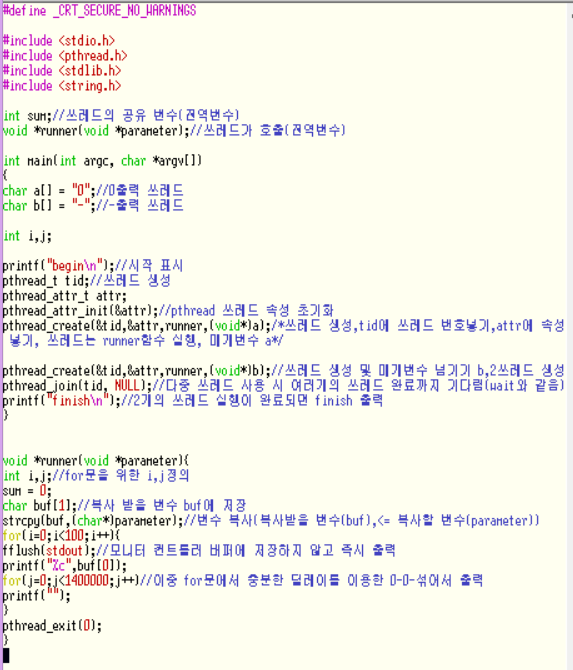
**운영체제(Multithread, Synchronization)**

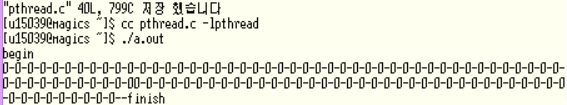
과제 6

**2015253039 권진우**

**1. (multithreaded program 작성) 다음과 같이 동작하는 multithreaded program을 UNIX/Linux의 pthread 라이브러리 사용 (교과서 및 thread 참고자료를 참고) (컴파일할 때에 –lpthread 옵션 사용)**



<코드>



<출력 결과>

<main함수에서 begin으로 시작해서 0-0-0-0-순으로 100회 출력, pthread사용하여 2개의 쓰레드 사용하기, 이중 for문을 이용하여 0-0-가 섞여서 출력, 2개의 thread는 같은 thread body를 사용create 함수를 이용하여 쓰레드를 생성, 시간차를 두어 섞어서 출력하며, exit함수로 쓰레드 종료>

\* atoi : 문자형을 정수형으로 변환하는 함수

**2. (synthronization 프로그래밍) producer-consumer 프로그램은 다음과 같이 동작한다.**

§ producer와 consumer는 각각 thread로서 동작한다.

§ producer와 consumer는 shared bounded buffer를 통해서 메시지를 주고받는다.

§ producer는 임의의 시간 지연 후 메시지를 보내는 동작을 반복한다.

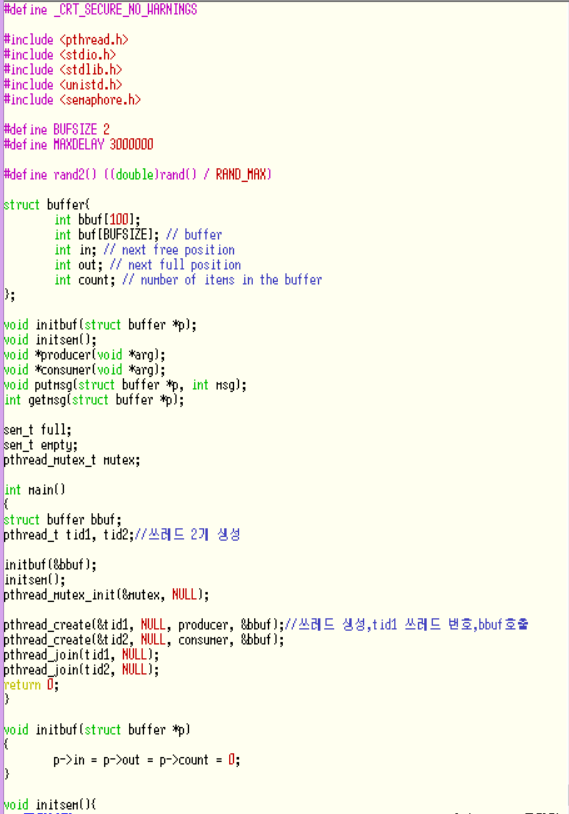
§ consumer는 임의의 시간 지연 후 메시지를 받는 동작을 반복한다.

