

#### 4. Homework

12201922 이규민

0) Open two Cygwin/terminal windows and type cli.c and serv.c at each window (the code is in Section 1).

[cli.c]

```
#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>
#include <sys/time.h>
#include <unistd.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "165.246.222.68"

int main(){
    int x,y;s
    struct sockaddr_in serv_addr;
    char buf[50];

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

    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = PF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(SERV_ADDR);
    serv_addr.sin_port = htons(SERV_TCP_PORT);
```

```
/* open a tcp socket*/
if ( (x =socket(PF_INET, SOCK_STREAM,0)) < 0){
    printf("socket creation error\n");
    exit(1);
}
printf(" socket opened successfully. socket num is %d\n", x);

/* connect to the server */
if (connect(x, (struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){
    printf("can't connect to the server\n");
    exit(1);
}

/* send input str to the server */
printf("now i am connected to the erver. enter a string to send\n");
scanf("%s", buf);
write(x,buf,strlen(buf));

printf("now let's read from the server\n");
y=read(x,buf,50);
buf[y]=0;
printf("what echoed from the server is %s\n",buf);
close(x); // disconnect
}
```

[serv.c]

```
#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>
#include <sys/time.h>
#include <unistd.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "165.246.222.68"

int 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");
    bzero((char *)&serv_addr, sizeof(serv_addr));
    serv_addr.sin_family=PF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
    serv_addr.sin_port=htons(SERV_TCP_PORT);
```

```
    if((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
        printf("socket creation error\n");
        exit(1);
    }
    printf("socket created successfully. socket num is %d\n", s1);

    x=bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    if (x < 0){
        printf("binding failed\n");
        exit(1);
    }
    printf("binding passed\n");

    listen(s1, 5);
    xx = sizeof(cli_addr);
    s2 = accept(s1,(struct sockaddr *)&cli_addr,&xx);
    printf("we passed accept. new socket num is %d\n", s2);

    y=read(s2,buf,50);
    buf[y]=0;
    printf("we got %s from cli\n",buf);
    printf("what do you want to send to cli? enter your string\n");
    scanf("%s", buf);
    write(s2,buf,strlen(buf));
    close(s2);    // disconnect the connection
    close(s1);    // close the original socket
}
```

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. Use "ifconfig"(in ubuntu) or "ipconfig"(in cygwin) to find out the IP address of your virtual machine or PC. 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 terminals: one for the server and the other for the client. Compile both and 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.)

```
kyumin@DESKTOP-NUDFAPK ~
$ ipconfig

Windows IP 구성

무선 LAN 어댑터 로컬 영역 연결 * 1:

    미디어 상태 . . . . . : 미디어 연결 끊김
    연결 별 DNS 접미사 . . . . . :

무선 LAN 어댑터 로컬 영역 연결 * 14:

    미디어 상태 . . . . . : 미디어 연결 끊김
    연결 별 DNS 접미사 . . . . . :

무선 LAN 어댑터 Wi-Fi 2:

    연결 별 DNS 접미사 . . . . . : inha.ac.kr
    링크-로컬 IPv6 주소 . . . . . : fe80::af77:405f:2ea4:655f%24
    IPv4 주소 . . . . . : 165.246.222.68
    서브넷 마스크 . . . . . : 255.255.255.0
    기본 게이트웨이 . . . . . : 165.246.222.1

이더넷 어댑터 Bluetooth 네트워크 연결 2:

    미디어 상태 . . . . . : 미디어 연결 끊김
    연결 별 DNS 접미사 . . . . . :
```

```
#define SERV_TCP_PORT 34357
#define SERV_ADDR "165.246.222.68"
```

```
#define SERV_TCP_PORT 34357
#define SERV_ADDR "165.246.222.68"
```

cygwing에서 ipconfig 명령어를 통해 나의 ip를 확인한 후 cli.c와 serv.c의 ip를 수정해줬다. 나의 pc가 보내고, 나의 pc가 받아야하므로 두 ip를 동일하게 입력해줬다. port번호의 경우 10000 이하는 시스템의 포트이므로 10000~60000 중 임의로 숫자를 입력했다.

```
kyumin@DESKTOP-NUDFAPK ~  
$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
now i am connected to the erver. enter a string to send  
hello  
now let's read from the server  
what echoed from the server is hi
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
we got hello from cli  
what do you want to send to cli? enter your string  
hi
```

서버가 수신을 하고 있을 때 클라이언트가 정보를 보내야하므로 serv를 먼저 실행한 후 cli를 실행했다. cli에서 hello라는 메시지를 입력했고, serv 창을 보니 hello 메시지가 정상적으로 수신되었다. 그리고 hi라는 메시지를 입력하니 cli에서도 정상적으로 입력을 받았다.

