lec

포트#는 10000이상의 랜덤 숫자를 사용하시오

10000이하는 시스템 포트

포트넘버는 서버만 사용자가 지정합니다. 서버가 그 포트 넘버에서 클라이언트를 기다리도록 하기 위해서입니다. 클라이언트 프로그램에서 포트넘버는 클라이언트의 포트가 아니고 이 클라이언트가 연결하려는 서버의 포트/ip 주소입니다. 따라서 클라이언트에 있는 서버 포트/ip 주소는 서버에 있는 서버 포트/ip 주소와 같습니다. 클라이언트는 서버 포트/ip 주소를 변경함으로써 다른 서버에 접속할 수 있습니다 (숙제 2-1, 2-2 참조). 클라이언트 자신의 포트/ip 주소는 따로 지정하지 않아도 시스템이 자동으로 임의의 포트번호를 지정해주기 때문에 프로그램에는 나타나지 않습니다(ip 주소도 클라이언트가 있는 컴퓨터의 ip 주소를 시스템이 자동으로 사용).

hw

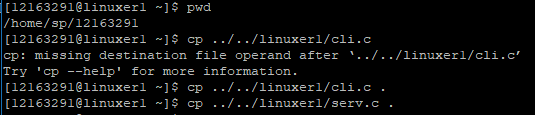
0) Copy cli.c and serv.c from linuxer1(151 server) or linnuxer2(152 server) using cp command as follows. Do not type by yourself the client and server code in this lecture note which is not accurate.

$ pwd

/home/sp1/12345

$ cp ../../linuxer1/cli.c **.**

$ cp ../../linuxer1/serv.c **.**



1) Adjust port and IP address for both cli.c and serv.c. SERV\_ADDR in cli.c and serv.c should be the IP address of the linux server you are using. cli.c will use this SERV\_ADDR to access the server while serv.c will use this SERV\_ADDR to set its own IP address. Pick a port number in the range of [10000..65535]. You need two putty windows: one for the server and the other for the client.

Run the server first and then the client.

(\* Sometimes, you have binding failure when running the server. It happens because the server port number is blocked temporarily. Wait for 10 seconds and retry or use a different port number.)

서버를 먼저 실행한 후, -listen상태의 server

클라이언트를 실행시켜 서버에 연결합니다.

|  |  |
| --- | --- |
|  |  |
| client | Server |

두 프로그램이 잘 소통하는 것을 확인할 수 있었습니다.

하지만, 맨앞의 한 단어만 읽어오는 것을 확인할 수 있었습니다.

1-1) Modify cli.c and serv.c such that they can talk in sentence (not just in word as in the current implementation).

문장을 받게 하기 위해, cli.c 와 serv.c의 **scanf**()로 문장을 받는부분을 모두

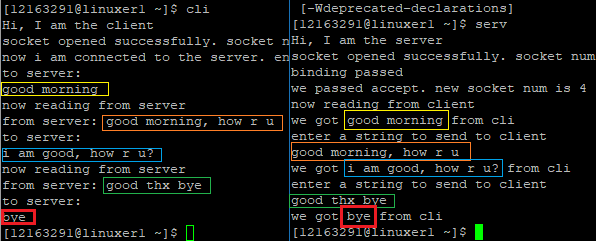
**gets**(buf) 로 **변경해주었습니다**.

|  |  |
| --- | --- |
|  |  |
| client | Server |

2) Modify cli.c and serv.c such that they can keep talking until the client sends "bye". **Use a finite loop.**

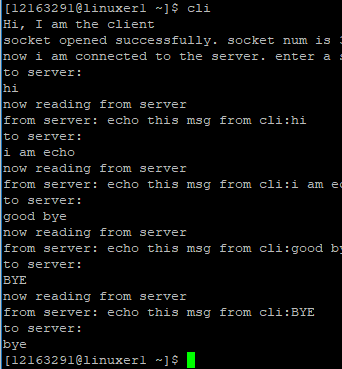
\* serv에서 cli에서 받은게 bye면 인것만 처리하는게 아니고,

client에서 serv로 보내려고 하는게 bye인것도 처리해야 양쪽에서 끝이난다 !

client (좌측)에서 serv로 bye를 보내니 양쪽의 연결이 끊기며 프로그램이 종료되는 것을 확인할 수 있었습니다.

