**2016025469 컴퓨터공학과 서건식**

**운영 체제 HW#3**

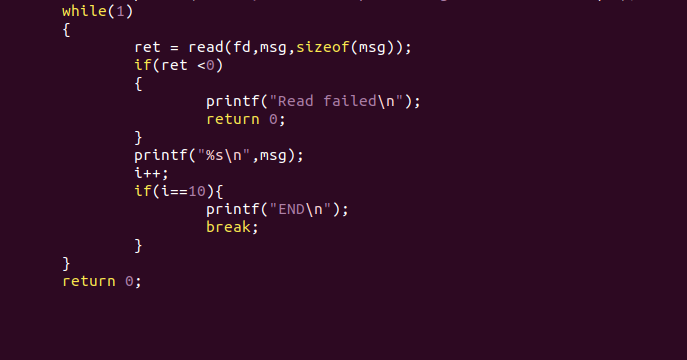
**제출 일자 : 2020/04/11**

1. **과제 A**

* **Named Pipe**

1. **프로그램 설명**
2. **reader\_Aprocess.c**

****

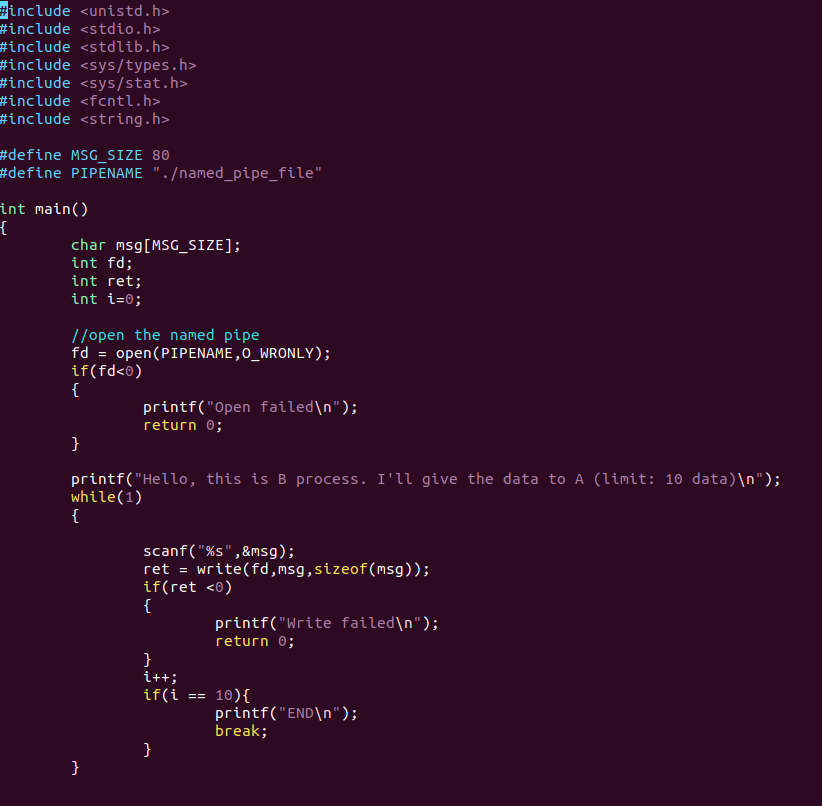
****

**Named pipe를 통해 데이터를 받은 프로세스입니다.**

**메커니즘은 2번에 기술하도록 하겠습니다.**

**추가된 부분은 While문 이며, 파일디스크럽터의 메시지를 받아와서 ret이 0보다 큰 경우에 메시지를 출력합니다. 최대 10번까지 읽습니다.**

1. **Writer\_Bprocess.c**

