5장 프로세스의 생성과 소멸

Section 1 : 프로세스의 이해.

- 프로세스란 무엇인가?

  프로세스란 "실행 중에 있는 프로그램"을 의미한다. 어떤 프로그램이 하드디스크에 저장되어 있다고 한다면 이는 아직 프로세스 상태가 아닌 그냥 프로그램이다. 그러나 이를 실행시키는 순간 프로그램은 하드디스크에서 메모리 위로 할당이 된다. 이 상태부터 프로그램을 프로세스라고 부를 수 있는 것이다.

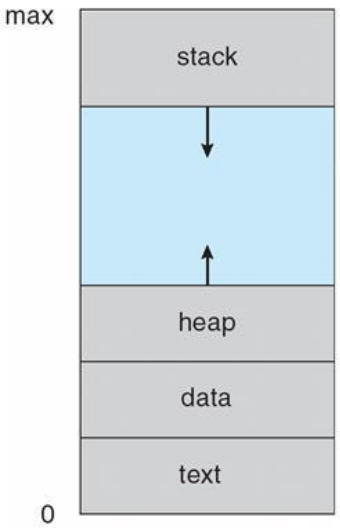
- 프로세스를 구성하는 요소

  1. Execution of "C" Program

  프로그램이 실행될 때 구성되는 메모리 공간에는 4가지가 있다.

  - Data 영역 : 전역변수나 static 변수 할당
  - Stack 영역 : 지역변수 할당과 인자값 저장.
  - Heap 영역 : 동적 할당 변수 저장.
  - Code 영역 : 실행 파일을 구성하는 명령어들이 올라가는 메모리 영역.

  이 네 가지 영역을 하나로 묶은 것이 프로그램 실행 시 만들어지는 메모리 공간의 대략적인 구성이다. 또한 이것이 프로세스의 실체이다.



만약 세가지 프로세스가 만들어진다면? 각기 작동하는 메모리 구조가 세 개 생성될 것이다. 즉, 프로세스의 개수 만큼 위와 같은 메모리 구조가 생성될 것이다.

2. Register Set

프로세스 구성 요소로 또한 생각해봐야 하는 것은 CPU 내의 레지스터들이다. 프로그램 실행을 위해서는 레지스터의 존재가 절대적이다.

CPU가 어떤 프로그램을 실행 중, 즉 프로세스를 실행 시키고 있다면 CPU 내부에 있는 레지스터들은 프로세스 실행을 위해 필요한 데이터들로 채워지게 된다. 따라서, 레지스터들의 상태까지도 프로세스의 일부로 포함시켜 말할 수 있다. -> 컨텍스트 스위칭의 고려 요소가 된다.

Section 2 : 프로세스의 스케줄링과 상태 변화.

- 프로세스의 스케줄링

CPU는 하나인데, Windows는 여러 개의 프로그램이 동시에 실행하는 것을 지원한다.

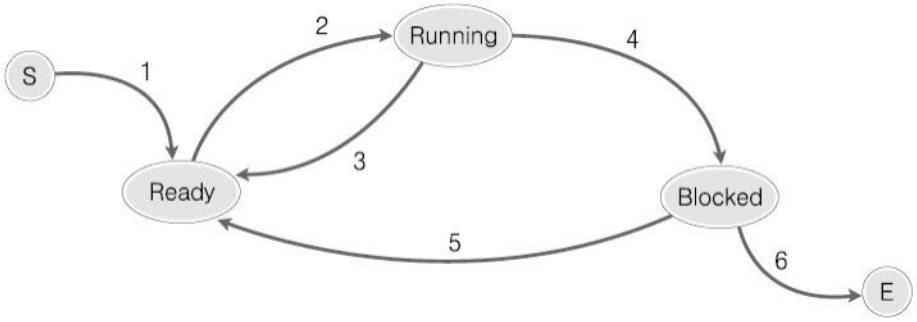
➔ 하나의 CPU가 여러 개의 프로세스를 번갈아 가면서 실행해야만 한다.

➔ CPU가 매우 빠르기 때문에 사용자 입장에서는 동시에 실행 되는 것처럼 느껴짐.

그런데 프로세스 중에는 긴급하게 처리되어야 할 중요한 프로세스도 있을 것이고 그에 비해서 중요도가 낮은 프로세스도 있을 것이다. 이 들에게 똑같이 CPU를 할당한다는 것은 크나큰 손해이다. 따라서 OS는 CPU를 할당하는 우선 순위와 순서를 정한다.

이 과정에서 순서를 정하는 알고리즘을 스케줄링 알고리즘, 순서를 매기는 일을 스케줄링이라고 한다. 그리고 이를 진행하는 소프트웨어적 OS요소를 스케줄러라고 칭한다.

프로세스의 상태 변화.



각 프로세스는 상태를 갖고, 시간에 따라 이 상태는 계속 변화한다. 각각의 상태는 다음을 의미한다.

Start : 프로세스가 생성된 상태.

Ready : 실행을 기다리고 있는 상태.

Running : 현재 실행되고 있는 상태. (CPU마다 1개의 프로세스 밖에 존재할 수 없을 것이다.)