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

[cli.c]

```
/* send input str to the server */
printf("now i am connected to the server. enter a string to send\n");
fgets(buf, 50, stdin);
buf[strlen(buf) - 1] = '\0';
write(x,buf,strlen(buf));

printf("now let's read from the server\n");
y=read(x,buf,50);
buf[y]=0;
printf("what echoed from the server is %s\n",buf);
close(x);    // disconnect
}
```

[serv.c]

```
//socket을 통해 들어오는 연결을 듣는다.
listen(s1, 5);
xx = sizeof(cli_addr);
s2 = accept(s1,(struct sockaddr *)&cli_addr,&xx);
printf("we passed accept. new socket num is %d\n", s2);

y=read(s2,buf,50);
buf[y]=0;
printf("we got %s from cli\n",buf);
printf("what do you want to send to cli? enter your string\n");
fgets(buf, 50, stdin);
buf[strlen(buf) - 1] = '\0';
write(s2,buf,strlen(buf));
close(s2);    // disconnect the connection
close(s1);    // close the original socket
}

"serv.c" 57L, 1542B                                     52,2-5
```

위 두 코드에서 사용자의 입력을 받는 함수인 scanf를 제거하고, fgets를 이용하여 입력을 받았다. Scanf는 단어 단위로 입력을 받으므로 문장을 입력 받지 못한다. 그러나 fgets의 경우 사용자가 엔터를 누를 때까지 입력을 받으므로 문장 전체를 입력 받을 수 있다. Fget는 마지막에 개행문자까지 입력되므로 buf의 마지막 문자를 'W0'으로 수정을해줬다.

[결과]

```
kyumin@DESKTOP-NUDFAPK ~  
$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
now i am connected to the erver. enter a string to send  
hello world  
now let's read from the server  
what echoed from the server is my name is kyumin
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
we got hello world from cli  
what do you want to send to cli? enter your string  
my name is kyumin
```

Serv를 실행한 후 cli를 실행했다. 그리고 cli에서 hello world를 입력하니 serv에서 정상적으로 수신했다는 메시지가 나타난다. 그리고 my name is kyumin을 입력하니 cli에서 정상적으로 수신했다는 메시지가 나타난다.

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

[cli.c]

```
for(int i = 0; i < 10; i++)
{
    /* send input str to the server */
    printf("now i am connected to the server. enter a string to send\n");
    fgets(buf, 50, stdin);
    buf[strlen(buf) - 1] = '\0';
    if(strcmp(buf, "bye") == 0) break;
    write(x,buf,strlen(buf));

    //socket의 메 시 지 출 력
    printf("now let's read from the server\n");
    y=read(x,buf,50);
    buf[y]=0;
    printf("what echoed from the server is %s\n",buf);
}
close(x);    // disconnect
}
```

[serv 결과]

```
kyumin@DESKTOP-NUDFAPK ~
$ serv
Hi, I am the server
socket created successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
we got hello from cli
what do you want to send to cli? enter your string
hi
we got  from cli
what do you want to send to cli? enter your string
```

`if(strcmp(buf, "bye") == 0) break;`코드를 추가하였다. Bye가 입력되면 cli는 for문을 탈출하고, x를 close 한다. 이 때 serv를 보면 화면에 공백이 출력된다.

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, run the client in ubuntu and run the server in cygwin terminal. Make sure the client specify the IP and port number set in the cygwin server.)

---

 165.246.222.68 - PuTTY

```
hello worldmy name is kyumin
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
we got [REDACTED] from cli  
what do you want to send to cli? enter your string  
hello world  
we got my name is kyumin  
from cli  
what do you want to send to cli? enter your string
```

Putty를 실행하고 ip와 port를 serv.c와 동일하게 입력해줬고, Connection type은 Telnet으로 설정 후 실행했다. 그리고 위 사진과 같이 서로 입력한 내용이 출력 되는 것을 볼 수 있다.



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):

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

where \r is a backslash character.

```
for(int i = 0; i < 10; i++)
{
    /* send input str to the server */
    /*
    printf("now i am connected to the server. enter a string to send\n");
    fgets(buf, 50, stdin);
    buf[strlen(buf) - 1] = '\0';
    if(strcmp(buf, "bye") == 0) break;
    write(x,buf,strlen(buf));
    write(x,"GET / HTTP/1.1\r\nHOST:www.inha.ac.kr\r\n\r\n",45);

    //socket의 메 시 지 출 력
    printf("\nnow let's read from the server\n");
    y=read(x,buf,50);
    buf[y]=0;
    printf("what echoed from the server is %s\n",buf);
    */

    write(x,"GET / HTTP/1.1\r\nHOST:www.inha.ac.kr\r\n\r\n",45);
    y=read(x,buf,50);
    buf[y]=0;
    printf("%s\n", buf);
}
close(x); // disconnect
```

```
kyumin@DESKTOP-NUDFAPK ~
$ cli
Hi, I am the client
socket opened successfully. socket num is 3
HTTP/1.1 200 OK
Date: Mon, 13 May 2024 01:46:06 G
MT
Content-Type: text/html; charset=utf-8
Content
-Length: 1875
Connection: keep-alive
Server: Apache
Set-Cookie: JSESSIONID=E80184FDCA4EB39FE3ADF0A9D46CE757; Path=/; HttpOnly
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
```

GET 명령어를 write 함수를 통해서 보내고, 받은 내용을 출력하는 과정을 10번 반복하였다. 그 결과는 위와 같다.

