**2018008304 컴퓨터소프트웨어학부 박경하**

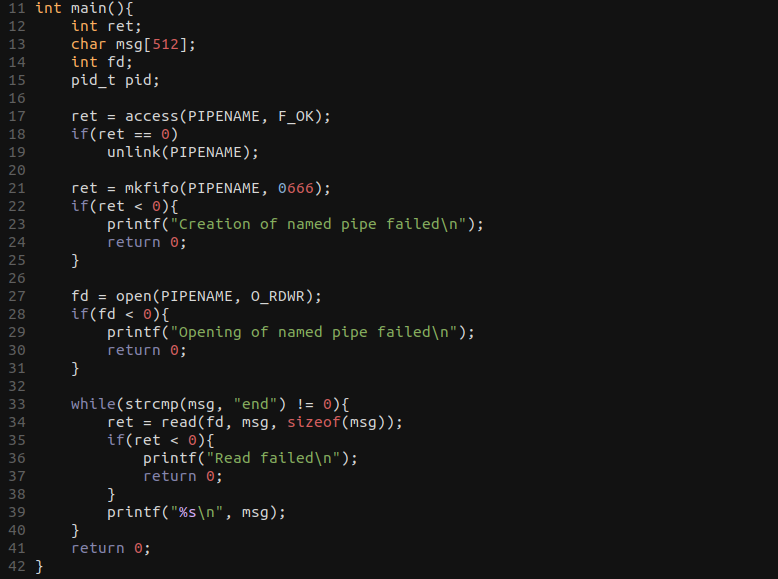
**운영 체제 HW#3**

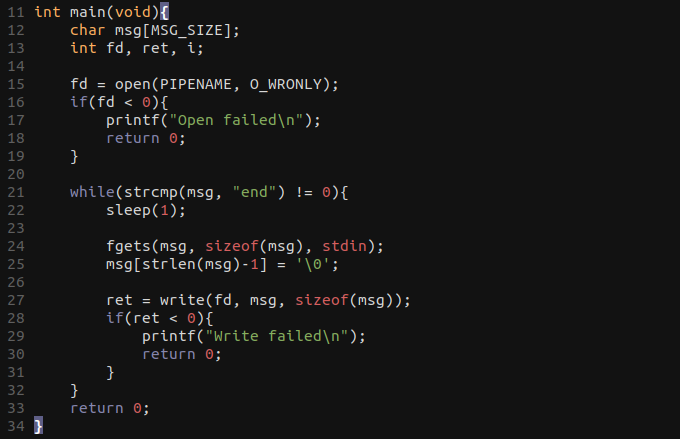
**제출 일자: 2021/03/30**

1. **과제 A**

* **Named Pipe**

1. **프로그램 설명**

****

****

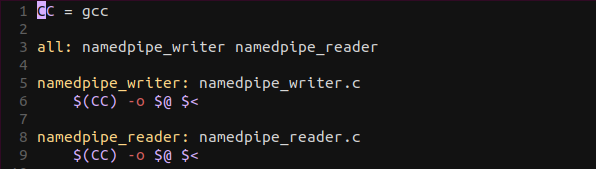
첫번째 사진은 reader콛 코드, 두번째 사진은 writer 코드입니다. Reader 코드를 먼저 보면, 우선 access 함수를 이용해 pipe가 존재하는지 조사한 뒤에, 만약 해당 이름의 Pipe가 있다면 unlink합니다. 그리고 ‘mkfifo’함수를 이용하여 named pipe을 생성하는데 이 때 관리권한을 0666으로 줌으로써 읽고 쓰기가 모두 가능하게끔 합니다. 과제 조건을 해결하기 위해서 while문을 이용하였으며, 만약 ‘end’을 입력했을 시에 종료되도록 설계하였습니다.

Writer 코드는 쓰기 권한만 주기 위해 O\_WRONLY로 named pipe파일을 엽니다. 그 뒤에 마찬가지로 과제 조건을 위해 while문을 사용하였으며 ‘fgets’함수로 문장을 입력 받은 뒤, write을 이용하여 그 문장을 파일에 작성하며 ‘end’ 입력 시에는 종료하게 됩니다.

