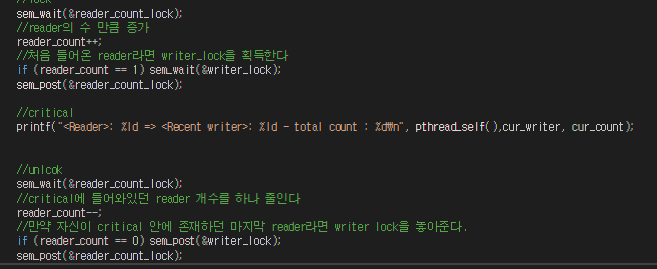
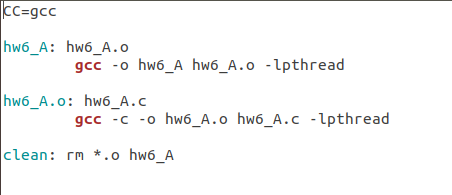
**2019062833 컴퓨터소프트웨어학부 김유진**

**운영 체제 HW#6**

**제출 일자 : 2021/05/04**

1. **과제 A**
2. **자료구조 설명**
   1. Semaphore
      1. 운영체제의 리소스를 경쟁적으로 사용하는 다중 프로세스에서 행동을 조정하거나 또는 동기화 시키는 기술
      2. 세마포어는 운영체제 또는 커널의 한 지정된 저장장치 내 값으로서, 각 프로세스는 이를 확인하고 변경 가능
      3. 확인되는 세마포어의 값에 따라, 그 프로세스가 즉시 자원을 사용할 수 있거나, 또는 이미 다른 프로세스에 의해 사용 중이라는 사실을 알게 되면 재시도하기 전에 일정 시간을 기다려야만 함
      4. 세모포어와 뮤텍스의 차이
         1. 세마포어(Semaphore) :
            * 공유된 자원의 데이터를 여러 프로세스가 접근하는 것을 막는 것
            * 시스템 범위에 걸쳐있고 파일 시스템 상의 파일 형태로 존재
         2. 뮤텍스(Mutex) :
            * 공유된 자원의 데이터를 여러 쓰레드가 접근하는 것을 막는 것
            * 프로세스 범위를 가지며 프로세스가 종료될 때 자동으로 Clean up
3. **함수 설명**
   1. 메인
      1. sem\_init 함수로 세마포어 초기화를 진행한 후 2개의 reader와 5개의 writer 스레드를 생성한다
      2. 각 스레드에게 역할에 맞는 함수를 실행하도록 하고 join 함수로 스레드가 종료될 때까지 기다린다.
      3. 모든 스레드가 종료되면 세마포어를 파괴한다
   2. Writer
      1. 정해진 횟수만큼 critical section에 진입한다.
      2. critical seciton에서 공유변수인 cur\_writer와 cur\_count 값을 변경한다
      3. critical seciton에 진입하기 전에는 sem\_wait 함수를 호출하고 이 후에는 sem\_post 함수를 호출한다.
   3. Reader
      1. 정해진 횟수만큼 critical section에 진입한다.
      2. critical seciton에서 공유변수인 cur\_count와 cur\_writer 값을 읽어서 출력한다.
      3. critical seciton에 진입하기 전에 locking 작업을 해준다
         1. reader 우선인 first reader-writer 방식을 사용하므로 위와 같은 코드를 적어준다
4. **프로그램 구조 설명**
   1. 메인 함수에서 스레드를 생성하고 각 스레드가 주어진 함수를 실행하게 된다.
   2. writer 스레드의 경우 writer()를 실행하게 되고 그 내부에서 주어진 횟수만큼 공유변수에 접근하여 값을 변경한다.
   3. reader 스레드의 경우 reader()를 실행하게 되고 그 내부에서 주어진 횟수만큼 공유 변수의 값을 읽어서 출력한다.
   4. 어느 스레드가 먼저 불릴지는 알 수 없으므로 무작위로 스레드가 동작하게 되어 코드의 순서는 보장하지 못한다.
5. **프로그램이 어떻게 First Reader-Writers Problem을 해결하는지 설명**