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.

[cli.c]

```
//프로세스 복제
int fp = fork();

for(int i = 0; i < 10; i++)
{
    if(fp != 0)
    {
        /* send input str to the server */
        //printf("now i am connected to the server. enter a string to send\n");
        fgets(buf, 50, stdin);
        buf[strlen(buf) - 1] = '\0';
        if(strcmp(buf, "bye") == 0) break;
        write(x, buf, strlen(buf));
    }
    else
    {
        //socket의 메시지 출력
        //printf("now let's read from the server\n");
        y = read(x, buf, 50);
        buf[y] = 0;
        printf("what echoed from the server is %s\n", buf);
    }
}
close(x); // disconnect
```

기존 코드는 클라이언트와 서버가 한 번씩 말할 수 있었다. 응답을 받아야 다시 메시지를 보낼 수 있었다. Socket의 메시지를 쓰고 읽는 과정을 유한 루프로 만들기 위해서 for문을 10번 반복했는데 for 문에 진입하기 전에 fork함수를 사용하여 프로세스를 복제했다.

그리고 for문에 진입하면 부모 프로세스는 쓰기만 반복하고, 자식 프로세스는 읽기만 반복한다.

[serv.c]

```
//프로세스 복제
int fp = fork();

//socket의 메시지를 읽고 출력
for(int i = 0; i < 10; i++)
{
    if(fp == 0)
    {
        y=read(s2,buf,50);
        buf[y]=0;
        printf("we got %s from cli\n",buf);
    }
    else
    {
        //printf("what do you want to send to cli? enter your string\n");
        fgets(buf, 50, stdin);
        buf[strlen(buf) - 1] = '\0';
        write(s2,buf,strlen(buf));
    }
}
close(s2); // disconnect the connection
close(s1); // close the original socket
```

Serv.c 코드도 동일한 방법으로 수정했다. Socket의 메시지를 쓰고 읽는 과정을 유한 루프로 만들기 위해서 for문을 10번 반복했는데 for 문에 진입하기 전에 fork함수를 사용하여 프로세스를 복제했다.

그리고 for문에 진입하면 자식 프로세스는 쓰기만 반복하고, 부모 프로세스는 읽기만 반복한다.

[결과]

```
kyumin@DESKTOP-NUDFAPK ~  
$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
hello  
world  
what echoed from the server is hi  
what echoed from the server is system programming
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
we got hello from cli  
we got world from cli  
hi  
system programming
```

위 사진이 cli.c 코드를 실행한 화면이고, 아래 사진이 serv.c를 실행한 화면이다.

Cli를 실행했을 때 hello와 world를 입력했고, serv의 화면을 보면 정상적으로 두 단어가 출력된 것을 볼 수 있다.

serv에서도 hi와 system programming이라는 단어를 입력했고, cli 화면에서 정상적으로 두 메시지가 출력되었다.

Client와 server가 한 번씩만 번갈아 가면서 말하지 않고, 한쪽에서 두 번 이상 말할 수 있게 되었다.

## Week 12 homework

3-1) Write a program that reads "login.txt" and answers whether a given name is in this list or not For example, if "login.txt" has

kim

hong

steve

ted

Then you program(ex31) should answer

\$ ex31

enter name: ted

ted is legal user

enter name: kim2

kim2 is not legal user

.....

