

lect11 HW

시스템프로그래밍 1 분반 12180626 성시열

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

```
[12180626@linuxer1 ~]$ cp ../../linuxer1/cli.c .
[12180626@linuxer1 ~]$ cp ../../linuxer1/serv.c .
[12180626@linuxer1 ~]$ ls
cli.c      ex11.c  ex33.c  ex6.c  f2      fabc     mycp.c   path
cphw4     ex13    ex3.c   ex7    f3      hw33    myecho   path.c
cphw4.c   ex13.   ex4     ex7.c  f4      hw4     myecho.c serv.c
d1        ex13.c  ex44    ex8     f5      hw4.c   myexec   sw2.wav
d2        ex1.c   ex444   ex.8    f6      mycat   myexec.c swvader03.wav
d3        ex2     ex444.c ex8.c   f7      mycat2  mysh     x
echo      ex22    ex44.c  ex9     f8      mycat2.c mysh.c   y.c
ex0       ex22.c  ex4.c   ex9.c   f9      mycat3  myxxd
ex0.c     ex2.c   ex5     f1      f91     mycat3.c myxxd.c
ex1       ex3     ex5.c   f1.c    f92     mycat.c  newhw4.c
ex11      ex33    ex6     flout   f93?    mycp     outfiles
```

```
$ cp ../../linuxer1/cli.c .
```

```
$ cp ../../linuxer1/serv.c .
```

코드를 통해 강의노트에 있는 코드를 타이핑 하지 않고 시스템 내에 있는 client와 server 프로그램을 나의 로그인 디렉토리에 복사하였다.

ls를 통해 cli.c 와 serv.c 파일이 복사되어 생성됨을 알 수 있었다.

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.

```
[12180626@linuxer1 ~]$ vi cli.c
[12180626@linuxer1 ~]$ vi serv.c
```

로그인 디렉토리로 복사한 cil.c와 serv.c 파일을 vi를 활용해 port 번호를 수정한다.

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>

#define SERV_TCP_PORT 62608
#define SERV_ADDR "165.246.38.151"

void main(){
    int x,y;
    struct sockaddr_in serv_addr;
    char buf[50];
    printf("Hi, I am the client\n");
}

```

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>

#define SERV_TCP_PORT 62608
#define SERV_ADDR "165.246.38.151"

void main(){
    int s1,s2, x, y;
    struct sockaddr_in serv_addr, cli_addr;
    char buf[50];
    socklen_t xx;

    printf("Hi, I am the server\n");
}

```

port 번호를 10000부터 65535 사이의 숫자 중 하나인 62608로 하고, client의 port 번호와 server의 port 번호를 동일하게 하여, 내 컴퓨터에서 client와 server 역할을 동시에 하게 한다.

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

```

[12180626@linuxer1 ~]$ gcc -o cli cli.c
[12180626@linuxer1 ~]$ gcc -o serv serv.c

```

cli와 serv 두 파일을 모두 컴파일 하였다.

```

[12180626@linuxer1 ~]$ serv
Hi, I am the server
socket opened successfully. socket num is 3
binding passed

```

serv를 실행시켜본 결과 해당 문장이 출력되었고, client를 기다리는 모습이다.

```
[12180626@linuxer1 ~]$ serv
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
now reading from client

12180626@linuxer1:~
login as: 12180626
12180626@165.246.38.151's password:
Last login: Sun Jul 3 15:57:11 2022 from 183.91.239.24
[12180626@linuxer1 ~]$ cli
Hi, I am the client
socket opened successfully. socket num is 3
now i am connected to the server. enter a string to send
```

새로운 putty 창을 열어 로그인한 뒤 cli를 실행시켜 보았다. 해당 문장이 출력됨과 동시에 serv 쪽 putty 창에서도 새로운 문장이 출력된 것을 알 수 있다.

```
now i am connected to the server. enter a string to send
Hello
```

client 쪽에서 Hello 라는 메시지를 보내보았다.

```
we got Hello from cli
enter a string to send to client
Hi
```

server 쪽에서 Hello 라는 메시지를 받고, client에 보낼 메시지를 입력하는 입력창에 Hi라는 메시지를 입력하였다.

```
now reading from server
from server: Hi
```