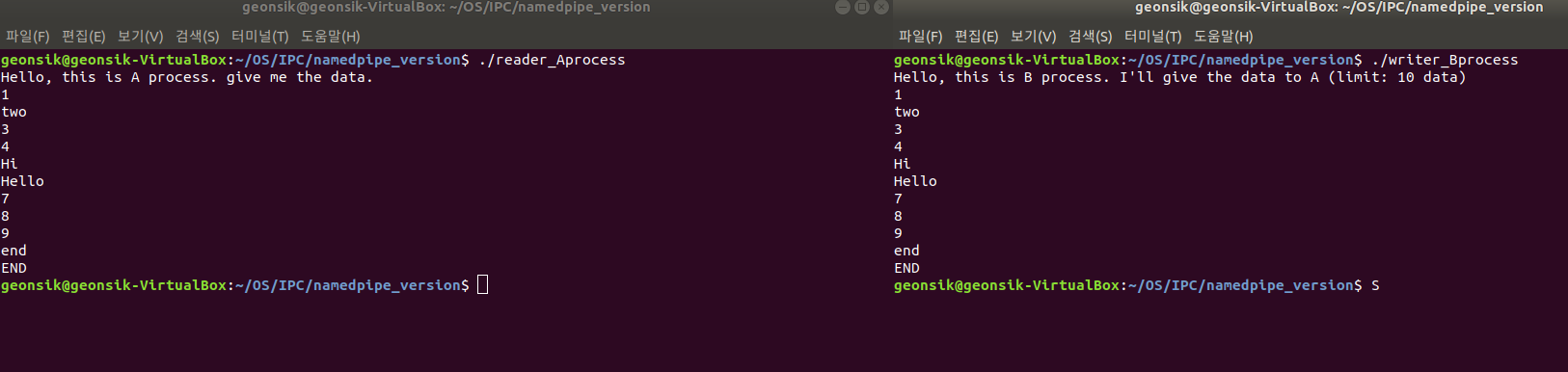
****

**이 또한 named\_pipe\_file을 파일디스크럽터로 open후에 while문에서 10번까지 입력하는 프로세스입니다.**

1. **IPC 메커니즘 설명**

**Reader\_Aprocess 에서 mkfifo를 통해 파이프라인을 형성해야 하기 때문에 해당 실행파일을 먼저 실행하고, writer\_Bprocess를 실행한 후에 여기서 해당 파이프에 문자를 입력하면, A 프로세스에서 해당 값을 받습니다. 동기화가 되기 때문에 동기화 문제가 없습니다.**