[ex31.c]

```
void main()
{
    char buf[20], name[20];
    FILE *fp;

    for(int i = 0; i < 10; i++)
    {
        //이름 입력
        printf("enter name: ");
        scanf("%s", name);
        fp = fopen("login.txt", "r");

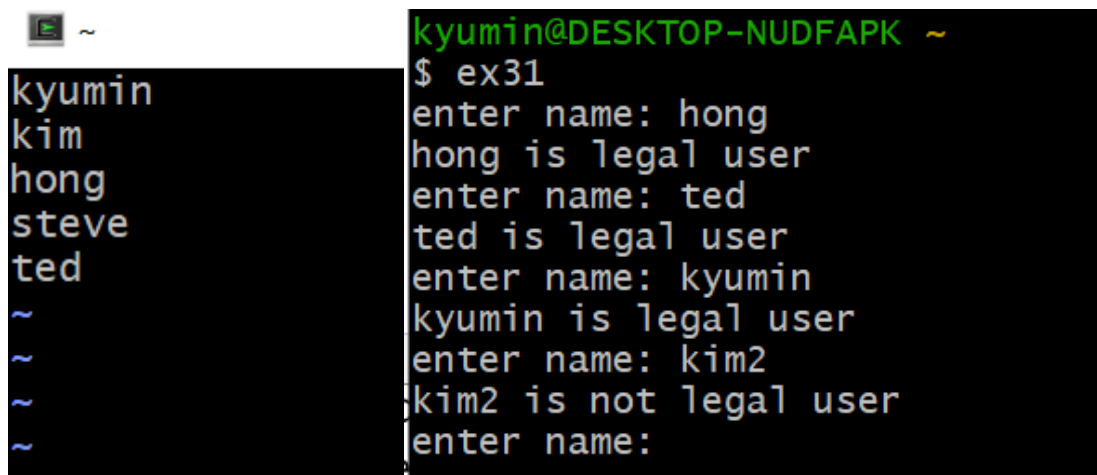
        while(1)
        {
            if(fgets(buf, 20, fp) == NULL)
            {
                printf("%s is not legal user\n", name);
                break;
            }
            buf[strlen(buf) - 1] = '\0';
            if(strcmp(name, buf) == 0)
            {
                printf("%s is legal user\n", name);
                break;
            }
        }
    }
    fclose(fp);
}
```

우선 사용자로부터 이름을 입력 받고, name 변수에 저장한다.

그리고 fopen 함수를 사용하여 login.txt 파일을 읽기 모드로 열었다. 파일의 내용을 한 줄씩 읽고 buf에 저장하기 위해서 fgets를 사용했다. 무한 루프인 while 내부에서 fgets를 사용하고, 읽은 내용이 없을 경우 일치하는 이름이 없다는 메시지를 출력하고 탈출한다.

여기서 scanf는 줄바꿈이 들어가지 않지만, fgets는 줄바꿈이 들어가므로 buf의 마지막 문자를 'W0'으로 바꿔주었다.

만약 동일한 이름이 존재한다면 메시지를 출력하고 바로 while문을 탈출한다.



```
kyumin
kim
hong
steve
ted
~
~
~
~

kyumin@DESKTOP-NUDFAPK ~
$ ex31
enter name: hong
hong is legal user
enter name: ted
ted is legal user
enter name: kyumin
kyumin is legal user
enter name: kim2
kim2 is not legal user
enter name:
```

왼쪽 사진은 login.txt 내용이고, 오른쪽은 이름을 검색해본 결과다.

3-2) Write a server that speaks only with registered user in "login.txt" file. The server should open login.txt file before chatting and remember the names of legal users.

```
cli -> serv: hello
serv -> cli: name?
cli -> serv: kim
serv -> cli: ok. now we can talk
.... cli and serv chats ....
```

```
cli -> serv: hello
serv -> cli: name?
cli -> serv: hong
serv -> cli: not legal user. disconnecting.
```

[serv.c]

```
//socket을 통해 들어오는 연결을 듣는다.
listen(s1, 5);
xx = sizeof(cli_addr);
s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
printf("we passed accept. new socket num is %d\n", s2);

//사용자 확인
int login = open("login.txt", O_RDONLY | O_CREAT, 00777);
char name[20];
y = read(login, name, 20);
name[y - 1] = '\0';

//client 입력 대기
y = read(s2, buf, 50);
buf[y] = 0;
printf("cli -> serv : %s\n", buf);

//이름 질문
write(s2, "name?", 5);
```

```

//이름 확인
y = read(s2, buf, 50);
buf[y] = 0;
printf("cli -> serv : %s\n", buf);
if(strcmp(name, buf) != 0)
{
    write(s2, "not legal user. disconnecting.", 31);
    close(s2);    // disconnect the connection
    close(s1);    // close the original socket
    exit(0);
}

write(s2, "ok. now we can talk", 14);

//프로세스 복제
int fp = fork();

//socket의 메시지를 읽고 출력
for(int i = 0; i < 10; i++)
{
    if(fp == 0)
    {
        y=read(s2,buf,50);
        buf[y]=0;
    }
}

```

```

kyumin@DESKTOP-NUDFAPK ~
$ cat login.txt
kyumin

```

Listen 함수를 이용해서 client와 연결이 된다면 우선 사용자 확인이 필요하다.

우선 처음에 open 함수를 사용해서 “login.txt” 파일을 열고, 그 내용을 캐릭터 name 배열에 저장한다. 이 때 `name[y - 1] = '\0';`라는 코드가 있는데 이는 파일의 내용을 read할 때 마지막에 개행문자가 들어있으므로 “kyuminWn”이라는 메시지가 name에 저장된다. 그러나 client로부터 입력되는 메시지에는 개행문자가 없으므로 마지막 문자를 종료로 바꿔주었다.

