System Programming 분야는 속도와 안정성에 더 많은 관심을 가져야 한다.

Windows System Programming:
Windows 상에서 System Programming 분야를 개발하고자 하는 개발자는 Windows의 다양한 특징 뿐 아니라, System Programming 분야에서의 방대한 특징들을 알고 있어야 좋은 설계를 할 수 있고 안전하게 개발을 완료할 수 있을 것이다. 그런데 모든 기술을 하나하나 깊게 이해하기는 시간도 오래 걸리고 너무나 지치는 과정이 될 수 있다.

넓고 얕은 지식:
이에 우리는 "넓고 얕은 지식"을 모아서, 전반적이고 주요한 특징들을 가볍게 살펴보고자 한다. 숲을 볼 수 있는 눈을 기른 후에, 숲 속으로 들어가서 나무들을 각각 세심하게 살펴볼 수 있는 힘을 기르기 위함이다.
General이 더 중요한가? Special이 더 중요한가?
사람마다 의견이 다르겠지만, 탄탄한 General 위에서 각각의 세부적인 Special을 쌓아가는 것이 보다 안정적이라고 생각되어 이 강의를 준비했다.

자, 우리 거대한 Windows System Programming의 세계를 가볍게 여행해 보자!

■ 스레드 [Thread]

①컴퓨터 프로그램 수행 시 프로세스 내부에 존재하는 수행 경로, 즉 일련의 실행 코드. 프로세스는 단순한 껍데기일 뿐, 실제 작업은 스레드가 담당한다. 프로세스 생성 시 하나의 주 스레드가 생성되어 대부분의 작업을 처리하고 주 스레드가 종료되면 프로세스도 종료된다. 하나의 운영 체계에서 여러 개의 프로세스가 동시에 실행되는 환경이 멀티태스킹이고, 하나의 프로세스 내에서 다수의 스레드가 동시에 수행되는 것이 멀티스레딩이다. ②나무 데이터 구조(tree data structure)에서 상위 노드의 식별과 나무 내의 정보 탐색을 촉진하는 포인터. ③게시판이나 토론의 장에서 최초 메시지에 대한 댓글들의 연속. 스레드는 수없이 많은 각각의 토론들을 관련 있는 것끼리 찾아서 읽거나 참여하는 데 도움이 된다.

[네이버 지식백과] 스레드 [thread] (IT 용어사전, 한국정보통신기술협회)

■ 멀티-스레드 [Multi-Thread]

1 개의 응용 프로그램이 스레드(thread)로 불리는 처리 단위를 복수 생성하여 복수의 처리를 병행하는 것. 즉, 응용 프로그램 내에서의 다중 작업(multitasking) 처리를 말한다. 다중 작업과 같이 중앙 처리 장치(CPU)의 처리 시간을 매우 짧은 단위로 분할하여 복수의 스레드에 차례로 할당함으로써 복수의 처리가 동시에 이루어지는 것처럼 보인다.

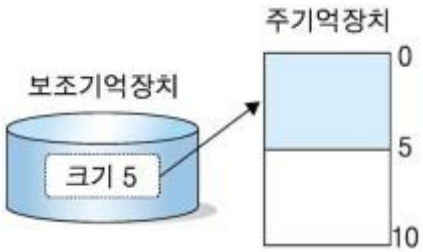
[네이버 지식백과] 멀티 스레드 [multi-thread] (IT 용어사전, 한국정보통신기술협회)

■ 가상 메모리 [Virtual Memory]

프로그램이 실행되기 위해서는 주기억장치로 들어가야 하는데, 실행될 프로그램이 주기억장치보다 크거나 여러 개인 경우에는 주기억장치 공간의 부족으로 인해 프로그램이 제대로 실행되지 못할 수 있다. 그래서 당장 실행에 필요한 부분만 주기억장치에 저장하고, 나머지는 보조기억장치에 두고 동작하도록 하여 이런 문제를 해결할 수 있는데, 이런 개념을 가상 메모리라 하며 운영체제에서 지원한다.

1) 가상 메모리

프로그램이 실행되려면 우선 주기억장치에 들어가야 하는데, 다음과 같이 크기 5 의 프로그램이 실행을 위해 크기 10 의 주기억장치로 들어갈 때는 아무런 문제가 발생하지 않는다.