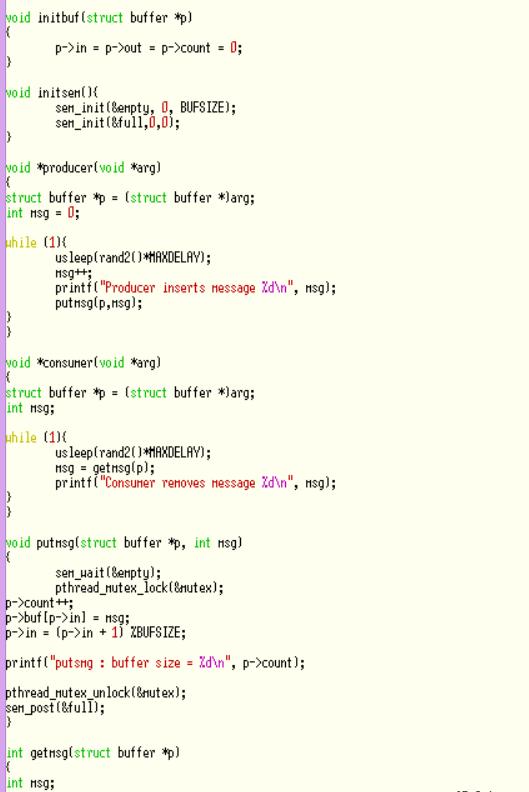
§ 지연 시간은 난수발생 함수를 사용하여 0과 MAXDELAY 사이의 시간을 얻어서 사용하며 시간 지연은 usleep 함수를 이용한다.

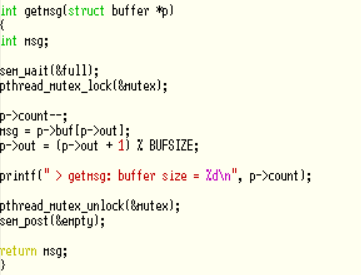
-동기화가 적용 되지 않은 프로그램이 주어지고 이 프로그램 동작을 설명하고 동기화를 적용하기

**동작 설명** : <프로그램은 2개의 쓰레드로 Producer 쓰레드와 Consumer쓰레드로 구성되며 Producer 쓰레드는 메시지를 버퍼를 통하여 전송하고 Consumer 쓰레드는 버퍼로부터 메시지를 받아오는데 두 쓰레드 간의 동작 시간을 맞춰주는 동기화가 이뤄지도록 만들어 주는 것이 목표입니다. 이 때 코드의 버퍼 사이즈는 2로 설정되어있으며 동기화를 사용하지 않을 경우 Producer 쓰레드는 Consumer 쓰레드의 메시지 수신과는 상관 없이 메시지를 전송하게 되며 Count변수에 대한 원치않는 결과가 나올 수도 있습니다. 세마포나 Mutex,조건 변수 쓰레드 동기화를 이용하여 쓰레드 간의 동작 순서를 맞춰줌과 동시에 CS구역(putmsg, getmsg)에 Lock을 검으로 인하여 Count 변수에 경쟁조건으로 인한 잘못된 결과가 발생하는 것을 막아줍니다.>