2-1) Try to talk with the other student. Note that one of you would be the client and the other the server. SERV\_ADDR and SERV\_TCP\_PORT in cli.c should match to those in serv.c of the other student.

(If you prefer to work alone, modify your cli.c such that it connects to 165.246.38.136 with port number 19924. The server program at 165.246.38.136:19924 will simply echo whatever message you send. You stop the chatting with "bye".)

**fd**

저는 165.246.38.136 주소의 port# 19924를 사용하였습니다.

cli.c 에서



로 포트번호와 접속 서버 주소를 변경한 후, 실행하였고,

앞서 설명된 것과 같이. echo를 진행하다

bye를 입력하니 프로그램이 끝나는 것을 확인할 수 있습니다.

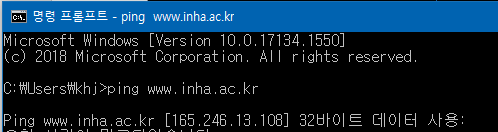
2-2) Modify cli.c such that it connects to Inha web server and read the web page. Inha web server domain name is www.inha.ac.kr and port number is 80.

You can find the IP address for www.inha.ac.kr with ping command.

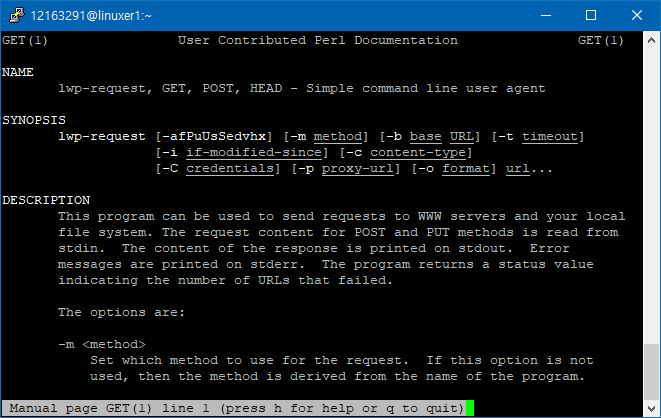
To receive the web page from a web server use GET command:

GET / HTTP/1.1\r\nHOST: www.inha.ac.kr\r\n\r\n

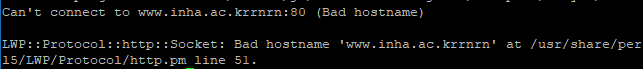
where \r is <CR> character and \n is <LF> character.



cmd창에서 ping 명령어를 통해 인하대의 IP주소를 알아내었습니다.



man GET으로 대략적 설명을 읽어보았습니다.

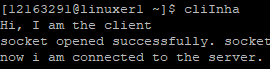


원래는

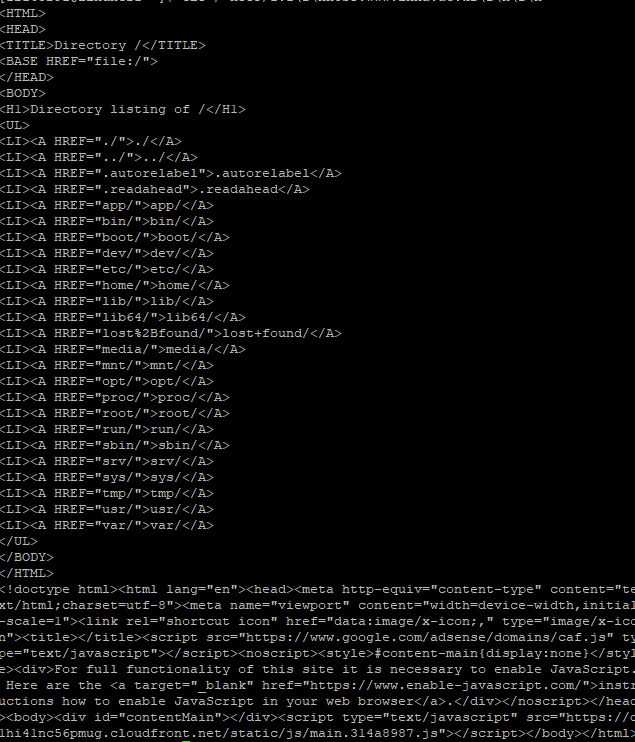
GET함수를 사용하여, inha 사이트의 정보를 읽어오려면 그것이 불가능 했습니다.



cli의 연결 포트번호와, 연결 주소를 위와같이 변경하였습니다.