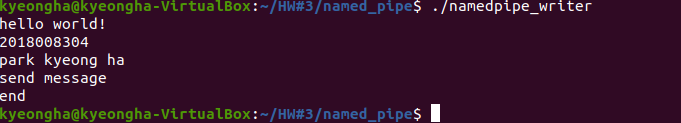
1. **IPC 메커니즘 설명**

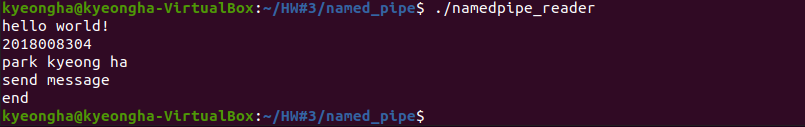
Named pipe는 writer와 reader 총 2개의 프로그램이 있으며, 한 프로그램은 데이터를 쓰고 다른 프로그램은 데이터를 읽기 위해 존재합니다. 일반 pipe는 다른 프로세스가 접근할 수 없다는 단점을 가지고 있기 때문에 이를 보완하기 위해 named pipe을 이용할 수 있습니다. Named pipe는 부모 자식간의 경우가 아니더라도 pipe을 사용할 수 있다는 장점이 있으며, Mkfifo함수를 통해 FIFO 파일을 이용하여 pipe 역할을 대신합니다. 따라서 Mkfifo라는 pipe를 통해 두 프로그램이 한 방향으로 데이터를 보내고 읽게 되는 방식입니다.

1. **컴파일 방법 설명**

****

Makefile은 위와 같으며, writer와 reader 프로그램을 각각 만들어 컴파일하였습니다.

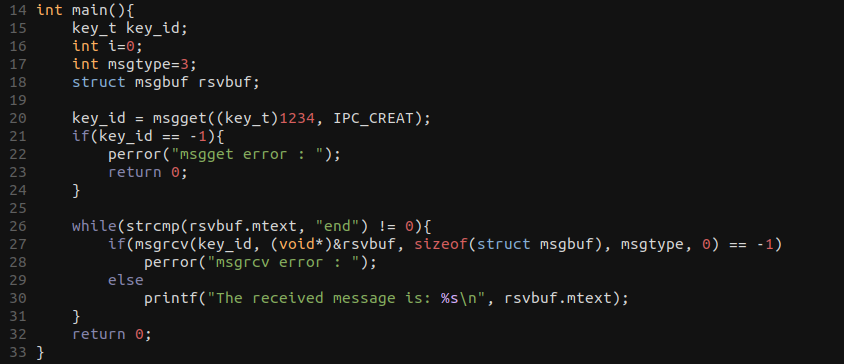
****

****

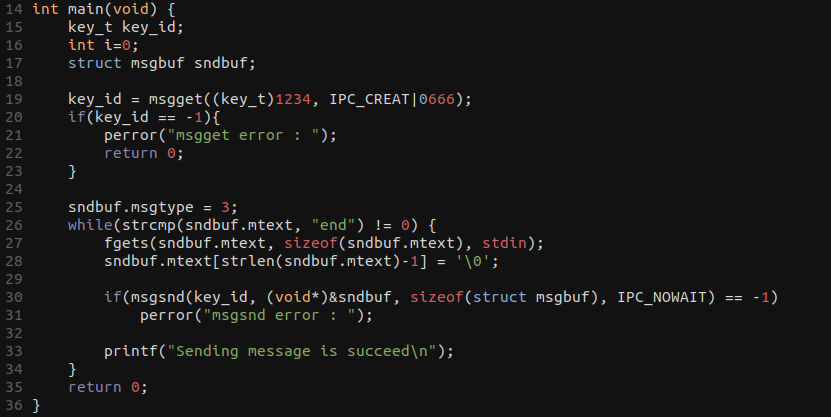
위는 실행결과로서, end라는 문장을 입력하기 전까지 계속 반복하여 각각 데이터를 쓰고 읽어서 출력하는 모습을 보여주고 있습니다.

* **Message Queue**

1. **프로그램 설명**



이것은 consumer의 코드이며, main함수 위에 msg의 타입과 내용을 저장하는 msgbuf 라는 구조체가 정의되어 있습니다. Msgtype을 통해 메세지가 어떤 타입을 갖는지 알 수 있게 됩니다. Main 함수에서는 key 값을 msgget의 인자로 넣어 message queue의 id를 반환해 저장한 뒤, while문을 이용하여 message queue에 있는 데이터를 읽어옵니다. 이 때, 만약 받아온 메세지가 ‘end’라면, 문장을 출력해준 뒤 종료하게 됩니다.



