멀티스레드와 동기화

김현수

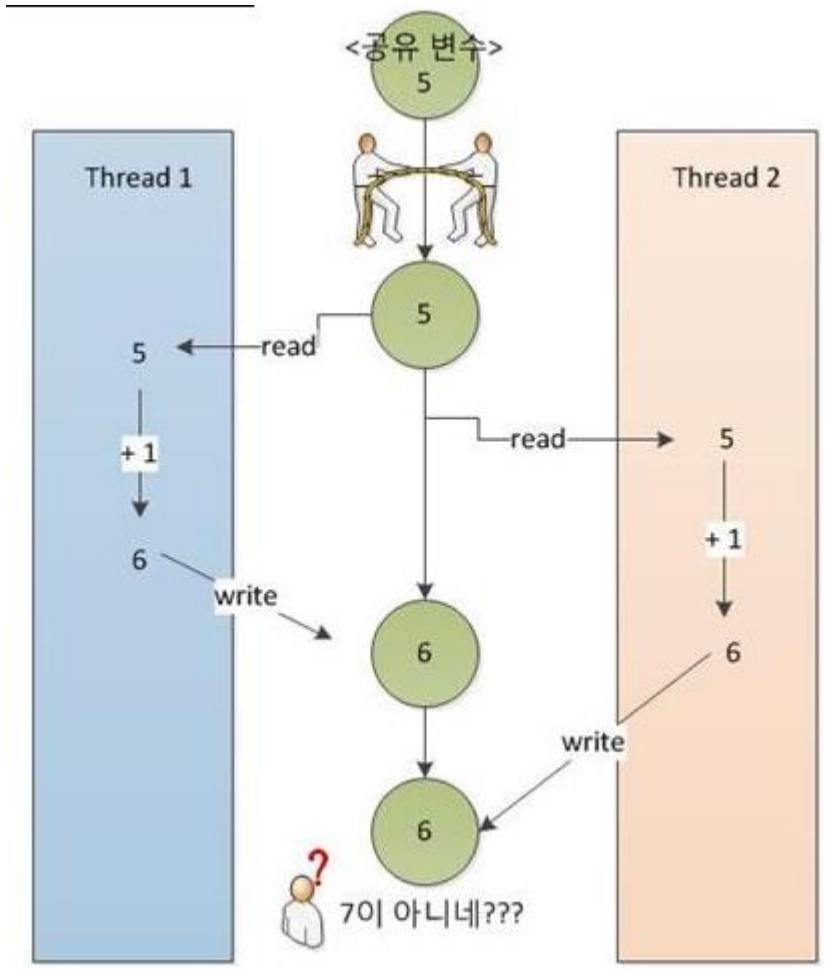
멀티스레드의 장점

- 스레드 : 스택을 제외한 메모리를 공유!
- 멀티 스레드 : 하나의 프로세스, 여러 개의 스레드
  - Context switching할 때 공유하고 있는 메모리만큼의 메모리 자원을 아낄 수 있다.
  - 메모리를 공유하기때문에 응답 시간이 빠르다.