Blocked : I/O 작업등으로 Ready 상태가 되기를 기다리고 있는 상태.

Exit : 프로세스가 끝나고 종료를 기다리고 있는 상태.

또한 상태가 변화하는 상황은 다음과 같다.

1. Start -> Ready : 프로세스는 생성과 동시에 Ready상태로 들어간다. Ready상태에서 스케줄러에 의해 선택되기를 바라고 있는 상태이다.
2. Ready -> Running : Ready 상태에 있던 프로세스 중 스케줄러에 의해 선택된 프로세스가 CPU에 의해서 실행되고 있는 상황이다.
3. Running -> Ready : 프로세스는 생성과 함께 우선순위가 매겨지고 이는 시간에 따라 변화한다. Running 상태의 프로세스는 자신보다 우선순위가 높은 프로세스가 나타나면 Running 상태를 양보하고 Ready상태로 들어가 우선순위가 높아지기를 기다린다.
4. Running -> Blocked : 실행 중의 프로세스가 실행을 멈추는 상태로 돌입하는 것이다. 주로 I/O접근에 의해서 발생한다. I/O접근은 CPU에 비해 현저히 속도가 낮기 때문에 이를 CPU가 기다리는 상황이 발생하지 않게 하기 위함이다.
5. Blocked -> Ready : I/O접근이 끝난 프로세스는 다시 Ready 상태가 되어 Running이 되기를 기다린다.

Section 3 : 컨텍스트 스위칭

Register Set을 말하면서 CPU내에 존재하는 레지스터 들은 현재 실행 중에 있는 프로세스의 정보로 채워지고, 이러한 상태까지 우리는 프로세스의 일부라고 말할 수 있다고 하였다. 그런데 스케줄러가 등장하면서 고려해야할 상황이 생겼다.

한 프로세스가 끝나지 않은 상태에서 다른 프로세스의 우선순위가 더 높아져 Running 프로세스가 바뀔 경우, 레지스터에 저장된 정보를 다른 프로세스의 정보로 바꿔주어야 하기 때문이다. 그러나 아직 기존 프로세스도 실행이 끝나지 않았으므로, 프로세스가 다시 Running되었을 때를 대비하여 어딘가에 그 정보를 저장해 두어야 한다.

이렇게 프로세스가 바뀌면서 기존 프로세스의 레지스터 정보를 저장하고 저장했던 정보를 다시 불러오는 것은 컨텍스트 스위칭(문맥 교환)이라고 한다. 실행 중인 프로세스의 문맥을 저장하고 그 전의 문맥을 불러오기 때문이다.

다만 이 컨텍스트 스위칭은 시스템에 많은 부담을 준다. 그렇기 때문에 이를 극복하고자 이후에 쓰레드라는 이슈가 떠오른다.

Section 4 : 프로세스의 생성

```
515     WINBASEAPI
516     BOOL
517     WINAPI
518     CreateProcessW(
519         _In_opt_ LPCWSTR lpApplicationName,
520         _Inout_opt_ LPWSTR lpCommandLine,
521         _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes,
522         _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes,
523         _In_ BOOL bInheritHandles,
524         _In_ DWORD dwCreationFlags,
525         _In_opt_ LPVOID lpEnvironment,
526         _In_opt_ LPCWSTR lpCurrentDirectory,
527         _In_ LPSTARTUPINFOW lpStartupInfo,
528         _Out_ LPPROCESS_INFORMATION lpProcessInformation
529         );
```

1. lpApplicationName : 생성할 프로세스의 실행파일 이름을 전달.
2. lpCommandLine : 실행될 프로그램에 전달 인자를 넣어준다. 첫 번째 인자에 NULL을 전달하고 이 인자에 실행파일의 이름을 더불어 전달 할 수 있다. 이 경우, 표준 검색 경로를 기준으로 실행 파일을 찾게 된다.
3. lpProcessAttributes : 보안 속성 지정 인자.

4. lpThreadAttributes : 쓰레드 보안 속성 지정 인다.
5. hInheritHandle : 전달인자가 TRUE인 경우, 부모 프로세스가 소유하는 핸들 중 일부를 상속한다.
6. dwCreationFlag : 생성하는 프로세스의 특성(우선순위 등)을 결정 짓는 옵션.
7. lpEnvironment : 프로세스마다 환경 블록이라는 메모리 블록을 관리하는데, 이를 통해 프로세스가 실행에 필요한 문자열을 저장할 수 있다. 이 인자를 통해서 생성 프로세스의 환경 블록을 지정한다. NULL일 경우, 부모 프로세스에서 상속받는다.
8. lpCurrentDirectory : 생성하는 프로세스의 현재 디렉터리 설정 인자. NULL일 경우 현재 디렉터리를 상속받는다.
9. lpStartupInfo : STARTUPINFO 구조체 변수를 초기화한 다음에 이 변수의 포인터를 인자로 전달한다. 이 구조체는 생성 프로세스의 속성을 지정할 때 사용된다.
10. lpProcessInformation : 생성하는 프로세스 정보를 얻기 위해 사용되는 인자이다.