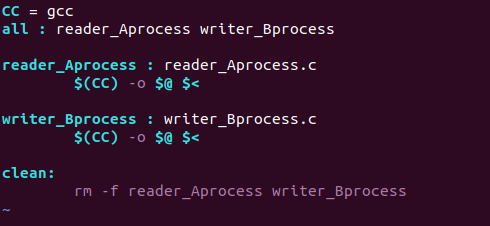
**실행결과입니다.**



1. **컴파일 방법 설명**

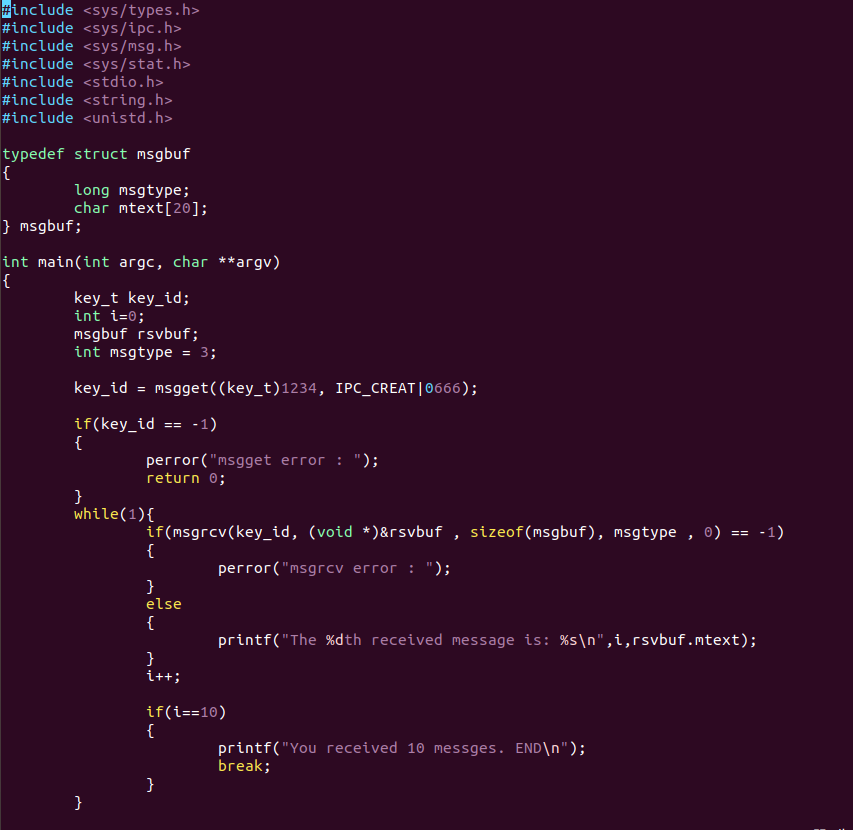
**Makefile을 만들어 컴파일하였습니다.**

**Makefile은 밑에 첨부하겠습니다.**

****

* **Message Queue**

1. **프로그램 설명**
2. **reader\_Aprocess.c**

****

**구조체는 msgbuf로 선언하였습니다. 메커니즘은 2번에 기술하도록 하겠습니다.**

**반복문에서, 리시브버퍼 함수를 실행할 때 오류가 없다면 리시브버퍼의 내용을 출력합니다. 10번째 메시지를 받으면 종료합니다.**

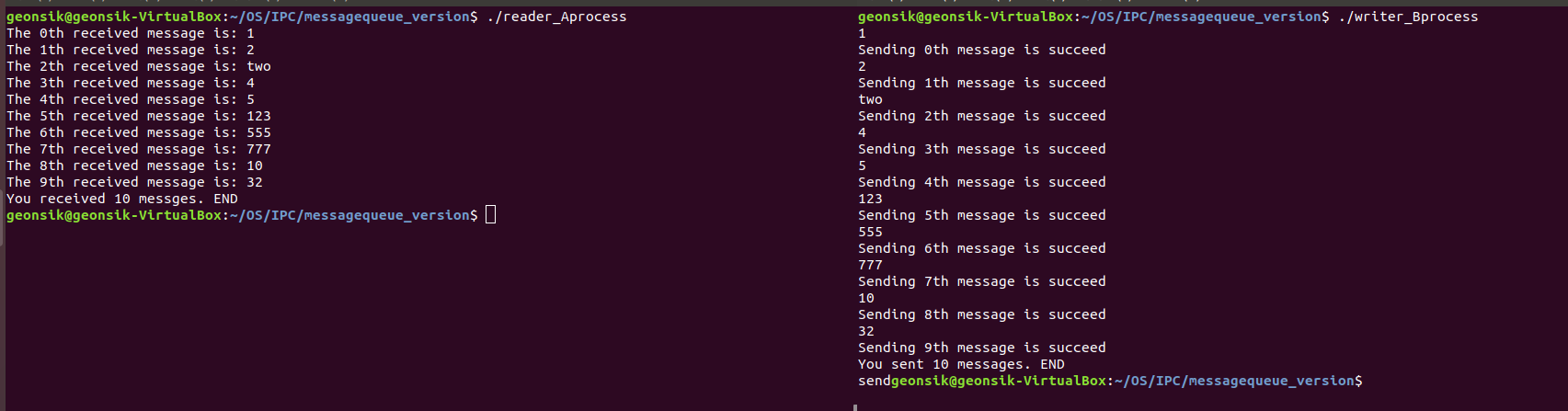