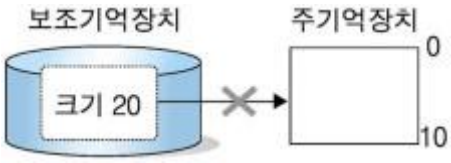


[그림 5-42] 크기 5 의 프로그램이 주기억장치로 진입

하지만 다음의 두 경우에는 문제가 발생한다.

- **주기억장치보다 프로그램의 크기가 큰 경우**

주기억장치 크기는 10 인데 프로그램의 크기가 20 인 경우로, 프로그램이 주기억장치에 들어갈 수 없어 프로그램이 실행되지 못한다.

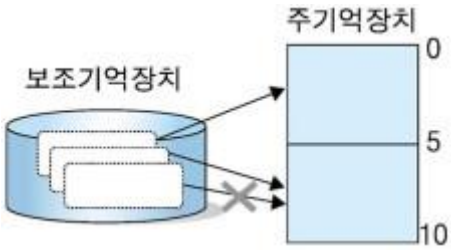


[그림 5-43] 주기억장치보다 프로그램의 크기가 큰 경우

- **실행하려는 전체 프로그램의 크기가 주기억장치보다 클 때**

크기 5 의 프로그램 3 개가 동시에 실행되고자 하는 경우로 두 개의

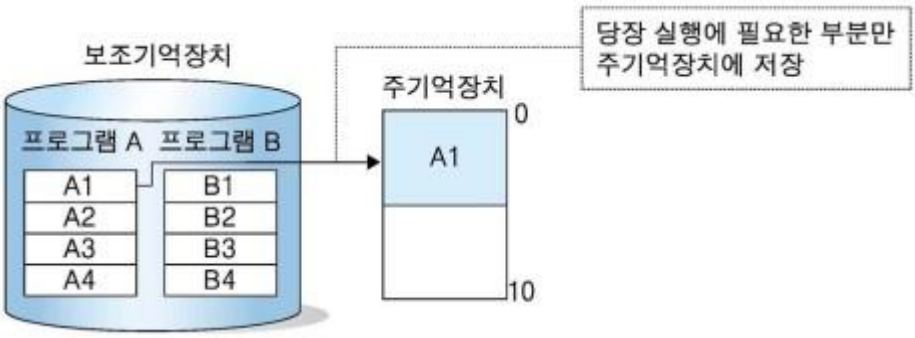
프로그램은 주기억장치에 들어가 실행될 수 있으나 주기억장치에 들어가지 못한 나머지 한 프로그램은 실행되지 못한다.



[그림 5-44] 실행하려는 전체 프로그램의 크기가 주기억장치보다 클 때

최근 프로그램의 크기가 커지고, 동시에 여러 프로그램이 실행되는 경우가 많으므로 이와 같은 문제는 자주 발생하게 된다. 이런 문제를 해결할 수 있는 방법이 바로 가상 메모리(virtual memory)다. 아무리 여러 프로그램이 실행되고자 하더라도 한 순간에 실행되는 프로그램은 하나뿐이고, 아무리 큰 프로그램이라 할지라도 한 순간에 중앙처리장치에 의해 실행되는 부분은 프로그램의 일부일 뿐이다. 따라서 당장 실행에 필요한 부분만 주기억장치에 저장하고, 당장 필요하지 않은 나머지 부분은 보조기억장치에 넣어 두고 실행하면 되는데, 이런 개념이 가상 메모리다. 결국 가상 메모리를 사용하면 사용자 입장에서는 실제 주기억장치보다 큰 주기억장치를 가지고 있는 것처럼 느끼게 된다.

다음은 크기 10 의 주기억장치에서 크기 20 의 프로그램 A 와 프로그램 B 를 실행하는 동작으로, 당장 실행에 필요한 프로그램 A 의 일부인 A1 을 주기억장치에 올려 실행하고 나머지는 보조기억장치에 두는 동작이다. 물론 A2 가 실행에 필요하게 되면 주기억장치로 올리면 된다.



