

PCB & Context Switching

Process Management

CPU가 프로세스가 여러개일 때, CPU 스케줄링을 통해 관리하는 것을 말함

이때, CPU는 각 프로세스들이 누군지 알아야 관리가 가능함

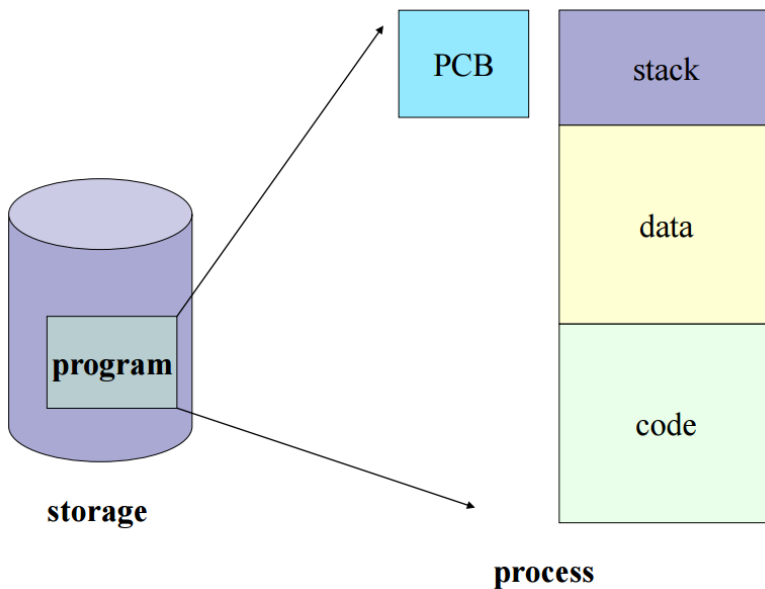
프로세스들의 특징을 갖고있는 것이 바로 **Process Metadata**

- Process Metadata
 - Process ID
 - Process State
 - Process Priority
 - CPU Registers
 - Owner
 - CPU Usage
 - Memory Usage

이 메타데이터는 프로세스가 생성되면 **PCB(Process Control Block)**이라는 곳에 저장됨

PCB(Process Control Block)

프로세스 메타데이터들을 저장해 놓는 곳, 한 PCB 안에는 한 프로세스의 정보가 담김



다시 정리해보면?

프로그램 실행 → 프로세스 생성 → 프로세스 주소 공간에 (코드, 데이터, 스택) 생성
→ 이 프로세스의 메타데이터들이 PCB에 저장

PCB가 왜 필요한가요?

CPU에서는 프로세스의 상태에 따라 교체작업이 이루어진다. (interrupt가 발생해서 할당받은 프로세스가 waiting 상태가 되고 다른 프로세스를 running으로 바꿔 올릴 때)

이때, 앞으로 다시 수행할 대기 중인 프로세스에 관한 저장 값을 PCB에 저장해두는 것이다.

PCB는 어떻게 관리되나요?

Linked List 방식으로 관리함

PCB List Head에 PCB들이 생성될 때마다 붙게 된다. 주소값으로 연결이 이루어져 있는 연결리스트이기 때문에 삽입 삭제가 용이함.

즉, 프로세스가 생성되면 해당 PCB가 생성되고 프로세스 완료시 제거됨

이렇게 수행 중인 프로세스를 변경할 때, CPU의 레지스터 정보가 변경되는 것을 Context Switching이라고 한다.

Context Switching

CPU가 이전의 프로세스 상태를 PCB에 보관하고, 또 다른 프로세스의 정보를 PCB에 읽어 레지스터에 적재하는 과정

보통 인터럽트가 발생하거나, 실행 중인 CPU 사용 허가시간을 모두 소모하거나, 입출력을 위해 대기해야 하는 경우에 Context Switching이 발생

즉, 프로세스가 Ready → Running, Running → Ready, Running → Waiting처럼 상태 변경 시 발생!

Context Switching의 OverHead란?

overhead는 과부하라는 뜻으로 보통 안좋은 말로 많이 쓰인다.

하지만 프로세스 작업 중에는 OverHead를 감수해야 하는 상황이 있다.

프로세스를 수행하다가 입출력 이벤트가 발생해서 대기 상태로 전환시킴
이때, CPU를 그냥 놀게 놔두는 것보다 다른 프로세스를 수행시키는 것이 효율적

즉, CPU에 계속 프로세스를 수행시키도록 하기 위해서 다른 프로세스를 실행시키고 Context Switching 하는 것
CPU가 놀지 않도록 만들고, 사용자에게 빠르게 일처리를 제공해주기 위한 것이다.