실습 과제6

사이버보안학과

201620641

유 상 정

1. 과제 소개
2. 서버에 접속한 클라이언트의 호스트명, IP, port 정보를 확인후 client에게 보내는 server & client 프로그램 작성
3. 다양한 IPC 기법중, Shared memory를 통한 문자열 입출력 프로그램 작성
4. b에서 작성한 프로그램에서, 세마포어를 사용해 동기화를 하여 각 프로그램이 지정한 문자열을 정확히 출력하는 프로그램 작성
5. 실행화면 설명

2-1. 서버에 접속한 클라이언트 정보 보내주기

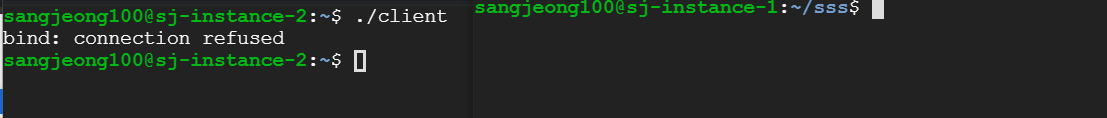
<환경 – sj-instance-1: server, sj-instance-2: client로 사용, Goggle Cloud Platform 이용>

Server -> OS: Ubuntu 18.04, ip: 10.178.0.3

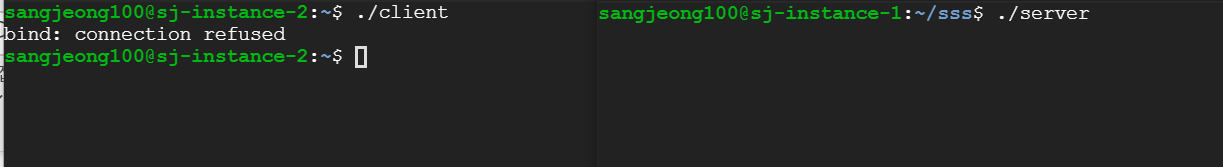
Client -> OS: Ubuntu 19.04, ip: 10.178.0.2