   1. reader가 수행되는 동안 writer가 critical section에 진입하는 것을 막음으로써 First reader-writer 문제를 해결
   2. 그러나 이 방법은 writer에게 starvation을 야기할 수 있다
6. **컴파일 방법**
   1. Makefile 사용**:** 
7. **과제 B**
8. **LR solution**
   1. Dining-Philosophers problem을 해결하기 위해 고안된 방법이다
   2. 철학자를 L과 R 타입으로 나누어 진행하는 것을 말한다
   3. L타입의 철학자는 먼저 왼쪽부터 젓가락을 잡게 되고 R타입의 철학자는 먼저 오른쪽부터 젓가락을 잡게 된다.
   4. 홀수 번째 철학자는 L타입으로 배정하고 짝수 번째 철학자는 R타입으로 배정한다.
9. **자료구조 설명**
   1. 세마포어에 대한 추가 설명: 함수
      1. sem\_init() : sem 매개 변수로 넘겨받은 이름이 명명되지 않은 세마포어를 초기화
      2. sem\_wait(): 잠금 수행을 통하여 sem로 참조된 세마포어를 잠근다. 만일 현재의 세마포어 값이 0 이라면, 호출하는 스레드는 세마포어를 잠그거나 시그널에 의해서 호출이 중단되기 전까지 sem\_wait()로부터 돌아오지 않는다. 함수 수행이 성공이라면 세마포어의 상태는 잠금상태가 되며, the sem\_post() 함수가 실행이 되어 성공적으로 반환하기 전까지 잠금상태를 유지한다.
      3. sem\_destroy(): 이름이 명명되지 않은 세마포어를 파괴한다
10. **함수 설명**
    1. 메인
       1. sem\_init 함수를 이용해서 세마포어를 초기화한다
       2. 5개의 철학자 스레드를 생성한 뒤 각 스레드에게 chopstick 함수를 실행하도록 한다.
       3. 모든 스레드가 끝날 때까지 기다린 후 세마포어를 모두 파괴한다
    2. chopstick
       1. 타입을 구분해주기 위해 자신이 몇번 째 철학자인지 판별한다
       2. 자신의 타입이 L 이라면 가장 먼저 왼쪽 젓가락에 대한 세마포어를 잡고 R 타입이라면 먼저 오른쪽 젓가락에 대한 세모포어를 잡는다.
       3. 이 후 나머지 한쪽의 젓가락을 획득하는데 두 젓가락을 모두 잡는데 성공했다면 식사를 시작한다
       4. 식사를 끝마치면 두 젓가락을 모두 놓고 종료한다.
11. **프로그램 구조 설명**
    1. 메인 함수에서 생성된 각각의 스레드들이 chopstick 함수를 실행한다.
    2. 동시에 5개의 스레드들이 젓가락을 규칙에 맞게 잡게 되고 양 젓가락을 모두 획득한 철학자는 식사를 한 후 두 젓가락을 내려 놓는다
    3. 모든 철학자들이 식사를 마치게 되면 스레드가 모두 종료되고 프로그램은 끝이 난다.
12. **프로그램이 어떻게 Dining-Philosophers Problem을 해결하는지 설명**
    1. 홀수 번째 철학자는 왼쪽 젓가락을 먼저 획득하고 짝수 번째 철학자는 오른쪽 젓가락을 먼저 획득하려 하기 때문에 옆에 붙어 앉은 철학자 둘은 같은 젓가락을 먼저 잡아야 하게 된다.
    2. 즉 둘 사이의 젓가락을 잡고 있는 사람의 식사가 끝나기 전까지는 다른 한 명은 젓가락을 잡으려는 시도를 할 수 없게 된다
    3. 그렇기 때문에 항상 1번째와 2번째 중 한 명, 3번째와 4번째 중 한 명, 5번째와 0번째중 한 명 만이 젓가락을 잡고 있게 되어서 데드락이 발생하지 않게 된다.
13. **컴파일 방법**
    1. Makefile 사용**:** 