

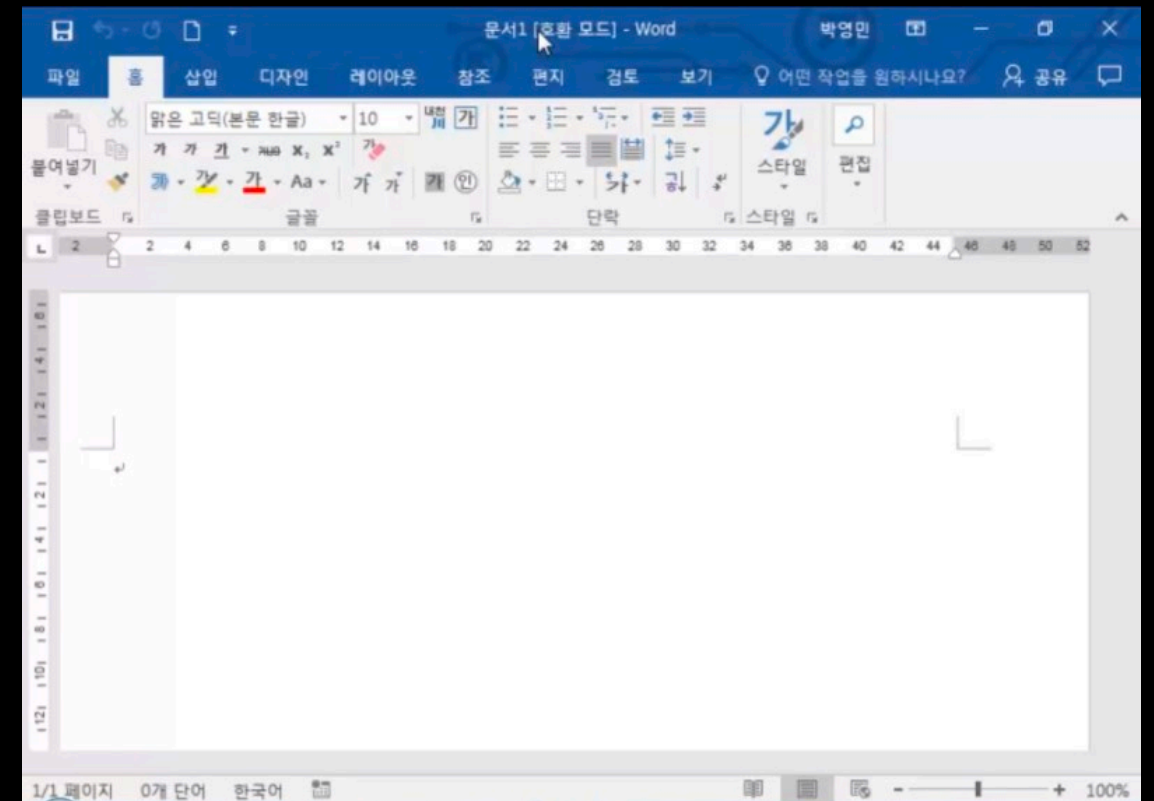
# 프로세스와 스레드

CS 01.

