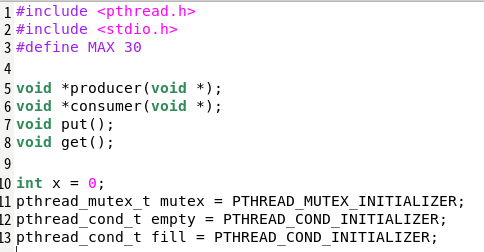
**학번: 12172133, 학년: 3, 학과: 컴퓨터공학과, 이름: 이종혁**

**제출일: 2021.05.29**

1. **synch\_pthread.c**

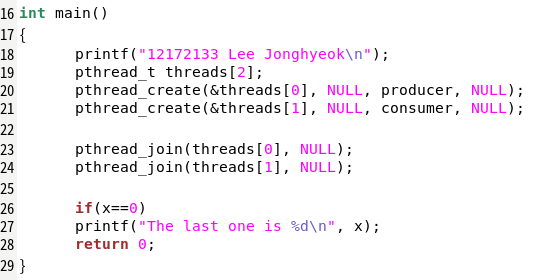
* 본 프로젝트의 소스 코드 c 파일은 동일 폴더에 있으므로, 캡쳐 사진을 이용해 설명하도록 한다.

1. **전역 변수 설정**



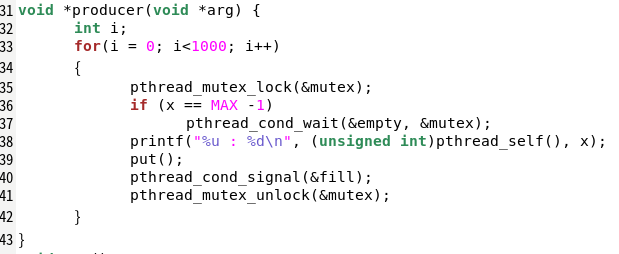
* 본 프로젝트 조건으로 x의 값이 버퍼의 크기로서 최대값 30이라고 주어졌으므로, MAX를 30으로 선언해준 다음 이것이 버퍼의 천장 개념을 하게끔 하였다.
* 본 프로젝트의 함수는 os 수업의 강의자료를 참고하여 작성하였는데, put, get함수는 각각 producer, consumer함수에서 호출될 것이다.
* 변수 mutex는 공유 변수(x)를 보호하기 위해 선언된 것이고(상호 배제), empty와 fill은 생산자 소비자 문제를 위해 선언되었다.

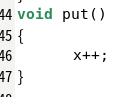
1. **Main()**



* main함수는 쓰레드를 생성하고 끝나기를 기다리는 게 역할이다. pthread\_create의 세번째 인자는 분기 목적의 쓰레드 함수 자리 이므로 당연히 producer와 consumer가 들어가게 된다.

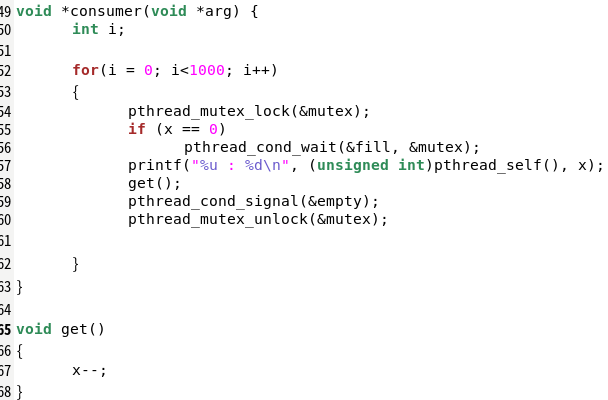
1. **producer**





* mutex를 활용하여 lock을 공유 자원을 보호한 다음, put함수를 부른다.
* pthread 문제에서는 x의 버퍼 값이 최대 30이라는 점을 버퍼가 수용할 수 있는 개수가 30이라는 것으로 해석하였다. 때문에 if 조건문에 x == 29(MAX-1)를 주어 0~29의 30개 값이 출력될 수 있도록 하였다.
* put함수는 위에서 설명했듯이 x값을 증가시키는 용도이다.
* put함수가 마무리되면, cond\_signal 함수를 통해 fill 쓰레드를 깨워준 후, pthread\_mutex\_unlock으로 다른 쓰레드가 공유자원에 접근할 수 있게 허용해준다.

