스마트시스템 운영체제
(LD01600)

김준철

정보시스템공학과

greensday@sungshin.ac.kr

4주차 강의 -1

	주차	강의 목차
9.2	1	과목소개 / 운영체제 개요
9.9	2	컴퓨터 시스템 구조
9.16	3	프로세스와 스레드1
9.22	4	프로세스와 스레드2, CPU스케쥴링1
10.7	5	CPU스케쥴링2
10.14	6	프로세스 동기화
10.21	7	교착 상태
10.28	8	중간고사
11.4	9	물리 메모리 관리
11.11	10	가상메모리 기초
11.18	11	가상메모리 관리
11.25	12	입출력시스템1
12.2	13	입출력시스템2, 파일시스템1
12.9	14	파일시스템2
12.16	15	기말고사

휴강(9.30)
(추석)

ch.03 프로세스와 스레드2

Process and Thread

- **Process(프로세스)**
  - Process는 독자적인 address space를 갖음
  - 다른 process와 자원을 공유하지 않음
    - process간 **protection**
  - 하나 이상(여러 개)의 thread를 가짐
- **Thread(스레드)** CPU의 일 처리(수행)의 기본 단위 (lightweight process)
  - Process 내에 존재하는 CPU의 작업 단위
  - 하나의 process에 여러 개의 thread가 존재할 수 있음
  - 동일 Process내의 thread들은 (process의) address space, resource를 공유함
    - thread간 **concurrency**
- **Hyper Thread(하이퍼스레드)**
  - Context switching 시 Hareware의 지원을 받는 구조(2개의 레지스터 세트)
  - Context switching 시 overhead의 감소가 목적

Thread의 개념

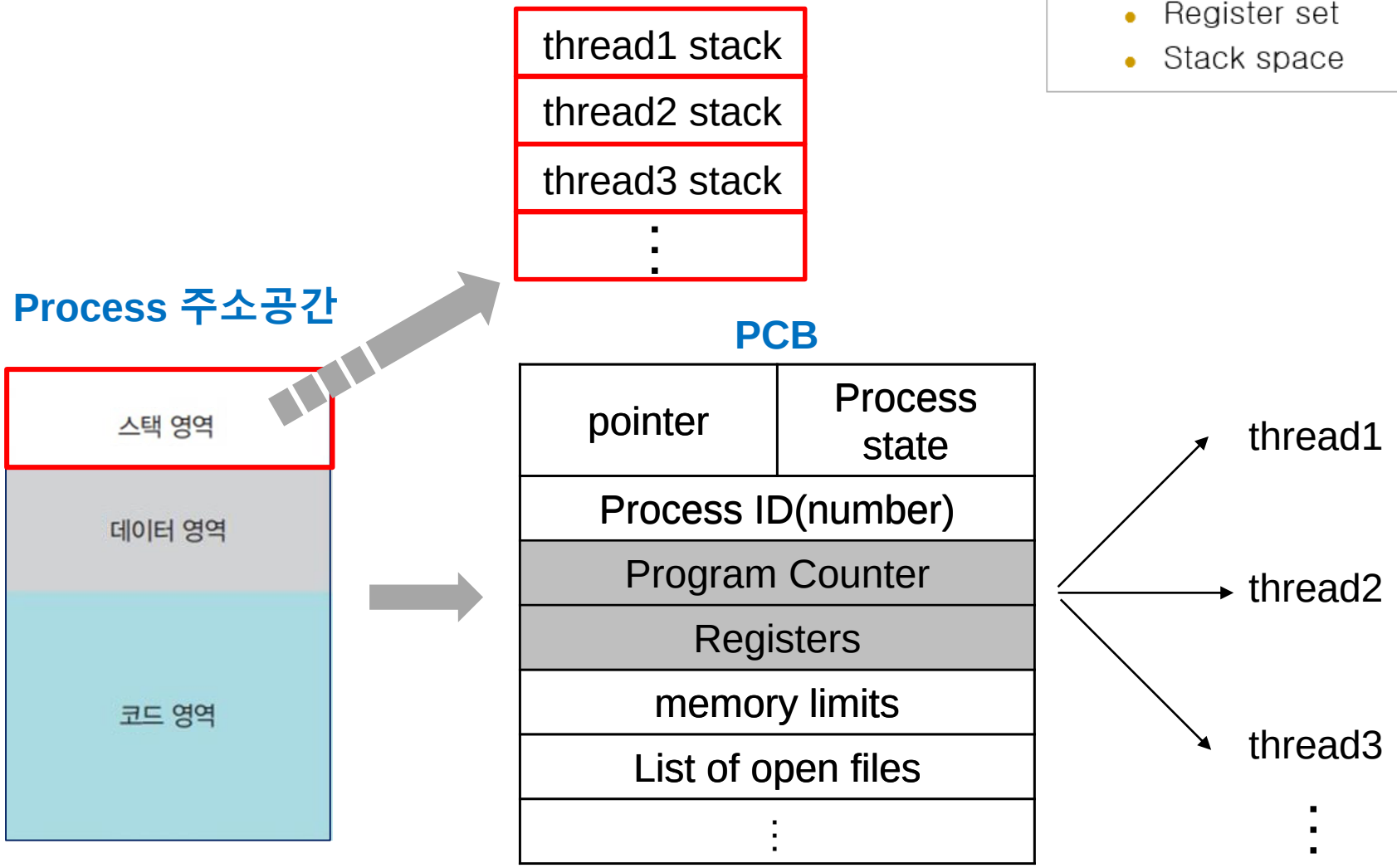
- **Thread(스레드)의 개념**
  - process의 내용 중 **CPU 수행에 관한 부분**만 분리한 실행 단위
    (register set 내용, stack 내용)
  - **process 하나는 한 개 이상의 thread로 나눌 수 있음**
  - 같은 process내의 thread들은 ┌ **동일한 memory 주소 공간을 공유**
    └ **process 관리 정보를 다른 thread들과 공유**
    - **독자적인 Register값들(GPR, PC, SP 등)이 필요함** (CPU 실행 상태 기록)
    - **별도의 stack 필요** (함수(프로시저)의 호출, 다른 실행들의 기록)

