스트 상자:      
     
스트 상자: 

12121518

유호균

OS 2차 과제

오퍼레이팅 시스템

-송민석 교수님

consumer, producer problem

1. **프로그램 인터페이스**

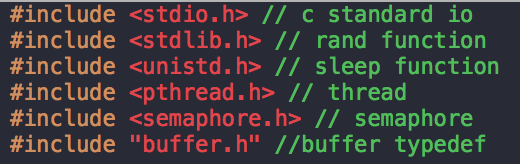
main 함수의 argument로 첫번째 인자는 프로그램 수행시간, 두번째 인자는 producer 쓰레드 갯수, 세번째 인자는 consumer 쓰레드 갯수를 지정한다. 다음은 예제이다.



ex) 10초동안 10개의 producer, 5개의 consumer 쓰레드로 프로그램을 실행한다.

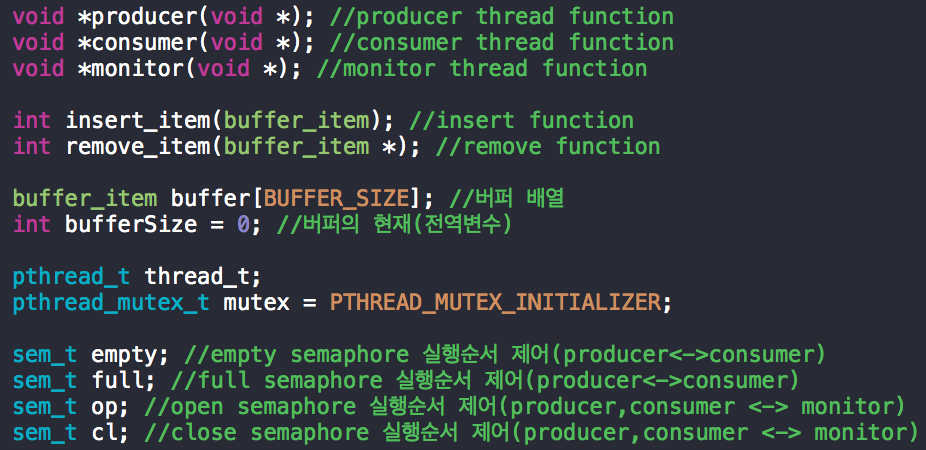
1. **프로그램 설명**

<헤더파일>



각각의 설명은 주석으로 대체한다. 이번 과제에서 중요한건 semaphore헤더를 인클루드하여 semaphore에 대해 다루었다는 것이다. Semaphore.h에는 semaphore\_t타입, sem\_init, sem\_wait, sem\_post등 semaphore에 관한 다양한 메소드들이 들어있다. 또한 멀티쓰레드를 다루기 때문에 pthread를 이용하여 저번 과제처럼 ptrhead\_create등을 이용하여 쓰레드를 생성하므로 pthread.h를 인클루드 하였다. 랜덤함수도 이용하기 때문에 stdlib.h를 인클루드 하였고 과제에서 typedef int buffer\_item, buffer size등 헤더가 주어졌으므로 인클루드 하여 메인 c 파일에서 사용하게 되었다.

<전역변수>



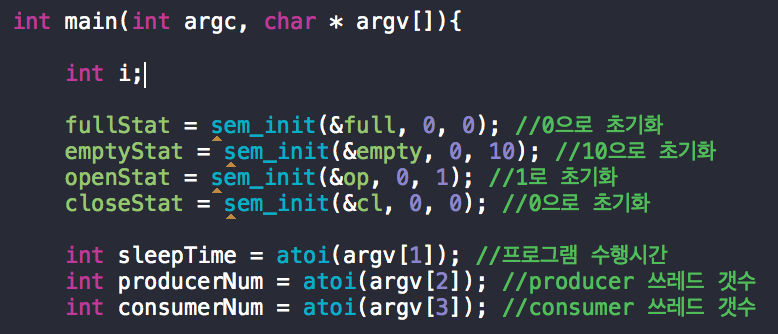
처음부분에 3가지 함수가 존재한다. Producer라는 producer 쓰레드가 생성되면 수행되는 함수이다. Consumer는 consumer쓰레드가 생성되면 수행하는 함수이고, monitor는 monitor쓰레드가 실행되면 수행하는 함수이다.