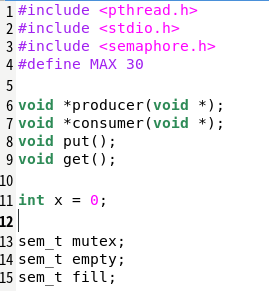
1. **consumer**



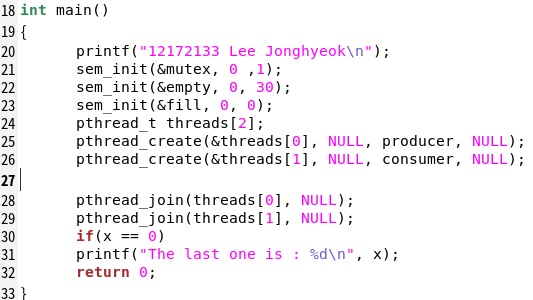
* consumer함수도 producer와 구조는 거의 같다. 다만, 이 함수는 소비자이므로 cond\_wait의 인자에는 fill이, cond\_signal의 인자에는 empty가 들어가게 된다. 그 외 mutex\_lock & unlock은 producer와 같은 메커니즘을 유지한다.
* get함수는 소비자에서 x를 감소시키는 함수이다.

1. **synch\_semaphore.c**

* **세마포어를 이용한 소스코드는 위 pthread를 이용한 코드를 조금 변형했을 뿐이므로, 각 지역변수의 역할은 거의 같다. 따라서 synch\_pthread에서 설명한 내용은 생략하고, 새로이 작성된 코드, 변수에 대한 설명을 하도록 한다.**

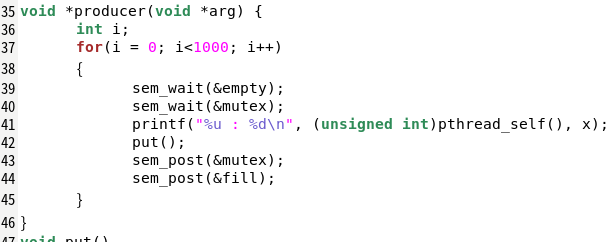
1. **지역변수 선언**

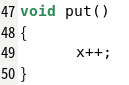
* pthread를 이용한 코드에서는 각 쓰레드 변수를 initializer로 대입해주었으나, 본 코드에서는 세마포어 헤더파일을 include하고 sem\_t type의 변수 mutex, empty, fill을 선언해주었다. 이들의 역할은 pthread를 이용한 위 코드와 다르지 않다.

1. **main()**

* main함수에서는 지역변수 mutex, empty, fill 변수에 대하여 semaphore initializing을 해준다.
* empty는 생산자가 발동되면 wait함수에 의해 하나씩 빈 공간이 줄어들고 소비자가 발생하면 증가하는 개념이므로, buffer의 max조건을 맞추기 위하여 30으로 초기화해준다.
* fill의 경우는 일단 생산자의 post함수에 의해 작동하기 시작하는 것으로 가정하고 0으로 초기화해준다.