- **thread의 구성**
  - Program counter
  - Register set
  - Stack 영역

- **thread가 동료 thread와 공유하는 부분**
  - Code 영역
  - Data 영역
  - OS 자원

Process와 thread

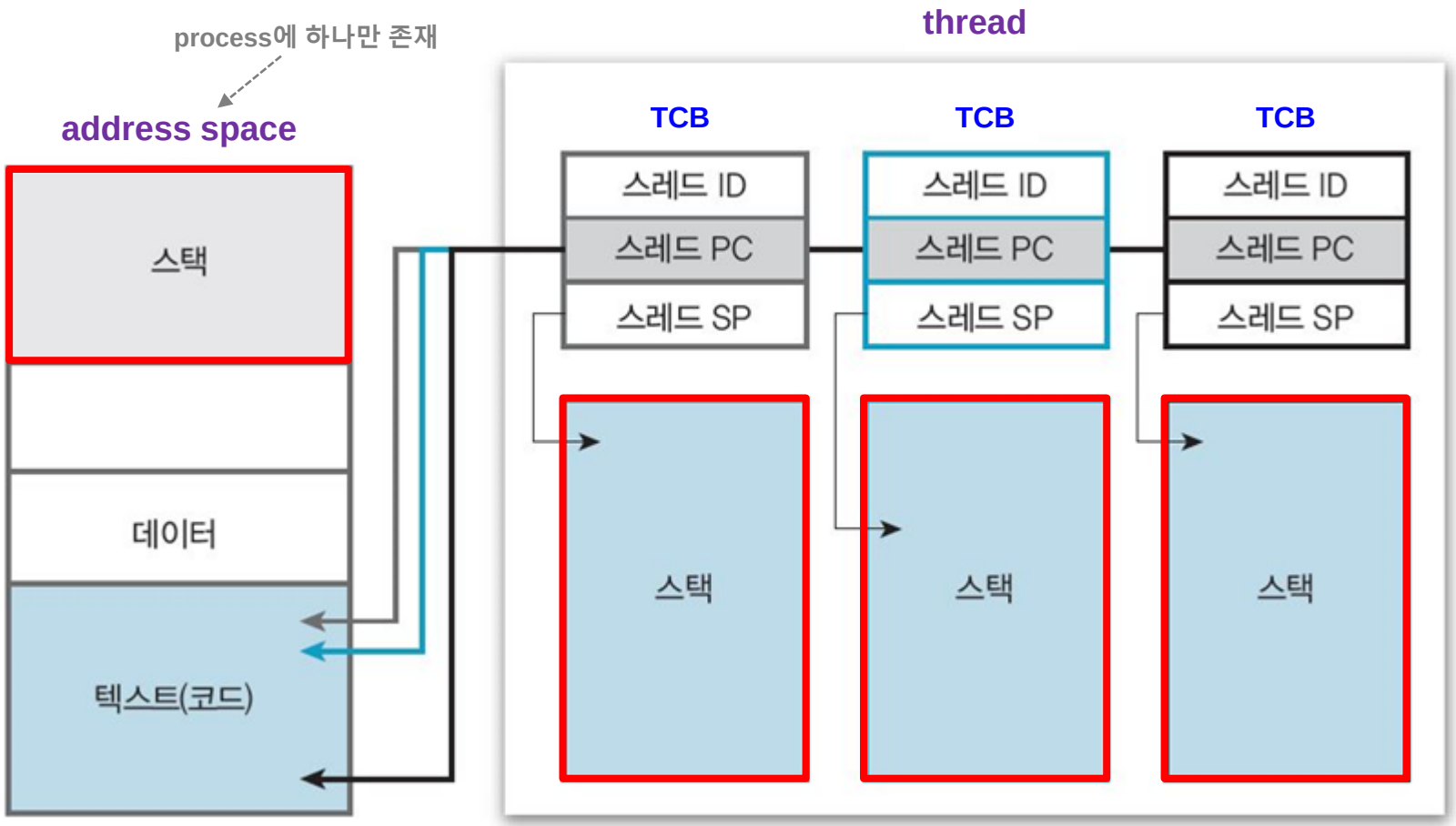
- thread의 구성