그 결과 client 쪽에서 Hi라는 메시지를 받은 것을 확인할 수 있다.
이후 프로그램이 종료된다.

```
Hi, I am the client
socket opened successfully. socket num is 3
now i am connected to the server. enter a string to send
Heloo my name is ssy.
```

메시지를 문장으로 보냈을 때,

```
now reading from client
we got Heloo from cli
```

첫 단어만 넘어온 것을 알 수 있다.

vi cli.c와 vi serv.c를 통해 입력을 scanf가 아닌 gets로 받는다.

```
[12180626@linuxer1 ~]$ vi cli.c
[12180626@linuxer1 ~]$ gcc -o cli cli.c
cli.c: In function 'main':
cli.c:37:4: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:640)
[-Wdeprecated-declarations]
[12180626@linuxer1 ~]$ cli
Hi, I am the client
socket opened successfully. socket num is 3
now i am connected to the server. enter a string to send
Hello my name is ssy.
now reading from server
from server: Hi my name is ssy.
```

```
[12180626@linuxer1 ~]$ vi serv.c
[12180626@linuxer1 ~]$ gcc -o serv serv.c
serv.c: In function 'main':
serv.c:51:4: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:640)
[-Wdeprecated-declarations]
[12180626@linuxer1 ~]$ serv
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
now reading from client
we got Hello my name is ssy. from cli
enter a string to send to client
Hi my name is ssy.
```

컴파일 후 실행해본 결과 문장을 주고받을 수 있음을 확인할 수 있다.

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

client가 bye라는 문자열을 보내기 전까지 유한루프를 통해 메시지를 주고받는 프로그램을 만든다. cli.c 코드수정을 위해 코드를 파악해보았다.

```
printf("now i am connected to the server. enter a string to send\n");
scanf("%s",buf);
write(x, buf, strlen(buf));
// read from server
printf("now reading from server\n");
y=read(x, buf, 50);
buf[y]=0;
printf("from server: %s\n", buf);
close(x); // disconnect the communication
```

해당코드 이 외에는 라이브러리 선언, 오류 처리, 변수 선언 등의 코드임을 알 수 있다.

```
int i;
for(i=0; i<20; i++){
    printf("now i am connected to the server. enter a string to send\n");
    gets(buf);
    write(x, buf, strlen(buf));
    if(strcmp(buf,"bye")==0) break;
    // read from server
    printf("now reading from server\n");
    y=read(x, buf, 50);
    buf[y]=0;
    printf("from server: %s\n", buf);
}
close(x); // disconnect the communication
```

위 코드를 20번 반복하는 for문을 작성하고 조건문과 strcmp를 통해 buf에 "bye"가 들어갈 경우 반복문을 탈출한다.

```

// read msg from client
int i;
for(i=0; i<20; i++){
    printf("now reading from client\n");
    y=read(s2, buf, 50);
    buf[y]=0;
    printf("we got %s from cli\n", buf);
    if(strcmp(buf, "bye")==0) break;
    // send msg to the client
    printf("enter a string to send to client\n");
    gets(buf);
    write(s2, buf, strlen(buf));
    if(strcmp(buf, "bye")==0) break;
}
close(s2); // disconnect the connection
close(s1); // close the original socket

```

serv.c 코드에서도 buf에 bye라는 코드를 받아 read했을 때 반복문을 탈출한다.

```

[12180626@linuxer1 ~]$ cli
Hi, I am the client
socket opened successfully. socket num is 3
now i am connected to the server. enter a string to send
a
now reading from server
from server: b
now i am connected to the server. enter a string to send
c
now reading from server
from server: d
now i am connected to the server. enter a string to send
e
now reading from server
from server: f
now i am connected to the server. enter a string to send
bye
[12180626@linuxer1 ~]$

```

```

[12180626@linuxer1 ~]$ serv
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
now reading from client
we got a from cli
enter a string to send to client
b
now reading from client
we got c from cli
enter a string to send to client
d
now reading from client
we got e from cli
enter a string to send to client
f
now reading from client
we got bye from cli
[12180626@linuxer1 ~]$

```

컴파일 한 뒤 메시지를 주고받아본 결과 bye를 입력했을 때 프로그램을 종료한 것을 확인하였다.