다음으로 producer 함수에서 insert\_item이란 작업을 수행하므로 함수를 선언하였고, consumerd서 remove\_item이란 작업을 수행하므로 함수를 수행하였다.

과제PDF의 헤더파일에서 buffer\_item과 buffer\_size가 주어졌으므로 c파일에서 선언하였고, thread를 만들기 위해 thread\_t를 선언하고 mutex\_t도 선언하였다.

총 4개의 세마포어를 다룬다. Empty, full 은 producer와 consumer의 순서를 통제하기 위하여 사용한다. 자세한 것은 아래에서 다루겠다. Op,cl은 monitor 쓰레드와 producer,consumer 쓰레드의 순서를 제어하기 위하여 선언하였다. 이것도 역시 아래에서 다루겠다.

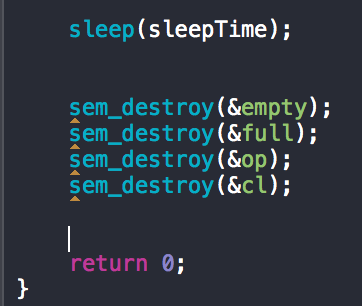
<메인함수>



세마포어를 초기화한다. 여기서 중요한 것은 세마포어 초기화시 할당하는 3번째 인자이다. 이 3번째 인자를 각각 다른 숫자로 초기화함에 주목해야 한다. Full semaphore는 초기에 buffer에 차 있는것이 없으므로 0으로 할당하고, empty는 buffer가모두 비어있으므로 10으로 할당, 초기 monitor쓰레드가 나중에 출력되어야 하므로 op semaphore에 1, cl semaphore에 0을 할당하였다. (op는 open, cl 은 close의 약자로 사용하였다.)

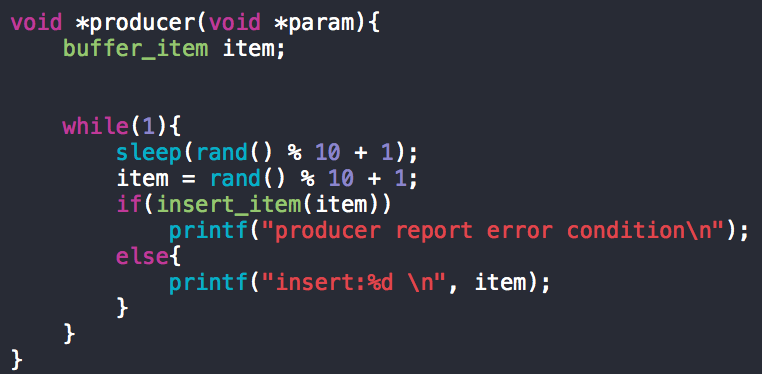


버퍼를 초기화하고, 입력받은 producer, consumer 쓰레드의 갯수만큼 쓰레드를 생성한다. Monitor 쓰레드도 역시 생성한다.