  - Program counter
  - Register set
  - Stack space



[그림] process와 thread

TCB(Thread Control Block)

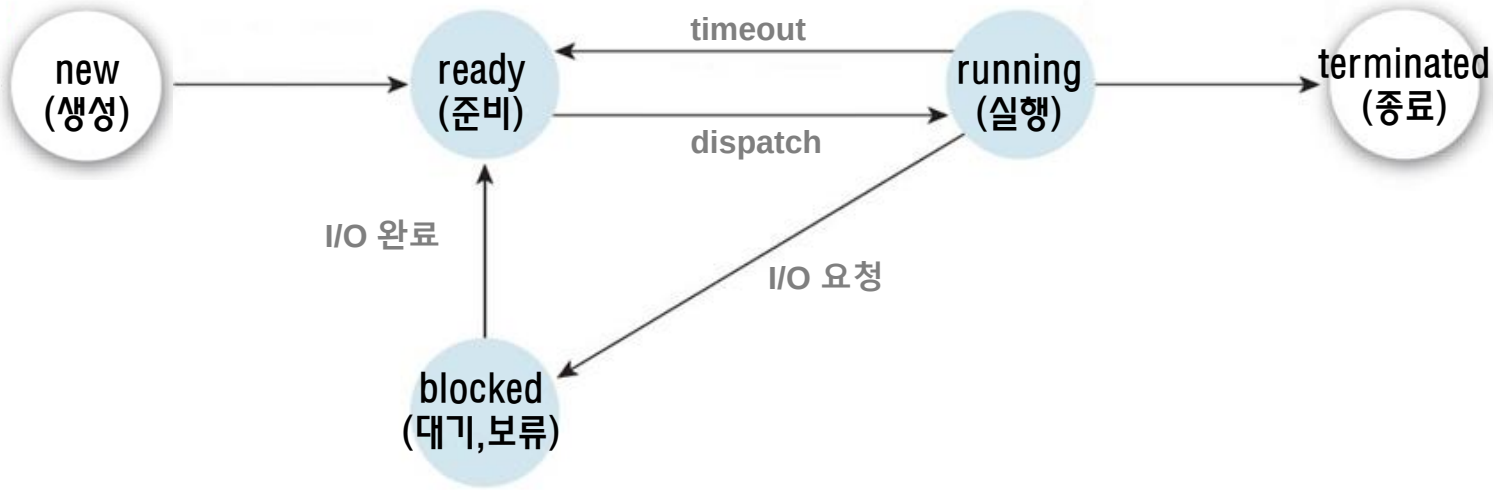
- thread의 구성
  - Program counter
  - Register set
  - Stack space



