Operating System: Concurrency and Threads

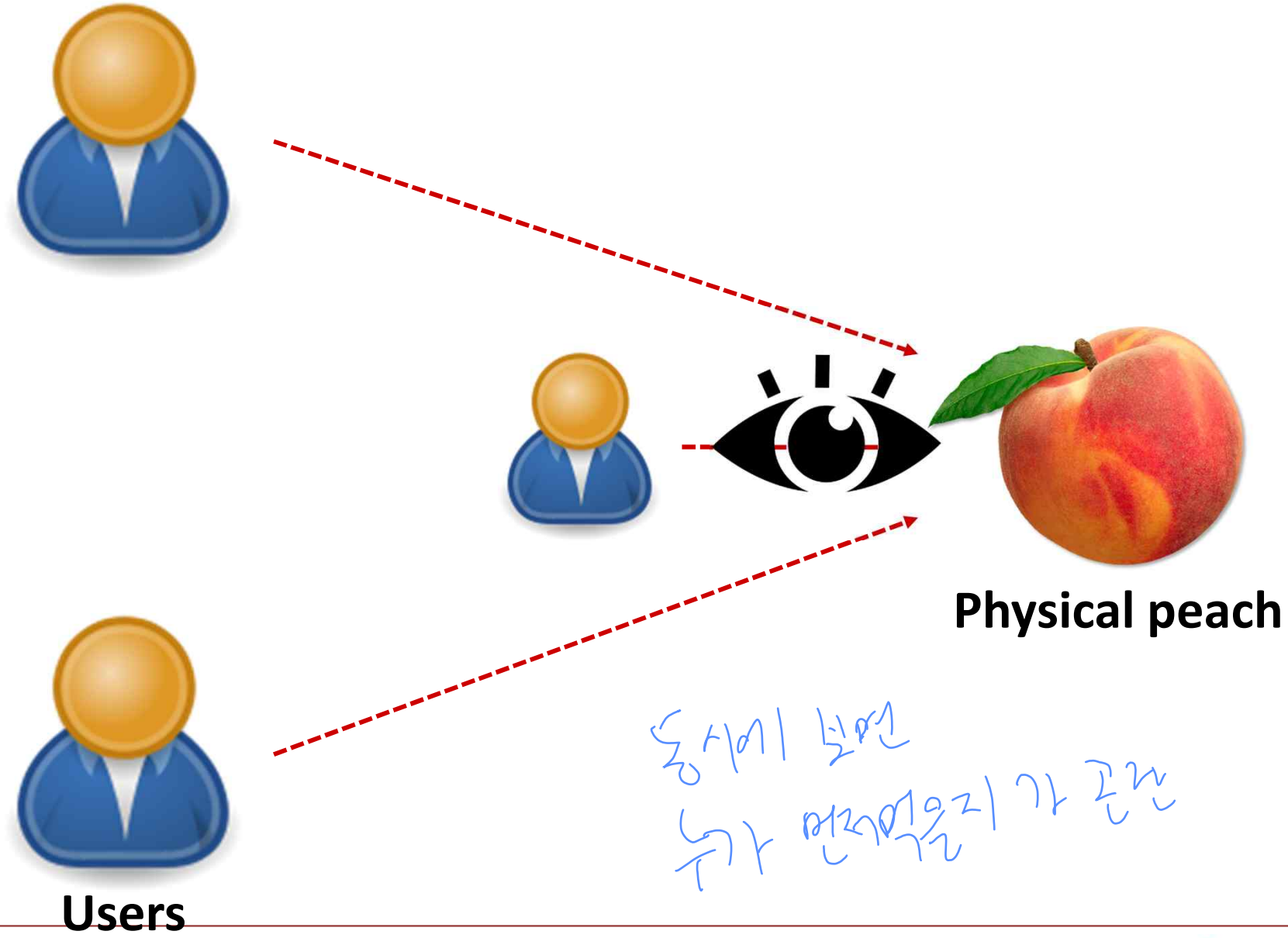
Sang Ho Choi (shchoi@kw.ac.kr)
School of Computer & Information Engineering
KwangWoon University

A Dialogue on Concurrency

- Professor: Anyhow, imagine there are a lot of peaches on a table, and a lot of people who wish to eat them

- Let's say we did it this way: each eater first identifies a peach visually, and then tries to grab it and eat it. What is wrong with this approach?

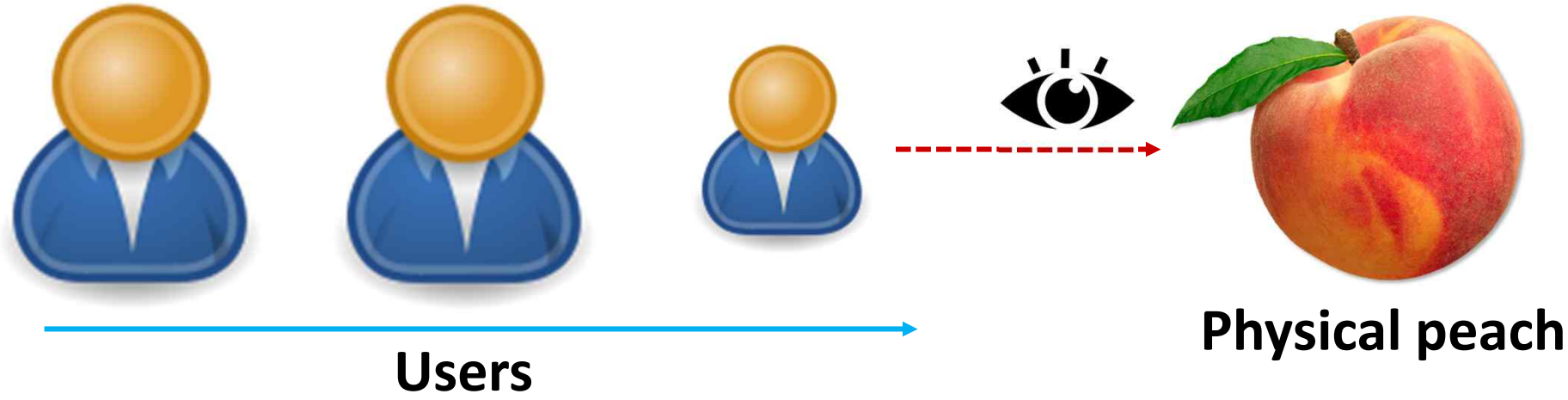
A Dialogue on Concurrency (Cont'd)