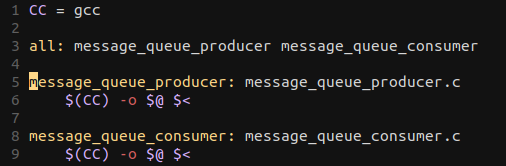
Producer의 코드로, consumer와 마찬가지로 main 함수 위에 msgbuf 구조체를 정의하였습니다. 또한, consumer 코드와 동일하게 msgget을 이용하여 메세지 id를 가져온 뒤에, 그 id의 mtext에 메세지를 msgsnd 함수를 이용해 저장하여 전송하게 됩니다.

1. **IPC 메커니즘 설명**

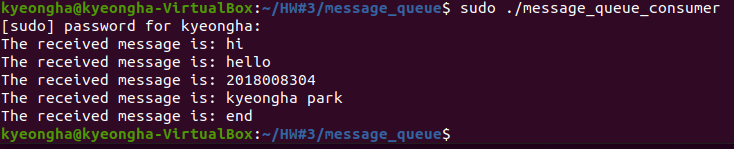
Message queue는 네트워크 채널을 통해 producer가 메세지를 보내고, consumer는 메세지를 받는 send와 recive 형태를 갖습니다.

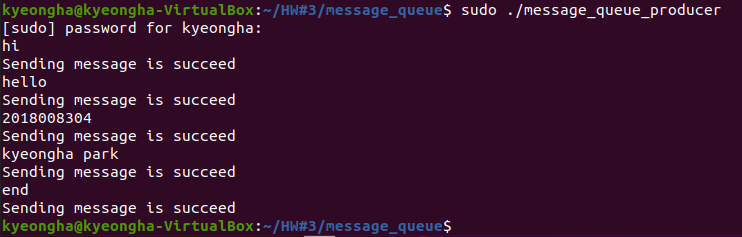
Message Queue는 커널에서 관리하여 key를 알면 어떤 프로세스에서도 접근이 가능해진다는 특징을 가지고 있습니다. Message queue와 pipe 모두 큐 자료구조로써 FIFO임에도 불구하고, Message queue는 메세지 타입을 통해 중간에 있는 데이터를 꺼낼 수 있다는 장점이 있습니다.

1. **컴파일 방법 설명**

****

Makefile은 위와 같으며, send 역할을 하는 producerr와 recive 역할을 하는 consumer을 각각 만들어 컴파일하였습니다.

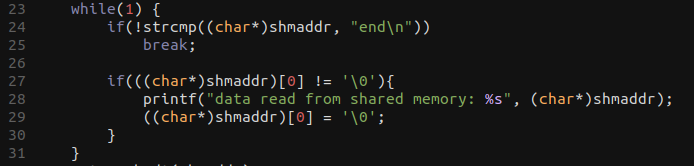
****

****

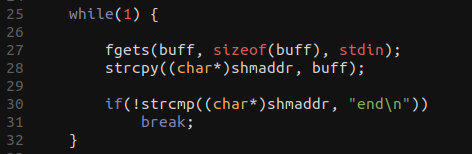
권한을 주기 위해 앞에 sudo를 붙여 실행하였으며, 마찬가지로 end라는 문장을 받기 전까지 문장을 받아 전달하고, 읽어서 출력하는 과정을 반복합니다.

* **Shared Memory**

1. **프로그램 설명**

****

위 사진은 reader 코드로, 과제 조건을 위해 추가로 입력한 부분입니다. shmget 함수를 이용해 key 값으로 shared memory의 id를 얻어와 shmid에 저장한 뒤, shmat 함수를 통해 공유메모리를 프로세스 메모리 공간에 부착합니다. 그 뒤에 나오는 부분이 사진부분이며, 만약 shmaddr에 아무 문장이 없다면 while문을 반복하다가 wirter에서 문장이 들어오게 되면 그 문장을 출력하고 shmaddr을 초기화해준 뒤에 다시 문장이 들어올 떄때까지 기다립니다. 그 뒤에 end 문장이 들어와 프로그램을 종료시키고자 하면, shmdt 함수를 통해 공유메모리를 프로세스 메모리 공간에서 떼어낸 뒤에 제거하는 작업을 거쳐 프로그램을 완전히 종료합니다.

****

Writer 또한 shmget 함수를 통해 공유메모리를 생성해준 뒤에, 프로세스 메모리 공간에 공유메모리를 부착합니다. 그 후, while문을 통해 end라는 문장이 나오기 전까지 문장을 입력받으며 shmaddr에 저장합니다. 그 뒤에 종료를 위해 공유메모리를 떼어내는 작업을 하여 종료합니다.

