멀티 스레드

* 옳바른 결과 => mutex
* 성능 향상

재작성 => mutex를 줄이자.. == Data Race를 줄이자.(Data Race를 잘 관리)

mutex없이 data race를 관리하자.

* 병렬 프로그램 작성시 주의할 점
* 컴파일러 문제 (최적화)
* CPU 문제 (최적화)

컴파일러의 최적화는 하드웨어의 전기사용량과 발열을 줄여준다.

이런 최적화 문제는 volatile을 사용하면 된다.

-volatile은 직접 메모리에 접근

-volatile을 했는데 잘 작동되지 않았다? 🡺 volatile을 잘못 쓴 것

volatile int \* a; => a가 가리키는 변수가 volatile

Int \* volatile a; => 변수의 주소를 담은 a가 volatile

위 둘은 서로 다르다!

Mutex(상호배제)를 사용했다면 volatile을 쓸 필요가 전혀 없다 => Mutex를 만남과 동시에 (Lock) 읽음

= 상호배제

피터슨 알고리즘, 빵집, 데커 등…

피터슨 알고리즘 🡺 어? 썼는데? 왜 또 안되고 더 느려지지? 🡺 CPU 문제(최적화)

피터슨 알고리즘의 문제

* 성능이 안나온다..(읽고 쓰는 공유 메모리가 많아서), 캐시 invalidate
* 실제 컴퓨터에서 오동작을 일으킴

해결책?

빵집 알고리즘 => 번호표.. (오버헤드가 크다?)

빵집 알고리즘 한번 써보자.. => 숙제 9월 24일까지 화요일 오전 10시까지

No<mutex>, With<mutex> 차이

정리

\* 멀티쓰레드에서 프로그램의 이상 동작

-C언어 자체의 문제

-컴파일러 문제가 아님(컴파일러는 잘못이 없다..)

-volatile로 해결

\* 근데 volatile을 사용함에도 문제가 있음

-어셈블리로 확인해 봐도 문제 없음

\* Lock을 사용할 경우의 성능 저하

-

Out Of Execution (컴퓨터 구조)

순서가 뒤죽박죽 바뀐다. => 오류가 생긴다.