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".)

현재 ip주소에 165.246.38.136, port 번호 19924에 원격으로 서버가 연결되어있다.

```
#define SERV_TCP_PORT 19924
#define SERV_ADDR "165.246.38.136"
```

그러므로 vi cli.c를 통해 client의 ip주소와 port 번호를 수정해주었다.

```
[12180626@linuxer1 ~]$ ./cli
Hi, I am the client
socket opened successfully. socket num is 3
Hi
echo this msg from cli:Hi
Hello
echo this msg from cli:Hello
How are you?
echo this msg from cli:How are you?
bye
GOODBYE USER
```

cli 프로그램을 컴파일 후 실행한 뒤 문자열을 전송하였을 때, 값이 잘 전송되는 것을 알 수 있다. bye를 전송했을 때는 내가 작성한 코드에 의해 GOODBYE USER를 출력하고 종료되는 것을 확인하였다.

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 (your code should send below string automatically using write() function – do not type it by hand):.

```
[12180626@linuxer1 ~]$ ping www.inha.ac.kr
PING www.inha.ac.kr (165.246.13.107) 56(84) bytes of data.
```

\$ ping www.inha.ac.kr를 통해 ip주소와 데이트 크기를 알 수 있다. port 번호는 80이다.

```
#define SERV_TCP_PORT 80
#define SERV_ADDR "165.246.13.107"
```

vi cli.c를 통해 client의 port 번호와 ip 주소를 변경한다.

```
printf("now i am connected to the server. enter a string to send\n");
strcpy(buf, "GET / HTTP/1.1\r\nHOST:www.inha.ac.kr\r\n\r\n");
write(x, buf, strlen(buf));
```

이후 클라이언트는 buf에

GET / HTTP/1.1WrWnHOST:www.inha.ac.krWrWnWrWn

위 문자열을 넣어줘야 하므로 gets() 대신 strcpy()를 사용한다.


```

now i am connected to the server. enter a string to send
now reading from server
from server:

<!doctype html>
<html lang="en">
  <head>
now i am connected to the server. enter a string to send
now reading from server
from server:
  <meta charset="utf-8">
  <meta name="title" co
now i am connected to the server. enter a string to send
now reading from server
from server: ntent="인하대학교">
  <meta name="keywords"
now i am connected to the server. enter a string to send
now reading from server
from server: content="인하, 인하대, 인하대학교, INHA,
now i am connected to the server. enter a string to send
now reading from server
from server: INHA UNIVERSITY, 인하공대, 공대, 공과대
now i am connected to the server. enter a string to send
now reading from server
from server: , 인하대병원, 사립대, 4년제, 仁河大
now i am connected to the server. enter a string to send
now reading from server
from server: , 인천, 하와이, 입학, 입시, 입학정보,
now i am connected to the server. enter a string to send
now reading from server
from server: 주안역, 정석학술정보관, 도서관, 인
now i am connected to the server. enter a string to send
now reading from server
from server: 대후문, 인하홍보동영상">
  <meta nam
now i am connected to the server. enter a string to send
now reading from server
from server: e="description" content="인천 미추홀구 인
now i am connected to the server. enter a string to send
now reading from server
from server: 로 100에 위치한 4년제 사립대학으로,
now i am connected to the server. enter a string to send
now reading from server
from server: 1954년 인하공과대학으로 설립되어 현
now i am connected to the server. enter a string to send
now reading from server
from server: 프론티어 학부대학·공과대학·자

```

컴파일 후 실행한 결과 학교 홈페이지(www.inha.ac.kr)의 서버로부터 메시지가 전달된 것을 알 수 있다.

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.

client 혹은 server 어느쪽에서든 먼저 채팅을 칠 수 있게 수정하고 무한의 process 수를 피하는 제한된 루프를 사용해야한다.