동시에 보면
누가 먼저일지 가 궁금

A Dialogue on Concurrency (Cont'd)

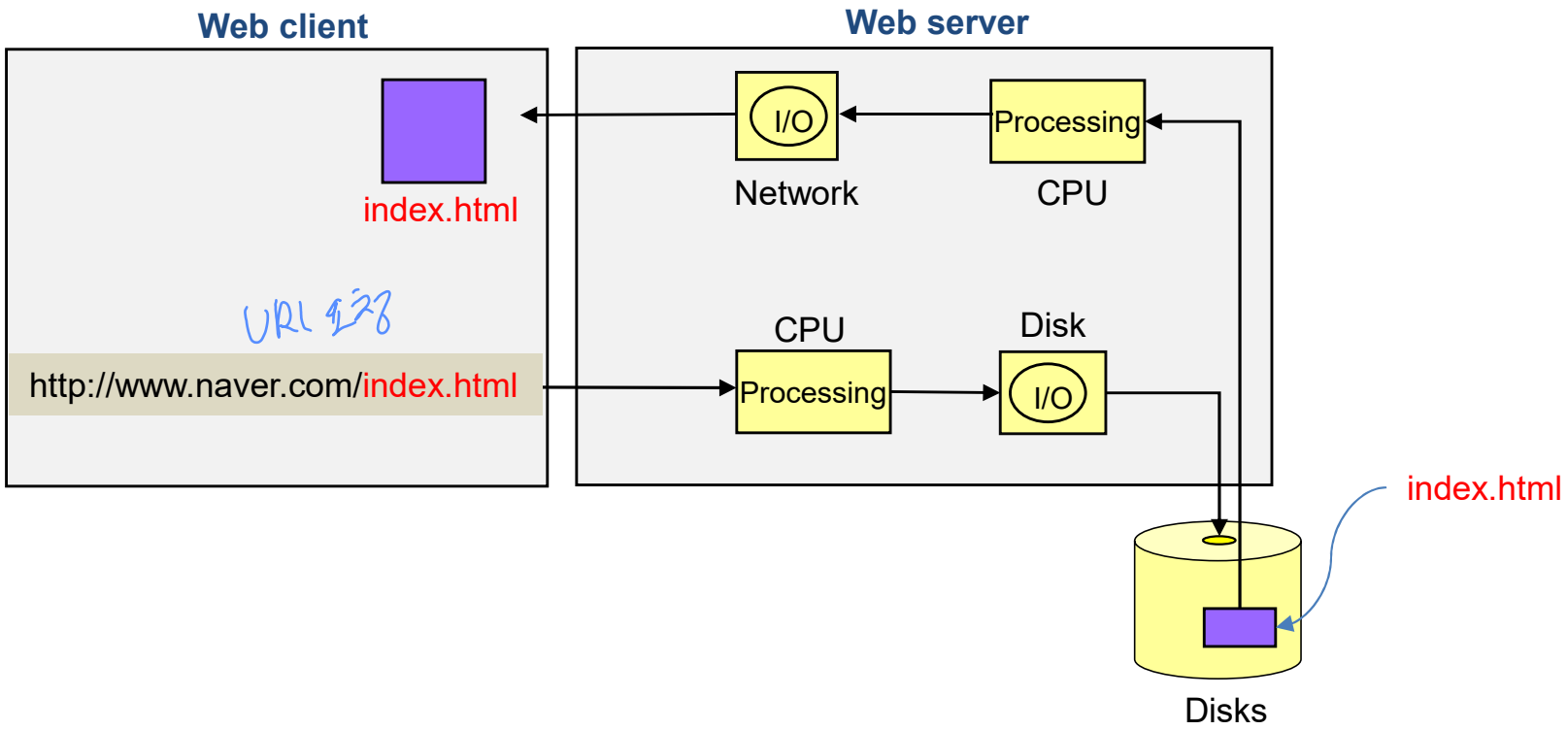
순차적으로 먹을수 있게



Why thread?