그리고 client로부터 socket을 받는다면, 이름을 묻는 메시지 “name?”을 보낸다.

client로부터 메시지(이름)을 받는다면 name 배열에 저장되어있던 메시지와 동일한 문자열인지 비교를 한다. 만약 다르다면 모든 socket을 닫고 연결을 종료한다. 올바른 이름이 입력되었다면 for문을 실행하여 통신을 시작한다.

[로그인에 성공한 결과]

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
cli -> serv : hello  
cli -> serv : kyumin  
cli -> serv : hi
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
hello  
serv -> cli : name?  
kyumin  
serv -> cli : ok. now we can talk  
hi
```

Login.txt 파일에는 “kyumin”이라는 메시지가 저장되어있다. Cli에서 kyumin을 입력하니 정상적으로 통신이 되는 것을 볼 수 있다.

[로그인에 실패한 경우]

```
kyumin@DESKTOP-NUDFAPK ~  
$ serv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
cli -> serv : hello  
cli -> serv : hong
```

```
kyumin@DESKTOP-NUDFAPK ~  
$
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ cli  
Hi, I am the client  
socket opened successfully. socket num is 3  
hello  
serv -> cli : name?  
hong  
serv -> cli : not legal user. disconnecting.  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :  
serv -> cli :
```

다른 이름이 입력된 경우 통신이 끊긴다. Cli의 경우 통신이 끊겼을 때 메시지를 주고 받는 for문을 빠져나오는 코드가 없어 위 사진에서는 “serv -> cli :” 메시지가 for문 반복 횟수만큼 출력된다. 이 문제는 통신이 끊겼을 때 따로 처리 방법을 만든다면 해결할 수 있다.



4) Implement simple ftp server and client. If the client doesn't follow the protocol, the server should stop the communication. This is not chatting program. Do not type "hello", "what file do you want?", etc. The server and client should automatically send or receive the ftp protocol messages. The user will only provide the file name to download. Do not use the code from Problem 3, which will make the coding very hard. Modify the code from Problem 1. Make sure your ftp server and client can handle any file size.

Simple ftp protocol

client => server: hello

server => client: what file do you want?

client => server: file name

server => client: file contents

[cli.c]

```
#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>
#include <sys/time.h>
#include <unistd.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "192.168.0.43"

int main(){
    int x,y;
    struct sockaddr_in serv_addr;
    char buf[50];

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

    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = PF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(SERV_ADDR);
    serv_addr.sin_port = htons(SERV_TCP_PORT);

    /* open a tcp socket*/
    if ( (x =socket(PF_INET, SOCK_STREAM,0)) < 0){
        printf("socket creation error\n");
        exit(1);
    }
    printf(" socket opened successfully. socket num is %d\n", x);
```

```

/* connect to the server */
if (connect(x, (struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){
    printf("can't connect to the server\n");
    exit(1);
}

//메시지 보내기 / 받기
strcpy(buf, "hello");
printf("client => server: %s\n", buf);
write(x, buf, strlen(buf));
y = read(x, buf, 50);
buf[y]=0;
printf("server: => client: %s\n", buf);
scanf("%s", buf);
write(x, buf, strlen(buf)); //파일 이름 전송
y = read(x, buf, 50);
buf[y] = 0;
printf("server: => client: %s\n", buf);

close(x); // disconnect
}

```

간단히 구현한 코드이기 때문에 1번 문제의 코드와 거의 비슷하다. 파일 이름만 사용자로부터 입력을 받아서 서버로 전송하면 되고, 이 때 파일 이름은 띄어쓰기가 없으므로 scanf 함수를 사용해도 문제가 없다.

[serv.c]

```

#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>
#include <sys/time.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "192.168.0.43"

int 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");

    bzero((char *)&serv_addr, sizeof(serv_addr));
    serv_addr.sin_family=PF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
    serv_addr.sin_port=htons(SERV_TCP_PORT);

    if((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
        printf("socket creation error\n");
        exit(1);
    }
    printf("socket created successfully. socket num is %d\n", s1);

```

```

x=bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
if (x < 0){
    printf("binding failed\n");
    exit(1);
}
printf("binding passed\n");

//통신 연결 대기
listen(s1, 5);
xx = sizeof(cli_addr);
s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
printf("we passed accept. new socket num is %d\n", s2);

//소켓 내용 읽기
y = read(s2, buf, 50);
buf[y] = 0;
printf("%s\n", buf);

write(s2, "what file do you want?", strlen("what file do you want?"));

y = read(s2, buf, 50);
buf[y] = 0;
printf("%s\n", buf);