[그림] TCB (Thread Control Block)

thread의 상태변화

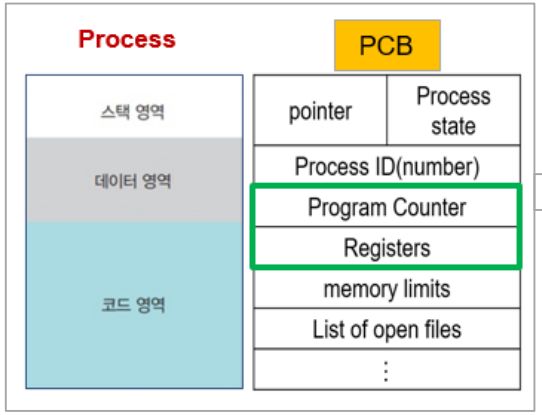
- thread의 상태변화



[그림] thread 상태변화

단일 thread와 다중(multi) thread

- process 관리 측면에서 단일 thread와 다중 thread



[그림] process와 PCB

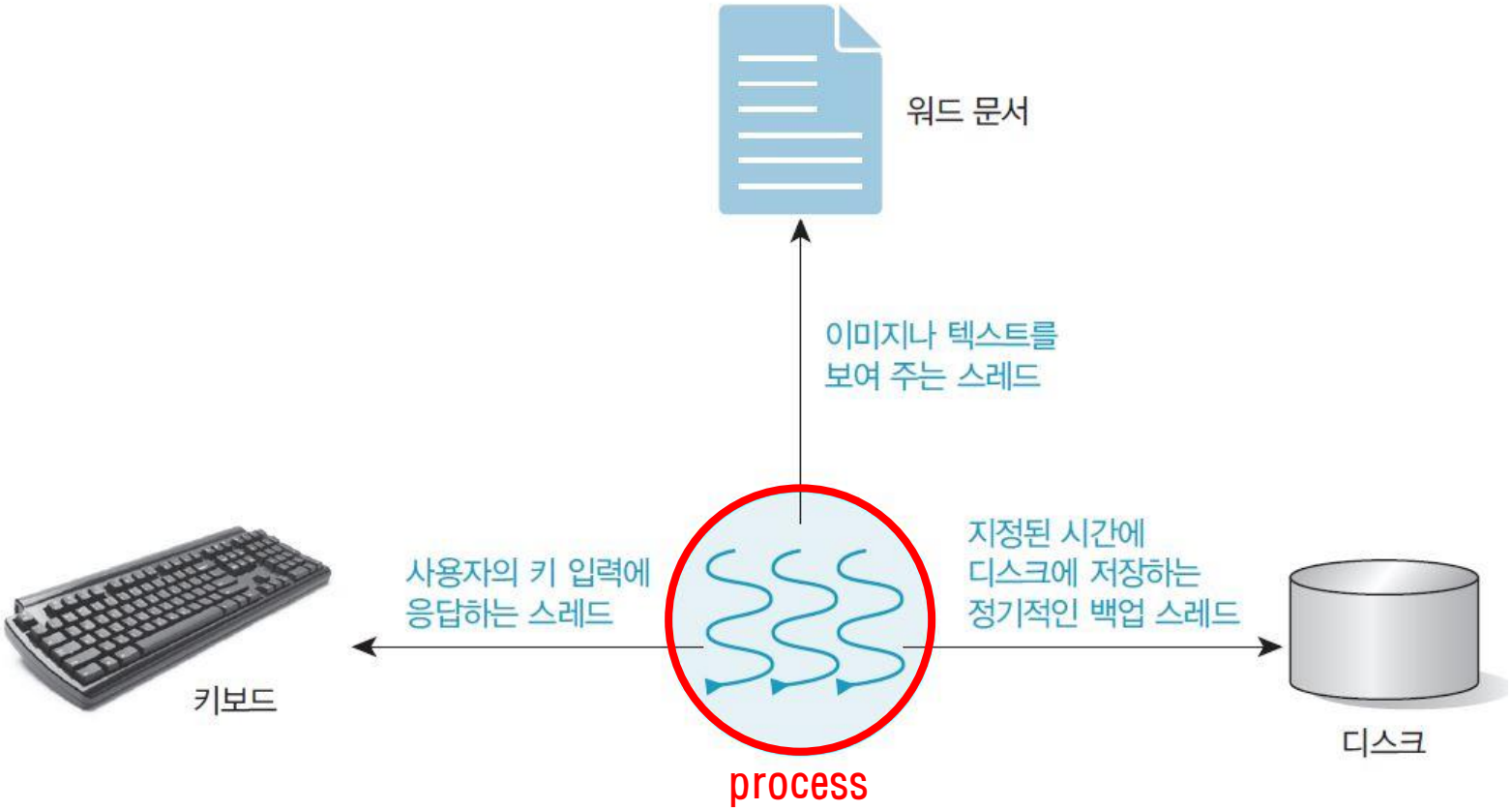
(a) 단일 thread를 가지는 process

(b) 다중 thread를 가지는 process

[그림] 단일 thread process와 다중 thread process

- multi(다중) thread : 프로그램 하나를 여러 실행 단위로 쪼개어 실행
- thread별로 실행 환경 정보가 따로 있지만 서로 많은 정보를 공유함
  - process 간의 context switching보다 thread 간의 context switching이 훨씬 경제적