멀티스레드의 문제점

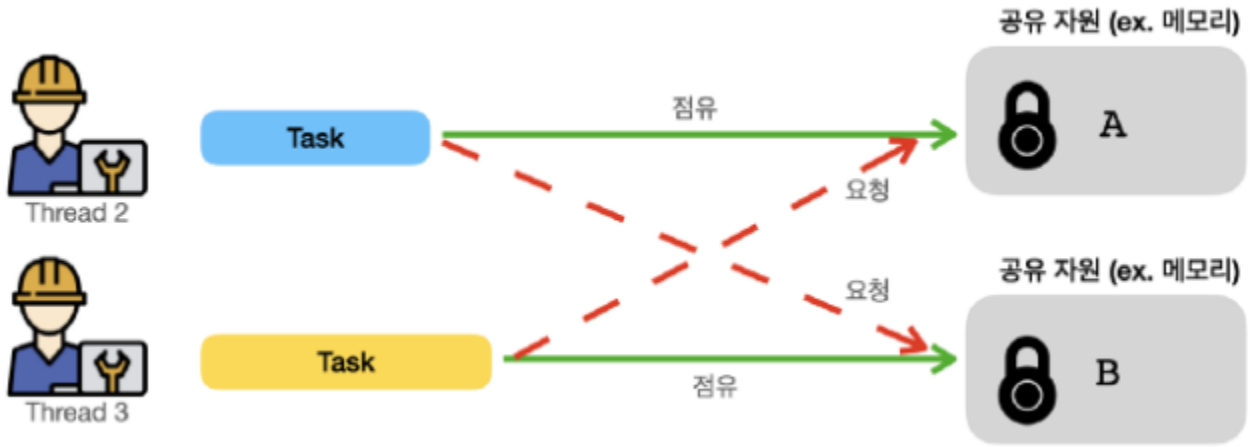
- Race Condition 경쟁 상태
- 서로 다른 쓰레드가 공유자원에 동시에 접근하는 경우
- 이미 사용 중인 공유자원에 접근할 경우

- 잘못된 값을 읽어오거나
- 잘못된 값으로 수정할 수 있다.
- -> 임계염역



멀티스레드의 문제점2

- Deadlock 교착 상태
- 각자가 서로 필요한 자원을 가지고 있어, 자원을 점유하면서 요청만 하는 상태

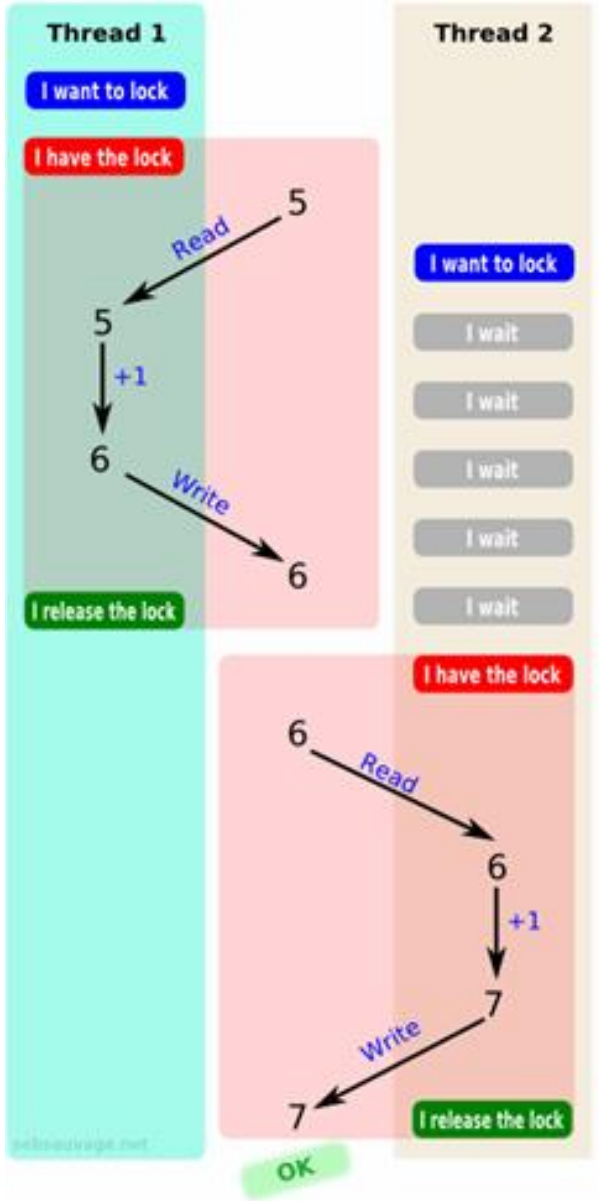


동기화

- 국어사전 : 작업들 사이의 수행 시기를 맞추는 것.
- 동기화 : 하나의 자원에 대한 **처리 권한**을 주거나 **순서를 조정**해주는 기법