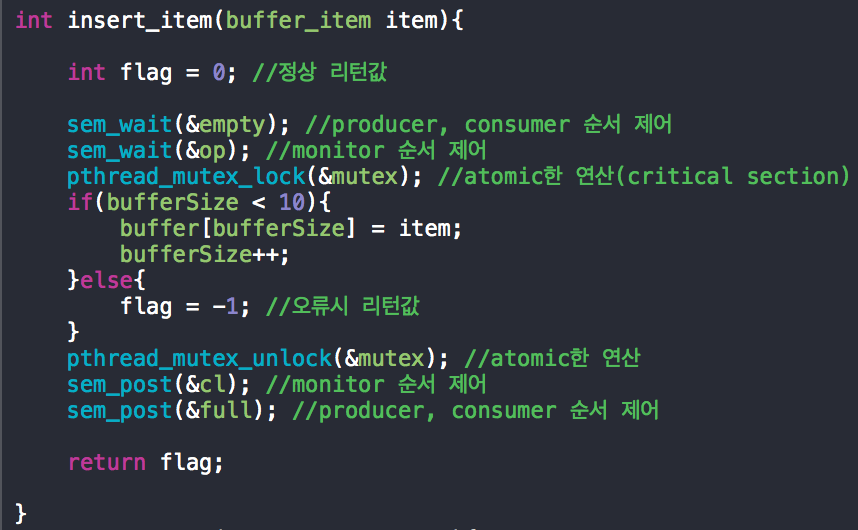


사용자가 입력한 시간만큼 프로그램을 수행하고, semaphore를 destroy 한 후에 프로그램을 종료한다.

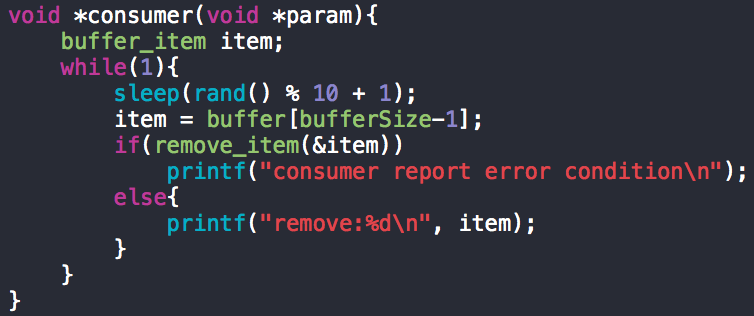
<함수 구현>



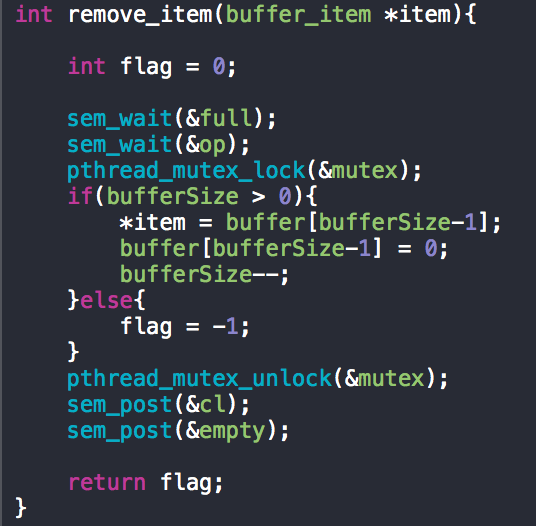
producer 쓰레드의 함수이다. Random한 시간마다 실행되고, random한 아이템이 삽입된다. Insert\_item함수가 0을 반환하면 아이템이 출력되고 -1을 반환하면 아이템이 출력되지 않고 producer report error가 출력된다. 에러메시지가 출력된느 경우는 insert\_item이 제대로 동작하지 않을 경우이다.



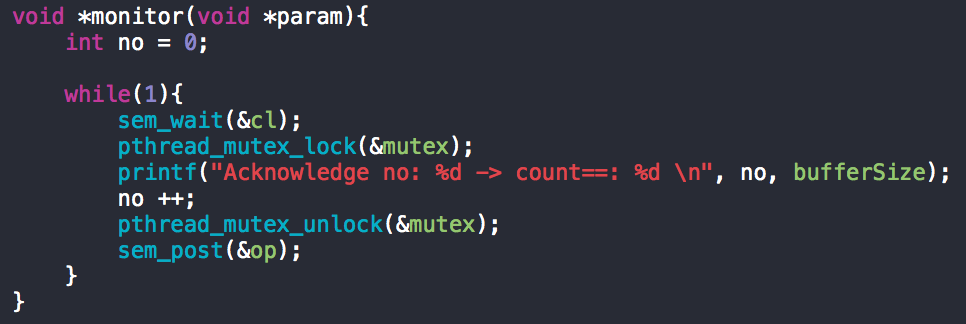
producer의 insert\_item함수 구현부분이다. 중요한 점은 semaphore를 사용하여 순서를 제어하고 mutex를 이용해 critical section 부분에 대해 atomic한 연산을 수행하여 공용자원(전역변수)의 consistency를 만족했다는 점이다. Producer와 consumer의 순서를 제어하는 부분은 수업시간에 설명한 Dining Philosopher problem문제를 해결하는 방법과 마찬가지로 full, empty semaphore와 mutex를 이용하여 구현하였고, monitor와의 순서 제어 부분은 별개의 semaphore를 만들어 제어하였다.