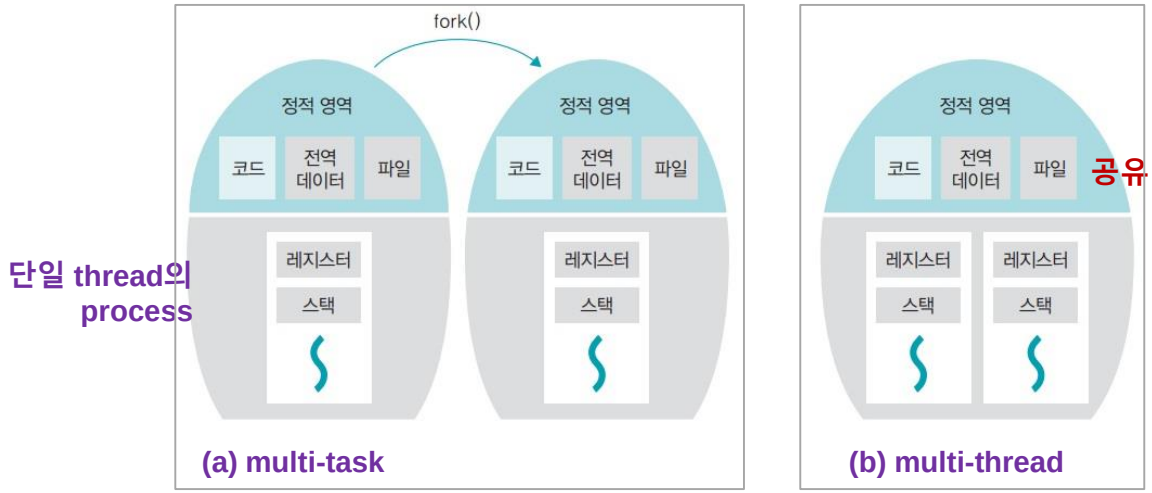
Multi-thread 사용의 예



[그림] multi- thread를 이용한 워드 편집기 프로세서

Multi-thread의 구조와 예

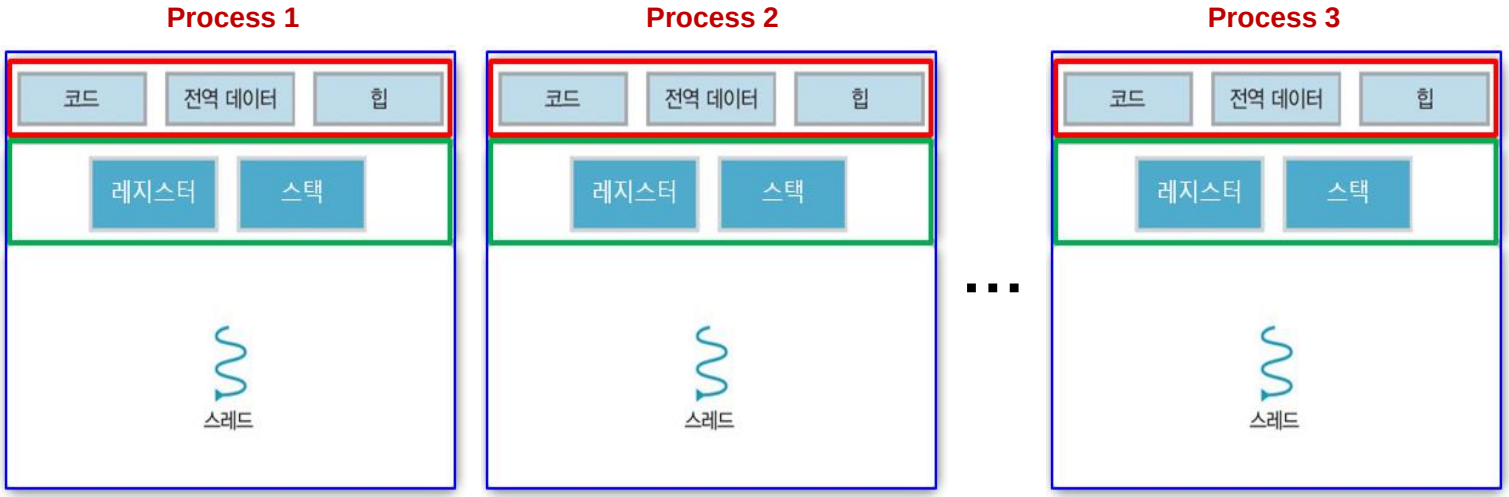
- **multi-task와 multi-thread**
  - multi-task : **여러 개의 process**로 구성된 것
  - multi-thread : 하나의 process에 **여러 개의 thread**로 구성된 것
- **multi-task과 multi-thread의 차이**
  - fork( ) 시스템 호출(**여러 개의 process 생성**)
    → 필요 없는 정적 영역이 여러 개가 됨
  - multi-thread 사용
    → **코드, 파일 등의 자원을 공유**(자원의 낭비를 막고 효율성 향상)