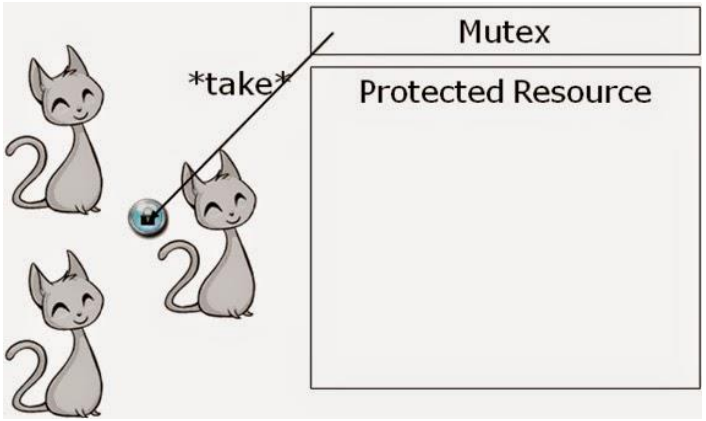
- 하지만! 병목현상으로 성능 저하 유발

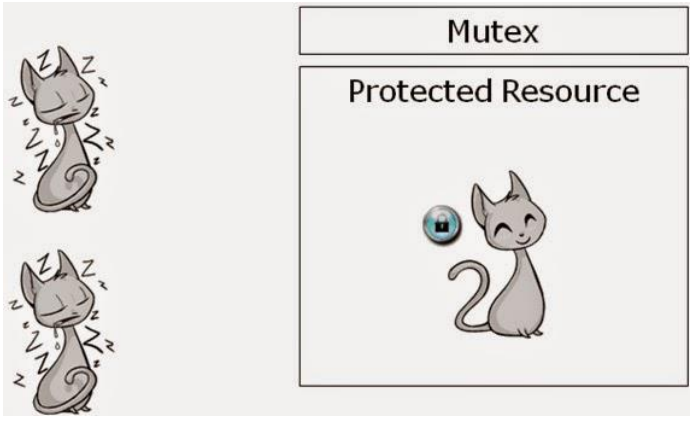
- 동기화가 필요한 부분에만 적용!



뮤텍스

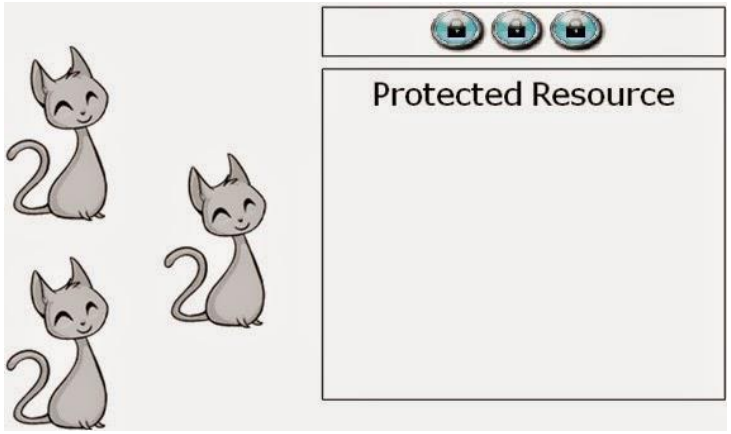
- Mutual Exclusion : 하나의 스레드에 처리 권한을 주는 것
- 임계 영역에 하나의 스레드만 접근 가능.
- 스레드가 나갈 때까지 다른 스레드는 접근 불가 : 상호 배제

