**운영체제 과제**

**유혁 교수님**

**2017320122 김정규**

**Writer의 starvation이란**

멀티스레드 프로그래밍에서 공유 자원에 접근할 때는 동시에 두 개 이상의 스레드가 자원을 변경시키지 않기 위해서 mutex를 사용한다. Mutex를 사용하면 공유 자원에 접근하는 스레드를 한 개로 제한하기 때문에 안전하지만, 어떤 경우는 비효율적이다. 예를 들어 여러 스레드가 공유 자원에 동시에 접근해야 하지만 그중 일부 스레드만 값을 변경하는 경우가 있다.

이런 경우 값을 읽기만 하는 스레드는 동시에 접근해도 상관없다. 하지만 어떤 스레드가 값을 변경하고 있으면, 다른 스레드는 공유 자원에 접근해서는 안 된다. 반대로 다른 스레드가 공유 자원을 읽는 중에는 값을 변경하는 스레드가 접근해서는 안 된다.

이때 사용하는 것이 readers-writer lock이다. Readers-writer lock은 여러 개의 reader와 한 개의 writer를 허용한다. 그래서 multiple-readers/single-writer lock(MRSW lock)이라고도 불린다. 즉, 이미 read lock이 잡혀있는 readers-writer lock에 read lock을 잡으면 바로 lock이 걸리고 다음 코드를 실행할 수 있지만, write lock을 잡으면, lock을 잡지 못하고 read lock이 풀릴 때까지 기다린다.

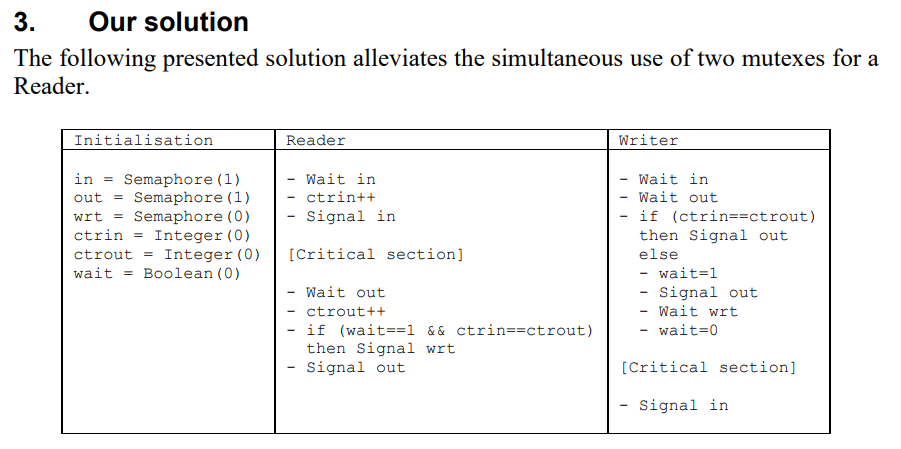
그렇다면 A thread가 read lock을 잡고, B thread가 write lock을 잡은 뒤 C thread가 read lock을 잡으면 어떻게 될까? 단순하게 생각해보면 A thread가 read lock을 잡고 있으니 B thread의 write lock은 잡히지 못하고 기다리고, C thread의 read lock은 잡힐 수 있기 때문에, A thread와 C thread의 코드가 실행될 것이다. 하지만 이런 구현의 경우 read lock이 지속적으로 잡히는 코드라면, write lock이 영원히 실행되지 못할 수도 있다. 이 현상을 write-starvation이라고 부른다.

**Solution**

readers-write lock의 구현하는 데에 있어서 이미 read lock이 잡혀서 read lock을 잡을 수 있는 상황에서도 write lock이 기다리고 있으면 새 read lock은 잡지 않고, write lock이 잡히는 것을 기다리는 방식으로 구현하는 솔루션이 있다. 이 경우 불필요하게 read lock을 못 잡고 기다리는 경우도 있기 때문에 효율성은 떨어질 수 있지만 먼저 기다리기 시작한 lock이 먼저 잡히기 때문에 더 공정할 것이다.

**Other solution**

다음 솔루션은 Faster Fair Solution for the Reader-Writer Problem - Vlad Popov and Oleg Mazonka (2013)에서 설명하는 솔루션이다.



이 솔루션의 주요 아이디어는 writer가 직접 reader들에게 작업 영역에 접근할 자신이 접근해야 함을 알리는 것이다. 이와 동시에 어떤 새로운 reader도 접근하여 읽을 수 없다. 영역을 나가는 모든 reader는 작성자가 대기하고 있는지 확인하고 마지막으로 떠나는 reader는 writer에게 지금 접근하여 자원을 변경해도 안전하다는 신호를 보낸다. 작업 영역에 대한 액세스를 완료한 후 writer는 대기 중인 reader에게 영역에 다시 접근할 수 있도록 허용했음을 알린다.

이 솔루션은 위의 솔루션의 writer가 접근해오면 다른 reader의 접근을 막고 writer에게 양보한다는 점에서 동일하다. 그러나 이 솔루션에서는 단순히 언제나 특정 접근을 막기만 하는 것이 아니라 reader와 writer의 상호작용적인 측면을 강화하여 일의 효율을 더 향상시킨 것으로 볼 수 있다.