[그림 5-45] 당장 실행될 부분만 주기억장치에 저장

[그림 5-45]에서 프로그램을 일정한 크기로 나누었는데, 이런 단위를 페이지(page)라 하고, 페이지 단위로 주기억장치에 올리며 동작하는 것을 페이징(paging)이라 한다.

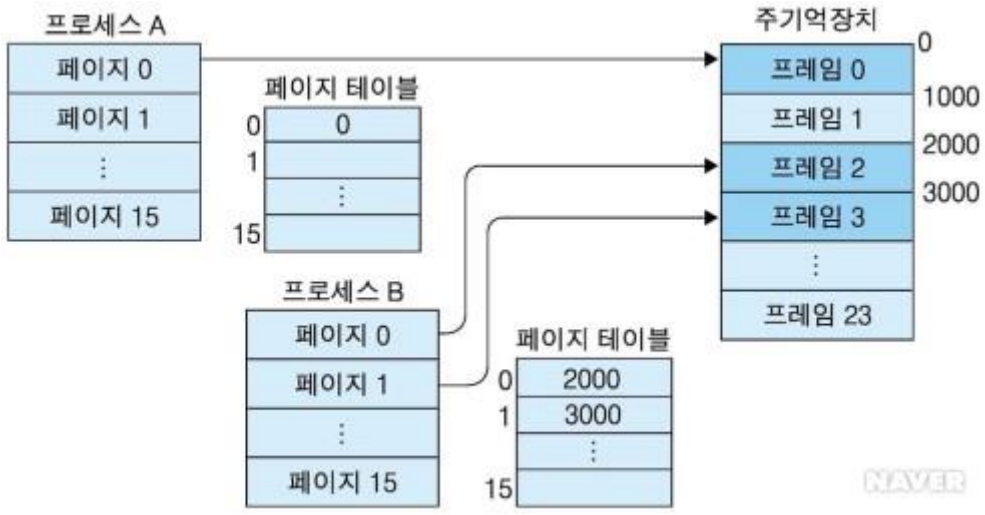
2) 페이징

페이징(paging)은 가상 메모리를 구현하는 한 방법으로, 가상 메모리 공간을 일정한 크기의 페이지(page)로 나누어 관리하는 방법이다. 이때 실제 주기억장치의 페이지에 해당하는 부분을 페이지 프레임(page frame)이라 한다.

그러면 페이징의 동작 원리에 대해 살펴보자. 우선 16 개의 페이지로 이루어진 프로세스 A 와 프로세스 B 가 실행중이고, 주기억장치는 24 개의 페이지 프레임 크기다. 그리고 각 페이지의 크기는 1000 이라 가정한다.

[그림 5-46] 16 개 페이지 크기의 프로세스들과 24 개 페이지 프레임 크기의 주기억장치

프로세스 A 의 페이지 0 과 프로세스 B 의 페이지 0, 1 이 당장 실행되어야 한다고 가정하자. 그러면 프로세스 A 의 페이지 0 이 주기억장치 페이지 프레임 0 에, 프로세스 B 의 페이지 0 과 페이지 1 이 주기억장치 페이지 프레임 2 와 3 에 저장된다고 하자. 이때 프로세스 마다 각 페이지가 주기억장치의 어느 프레임에 저장되는지를 나타내는 테이블을 운영체제가 관리하는데, 이를 페이지 테이블(page table)이라 한다.



[그림 5-47] 페이징 동작 예

프로세스 A 의 페이지 0 은 페이지 프레임 0 에 저장되므로 [그림 5-46]에서와 같이 페이지 테이블 0 번 항목에 페이지 프레임 0 의 시작 주소인 0 이 저장된다. 마찬가지로 프로세스 B 의 페이지 테이블 0 번 항목에 페이지 프레임 2 의 시작 주소인 2000 이, 1 번 항목에 페이지 프레임 3 의 시작 주소인 3000 이 저장된다.

임의의 페이지가 실행에 필요하면 우선 주기억장치에 해당 페이지가 있는지를 확인해서 있으면 그 페이지에 접근한다. 만약 해당 페이지가 주기억장치에 없다면 보조 기억장치로부터 그 페이지를 주기억장치 페이지 프레임에 저장해서 접근한다.

[네이버 지식백과] 가상 메모리 (컴퓨터 개론, 2013. 3. 10., 김종훈, 김종진)