먼저 이전 문제에서 변경하였던 ip 주소, port 번호, strcpy를 이전과 같이 수정한다.

```
int k;
k = fork();
int i;
for(i=0; i<20; i++){
    // read msg from client
    if(k==0){
        printf("now reading from client\n");
        y=read(s2, buf, 50);
        buf[y]=0;
        printf("we got %s from cli\n", buf);
        if(strcmp(buf, "bye")==0) break;
    }
    // send msg to the client
    else{
        printf("enter a string to send to client\n");
        gets(buf);
        write(s2, buf, strlen(buf));
        if(strcmp(buf, "bye")==0) break;
    }
}
close(s2); // disconnect the connection
close(s1); // close the original socket
```

```
int k;
k = fork();
int i;
for(i=0; i<20; i++){
    if(k==0){
        // send msg to the server
        printf("now i am connected to the server. enter a string to send\n");
        gets(buf);
        write(x, buf, strlen(buf));
        if(strcmp(buf, "bye")==0) break;
    }
    else{
        // read from server
        printf("now reading from server\n");
        y=read(x, buf, 50);
        buf[y]=0;
        printf("from server: %s\n", buf);
        if(strcmp(buf, "bye")==0) break;
    }
}
```

기존에는 아무런 조건 없이 모두 읽고 쓰기를 진행하였다면

이제는 fork()를 통해 파일을 복제하고,

server를 기준으로 파일이 자식 프로세스를 진행할 때 채팅 출력을, 부모 프로세스를 진행할 때 채팅 입력을 하게 한다.

client를 기준으로 파일이 자식 프로세스를 진행할 때 채팅 입력을, 부모 프로세스를 진행할 때 채팅 출력을 하게 한다.

추가로 "bye"라는 문자열이 입력되었을 때 처리를 수정해야한다.

단순히 break가 아닌 fork()로 인해 복제된 process들을 kill() 및 exit() 등으로 종료시켜야한다. 자식 프로세스에서는 실행되고 있는 프로세스의 부모 프로세스 ID를 return하는 getppid()를 활용하여 kill 함수를 사용한다.

부모 프로세스에서는 k에 자식의 프로세스가 return 되므로 kill의 argumentdp k를 넣어준다.

```
int k;
k = fork();
int i;
for(i=0; i<20; i++){
    // read msg from client
    if(k==0){
        printf("new reading from client\n");
        y=read(s2, buf, 50);
        buf[y]=0;
        if(strcmp(buf, "bye")==0){
            printf("GOODBYE USER\n");
            kill(getppid(), 15);
            close(s2);
            close(s1);
            exit(1);
        }
        else{
            printf("we got %s from cli\n", buf);
        }
    }
    // send msg to the client
    else{
        printf("enter a string to send to client\n");
        gets(buf);
        write(s2, buf, strlen(buf));
        if(strcmp(buf, "bye")==0){
            printf("GOODBYE USER\n");
            kill(k, 15);
            close(s2);
            close(s1);
            exit(1);
        }
    }
}
```

server 입장에서 자식 프로세스 중, 조건문을 활용하여 넘겨받은 값이 "bye"일 때 "GOODBYE USER"를 출력하고 부모 프로세스를 kill을 통해 정지시키고 exit()로 종료시키기 전에 close(s2), close(s1)을 진행한다. 그렇지 않을 때는 상대방에게 받은 문자열을 출력한다. server 입장에서 부모 프로세스 중, 조건문을 활용하여 gets()를 통해 입력한 값이 "bye"일 때 "GOODBYE USER"를 출력하고 자식 프로세스를 kill을 통해 정지시키고 exit()로 종료시키기 전에 close(s2), close(s1)을 진행한다. 그렇지 않을 경우는 따로 추가할 필요가 없다.

```

int k;
k = fork();
int i;
for(i=0; i<20; i++){
    if(k==0){
        // send msg to the server
        printf("now i am connected to the server. enter a string to send\n");
        gets(buf);
        write(x, buf, strlen(buf));
        if(strcmp(buf, "bye")==0){
            printf("GOODBYE USER\n");
            kill(getppid(), 15);
            close(x);
            exit(1);
        }
    }
    else{
        // read from server
        printf("now reading from server\n");
        y=read(x, buf, 50);
        buf[y]=0;
        if(strcmp(buf, "bye")==0){
            printf("GOODBYE USER\n");
            kill(k, 15);
            close(x);
            exit(1);
        }
        else{
            printf("from server: %s\n", buf);
        }
    }
}
close(x); // disconnect the communication

```

client 입장에서도 비슷하다.

