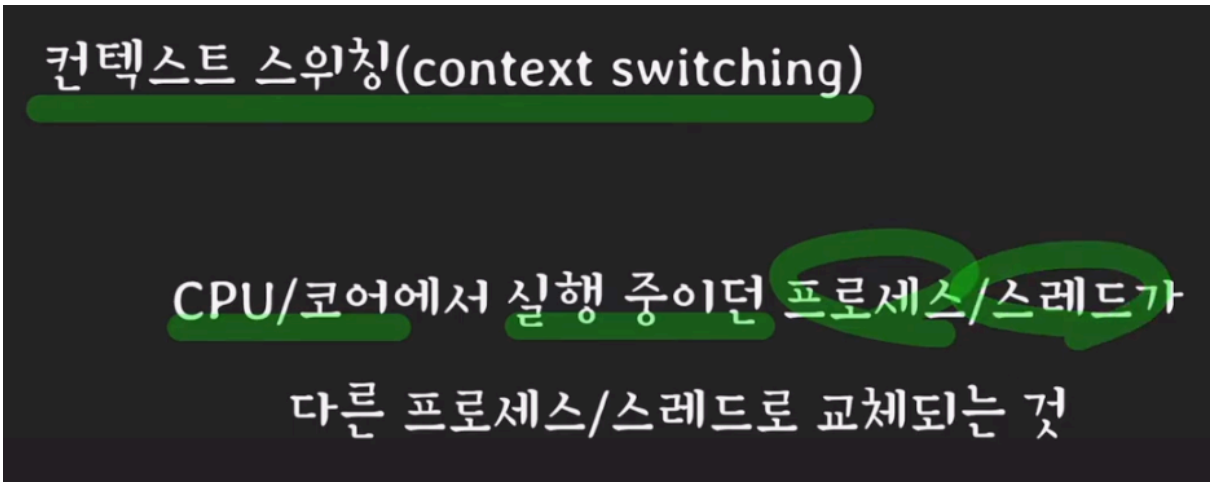
OS 2강

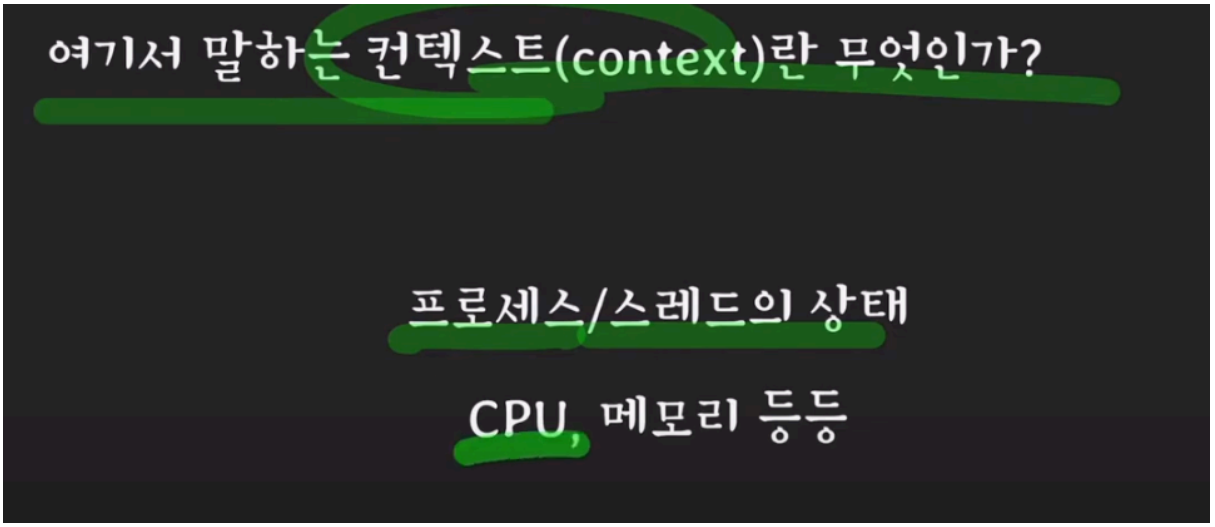
해당 내용은 유튜브 쉬운코드 님의 강의를 개인적인 공부 차원에서 정리하고자 작성한 자료입니다.