# 프로그램 VS 프로세스

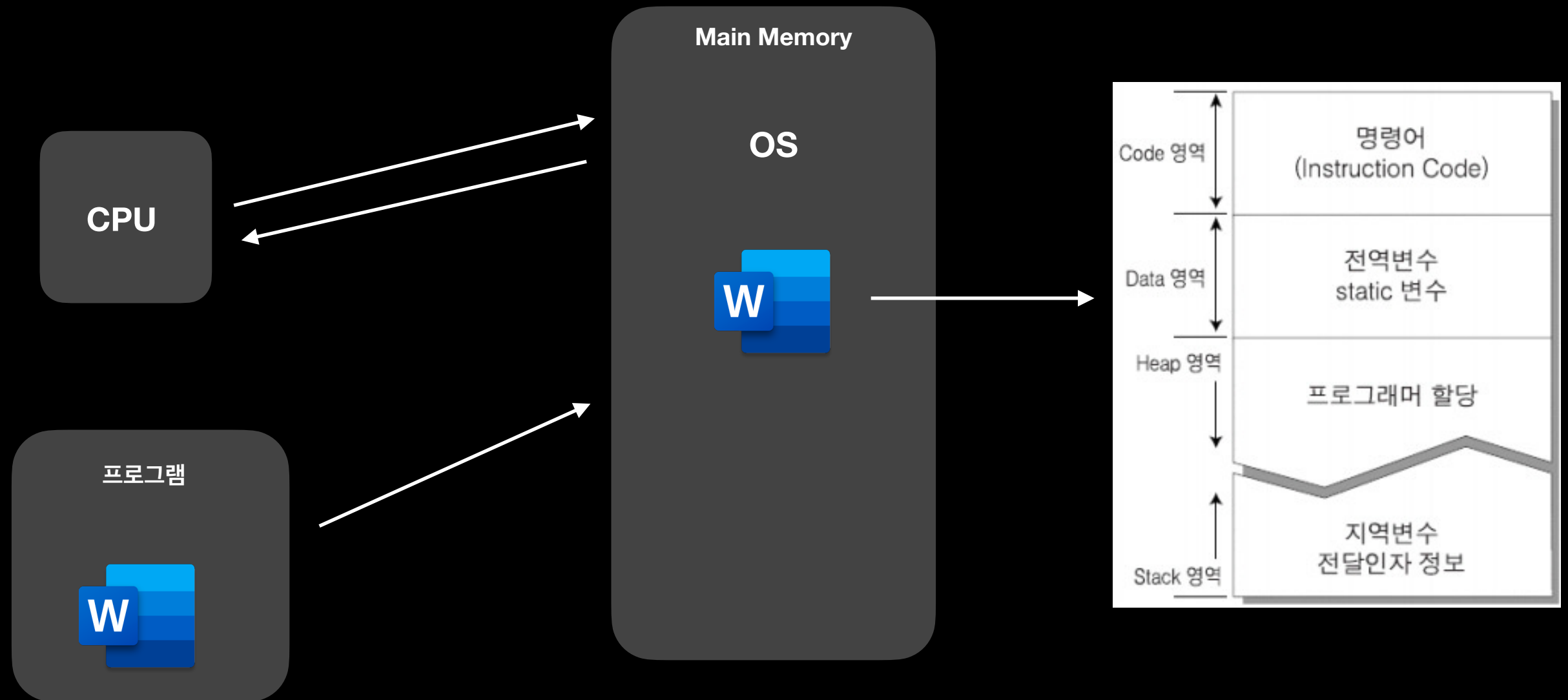


프로그램



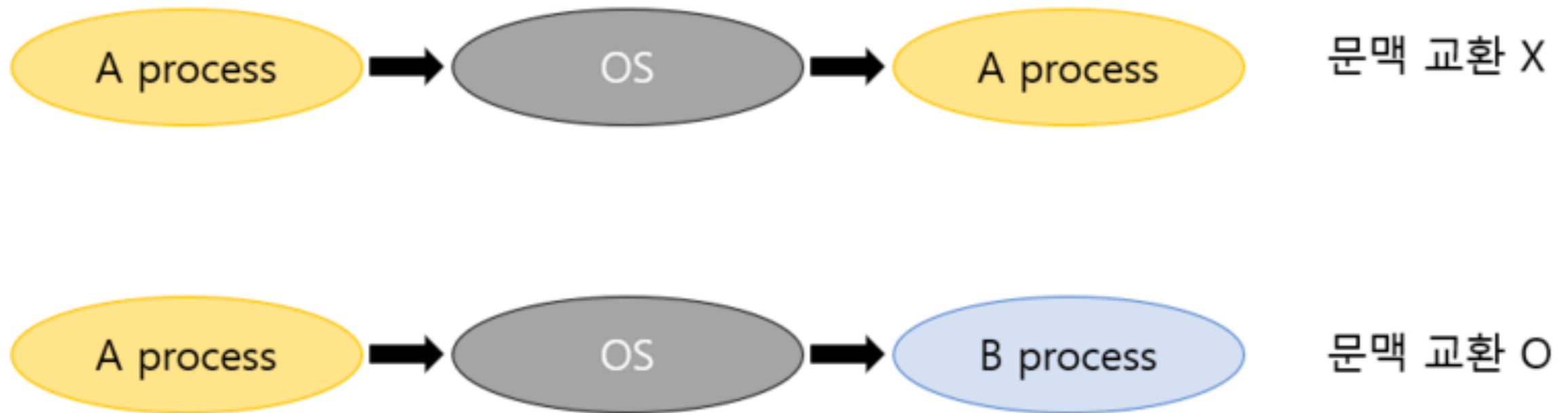
프로세스

# 프로그램 VS 프로세스



# 프로세스 