1. **Writer\_Bprocess.c**

****

**writer이며, scanf로 sndbuf 내의 멤버변수 mtext에 입력 후 msgsnd 함수로 버퍼에 sndbuf를 담습니다.**

**최대 10번까지 보낼 수 있습니다.**

**실행결과 입니다.**



1. **IPC 메커니즘 설명**

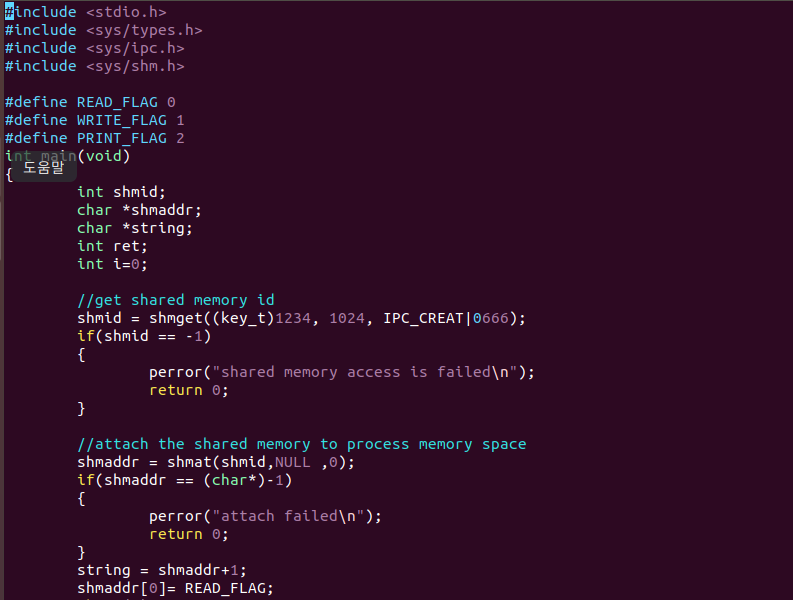
**Reader는 consumer, writer는 Producer 입니다. 커널에서 관리하는 메시지 큐에 Producer가 자료를 넣으면, 선입선출이기 때문에 consumer가 해당 값을 빼내 옵니다.**