컨텍스트 스위칭

해당 강의에서는 프로세스, 스레드의 컨텍스트 스위칭 비용에 대해 다루고 있다.

컨텍스트 스위칭의 개념은 다음과 같다

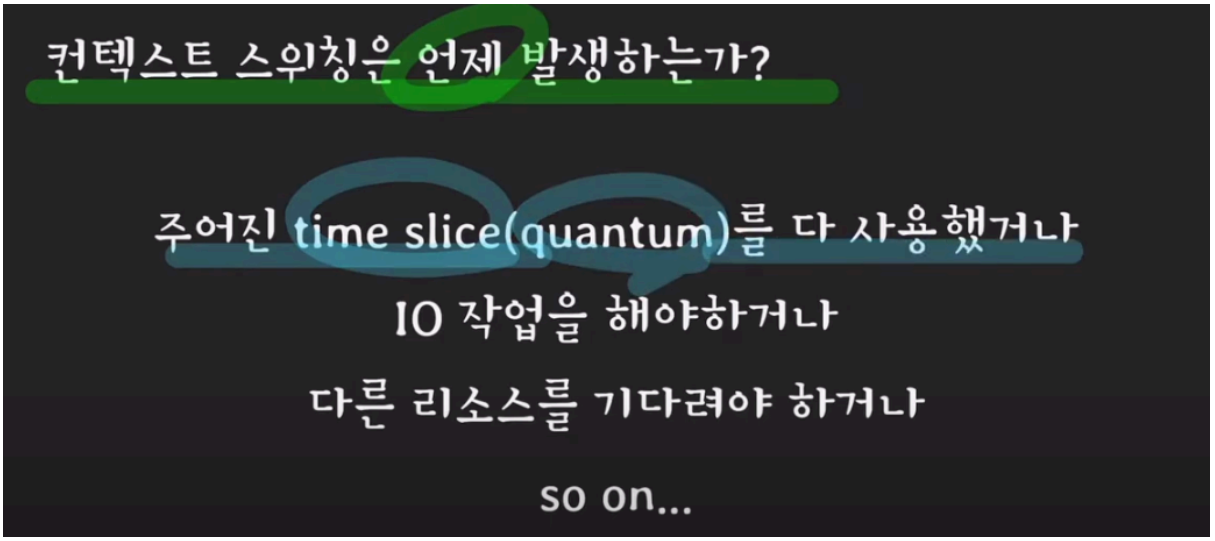


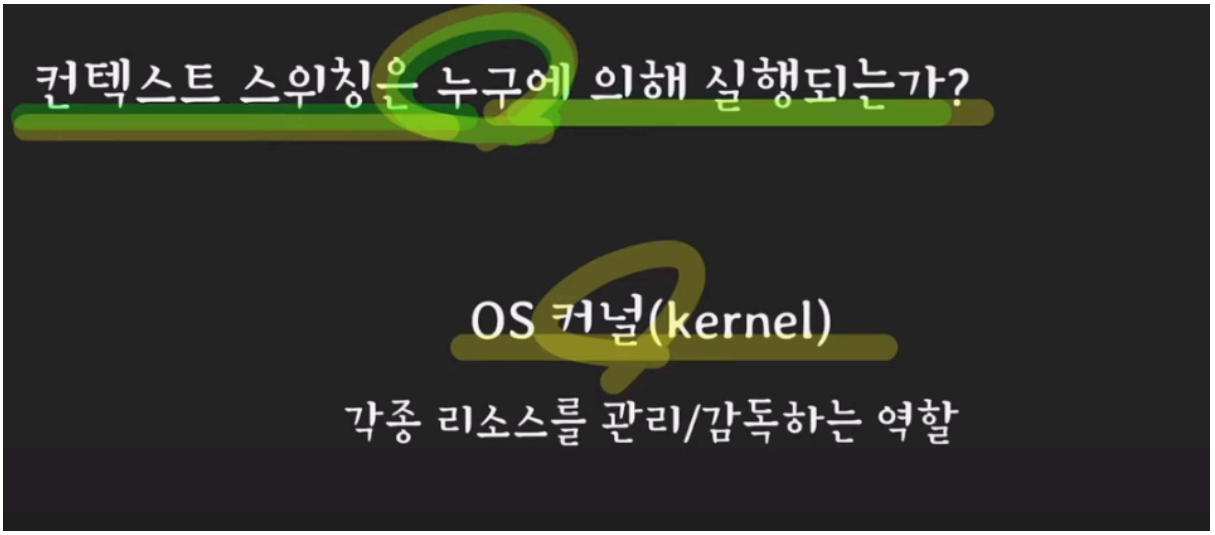