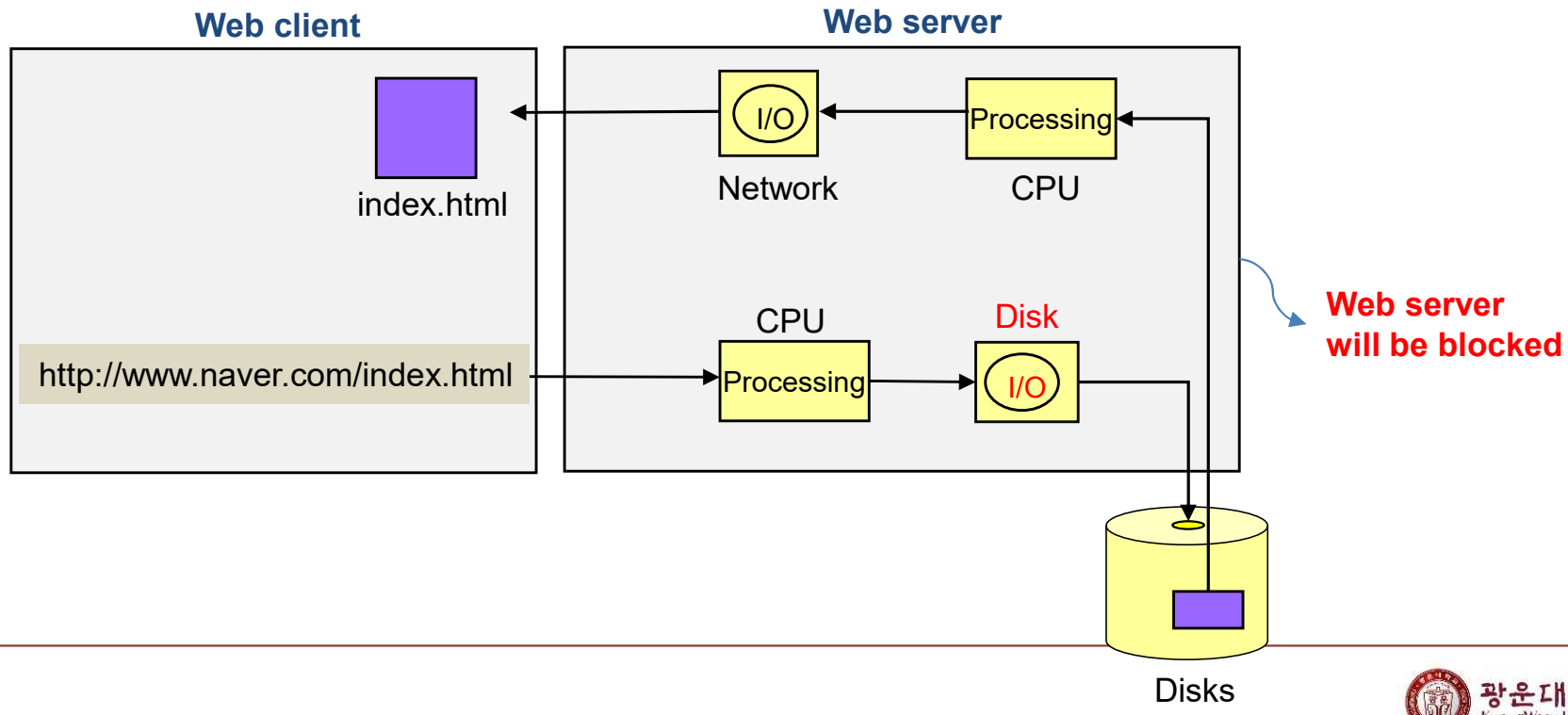
병렬처리가 가능하게 thread를 사용.

- Web server
  - a computer program that delivers web pages



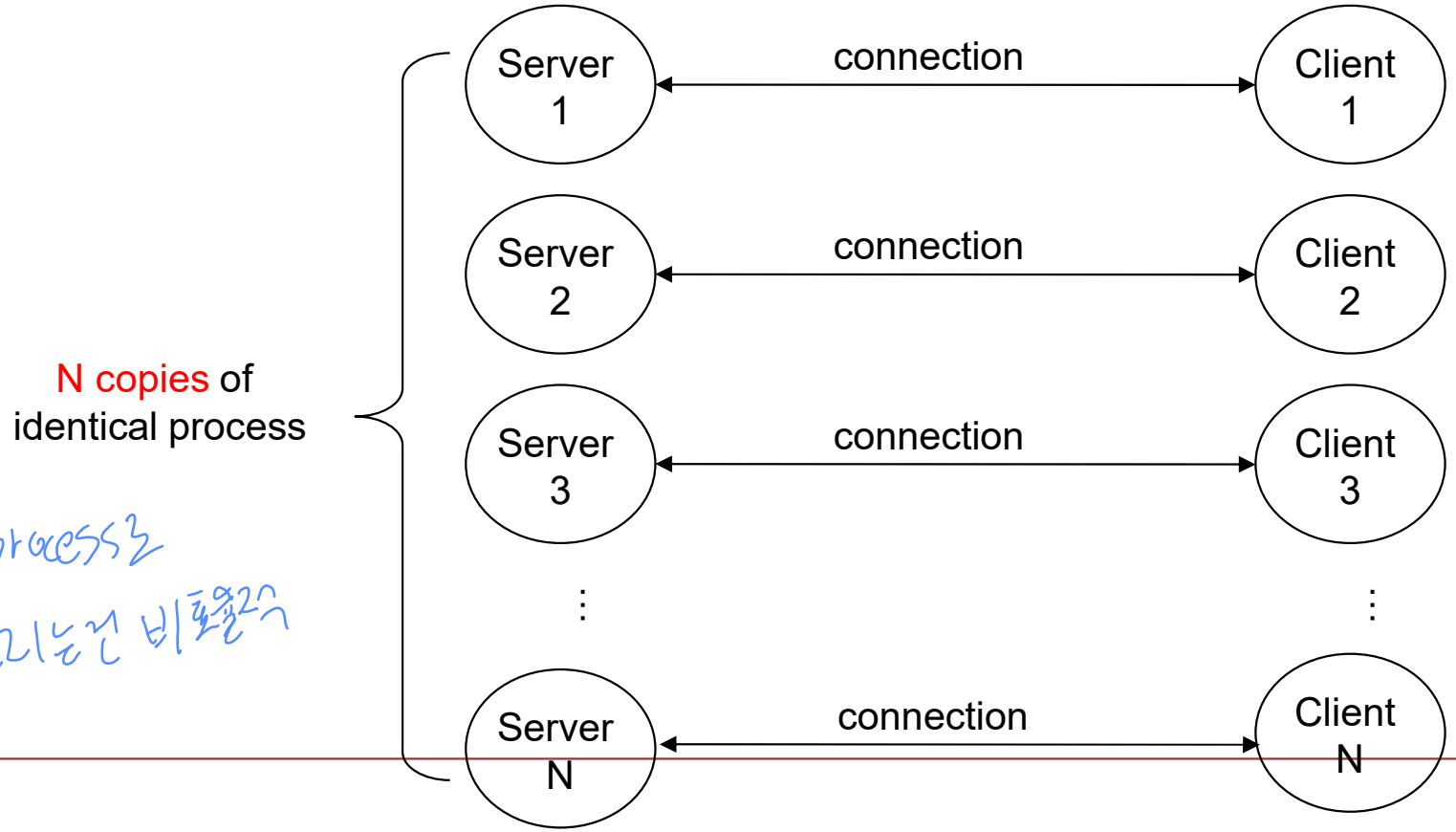
Why thread? (Cont.)