문맥

프로세스 문맥 : 현재 어디까지 실행되었는지를 파악하는 것



# 프로세스의 상태

NEW

Running

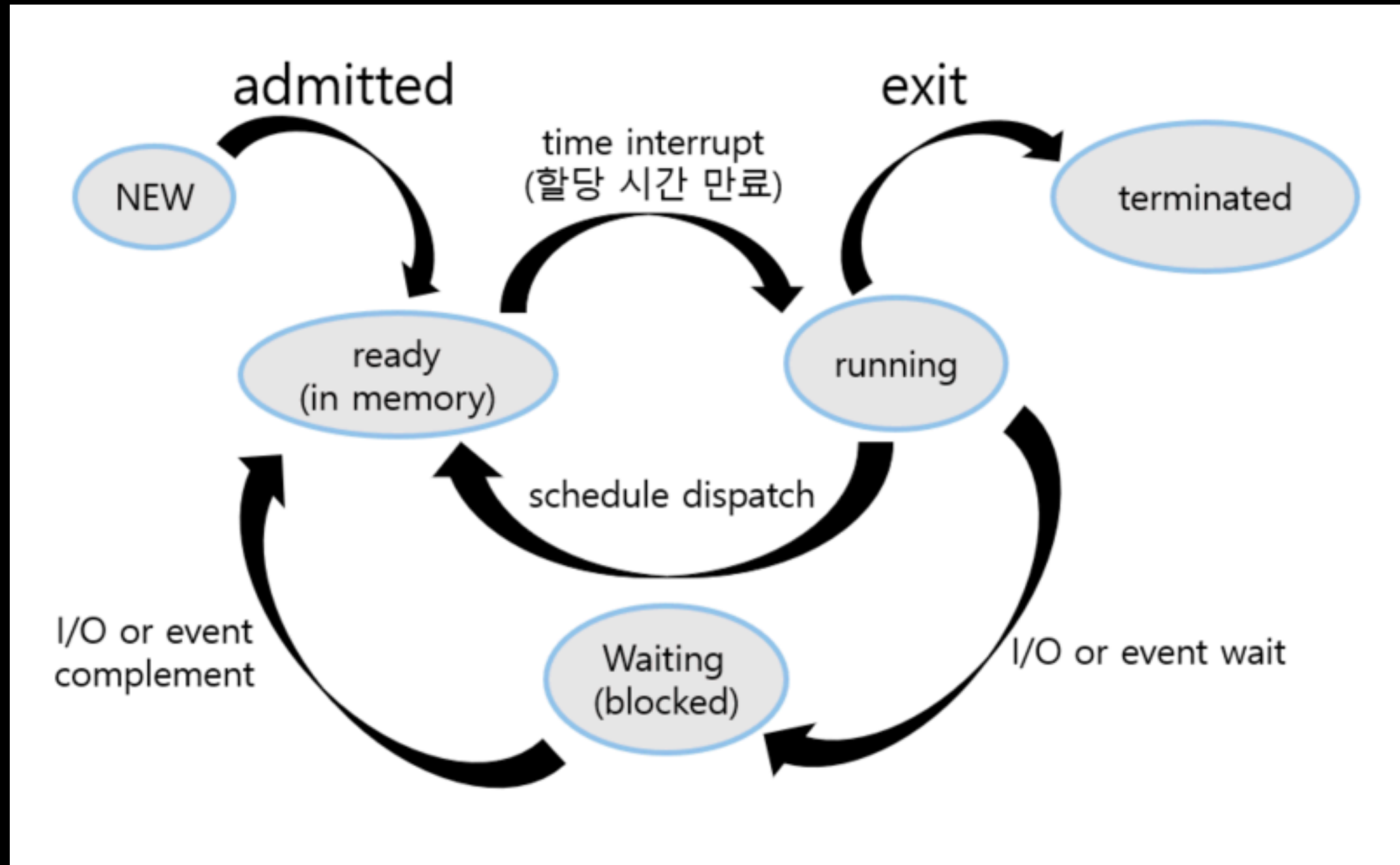
Ready

Blocked  
(Wait, Sleep)

