[Pre-lecture Assignment#5]

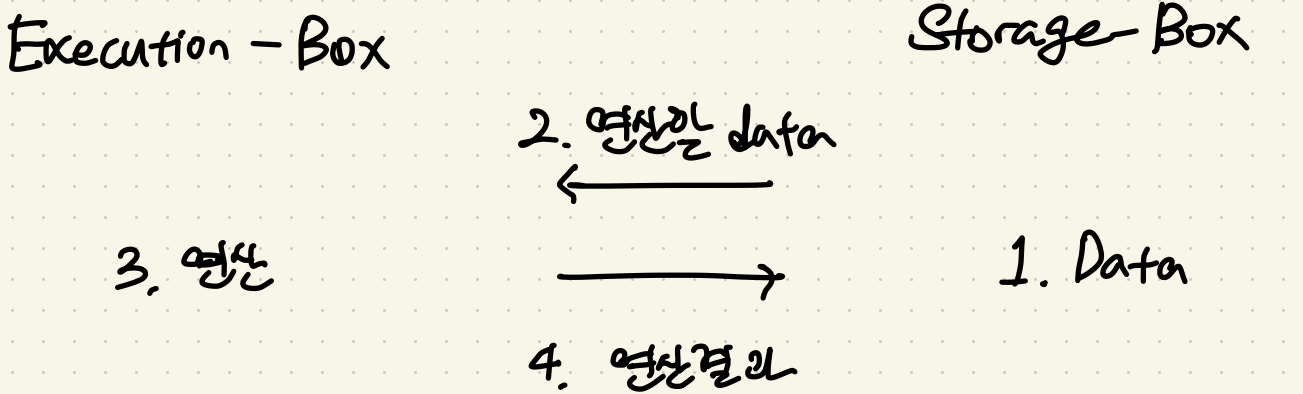
- Reading 1: Read Sections 3.5 IPC in Shared-Memory Systems (page 125-127), 6.1 Background (page 257-260). Note that reading assignments may appear in your homework and quiz.

- **Video Lecture 1** (00:00~33:00, around 33 minutes): Synchronization; Race condition

https://core.ewha.ac.kr/publicview/C0101020170403151644920369

- **No** submissions. However, pre-lecture assignments, as well as slides, class discussions, programming assignments, and textbook contents covering last week's lecture may appear in your next quiz.

데이터의 접근

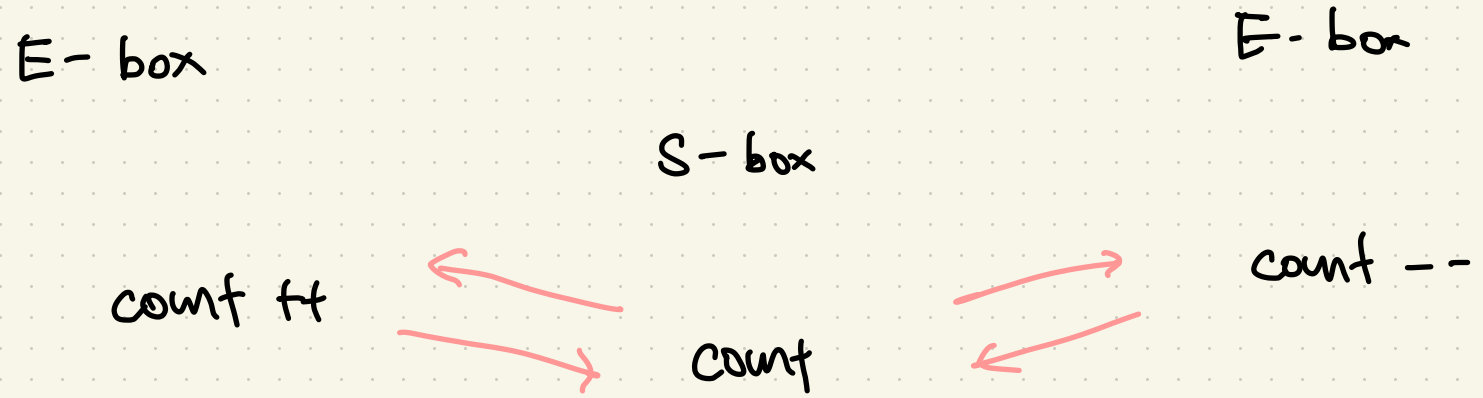


CPU, 컴퓨터내부, 프로세스

Memory, 디스크,
프로세스의 주소공간

이렇게 box가 나뉘어져
있음..