//입력된 파일명을 열고, 내용을 전송하기
int fd = open(buf, O_RDONLY | O_CREAT, 00777);
for(;;)
{
    y = read(fd, buf, 20);
    if(y == 0) break;
    write(s2, buf, strlen(buf));
}

close(s2);    // disconnect the connection
close(s1);    // close the original socket
}

```

Serv.c에서는 client로부터 입력받은 파일명을 오픈해서 내용을 전송해야한다. Client로부터 파일명을 전송 받으면 buf에 저장되고, open함수를 사용해서 buf에 저장된 문자열의 파일 이름을 오픈한다. 이 때 파일 내용은 수정이 필요없으므로 O\_RDONLY 옵션을 사용하여 읽기모드로 열었고, 파일이 없다면 빈 내용을 buf에 저장하기 위해서 O\_CREAT 옵션을 사용했다.

그리고 파일의 모든 내용을 읽기 위해서 무한 for문을 만들고, read 함수를 사용해서 파일 내용을 읽는다. 읽은 내용은 client로 바로 전송한다. 만약 파일의 모든 내용을 다 읽고, 읽은 내용이 없다면 break 함수를 사용해서 for문을 탈출한다.

그러나 cli.c 코드에서는 문제가 있다. 파일의 내용이 길다면 serv.c의 for문이 여러 번 돌면 write를 여러 번 한다. 그러나 cli.c에서 파일의 내용을 전송받아서 읽는 read 코드는 한 번밖에 없으므로 코드가 종료될 것이다. 이 문제를 해결하기 위해선 cli.c의 read 함수를 여러 번 반복할 수 있도록 for문 안에서 read를 하면 된다.

4-1) Modify your ftp client so that the client saves the file content under the same file name. For this purpose, run the client in a different directory if you run the server and client in the same machine.

[cli.c]

```
//메시지 보내기 / 받기
strcpy(buf, "hello");
printf("client => server: %s\n", buf);
write(x, buf, strlen(buf));
y = read(x, buf, 50);
buf[y]=0;
printf("server: => client: %s\n", buf);
scanf("%s", buf);
write(x, buf, strlen(buf)); //파일 이 를 전 송

//파일 내용 저장 하기
int fd = open(buf, O_RDWR | O_CREAT | O_TRUNC, 00777);
for(int i = 0; i < 10; i++)
{
    y = read(x, buf, 50);
    if(y == 0) break;
    write(fd, buf, y);
}

close(x); // disconnect
}
```

"ftpcli.c" 65L, 1514B 65,0-1 Bot

이전 문제에서 파일의 크기가 크면 메시지를 모두 받지 못한다는 문제가 있었지만 for문을 추가하여 해결하였다. 이 때 for문은 무한 루프로 만들어도 되지만 안전을 위해 10번에 제한을 두었다.

동일한 파일명을 생성하기 위해서 파일 디스크립터의 변수 이름을 fd라고 지정하였다. 그리고 내가 입력했던 파일명을 오픈하기 위해서 open 함수를 사용했고, 옵션은 **O\_RDWR | O\_CREAT | O\_TRUNC**을 사용했다.

For문 안에서는 server로부터 입력받은 내용을 for문을 돌면서 fd에 저장한다. 만약 서버에서 모든 내용을 다 보내면 client도 읽을 내용이 없으므로 y는 0이 되면서 for문을 탈출할 수 있게 된다.

[serv.c]

```

//소켓 내용 읽기
y = read(s2, buf, 50);
buf[y] = 0;
printf("%s\n", buf);

write(s2, "what file do you want?", strlen("what file do you want?"));

y = read(s2, buf, 50);
buf[y] = 0;
printf("%s\n", buf);

//입력된 파일명을 열고, 내용을 전송하기
int fd = open(buf, O_RDONLY | O_CREAT, 00777);
for(int i = 0; i < 10; i++)
{
    y = read(fd, buf, 50);
    if(y == 0) break;
    write(s2, buf, y);
}

close(s2); // disconnect the connection
close(s1); // close the original socket
}
"ftpserv.c" 741 1913B 74.0-1 Bot

```

이전 문제와 비슷한 코드이지만 이전 문제에서 오류가 있었다. Write함수 사용 시 보낼 문자의 길이를 buf에 크기로 지정했었다. 그러나 파일의 마지막 부분을 읽을 때 즉 y가 50보다 작을 때 보내지 말아야하는 메시지가 보내지는 문제가 있다. 그래서 길이를 y로 수정해줬다. 이외에는 모두 동일한 내용이다.

[실행 결과]

```

kyumin@DESKTOP-NUDFAPK ~/d1
$ ftpcli
Hi, I am the client
socket opened successfully. socket num is 3
client => server: hello
server: => client: what file do you want?
f1

kyumin@DESKTOP-NUDFAPK ~
$ ftpserv
Hi, I am the server
socket created successfully. socket num is 3
binding passed
we passed accept. new socket num is 4
hello
f1

```

이 문제는 server에 있는 파일을 client가 동일한 파일명과 동일한 내용을 가진 파일을 생성하는 것이다. 그러나 나는 동일한 pc로 진행하는데 원본 파일이 훼손되는 문제가 있을 수 있으므로 d1이라는 디렉토리 안에서 ftpcli를 실행해줬다.



5) Modify your ftp server such that it can handle multiple clients at the same time. Make sure the client can download an executable file also. Run the client in another directory so that you don't overwrite an existing file and for an executable file, confirm that the downloaded executable runs ok.