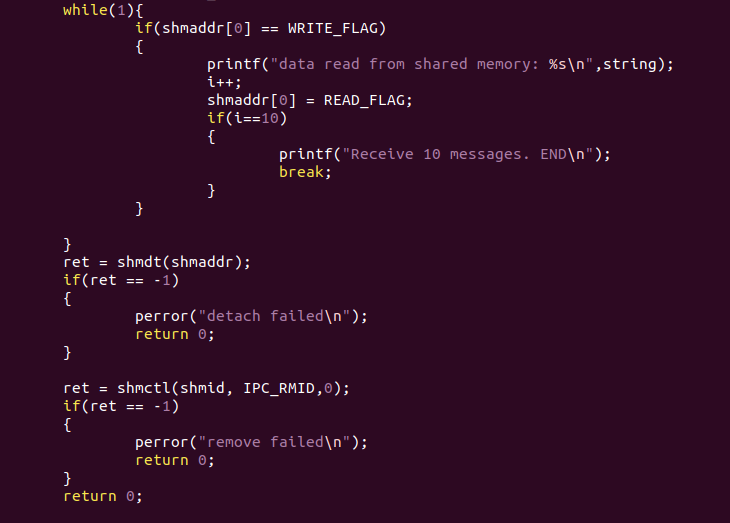
1. **컴파일 방법 설명**

**위와 동일합니다.**

* **Shared Memory**

1. **프로그램 설명**
2. **reader\_Aprocess.c**

****

****

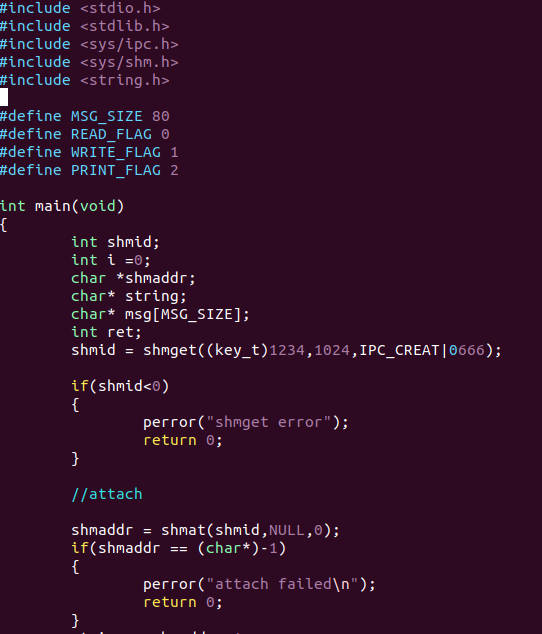
**Shared memory는 메시지큐나 파이프와 다르게 프로세스간 동기화가 이뤄지지 않습니다. 따라서 Shared memory에 Writer가 데이터를 입력했다는 Flag를 입력함으로써 동기화를 해줄 수 있습니다.**

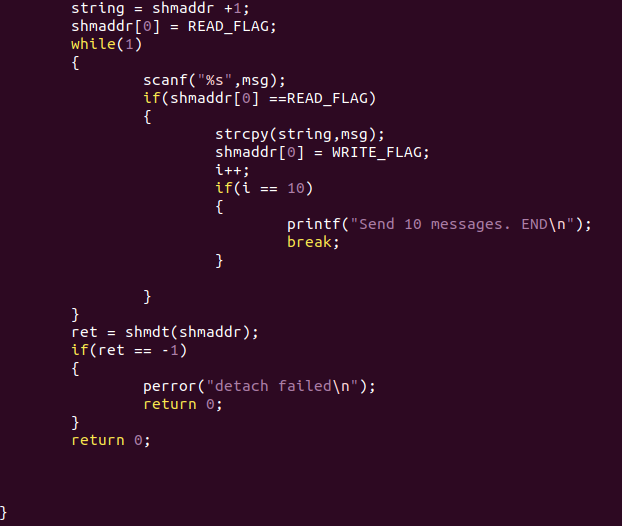
**Flag는 READ,WRITE가 있습니다. PRINT는 사용하지 않았습니다.**

**string이라는 변수는 문자를 저장한 곳을 가리키는 포인터 변수이며, shmaddr[0]+1 을 가리킵니다. shmaddr[0]에는 플래그를 삽입합니다. 처음에는 READ\_FLAG를 넣습니다.**

**while문에서, flag가 WRITE\_FLAG로 바뀔 때 까지 if문에서 대기합니다. writer에서 쓰기를 마치고 플래그를 바꾸면, 그 때 if문 값이 참이 되면서 데이터를 읽습니다. 읽자마자 플래그를 READ상태로 바꾸기 때문에, 다음 플래그가 올 때 까지 대기합니다.**