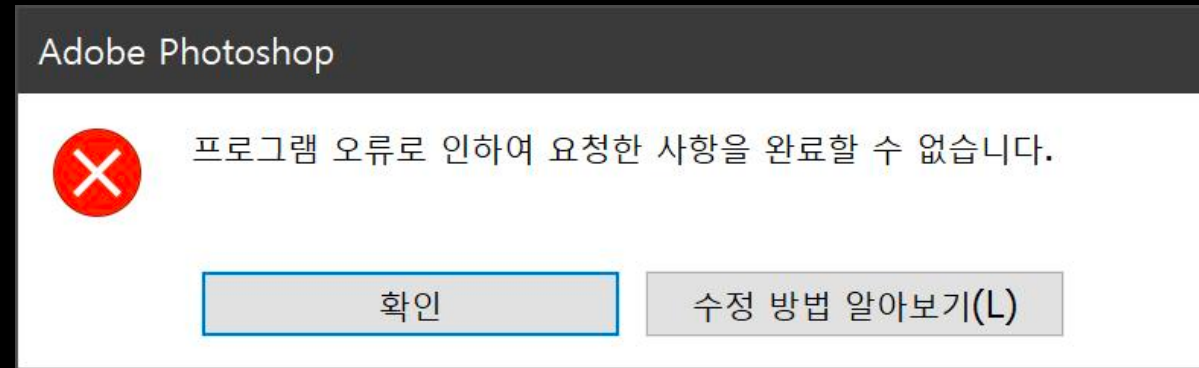
Suspended  
(Stopped)

Terminated

# 프로세스의 상태(전이도)



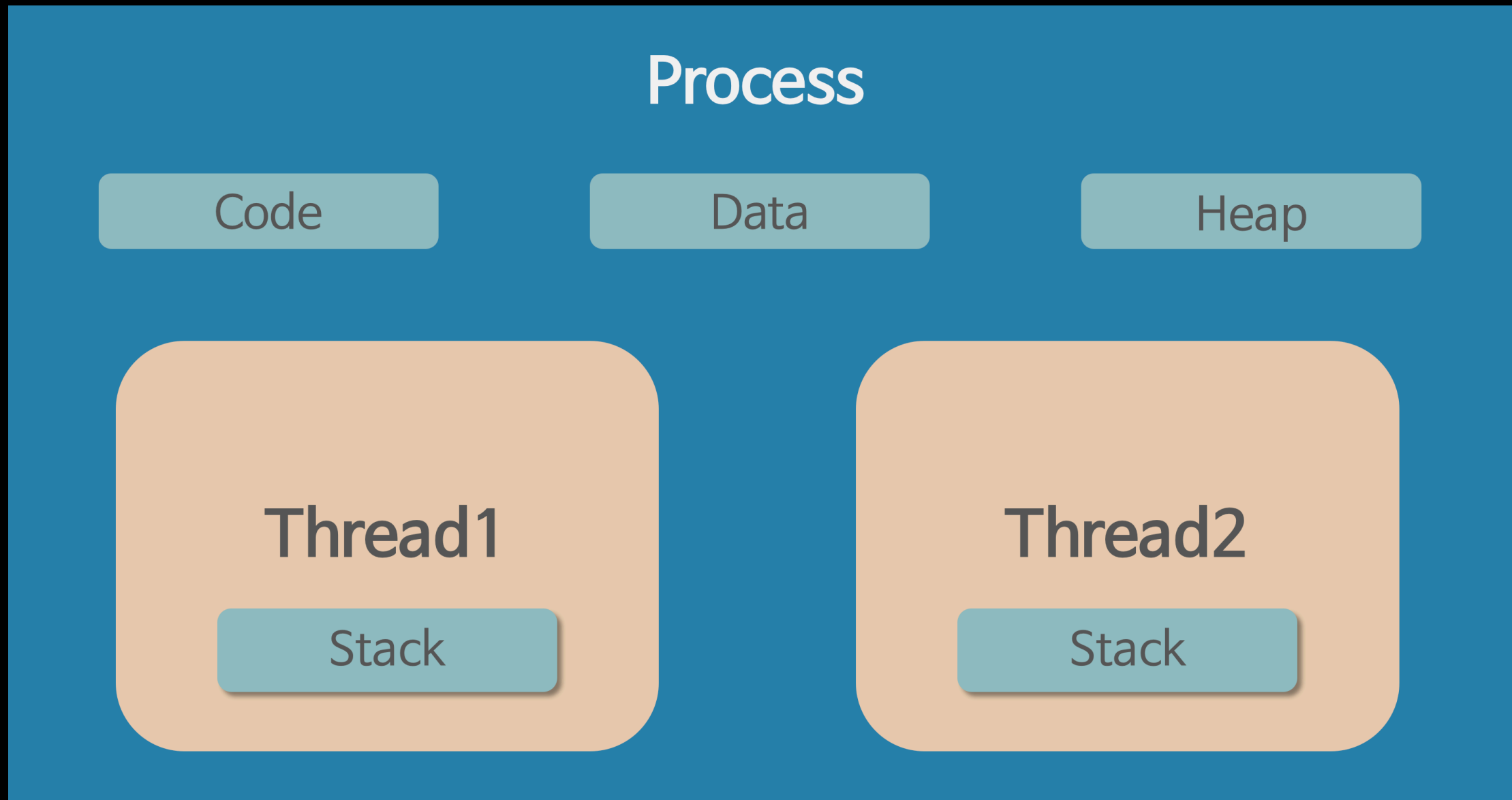
# PCB (Process Control Block)란?