```
x=bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
if (x < 0){
    printf("binding failed\n");
    exit(1);
}
printf("binding passed\n");

for(int SocketNum = 0; SocketNum < 10; SocketNum++)
{
    //통신 연결 대기
    listen(s1, 5);
    xx = sizeof(cli_addr);
    s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
    printf("we passed accept. new socket num is %d\n", s2);
    int ch = fork();
    if(ch == 0)
    {
        //소켓 내용 읽기
        y = read(s2, buf, 50);
        buf[y] = 0;
        printf("%s\n", buf);

        write(s2, "what file do you want?", strlen("what file do you want?"));
    }
}
```

32,17 53%

```
        y = read(s2, buf, 50);
        buf[y] = 0;
        printf("%s\n", buf);

        //입력된 파일명을 열고, 내용을 전송하기
        int fd = open(buf, O_RDONLY | O_CREAT, 00777);
        for(int i = 0; i < 10; i++)
        {
            y = read(fd, buf, 50);
            if(y == 0) break;
            write(s2, buf, y);
        }

        close(s1);
        close(s2);    // disconnect the connection
        exit(0);
    }
    else close(s2);
}
close(s1);    // close the original socket
}
```

82,0-1

여러명의 클라이언트와 통신하는 과제다. 이전 문제의 코드에서 for문을 만들고 listen 부터 반복되도록 만들면 된다. 그 이전까지는 s1의 기본적인 설정이 끝났으므로 반복할 필요가



없다. (반복문은 충분히 크기가 크거나 무한으로 돌도록 만들 수 있지만 현재 필요하지는 않으니 10번만 반복되도록 만들었다.)

client로부터 통신을 받아 s2를 생성한다면 ftp server는 프로세스 복제를 한다. 우선 부모 프로세스는 더 이상 할 작업이 없으므로 s2를 close하고 for문을 돌아 client의 통신을 대기한다.

자식 프로세스는 client로부터 수신받은 내용을 읽고, 메시지를 보낸다. client로부터 파일명을 입력받으면 그 파일을 오픈한다. 파일의 내용을 읽고 그 내용을 client에게 전달한다.

Client와 통신이 될 때마다 프로세스는 복제되므로 여러 클라이언트와 통신이 가능하다.

Client는 이전에 메시지를 보냈던 파일명으로 파일을 새로 만들고, 내용을 전달받으면 그 내용을 파일에 작성한다.

```
kyumin@DESKTOP-NUDFAPK ~/d1
$ cat f1
hello my boy
  hello my boy
    hello my boy
      hello my boy
        hello my boy
          hello my boy
hello my boy
  hello my boy
    hello my boy
      hello my boy
        hello my boy
          hello my boy
hello my boy
  hello my boy
    hello my boy
      hello my boy
        hello my boy
```

```
kyumin@DESKTOP-NUDFAPK ~/d2
$ cat f3
hello my boy
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ftpserv  
Hi, I am the server  
socket created successfully. socket num is 3  
binding passed  
we passed accept. new socket num is 4  
hello  
we passed accept. new socket num is 4  
hello  
f1  
f3
```

D1 디렉토리 안에 있는 클라이언트와 d2 디렉토리 안에 있는 클라이언트가 동시에 서버와 통신을 했고, 각각은 f1과 f3 파일을 요청했다. 그리고 각각 원하는 파일이 생성되었고, 내용을 확인해보면 정상적으로 전달받은 것을 확인할 수 있다.

5-1) Modify your ftp server in Problem 5 such that it can handle three commands: ls, get, put. For “ls” command, the server will show the file list in the current directory. For “get f1”, the server will send file f1. For “put f1” command, the server will store the file content sent by the client under the same file name f1. The server keeps handling a client’s command until it says “bye”.

cli ➔ server: hello

server ➔ cli: ok

cli ➔ serv: ls

serv ➔ cli: (the file list)

cli ➔ serv: get ex1.exe

server ➔ cli: the content of ex1.exe

cli ➔ serv: put f1

server ➔ cli: send f1

cli ➔ server: the content of f1 (the server will store them under file name f1)

.....

cli ➔ serv: bye

serv ➔ cli: bye

[ftpccli.c]