자식 프로세스는 getppid()를 통해 부모 프로세스를 kill하고 중지한다.

부모 프로세스는 k를 통해 자식 프로세스를 kill하고 중지한다.

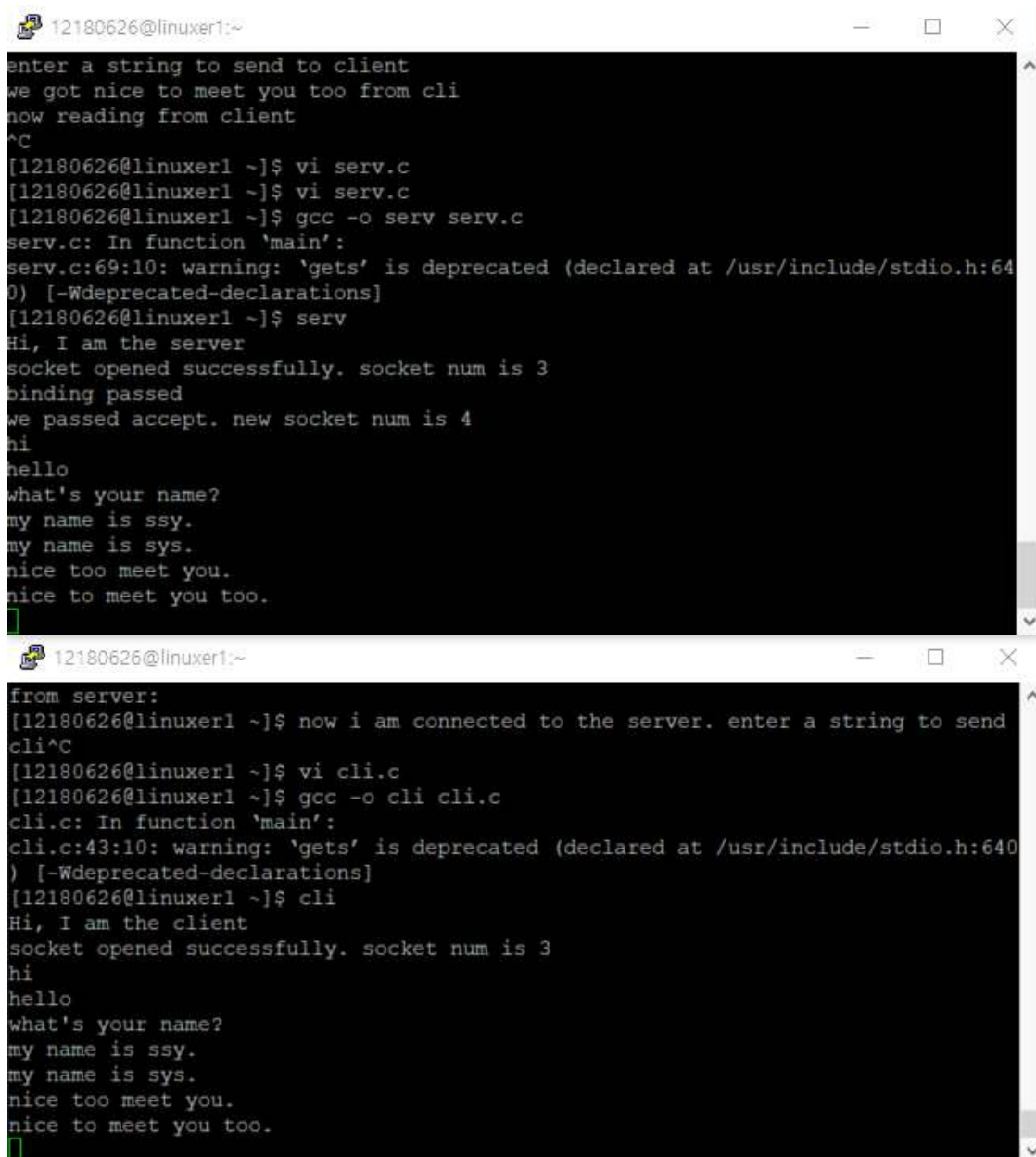
server의 자식 프로세스의 역할을, client의 부모 프로세스가

server의 부모 프로세스의 역할을, client의 자식 프로세스가 하고 있으므로 참고하여 코드를 작성한다.

```
12180626@linuxer1:~  
[12180626@linuxer1 ~]$ serv  
Hi, I am the server  
socket opened successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
enter a string to send to client  
now reading from client  
Hi  
enter a string to send to client  
we got Hello from cli  
now reading from client  
my name is ssy.  
enter a string to send to client  
we got nice to meet you. from cli  
now reading from client  
nice to meet you too.  
enter a string to send to client  
GOODBYE USER  
Terminated  
[12180626@linuxer1 ~]$ vi serv.c  
[12180626@linuxer1 ~]$   
  
12180626@linuxer1:~  
[12180626@linuxer1 ~]$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
now reading from server  
now i am connected to the server. enter a string to send  
from server: Hi  
now reading from server  
Hello  
now i am connected to the server. enter a string to send  
from server: my name is ssy.  
now reading from server  
nice to meet you.  
now i am connected to the server. enter a string to send  
from server: nice to meet you too.  
now reading from server  
bye  
GOODBYE USER  
Terminated  
[12180626@linuxer1 ~]$
```

컴파일 후 실행해본 결과, server 측에서 먼저 메시지를 보내도 서로 채팅이 가능하였고, bye를 쳤을 때, 양측에서 모두 채팅이 종료된 것을 확인하였다.

단, fork()에서 scheduler에 의해 프로세스가 실행되는데,