- Web server should service many clients at a time
- If web server performs an I/O operation?
  - It will be blocked
  - It cannot deal with requests from other clients



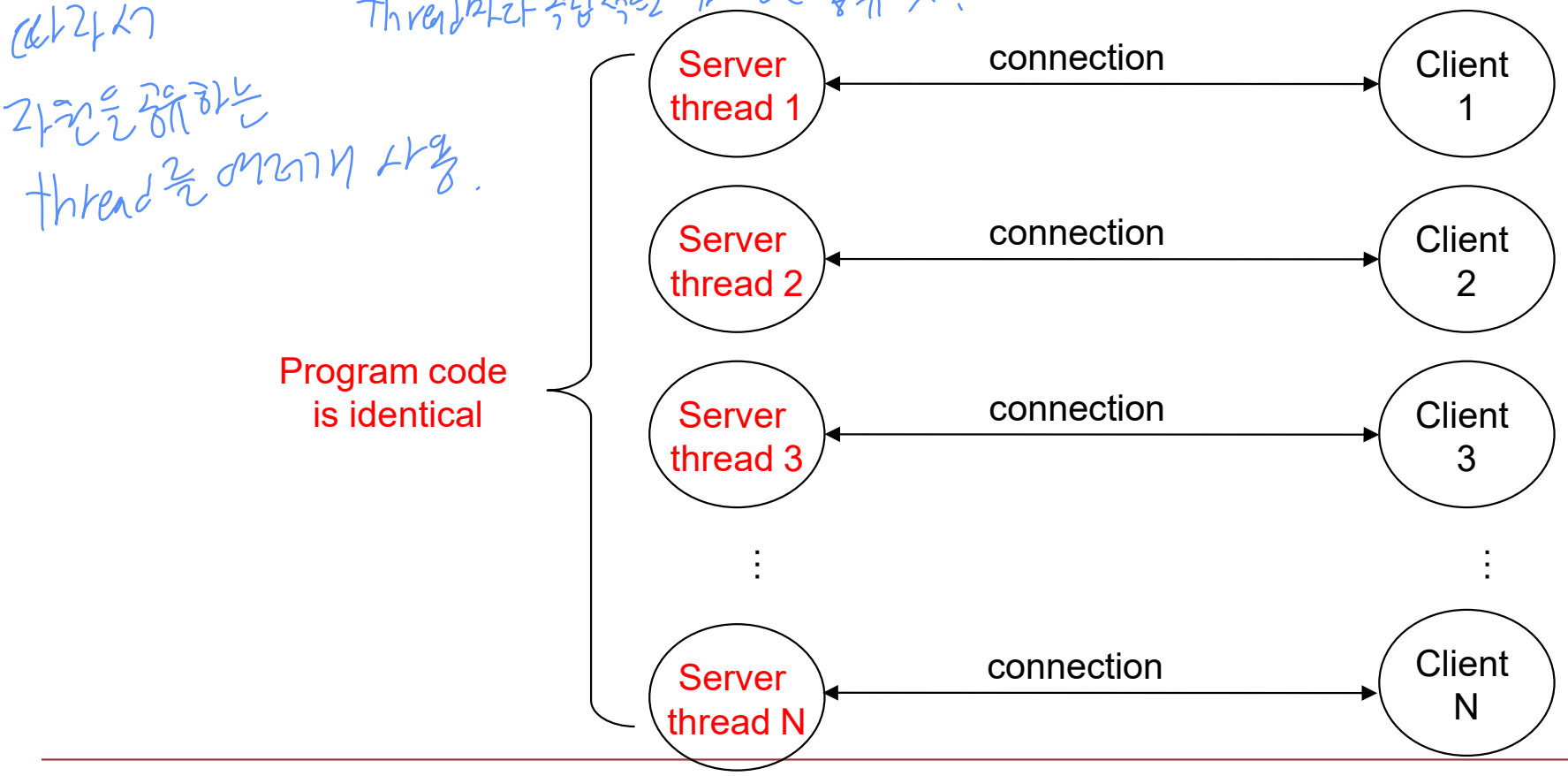
Why thread? (Cont.)