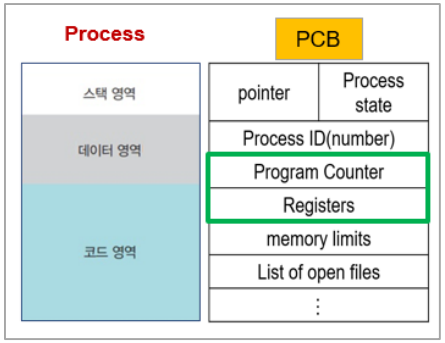


[그림] multi-task과 multi-thread의 구조

Context switching - processes

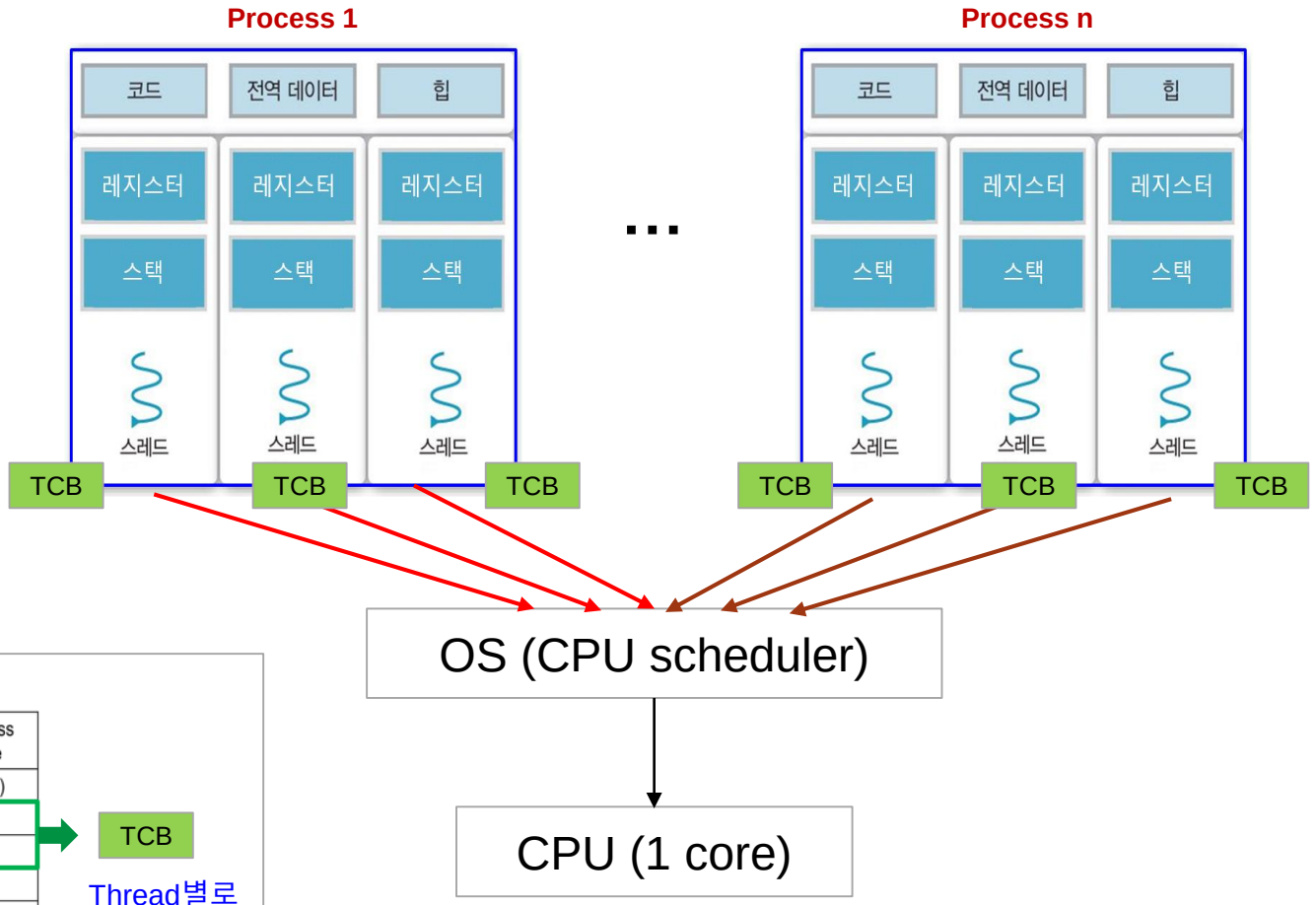
- Multiple processes





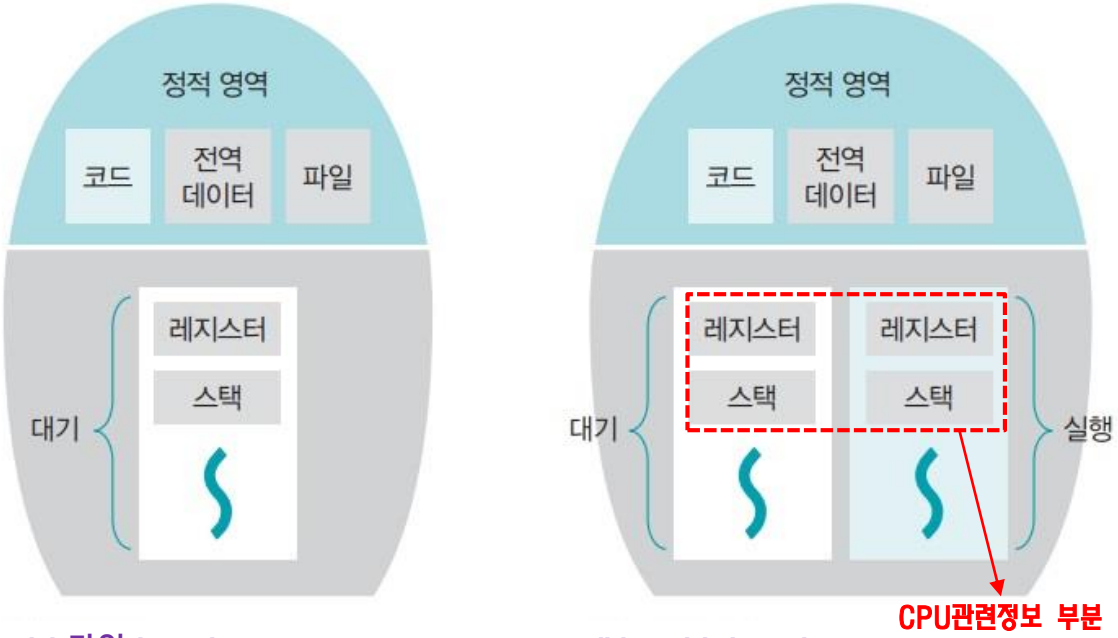
Context switching - threads

- Multiple threads



Multi-thread의 장단점

- multi-thread의 장점
  - 응답성 향상
  - 자원 공유
  - 효율성 향상

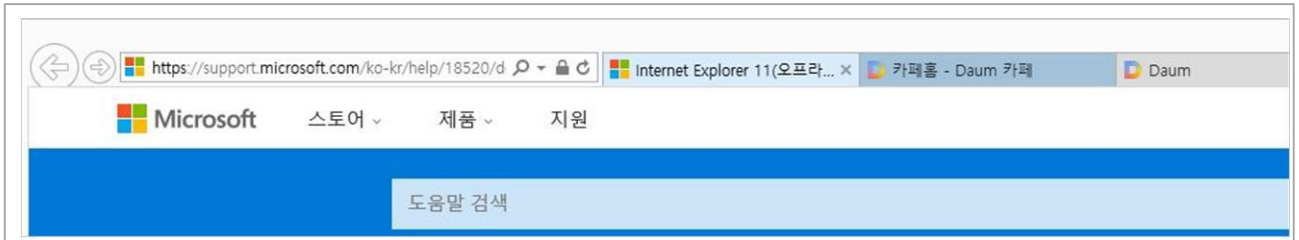


