1. 프로세스와 스레드의 차이.

- 프로세스는 OS로부터 자원을 할당 받는 작업의 단위
- 프로세스는 서로 완벽히 독립적인 공간을 가진다.
  (각자의 스택, 데이터 영역).

⇒ 멀티 프로세스 방식에서 독립적인 프로세스 끼리 자원을 주고 받는 통신 과정이 필요하다. (공유 메모리 등의 IPC)

- 스레드는 프로세스가 할당 받은 자원을 이용하는 실행의 단위.(흐름).
- 스레드는 각자의 스택을 갖고 데이터 영역을 공유한다.

⇒ 데이터 영역을 통해 쉽고 빠르게 통신.

2. 프로세스의 동기화.

- CPU 코어의 수가 증가하면, 성능이 비례하여 증가? No.
  프로세스 간 인터랙션을 하면, 동기화 문제가 발생 → 동기화를 위한 비용 ↑.
  ↳ Consistency가 깨진다.

⇒"동기화 문제 발생"에도 굳이 인터랙션을 하는 이유?

① 자원의 공유.

② 동시성 (Concurrency) : 멀티 프로그래밍과 관련.

- OS가 공유하는 코드 부분에 대해, 아무런 동기화 관련 조치 X이면?
  → Race Condition 문제 발생!

* Race Condition 이란? ⇒ 두 프로세스 간에 어떤 프로세스가 먼저 실행되느냐에 따라 결과값이 달라지게 되는 것. (Atomic하지 않다.)

⇒ Race Condition이 Critical Section에서 발생할 수 있다.

* Critical Section 이란? ⇒ 둘 이상의 프로세스 또는 스레드들이 공유하는 데이터에 접근하는 코드로서, 동시에 접근해서는 안되는 부분을 말함.
  ↳ 예를 들어, 시스템 콜에 의해 실행되는 커널 코드가 있음.

- 이 C-S 문제를 해결할 솔루션은 3가지 조건을 만족해야 올바른 솔루션임.

1. Mutual Exclusion : C-S에 1개의 프로세스만 진입해야 한다.
2. Progress : 둘 이상이 경쟁 시 승부가 나야만 한다. (아무도 진입 못하면 안됨).
3. Bounded Waiting : 경쟁에서 져서 대기 중인 프로세스는 언젠가는 C-S에 진입할 수 있어야한다. (일부 프로세스가 독점하면 안됨).