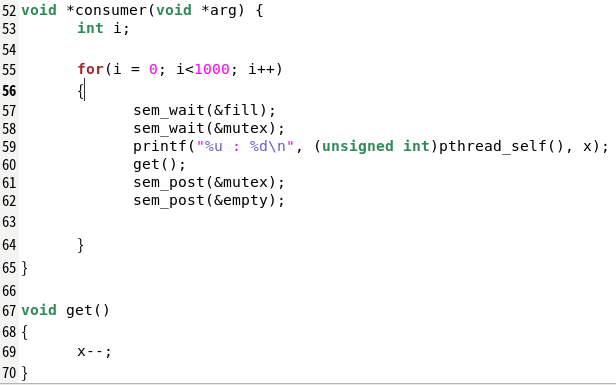
1. **producer**

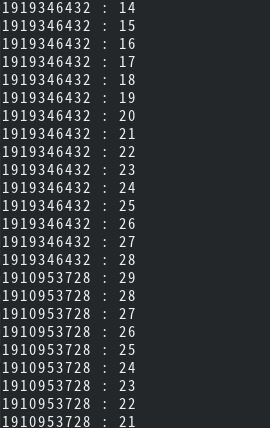
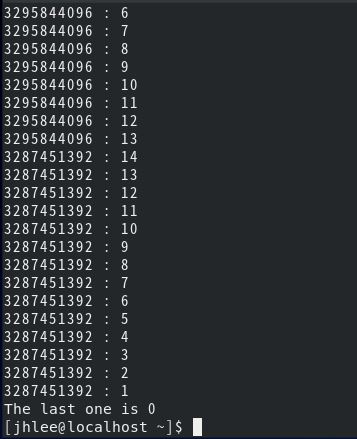




* mutex에 할당한 값은 1이었다. wait함수는 세마포어를 lock한 후 세마포어 카운트를 1 감소시키므로 mutex의 세마포어 카운트 값이 0이 되면, 해당 자원에는 접근하지 못하게 된다. 즉 block을 하여 공유자원 보호자의 역할을 하게 되는 것이다.
* put함수 이후 sem\_post는 세마포어 카운트를 증가시켜주는데, 생산자가 작동했으므로 fill의 세마포어 카운트를 늘려주어 블로킹을 풀어준다. 또한 mutex도 그 카운트를 늘려주어 락을 해지해주어 다른 스레드도 공유자원에 접근할 수 있도록 해준다.

1. **consumer**

* consumer의 내용은 producer와 다른 것이 거의 없다.
* 소비자이므로 wait함수를 통해 fill의 세마포어 카운트를 감소시키고, get함수가 실행 된 후 sem\_post함수를 통해 empty의 세마포어 카운트를 증가시킨다.
* mutex의 blocking 메커니즘 또한 producer와 같다.

1. **실행 화면**
2. **synch\_pthread.c**

* 좌측은 pthread를 이용한 코드의 출력 화면 중간 부분으로, 쓰레드 간 context 스위치가 일어나지만 x에 대한 잘못된 기작은 일어나지 않고 있는 것을 볼 수 있다.(동일한 수의 중복 출력과 같은 경우)
* 좌측 사진의 1919346432 스레드에서 1910953728로 넘어가는 과정에서 28 -> 29가 되었는데, 이 이유는 전 스레드의 생산자부분에서 x값이 늘어난 것이 후 스레드의 소비자에서 출력된 것이다.
* 과제 조건인 buffer의 크기의 max 값이 30일 때의 양수만을 출력하는 조건을 완료했으며, 이로서 프로젝트 목적을 달성하였다고 볼 수 있다.

1. 테이블이(가) 표시된 사진

   자동 생성된 설명**synch\_semaphore.c**

텍스트이(가) 표시된 사진

자동 생성된 설명

* 좌측은 semaphore을 이용한 코드의 출력 화면 중간 부분이다. 역시 쓰레드 간 context 스위치가 일어나면서도 부적절한 기작은 일어나고 있지 않다. 또한 0~30의 값이 출력되고 있다.
* 우측 사진은 쓰레드의 공유자원 x가 0이 되며 종료되었음을 보여준다. 이로써 과제 조건을 세마포어로 구현하는데 성공하였으며, 프로젝트 목적을 달성하였다고 볼 수 있다.