1. **IPC 메커니즘 설명**

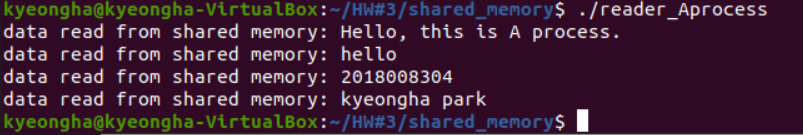
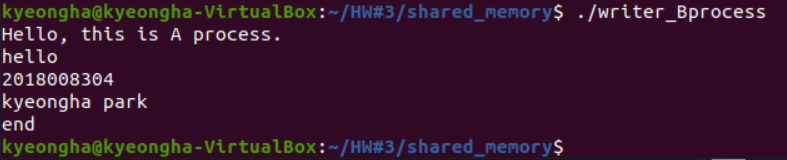
Shared memory는 여러 프로세스가 공동으로 사용하는 공유메모리를 만들어 데이터를 공유하게 되는 원리입니다. 따라서 shared memory을 만들어준 뒤, 그 메모리를 자신의 프로세스 메모리에 부착하여 연결한 후에 데이터를 집어 넣거나 가져오게 됩니다. 공유메모리 또한 같이 혼용하여 사용하는 경우를 막아야하기 때문에 lock이 굉장히 중요한 역할을 하게 됩니다.

1. **컴파일 방법 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

Makefile은 위와 같으며, writer와 reader 프로그램을 각각 만들어 컴파일하였습니다.

****

이는 실행결과로, end 문장이 나오기 전까지 문장을 받으면서 공유메모리에 저장하고 출력을 반복합니다..

1. **과제 B**
2. **멀티스레딩 내용**

스레드는 한 마디로 lightweight process라고 할 수 있으며, 운영체제가 제공하는 스케쥴링의 단위이다. 멀티스레드는 한 프로세스에 여러 개의 스레드가 존재하는 것을 의미하는데, User Address space를 보면 stack의 영역을 제외한 나머지는 모두 공유하여 사용한다. 즉, 스레드는 프로세스 내부에 존재하지만 각각 독립적인 stack 공간을 갖기 때문에 메모리 구성 또한 달라지게 된다.

스레드의 장점은 아래와 같이 정리할 수 있다.

1) Responsiveness(응답성)

CPU작업과 I/O작업의 성질을 봐야할 필요가 있는데, 시스템에 프로세스가 하나 있다고 하면, 다른 프로세스가 없기 때문에 run, waiting 상태만을 왔다 갔다 할 것이다.

(running상태와 waiting 상태의 literbal을 비교하면 waiting이 훨씬 길다, I/O을 받아오는 데에 시간이 걸림)

CPU가 작업하는 구간은 running 구간일텐데, 그 프로그램을 멀티스레드로 구현하면 wait구간에 CPU가 노는 것이 아니라 새로운 스레드를 실행할 여지가 있어지게 된다. -> 성능개선!

2) Resource sharing

스레드는 다 공유하기 때문에 별개를 fork()로 만드는 것보다 그냥 함수 호출 수준에서 스레드를 만드는 것이 경제적이다.

3) Economy

특히 전역변수를 공유하기 때문에 쓰레드들 간의 통신비용이 굉장히 적어질 수 있다. (프로세스 간의 정보는

massage를 주고 받아야하니까 상대적으로 비경제적)

4) Utilization of multiprocessor architectures

CPU가 8개인데 프로세스가 1개이면 나머지 CPU는 놀고 있을텐데 스레드를 7개 만들면 CPU을 사용할 수 있

기 때문에 멀티프로세서를 효율적으로 사용이 가능해진다.

5) Concurrent programming model

스레드를 사용하게 되면 동시에 발생하는 작업들을 만드는 것이 쉬워지며, 응용할 때 유익해진다.

멀티스레딩의 모델은 유저스레드와 커널스레드로 나눌 수 있으며, 유저레벨의 스레드는 커널레벨에 비해 빠르다는 장점이 있으나 1개의 스레드라도 waiting 상태에 들어가면 모든 스레드가 스케쥴링에서 제외된다는 단점이 있다.

한편, 커널레벨의 스레드는 모든 행위들이 커널에서 이루어지며 위와 같은 유저레벨 스레드의 단점을 해결할 수 있다.