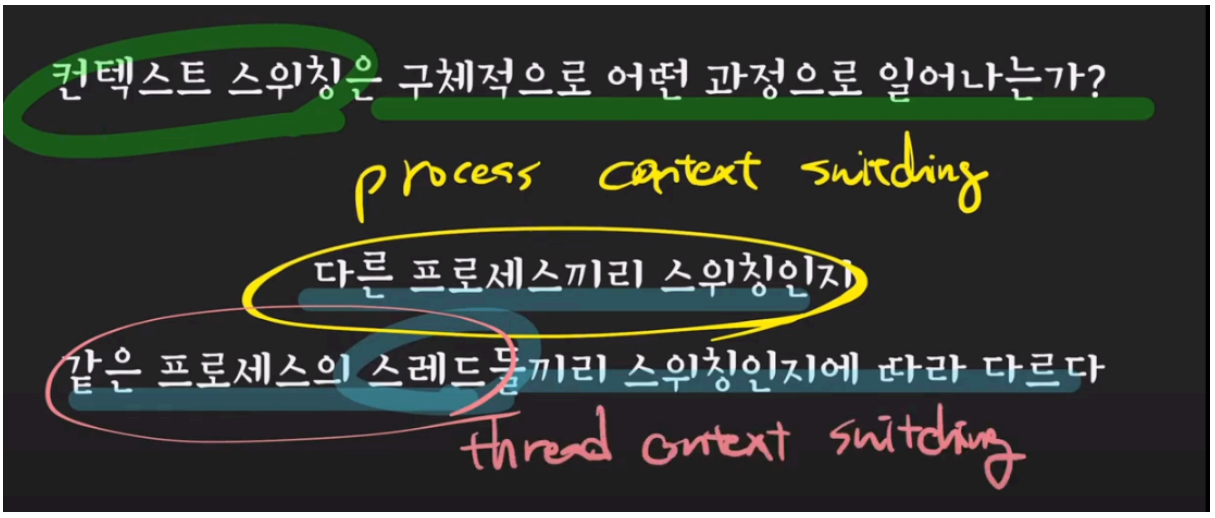
컨텍스트 스위칭이 필요한 이유는 1강에서 설명한 것 처럼

CPU를 여러 프로그램이 할당하기 위해서다

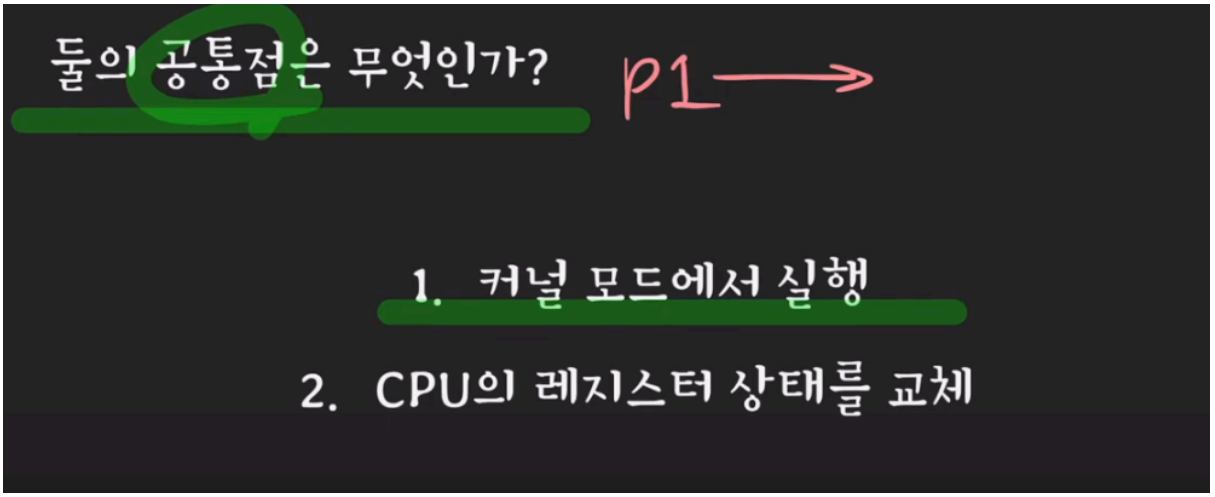
컨텍스트 스위칭이 발생하는 조건은 다음과 같다.