- Many server processes are required for concurrent services
  - Waste memory space
  - Time delay for process creation



Why thread? (Cont.)

- Threads are required
  - Code, data, and resources can be shared
  - Register values and stacks cannot be shared

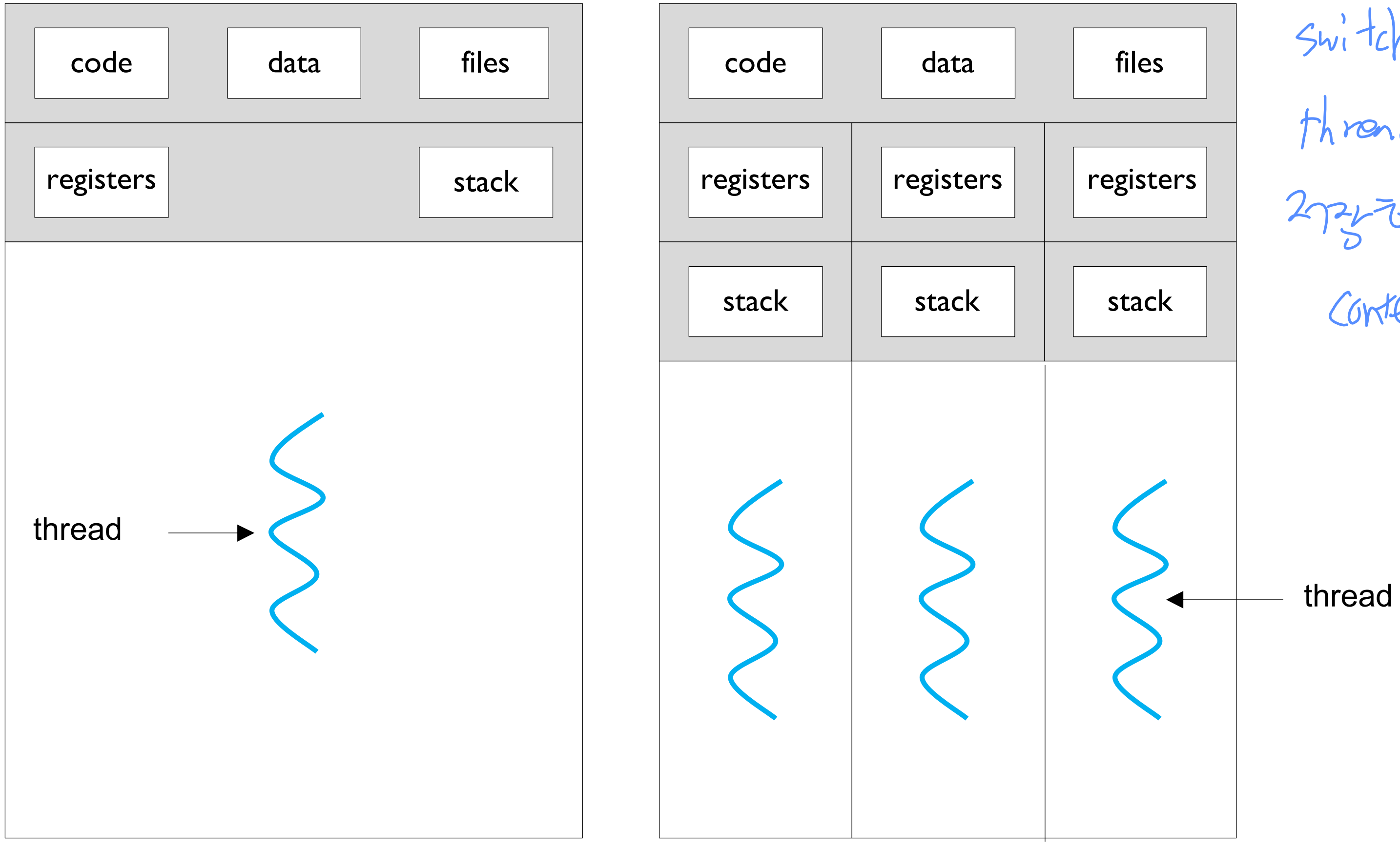


What is thread?

- Thread
  - a basic unit of CPU utilization
  - Each process can include many threads
- All threads of a process share
  - code, data, heap
  - open files
  - signal handlers
  - working environment (current directory, user ID, etc.)
- Each thread has it's own
  - stack
  - registers
  - thread ID

Thread vs. Process

- Single and Multi-threaded processes



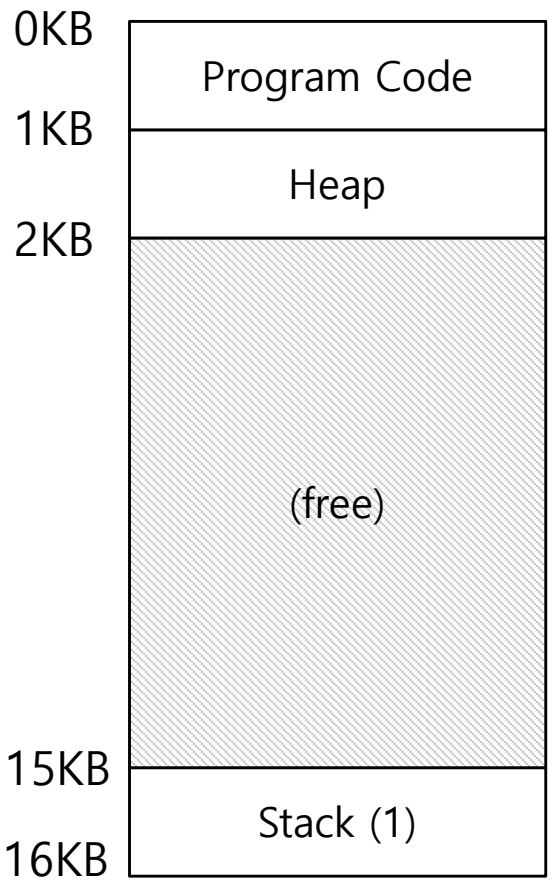
(a) single-threaded process (b) multi-threaded process