(a) 단일thread (b) multi-thread

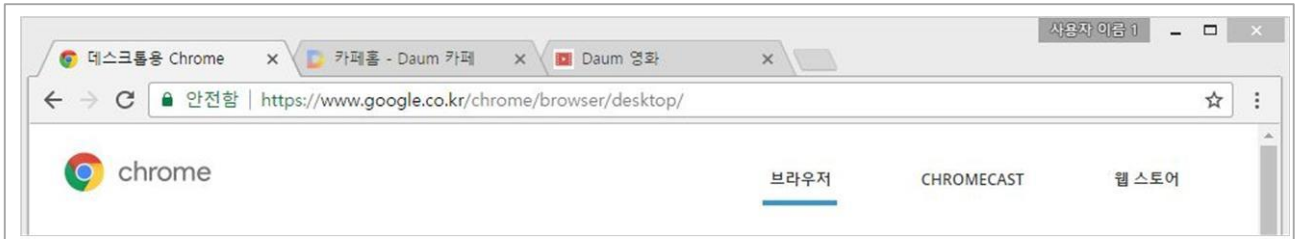
[그림] 단일 thread와 multi-thread의 구조

Multi-thread의 장단점

- multi-thread의 단점
  - 모든 thread가 자원을 공유하기 때문에 **한 thread에 문제가 생기면 전체 process에 영향을 미침**
  - 인터넷 익스플로러에서 여러 개의 화면을 동시에 띄웠는데 그중 하나에 문제가 생기면 인터넷 익스플로러 전체가 종료



(a) multi-thread를 이용하는 인터넷 익스플로러



(a) multi-tasking을 이용하는 크롬

[그림] multi tap 기능 구현 방식의 차이

감사합니다.