이후, putty의 한 창에서 인하대 서버와 연결을 시켜줍니다.



GET으로 얻은 결과입니다.

사실 이 문제를 제대로 이해하지 못한 것 같습니다, 죄송합니다

3) Modify the code such that the client and the server can talk in any order.

**Use a finite loop to avoid infinite number of processes.**

fork를 사용하여 문자열 보내는 부분이랑 받는부분 둘다 돌아가게

양방향 통신은 되는데 끝나는게 안됨

#include <signal.h>

kill(pid[i],SIGINT); //pid 부분에 getpid() 가 아닌 getppid()를 입력해야 합니다. !

kill이 들어가는 부분은 fork()==0 인 부분으로 child process이기 때문입니다.

1) 서로 원할 떄 순서 상관없이 소통하는 방법

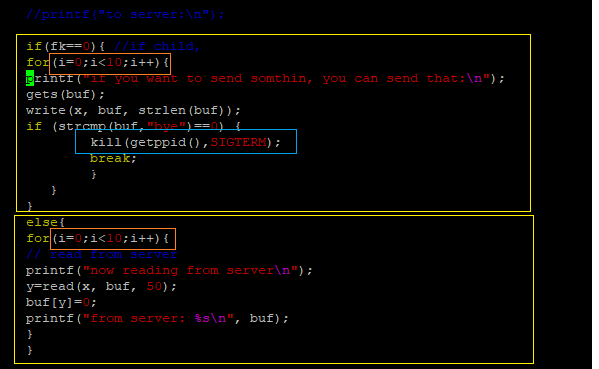
fork()를 사용하여, 두개의 proc를 돌리며 각각 듣고, 보내기를 할 수 있도록 합니다.

2) terminate하는 방법

원래 하던대로, bye의 입력을 terminate의 trigger로 사용하겠습니다.

kill() 함수와, SIGINT x SIGTER o 를 사용하여 bye가 들어오면 getppid를 받아 (child의 parent process의 pid를 kill 하기위해) parent process를 kill해주면,

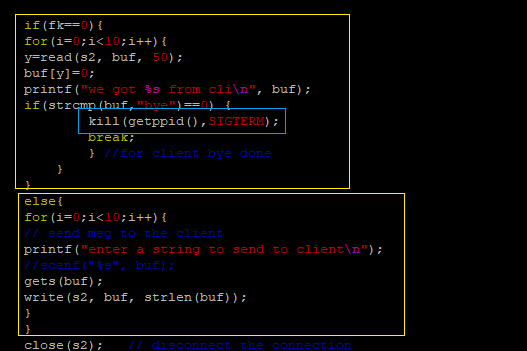
정상적으로 프로세스를 terminate 할 수 있습니다.



client코드입니다.

각 보내는부분, 받는 부분을 fork와 if문으로 나눠 동시에 실행시켜주었고,

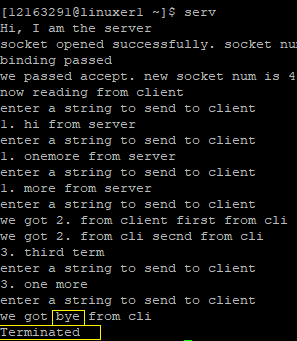
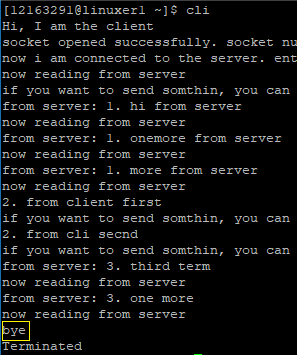
kill 함수를 이용하여 bye로 process를 terminate할 수있습니다.



server 코드입니다.

설명은 client부분과 동일합니다

그 실행결과는 다음과 같습니다.



서로 원할 때, 순서 상관없이 번갈아 가지않고

무엇인가를 보낼 수 있는 것을 확인하였습니다.

그리고, bye로 프로그램이 끝나는 것또한 확인할 수 있었습니다.