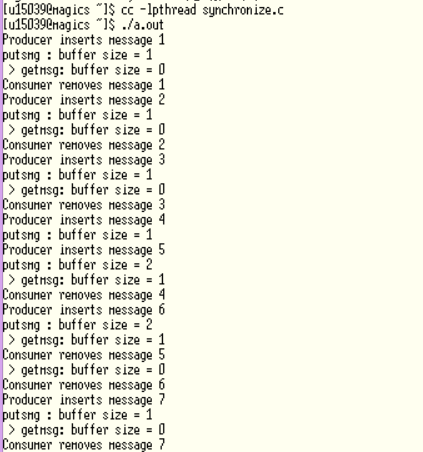
**1)Semaphore사용**







<세마포 사용 메시지 송/수신 코드>



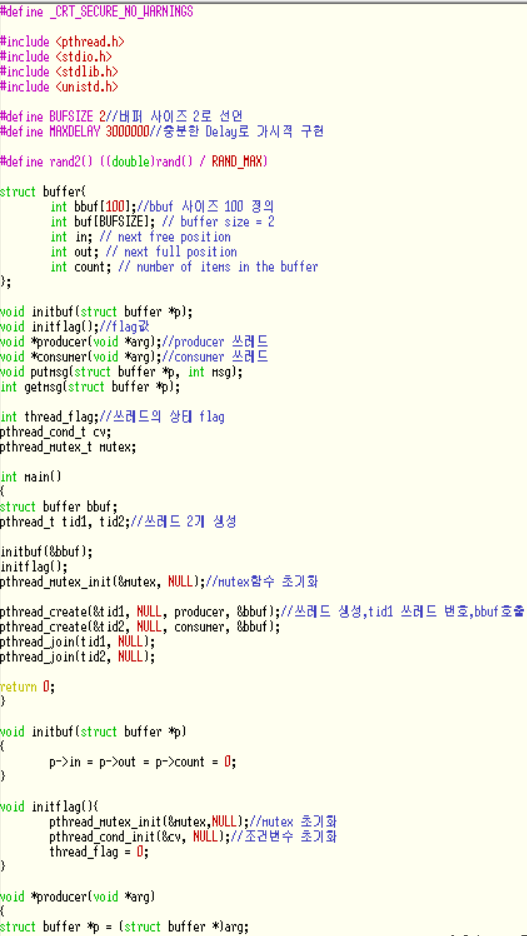
<세마포를 이용한 코드 출력>

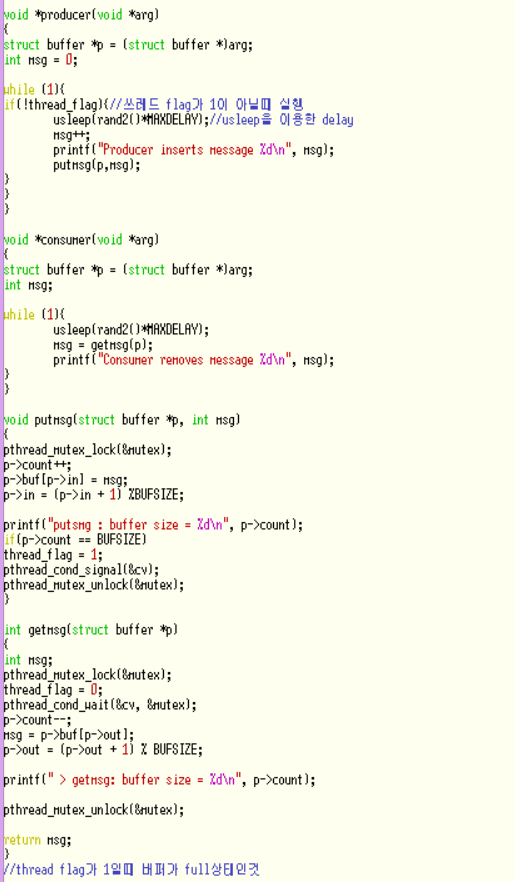
**-코드에 대한 설명 및 구성** : 쓰레드를 2개 생성 해주었고 1개는 메시지를 전송하는 Producer, 1개는 메시지를 수신하는 Consumer 쓰레드로 구성되었으며 세마포는 Mutex Lock보다 더욱 정교한 프로세스 동기화 도구인데 세마포 값은 사용가능한 자원의 수를 나타내주며 위의 코드에서는 BUFSIZE가 세마포의 값으로 사용되었습니다. putmsg와 getmsg에서 세마포를 구현하고 putmsg는 버퍼가 비어있을때(empty)까지 기다렸다가 mutex\_lock을 걸고 작업 수행 후 mutex\_lock을 해제함과 동시에 세마포값을 full로 바꿔줍니다. -> getmsg는 반대로 작업이 수행 되므로 서로 동기적으로 작업이 수행이 될 수 있습니다.

또한 pthread\_mutex\_lock(&mutex)와 pthread\_mutex\_unlock(&mutex)로 count변수 사용 시 상호배제를 구현하였습니다.

**2)Mutex와 Condition variable 사용**

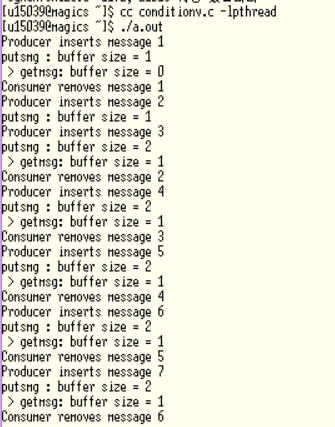
**<구현된 코드는 다음페이지에 있습니다.>**

****

****

<mutex와 condition variable을 이용한 동기화 구현 코드>

**<코드출력 및 설명 다음 장에 있습니다.>**



<Condition variable 코드 출력>

**-설명 및 구성 :** thread\_flag가 0이면 버퍼가 비었다는 것을 의미하여 putmsg가 가능하고 1이면 버퍼가 full이라는 것을 의미하여 getmsg가 이루어진다. mutex\_lock을 이용하여 상호배제를 구현하였고 CS구역의 동시 접근을 제어합니다. 프로세스가 2개인 경우에는 세마포를 사용하는 것이 조건변수(Condition variable)사용하는 것 보다 수월하였습니다. wait 와 signal 변수를 사용하며, wait는 다른 프로세스가 signal을 호출할 때까지 중지(block)되어 있고 signal 연산은 정확히 하나의 프로세스만 재개시킴으로써 wait된 하나의 프로세스를 실행시킵니다.

세마포의 V연산은 세마포 값을 변경 시키지만 (자원의 빈자리가 없으면 wait, 있으면 바로 진행)조건변수는 wait를 만나면 기다리고 signal이 호출 할 때 실행된다.