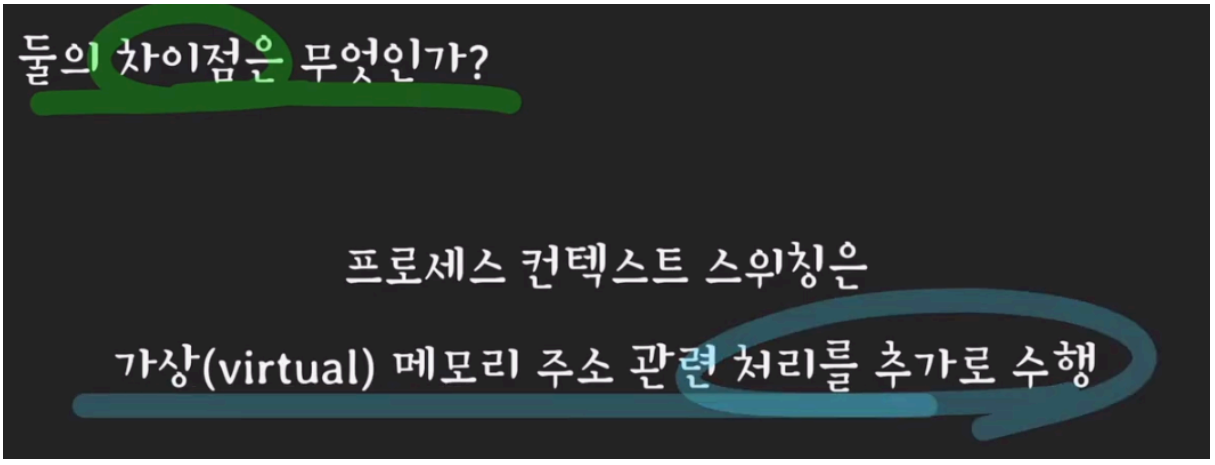
컨텍스트 스위칭은 다른 프로세스 / 스레드가 CPU를 사용하기 위한 작업이라고 했다.
그러면 작업이 바뀔때 이전 작업하던 내용을 CPU가 기억하고 있어야하는데
이게 컨텍스트 스위칭의 핵심적인 작업이다.



프로세스 컨텍스트 스위칭이 CPU 레지스터 상태를 교체하는건 당연히 일어나는 거지만

(서로 독립된 실행환경을 가지기 때문에)

스레드는 같은 프로세스의 주소 공간을 공유하지만 각각 본인의 독립적인 공간 (스택, CPU 레지스터 상태 등등)을 가지고 있기 때문에 스레드 컨텍스트 스위칭에도 해당 작업이 필요하다.



여기서 MMU, TLB, 페이지 테이블의 개념이 등장하는데

강의에서는 간단하게 하고 넘어갔기 때문에 조금 찾아봤다

각 프로세스는 독립적인

가상 메모리 주소 공간을 가지며, 이걸 운영체제는 페이지 테이블 이라는 자료 구조로 관리하게 됨.

프로세스가 메모리에 접근하려고 할때 CPU는 해당 프로세스의 **가상 주소**를 통해서 접근한다

이때 MMU는 페이지 테이블의 정보를 참고해 가상 주소 → 물리 주소로 변환한다.