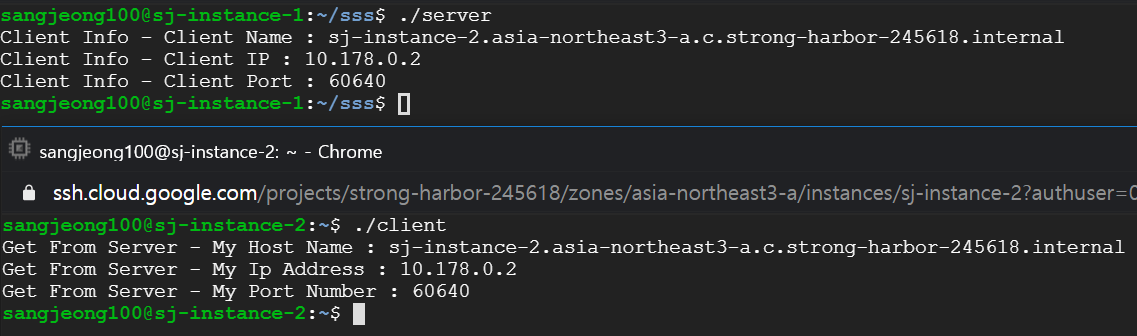
1. Server 작동 X, client 실행



1. server 작동



1. server 작동 후, client 실행, 그 후의 server와 client 화면

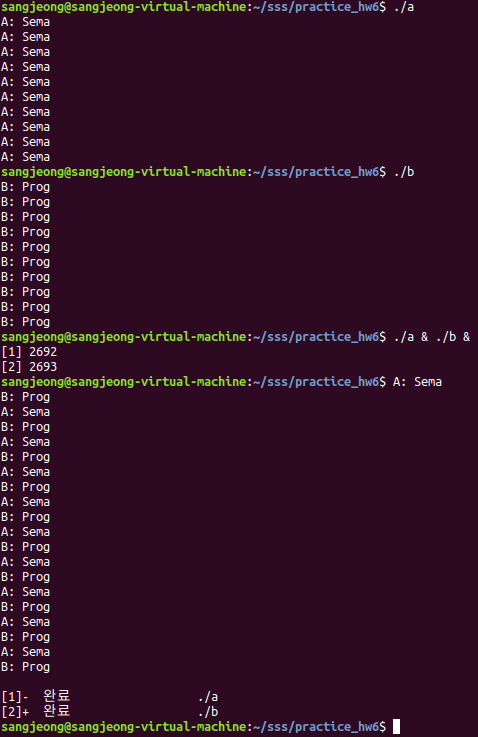
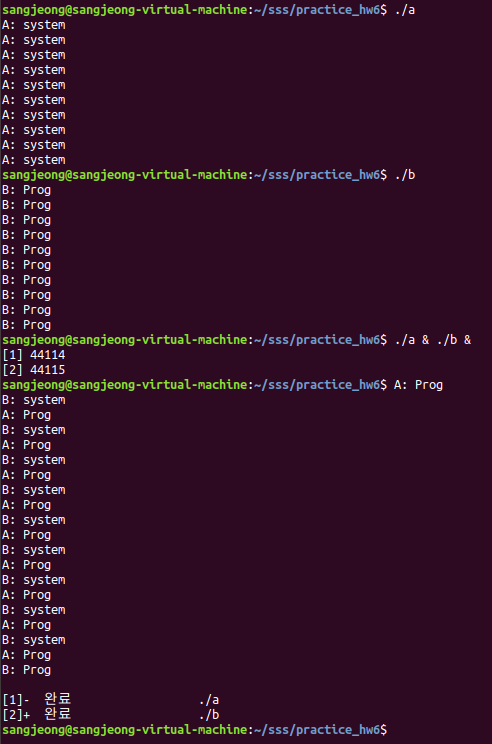


1번 코드는 server와 client 사이의 socket 통신 프로그래밍이다. server와 client는 socket을 생성하고, server는 client의 연결을 기다린다. 그리고 server와 client사이의 연결이 성공하게 되면, server는 client의 정보를 print하고, client에게 client의 host name, Ip address, Port Number를 알려준다. 그리고 server는 client와의 통신을 끊고 server 스스로도 종료한다. 이때, 데이터를 넘기고 받는 함수는 read, write 함수를 사용한다.

2-2. 프로세스간 통신 - 공유메모리

<환경 – VirtualMachine ubuntu 18.04>

<왼쪽 – semaphore 사용 X, 오른쪽 – semaphore 사용 O>



위의 실행화면은 share memory를 A process와 B process가 접근하는 경우, 동기화의 유무에 따라서 출력결과를 나타낸 것이다. 왼쪽에서 A process는 공유 메모리에 “system”을 쓰고, 1초뒤에 “system”을 읽는 process이고, B process는 공유 메모리에 “Prog”를 쓰고 1초뒤에 “Prog”를 읽는 process이다. 그런데, A, B를 동시에 실행하였을 때, 공유 메모리 접근에 대한 동기화 처리를 하지 않아서, A가 읽을 것은 B가 읽거나 B가 읽을 것을 A가 읽는 경우가 발생한다. 즉, 이러한 상황을 막기위해서 동기화 처리를 하여야 한다.

오른쪽의 실행화면을 보면, 동기화가 없는 경우와 실행 메커니즘은 똑같다. 다만 차이점은 semaphore를 사용하여 공유 메모리 접근에 대한 동기화 처리를 한 것이다. 위에서 보이는 것과 같이, A, B process를 동시에 수행할 때, 잘못 읽는 경우 없이, A는 “Sema”, B는 “Prog”를 출력함을 알 수 있다. 이 실습을 통해, 동기화의 중요성을 깨달았다.