## PCB

- 프로세스 번호(PID)
- 프로세스 상태(Status)
- 우선순위(Priority)
- 프로그램 카운터 값(PC)
- 메모리 포인터
- Context Data
- 할당받은 자원 목록
- 계정 정보: CPU를 사용한 시간 등
- 입출력 정보

# 스레드





# 프로세스와 스레드 차이점

| 차이점        | 프로세스      | 스레드               |
|------------|-----------|-------------------|
| 정의         | 실행중인 프로그램 | 프로세스의 실행 단위       |
| 생성 / 종료 시간 | 많은 시간 소요  | 적은 시간 소요          |
| 컨텍스트 전환    | 많은 시간 소요  | 적은 시간 소요          |
| 상호작용       | IPC 사용    | 공유 메모리 사용         |
| 자원 소모      | 많음        | 적음                |
| 독립성        | 각각 독립적    | 스택만 독립적이고 이외에는 공유 |

- IPC? :프로세스 간 통신(Inter-Process Communication, IPC)이란 프로세스들 사이에 서로 데이터를 주고받는 행위 또는 그에 대한 방법이나 경로

# 프로세스와 스레드 차이점

Process

Registers

Memory

Heap

Static

Code

Stack

Process

Thread

Registers

Thread

Registers

Thread

Registers

Memory

Heap

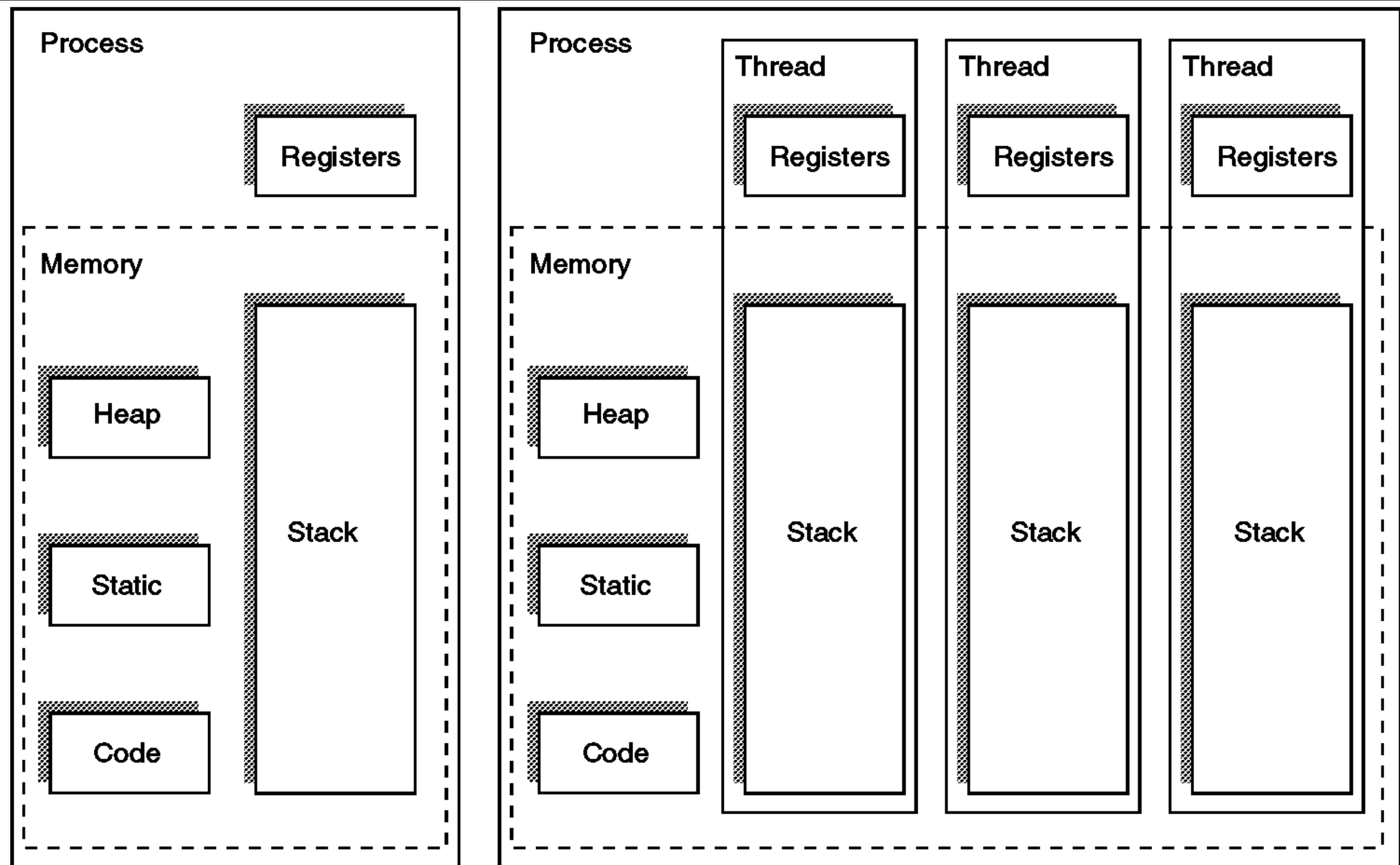
Static

Code

Stack

Stack

Stack



# 스레드 종류

## 유저레벨

프로세스 1개(사용자 스레드 N개) 당 커널 스레드 1개가 할당된다.

프로세스 내에 스레드 라이브러리가 있어서 커널의 도움 없이 스레드의 스케줄링을 할 수 있다.

스레드 정보(TCB, Thread Control Block)는 프로세스 내에서, 프로세스 정보(PCB, Process Control Block)는 커널에서 관리한다.