consumer 쓰레드가 수행하는 함수이고, remove\_item에 item의 주소값을 넘긴다는 점이 조금 다르다.



consumer 쓰레드에서 실행되는 remove\_item 함수이다. Insert\_item과 크게 다른점은 없으며 여기도 역시 op,cl semaphore를 사용하여 monitor쓰레드의 순서를 제어하였다. Item의 주소값을 받아왔으므로 pointer의 값으로써 삭제될 값을 넘겨서 후에 삭제될 아이템이 출력되도록 만든다. 배열의 끝값을 제거하였다.



monitor 쓰레드에서 사용하는 함수이다. Cl,op semaphore를 사용하여 consumer, producer이후에 monitor쓰레드가 동작하지 않으면 consumser, producer가 기다리는 것으로 순서를 제어하였다

1. **출력결과**  
   insert, remove, monitord의 순서를 제어했음을 알 수 있다. Insert, remove는 생성된 아이템의 인덱스가 아니라 생성된 아이템 값이다.( item = rand() % 10 + 1;)
2. **평가 및 개선방향**1. 과제에서 insert하는 부분은 정확히 설명하였는데, remove에 관한 부분이 조금 모호하다. 특별한 설명이 없어서 구현이 쉬운 끝값 제거를 선택했는데, 만약 조금 더 자세히 구현한다면 consumer 쓰레드의 구현에서 buffer\_item을 랜덤하게 주어지고, remove\_item함수에서는 주어진 값을 배열을 탐색하며 검색하고, 그 값을 삭제하면 조금더 자세한 프로그램이 될 수 있지 않을까 생각한다.   
   2. Count Semaphore 2개로 구현하였는데 생각해내기 꽤 까다로웠다. (producer,consumer ⬄ monitor)의 순서를 제어하므로굳이 count가 아니라 binary semaphore로 구현해도 될 것 같다는 생각이 들었다.
3. **기타 특이사항**Mac os의 XCODE에서 코드를 수정하고, 원격 리눅스 서버에서 gcc 컴파일 작업을 하였는데, sem\_init과 sem\_destroy 함수가 deprecated 되었다는 경고가 생겼다.  
   구글링해보니 sem\_init은 이름없는 세마포어, sem\_open은 이름있는 세마포어를 만드는 함수였다. Sem\_destroy역시 sem\_close와 비슷하다. mac의 xcode는 리눅스(GCC compiler)와 다르게 LVVM compiler를 default로 사용하므로 나오는 경고였다. 또한 리눅스에서도 pthread에 대한 컴파일을 수행할때 GCC 옵션에   
   –lpthread라는 옵션을 주어야 올바르게 컴파일 됨을 알 수 있다.
4. **참고자료   
   - Semaphore**  
   <http://www.joinc.co.kr/w/Site/system_programing/IPC/semaphores>  
   **- Mutex**  
   <http://www.joinc.co.kr/w/Site/Thread/Beginning/Mutex>  
   **- Dining philosopher problem**  
   <https://en.wikipedia.org/wiki/Dining_philosophers_problem>  
   **- In gcc, –lpthread Option**<http://stackoverflow.com/questions/23250863/difference-between-pthread-and-lpthread-while-compiling>