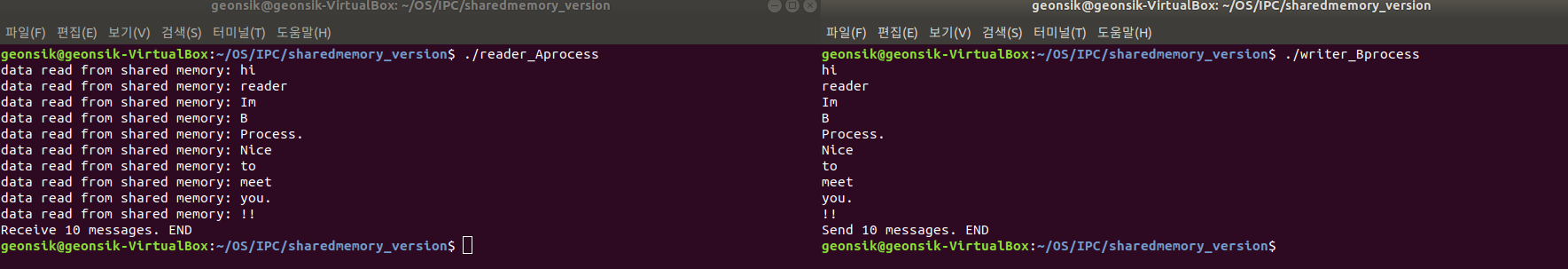
1. **Writer\_Bprocess**

****

****

**Reader와 거의 동일합니다만, 처음에 실행하는 것이 A process 이기 때문에 shmaddr[0]은 READ\_FLAG일것입니다. 그것이 아니더라도 while문 전에 첫 입력이기 때문에 flag를 교체해줬습니다. while문에서 READ FLAG일 경우 입력받은 값을 string에 저장하고, 플래그를 WRITE\_FLAG로 바꿔줍니다. 바꿔주는 동시에 A process가 읽게됩니다.**

**실행결과 입니다.**



1. **IPC 메커니즘 설명**

**기본적으로 shared memory는 동기화를 지원하지 않는 이유가, FLAG가 없다면 reader는 그대로 해당 번지에서 값을 가져와 읽어오기 때문에 아무리 writer가 입력값을 넣는다고 하더라도 reader는 읽어간 시간에 존재하는 데이터만을 가지고 오게 됩니다.**

**이것이 shared memory 가 플래그를 통해 데이터 입력의 유뮤를 판별해야 하는 이유입니다.**

1. **컴파일 방법 설명**

**위와 동일합니다.**

1. **과제 B**
2. **멀티스레딩 내용**

**쓰레드 : Lightweight process**

**\*쓰레드 또한 실행의 단위이며, 스케쥴링의 대상이다.**

**한 프로세스의 resources를 공유하며, 공유하는 것은 address space 중에서 code,data 영억을 공유한다. stack 공간과 processor context도 독자적으로 가진다.**

**이렇게 쓰레드를 생성하여 멀티스레딩을 통한 프로그래밍의 이점은, 응답성이 뛰어나다는 것이다. 만약 cpu와 process가 1개이고, 프로세스가 실행중이다가 I/O 디바이스의 처리로 wait상태가 되었다 하자. 쓰레드로 구현하게 되면 이 wait상태 에서도 CPU를 작동시키는게 가능하다. 또한 메모리 자원을 공유하며, 쓰레드간 데이터 전송이 아주 쉽고 저렴하다. 또한 cpu가 8개고 process가 1개일 때, 쓰레드를 8개 만들어서 cpu를 전부 사용 할 수 있도록 할 수 있다. 그리고 프로그램은 본래 sequential하게 돌아가는데, 쓰레드를 사용해서 동시에 작업하는 일,사건,작업을 설계하고 구현하는 것이 쉬워진다. (Concurrent programming model)**

**이러한 멀티스레딩은 웹 서버에서도 많이 사용되는데, process 를 사용자 수 만큼 fork 하는 것이 아닌 쓰레드를 사용자수 만큼 만든다.**

**또한 쓰레드는 유저 쓰레드와 커널 쓰레드로 구분 되는데, 일반적으로 유저 쓰레드는 Many to one 방식으로 커널에서 봤을 때는 한 프로세스로만 보인다. 커널 쓰레드는 One to one / Many to many 모델이 있다.**

**쓰레드에 대한 이슈들도 있으며, 예를 들면 User level 에서 일어나는 스택 오버플로우 이다. 싱글 쓰레드 일 경우 스택이 다른 영역을 침범하면 exception 이 발생하지만, 멀티쓰레드일 경우에는 커널이 이를 한 프로세스 로만 보기 때문에 독립적으로 만들어진 스택 영역을 서로 침범할 수가 있다. 이를 대비할 필요가 있다.**