Context switching

process 에서는
PCB에 context들 저장해서
switch 했는데
thread 또한 TCB라는곳에
저장하고 불러오는
context switch이 발생.

Thread vs. Process (Cont.)

- There will be one stack per thread

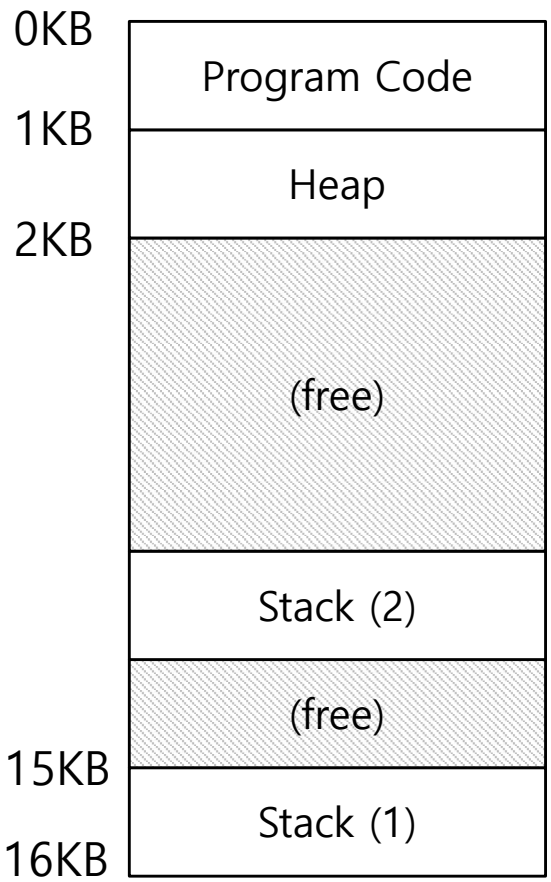


The code segment: where instructions live

The heap segment: contains malloc'd data dynamic data structures (it grows downward)

(it grows upward)
The stack segment: contains local variables arguments to routines, return values, etc.

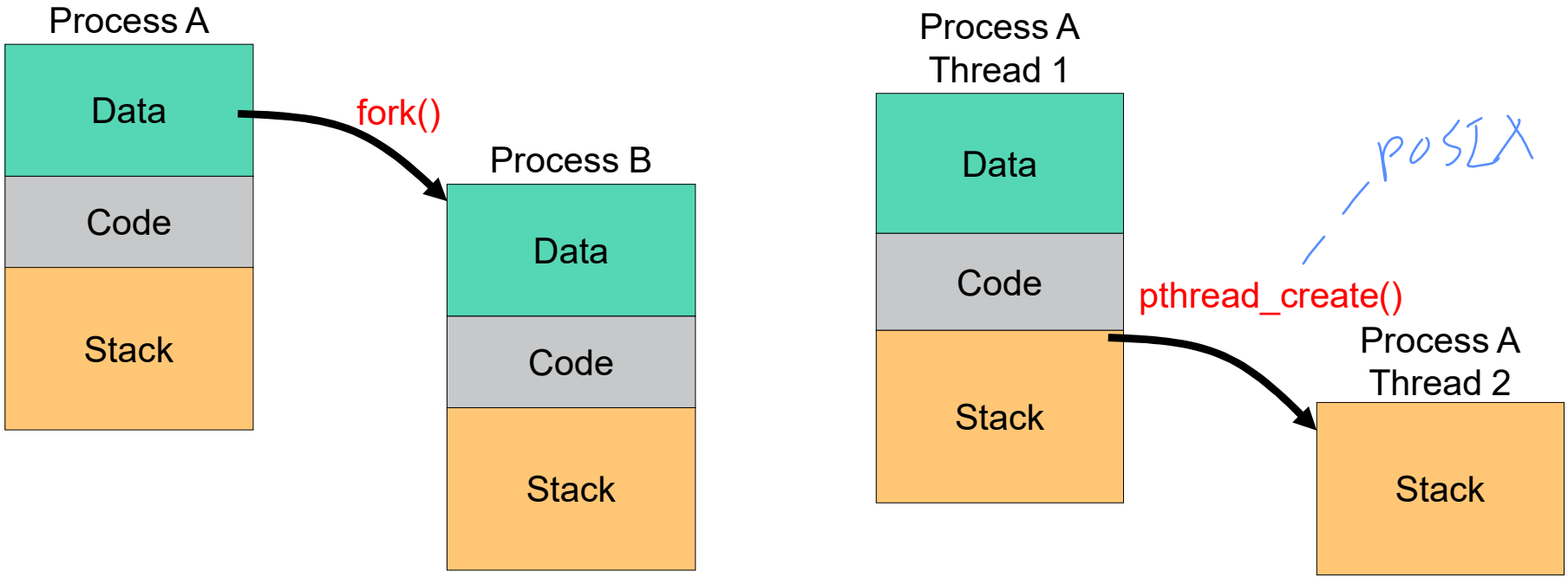
A Single-Threaded Address Space



Two threaded Address Space

Thread vs. Process (Cont.)