코드를 뜯어본 결과 serv.c에서는 부모 프로세스가 cli.c에서는 자식 프로세스가 먼저 실행된다. 코드는 이에 맞게 적절한 사용자의 입력을 유도하려고 printf()를 사용하여 출력문을 작성하였지만, 순서없이 서로 채팅을 할 때 어떤 프로세스가 먼저 실행될지 예측되지 않아 출력문을 상황에 맞게 출력하지 못하였다.
이에 사용자의 입력을 유도하는 printf()문을 모두 생략하고 입력받은 값만 출력하도록 코드를 수정하였다.



```
12180626@linuxer1:~  
enter a string to send to client  
we got nice to meet you too from cli  
now reading from client  
^C  
[12180626@linuxer1 ~]$ vi serv.c  
[12180626@linuxer1 ~]$ vi serv.c  
[12180626@linuxer1 ~]$ gcc -o serv serv.c  
serv.c: In function 'main':  
serv.c:69:10: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:640) [-Wdeprecated-declarations]  
[12180626@linuxer1 ~]$ serv  
Hi, I am the server  
socket opened successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
hi  
hello  
what's your name?  
my name is ssy.  
my name is sys.  
nice too meet you.  
nice to meet you too.  
[12180626@linuxer1 ~]$  
from server:  
[12180626@linuxer1 ~]$ now i am connected to the server. enter a string to send  
cli^C  
[12180626@linuxer1 ~]$ vi cli.c  
[12180626@linuxer1 ~]$ gcc -o cli cli.c  
cli.c: In function 'main':  
cli.c:43:10: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:640) [-Wdeprecated-declarations]  
[12180626@linuxer1 ~]$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
hi  
hello  
what's your name?  
my name is ssy.  
my name is sys.  
nice too meet you.  
nice to meet you too.  
[12180626@linuxer1 ~]$
```

컴파일 후 실행한 결과 실제 채팅처럼 서로 문자열을 주고 받는 것을 확인할 수 있다.