Race Condition



S-box를 공유하는 E-box가 여럿 있는 경우 Race Condition의 가능성이 있음

S-box → Memory Address space

E-box → CPU process

어떻게 해결하지? ⟶

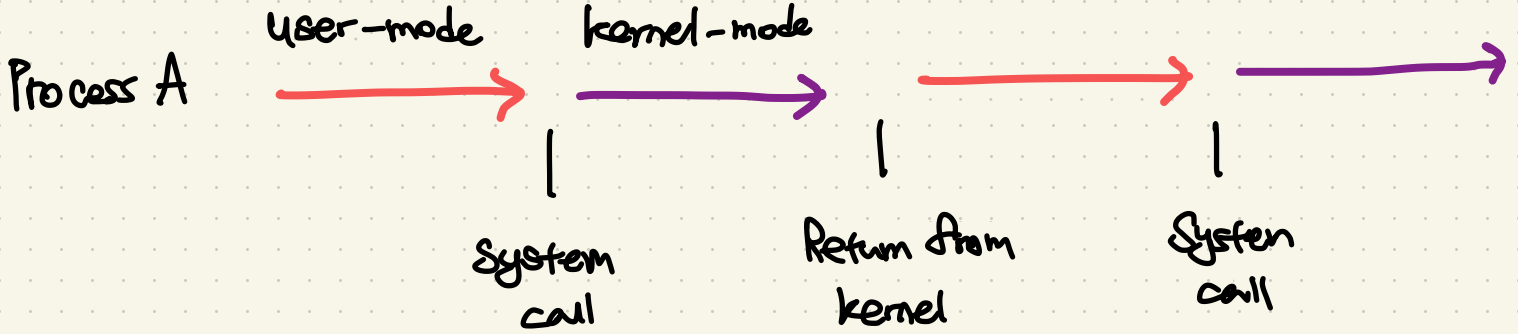
OS에서 race condition은 언제 발생하는가?