이 변환 과정을 가볍게 하기 위해서 TLB 라는 곳에 이 매핑 정보를 캐싱해 접근을 수월하게 함

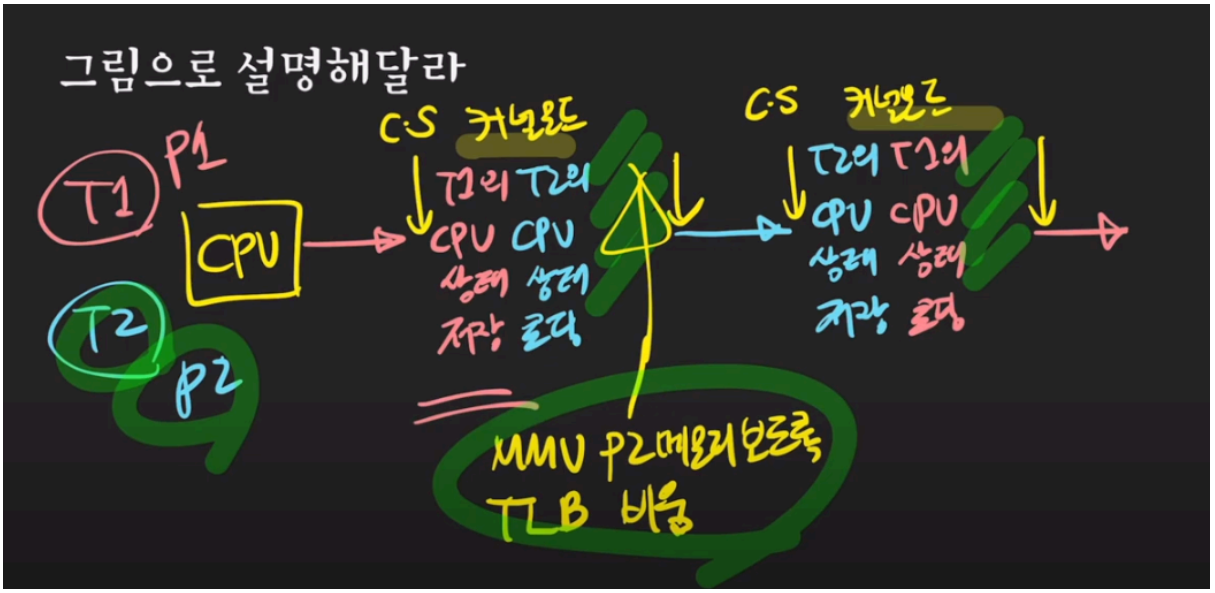
근데 만약에 컨텍스트 스위칭이 일어나면 현재 TLB에 있는 매핑 정보가 일치하지 않기 때문에 이걸 비워줘야 하는 작업을 하고, 새로운 정보를 채워줘야한다.

이렇기 때문에 오버헤드 (추가비용)이 발생하는 것

MMU : 가상 주소 → 물리 주소로 바꿔주는 하드웨어

TLB : MMU를 수행하기 위한 캐시 정보

페이지 테이블 : MMU가 가상 → 물리 로 바꾸는데 필요한 매핑 정보 (프로세스 당 하나 씩 존재)



이걸 이해하면 쉽게 이해된다.

스레드는 프로세스 “내의” 작업단위기 때문에 TLB를 비우고, 가상 메모리 처리 관련된 작업을 하지 않기 때문에 프로세스 컨텍스트 스위칭보다 훨씬 가볍다는거임

항목	프로세스 전환	스레드 전환 (동일 프로세스)
페이지 테이블 교체	✅ (다른 주소 공간)	❌ (동일한 주소 공간)
MMU 기준 주소 변경	✅ (예: CR3 레지스터)	❌
TLB flush 필요 여부	✅ 반드시 필요 (다른 매핑 정보)	❌ 필요 없음 (같은 매핑 유지)

✅ 3. 그럼 스레드 스위칭 시에는 뭐가 바뀌는가?

- 스택 포인터 (스레드는 각자 스택 가짐)
- 레지스터 상태
- 프로그램 카운터 (PC)
- 우선순위, 상태 등 스케줄링 정보

👉 하지만 **페이지 테이블, MMU, TLB는 그대로 유지됨**
→ 그래서 **스레드 컨텍스트 스위칭은 훨씬 가볍고 빠름**