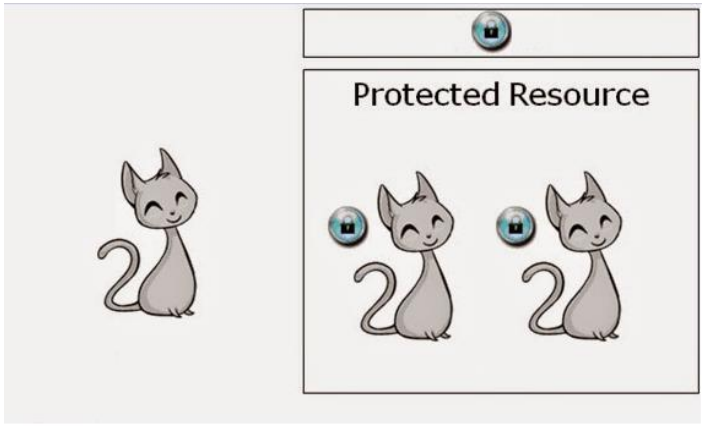


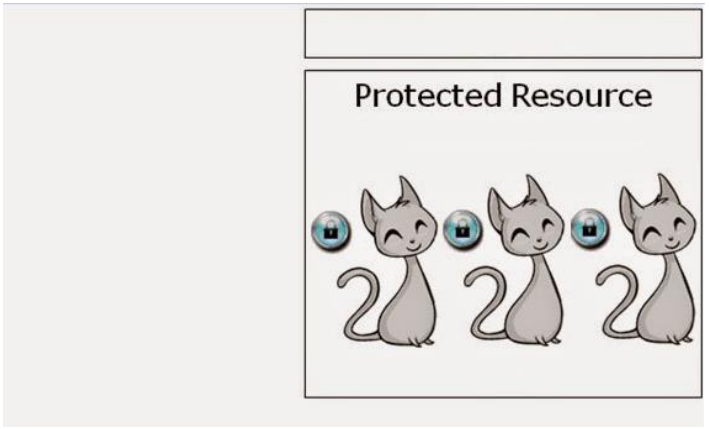


세마포어

- Semaphore는 뮤텍스와 비슷하지만 더 큰 개념
- 임계영역에서 특정 개수만큼 스레드가 접근 가능한 경우
- N개의 뮤텍스 -> 세마포어







뮤텍스 vs 세마포어

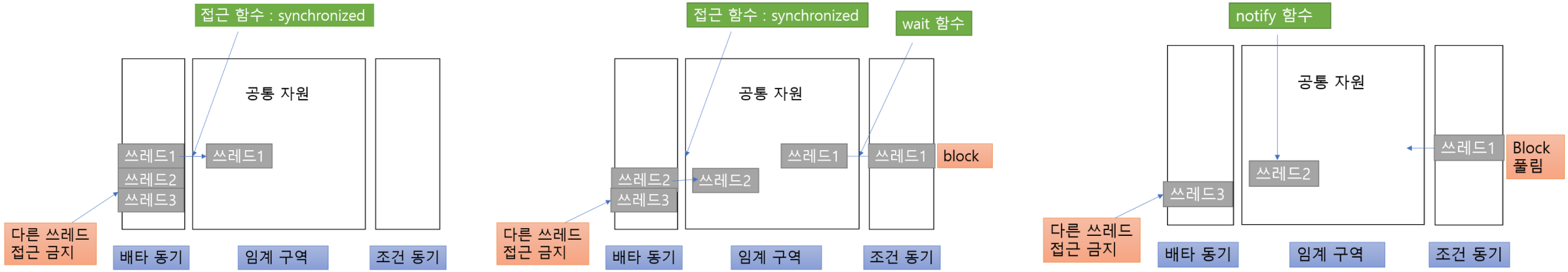
- 세마포어는 뮤텍스가 될 수 있다.
- 뮤텍스는 세마포어가 될 수 없다.

- 세마포어는 소유자가 없다.
- 뮤텍스는 소유자가 있다.

- 세마포어는 1개 이상 동기화
- 뮤텍스는 1개만 동기화

모니터

- 현대에 사용되는 동기화 도구 중 하나
- 뮤텍스와 마찬가지로 임계영역에 하나의 스레드만! : 상호배제
- Lock + Synchronized(Queue)



뮤텍스 vs 모니터

- 뮤텍스는 운영체제 커널, 프레임워크, 라이브러리에 의해서 제공
- 모니터는 프레임워크나 라이브러리 그 자체에서 제공

- 뮤텍스는 무겁고 느리다.
- 모니터는 가볍고 빠르다.

감사합니다!