1. kernel 수행 중 인터럽트 발생시

2. process가 system call을 하여 kernel mode로 수행 중인데 context switch가 일어나는 경우

3. Multiprocessor에서 shared memory 내의 kernel data

race condition 발생 예시

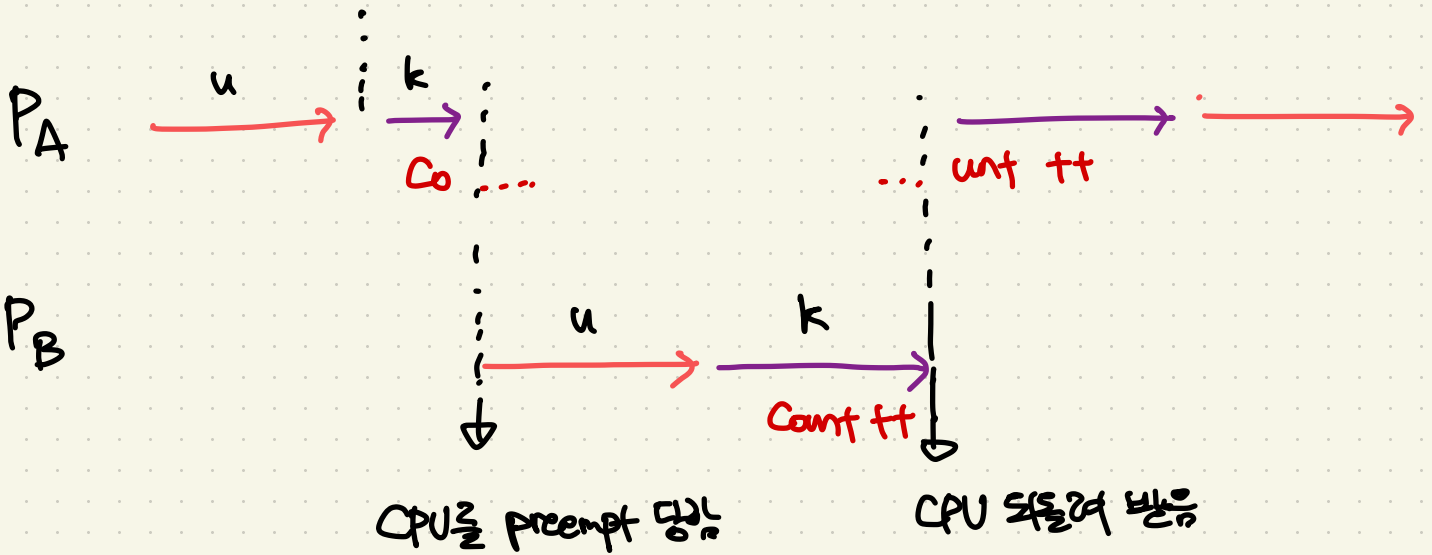


각 process의 address space 간에는 data sharing이 없음

그래서 system call을 하는 동안 kernel address space의 data를 access 하게 됨

이 작업 중간에 CPU를 preempt 해버린 race-C 발생

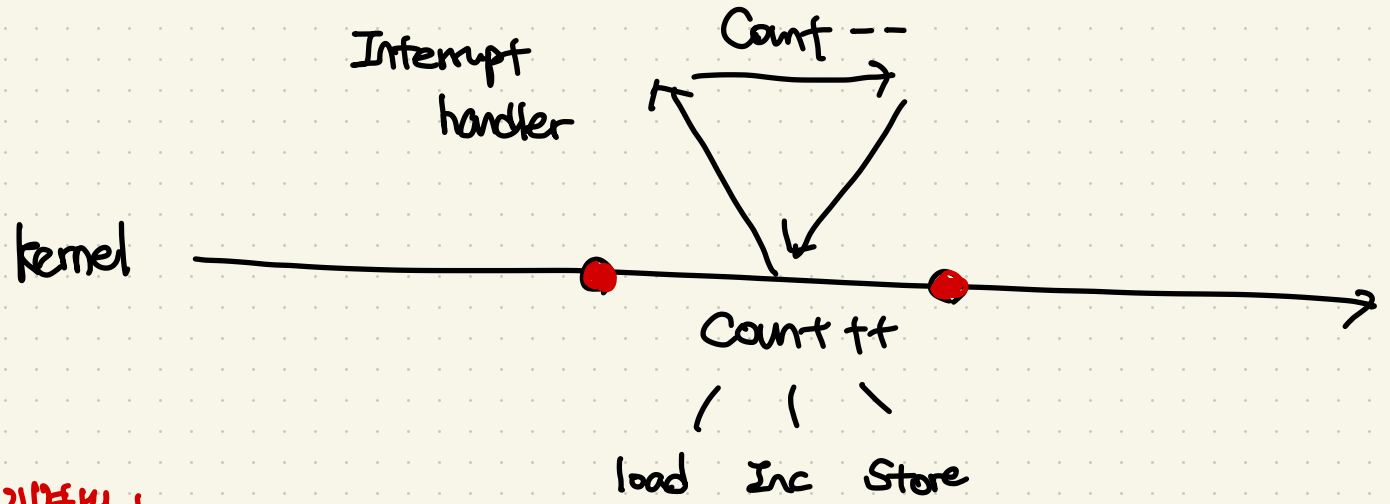
System call read()



→ 왜일까? : kernel mode에서 수행 중이면 CPU를 preempt 하지 않음!

k → u 일때 preempt

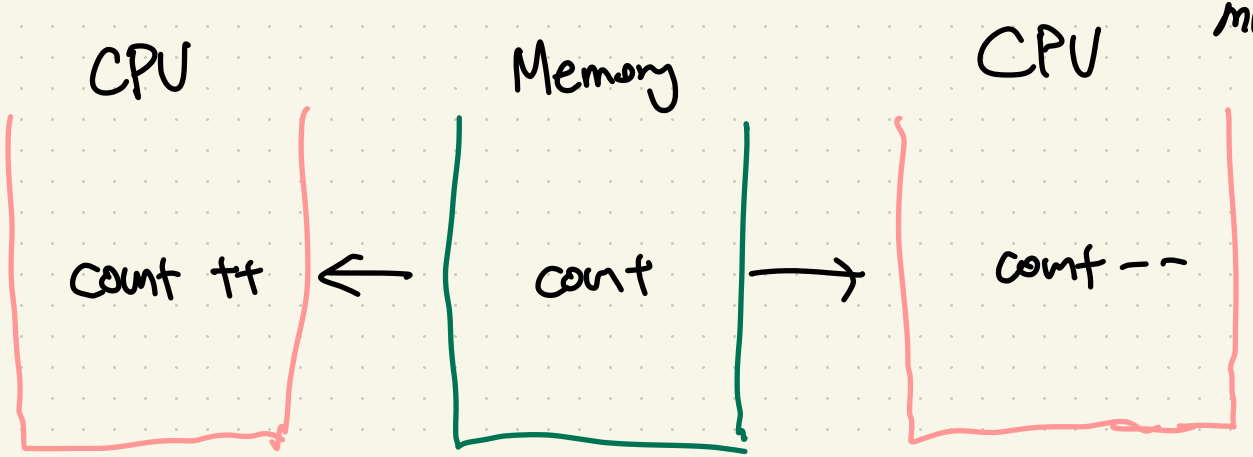
interrupt handler v.s. kernel



해결법 :

- → disable/enable interrupt.

multiprocessor



어떤 CPU가 마지막으로 count를 store 했는가 ...? → race-c

이거는 interrupt enable/disable로 해결 불가능

해결! 1) 한번에 하나의 CPU 만이 커널에 들어갈 수 있게 하는 방법

2) 커널 내부의 공유 데이터에 접근할때 마다 데이터에 대한 lock/unlock 을 하는 방법

모든 race condition은 결국 공유데이터의 접근 과정에서 발생한다.

따라서, 공유 데이터를 접근하는 코드인 Critical section 이 존재하는데,

하나의 process가 critical section에 있을때 다른 모든 process는

critical section에 들어갈 수 없도록 한다.