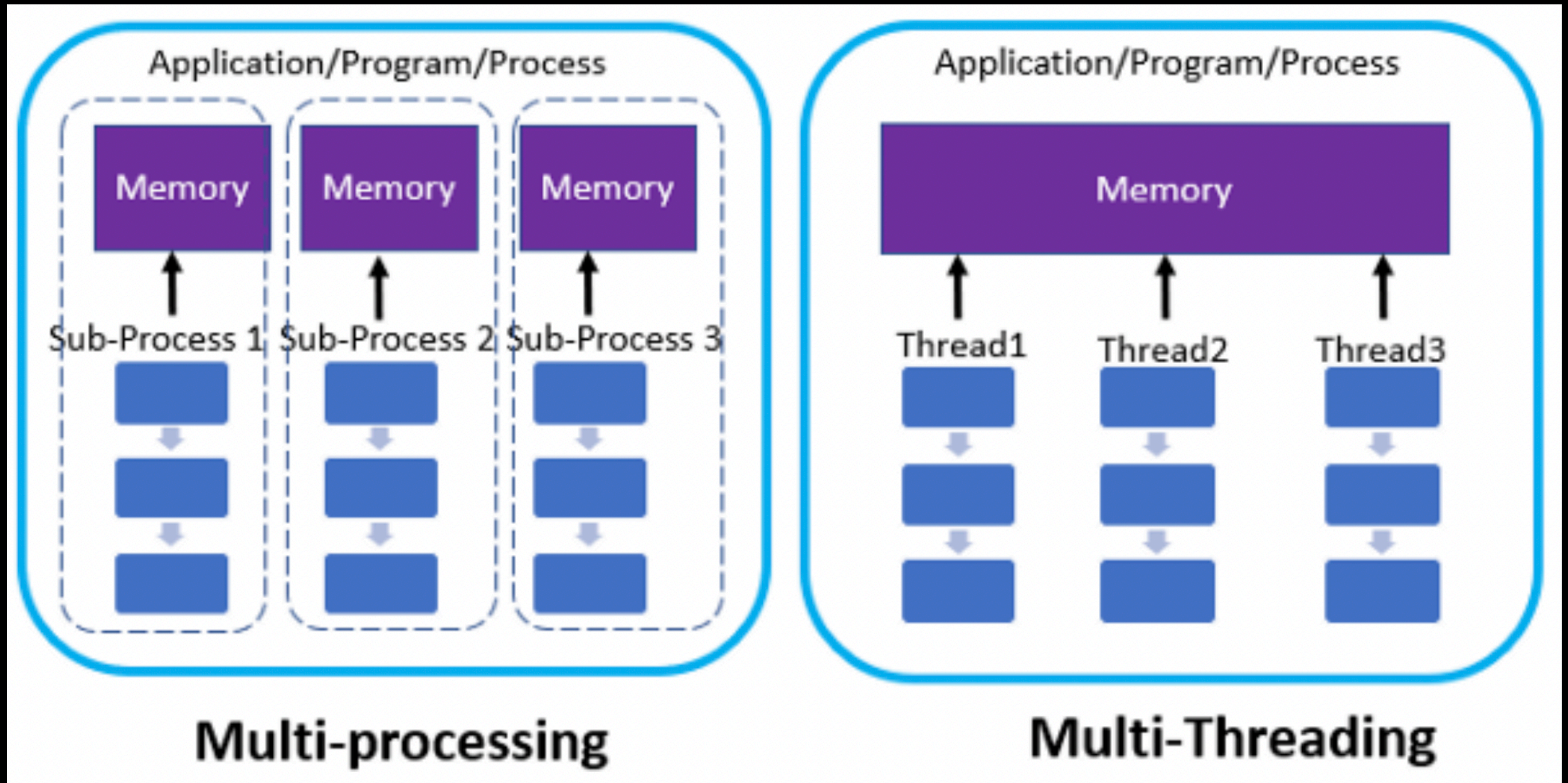
## 커널 레벨

프로세스 내의 사용자 스레드 1개 당 커널 스레드 1개가 할당된다. (사용자 스레드를 생성하면 할당할 커널 스레드를 1개 생성한다.)

프로세스 내에 스레드 라이브러리가 없어서 커널 스레드를 스케줄 하여 매핑된 사용자 스레드를 동작시킨다.

커널이 전체 TCB와 PCB를 관리한다.

# 멀티 프로세스 VS 멀티 스레드



# 장단점

| 종류 | 멀티 프로세스  | 멀티 스레드  |
|----|--|---|
| 장점 | <ul style="list-style-type: none"><li>-하나의 프로세스가 죽어도 다른 프로세스에는 영향을 끼치지 않음</li></ul>  | <ul style="list-style-type: none"><li>- 프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어들어 자원을 효율적으로 관리 가능</li><li>- Code, Data, Heap 영역을 공유하기 때문에 데이터를 주고 받는 것이 간단해지고 자원 소모가 적음</li><li>- 스레드 사이 작업량이 작아 문맥교환이 빠르며 시스템 처리량 증가</li></ul> |
| 단점 | <ul style="list-style-type: none"><li>- 각각 독립된 메모리 영역을 갖고 있어 작업량이 많을수록 오버헤드가 발생하고 문맥 교환(Context Switching)으로 인한 성능 저하를 유발</li><li>- 프로세스 사이의 통신이 복잡(IPC)</li></ul> | <ul style="list-style-type: none"><li>- 프로그램 디버깅이 까다로움</li><li>- 하나의 스레드에 문제가 생기면 전체적인 프로세스에 영향을 끼침</li><li>- 동기화 문제 발생(전역 변수를 이용하기 때문)</li><li>- 단일 프로세스 시스템에서 효과를 기대하기 어려움</li><li>- 다른 프로세스에서 스레드 제어 불가</li></ul>  |