- Creation of a new process using fork is expensive
  - time & memory
- A thread does not require lots of memory or startup time
  - sometimes called a lightweight process LWP



(a) Process creation

(b) Thread creation

Thread vs. Process (Cont.)

- Responsiveness
  - A process can continue running even if part of it is blocked
  - E.g. Web server can service for other web client in one thread, while a file is being loaded in another thread
- Resource Sharing
  - Memory and resources can be shared
- Economy
  - The overhead for process creation and context switching is low
- Utilization of MP Architectures
  - Parallelism in multiprocessor architecture is increased

User threads vs. Kernel thread

User mode와 Kernel mode를 나눈 객체로 유사.

- User thread
  - Is supported by user-level threads library
    - POSIX Pthreads
    - Win32 threads
    - Java threads

- Kernel thread
  - Is supported and managed directly by the operating system
    - Windows
    - Solaris
    - Linux
    - Mac OS

Linux threads

- Linux threads
  - clone() system call is used for create threads
  - Linux does not distinguish between processes and threads
  - Flag set of clone determines how much sharing between parent and child
    - Ex 1)

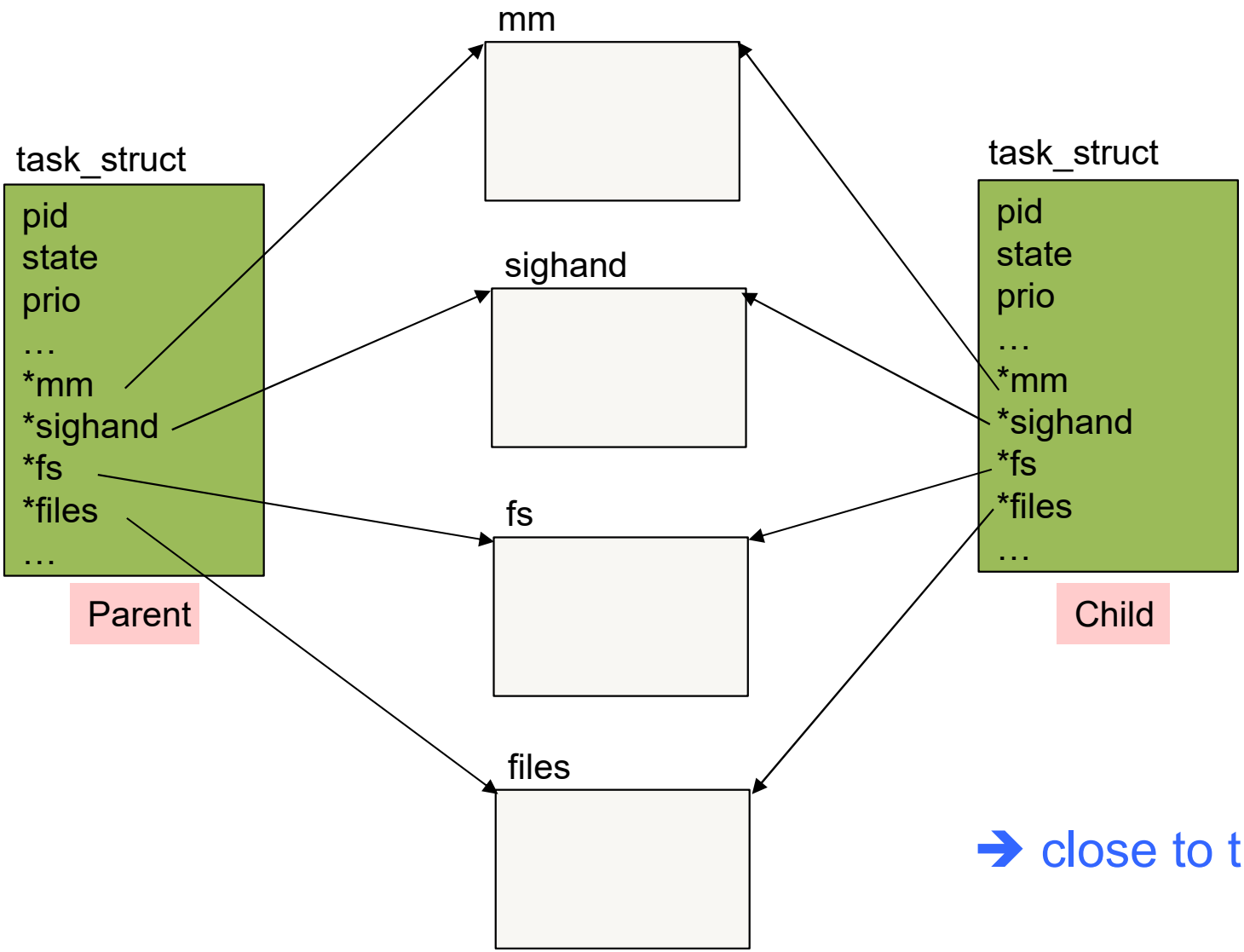
      clone(CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND, 0)

      ➔ close to thread
    - Ex 2) clone(CLONE_SIGHAND, 0)

      ➔ close to process

Linux threads (Cont.)

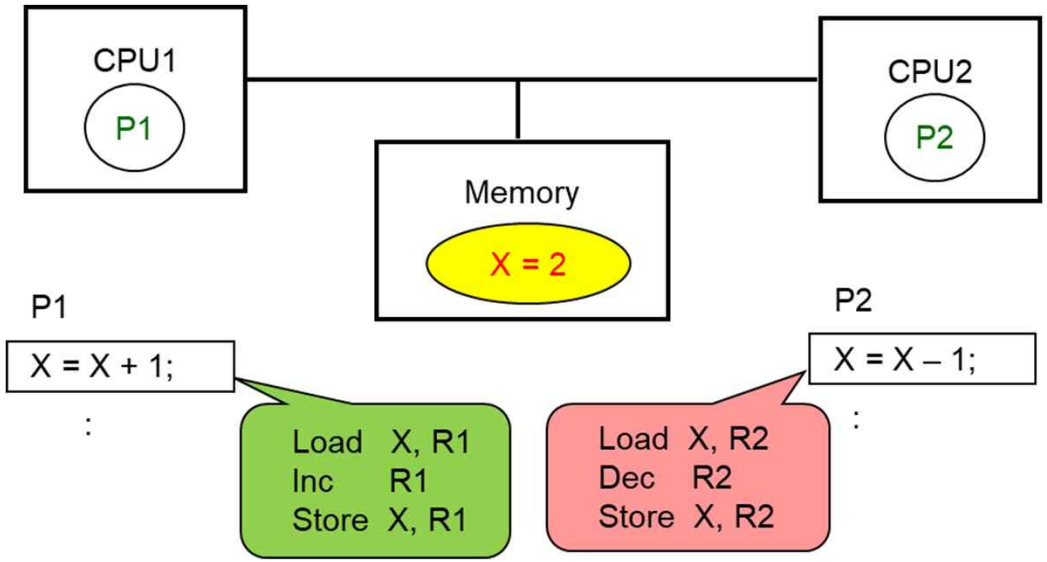
- clone(CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND, 0)



➔ close to thread

Race Condition

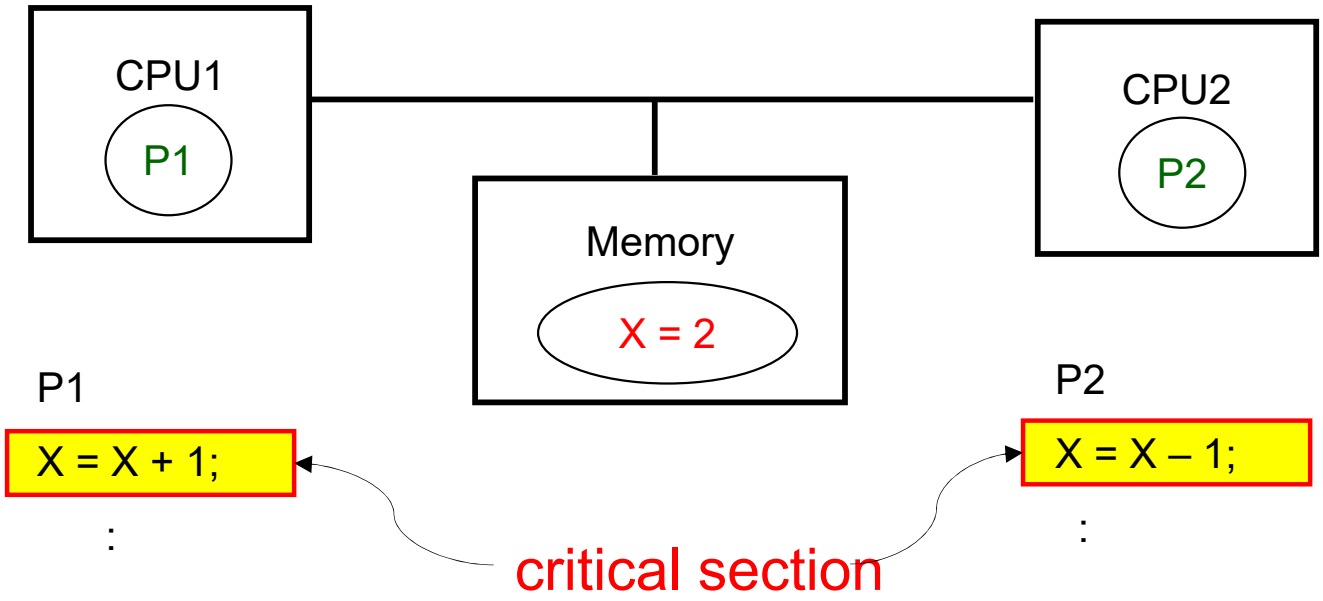
- Concurrent access to shared data may result in data inconsistency



- Race condition
  - Situation where several processes access & manipulate shared data concurrently
  - To prevent race conditions, concurrent processes must be synchronized
  - In uniprocessor environment, race condition can also occur by CPU scheduler

Critical Section Problem

- Critical section
  - A code segment in which the shared data is accessed



- Critical section problem
  - ensure that when one process is executing in its critical section, no other processes are allowed to execute in its critical section

Critical Section Problem (Cont.)

- It is required to design a protocol that the processes can use to cooperate
- General structure of a typical process Pi

general structure

```
do {
        entry section
        critical section
        exit section
        remainder section
} while (TRUE);
```

example

```
1   lock_t mutex;
2   . . .
3   lock(&mutex);
4   x = x + 1;
5   unlock(&mutex);
6   …
```

  - Entry section
    - code segment of requesting permission to enter the critical section
  - Exit section
    - code segment of notifying the exit of the critical section