```
#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>
#include <sys/time.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/wait.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "192.168.0.43"

int main(){
    int x, y, status;
    struct sockaddr_in serv_addr;
    char buf[50];

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

    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = PF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(SERV_ADDR);
    serv_addr.sin_port = htons(SERV_TCP_PORT);

    /* open a tcp socket*/
    if ( (x =socket(PF_INET, SOCK_STREAM,0)) < 0){
        printf("socket creation error\n");
        exit(1);
    }
    printf(" socket opened successfully. socket num is %d\n", x);

    /* connect to the server */
    if (connect(x, (struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){
        printf("can't connect to the server\n");
        exit(1);
    }
}
```

```
//메시지 보내기 / 받기
strcpy(buf, "hello");
printf("client => server: %s\n", buf);
write(x, buf, strlen(buf));
y = read(x, buf, 50);
buf[y] = 0; //필요한지 확인 필요
printf("server => client: %s\n", buf);

//명령어 전송 및 수신 반복
for(;;)
{
    printf("\nclient => server: ");
    fgets(buf, 50, stdin);
    buf[strlen(buf) - 1] = '\0';
    write(x, buf, strlen(buf));
    if(strcmp(buf, "bye") == 0) break;

    printf("server => client: ");
    y = read(x, buf, 50);
    buf[y] = 0;
    printf("%s\n", buf);
}
close(x); // disconnect
}
```

클라이언트에서는 메시지를 보내고 읽는 과정을 계속 반복한다.

[ftpserv.c]

```
#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>
#include <sys/time.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/wait.h>

#define SERV_TCP_PORT 34357
#define SERV_ADDR "192.168.0.43"

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

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

    bzero((char *)&serv_addr, sizeof(serv_addr));
    serv_addr.sin_family=PF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
    serv_addr.sin_port=htons(SERV_TCP_PORT);

    if((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
        printf("socket creation error\n");
        exit(1);
    }
    printf("socket created successfully. socket num is %d\n", s1);

    x=bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    if (x < 0){
        printf("binding failed\n");
        exit(1);
    }
    printf("binding passed\n");
```

```

for(int SocketNum = 0; SocketNum < 1; SocketNum++)
{
    //통신 연결 대기
    listen(s1, 5);
    xx = sizeof(cli_addr);
    s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
    printf("we passed accept. new socket num is %d\n", s2);
    int ch = fork();
    if(ch == 0)
    {
        //소켓 내용 읽기
        y = read(s2, buf, 50);
        buf[y] = '\0';
        printf("%s\n", buf);

        write(s2, "ok", 2);

        for(;;)
        {
            y = read(s2, buf, 50);
            buf[y] = 0;
            printf("%s\n", buf);

            //strtok을 사용하면 buf는 띄어쓰기가 \0으로 변환되므로
            //첫 번째 단어만 남게 됨.
            //그러므로 if문을 ls, get, put으로 나눌 수 있게 됨.
            char FileName[20];
            strcpy(FileName, strtok(buf, " "));
            strcpy(FileName, strtok(NULL, " "));
            printf("buf : %s, FN : %s\n", buf, FileName);

            if(strcmp(buf, "ls") == 0)
            {
                char *argv[10];
                strcpy(argv[0], "/bin/ls");
                argv[1] = 0;

                int x1 = fork();
                if(x1 == 0)
                {
                    //file descriptor 1(화면 출력) 삭제 후 s2를 복제
                    //1번에는 s2가 들어가므로 ls 명령어 실행 시
                    //화면에 출력되지 않고 s2에 출력됨.
                    close(1);
                    y = dup(s2);
                }
            }
        }
    }
}

```

서버에서는 클라이언트와 연결이 된다면 fork 함수를 사용해서 프로세스 복제를 하고 부모 프로세스는 s2를 닫고, 통신 대기를 반복함으로써 여러 클라이언트와 통신을 할 수 있다. 그러나 위 코드의 경우 여러 클라이언트와 통신할 필요가 없으므로 한 번만 반복되도록 하였다.

자식 프로세스는 입력받은 명령어가 확인하여 if문에 들어간다. 시간 관계상 ls 명령어 구현은 성공하지 못하여 제출하게 되었다. 다음 과제 제출 시 ls 구현 및 결과 제출할 예정이다.

```

        //명령어 실행
        y = execve(argv[0], argv, 0);
        if(y < 0)
        {
            perror("exec failed");
            exit(1);
        }
        close(s2);
        close(s1);
        exit(0);
    }
    else wait(&status);
}

else if(strcmp(buf, "get") == 0)
{
    printf("get\n");
    //입력된 파일명을 열고, 내용을 전송하기
    int fd = open(fileName, O_RDONLY | O_CREAT, 00777);
    for(;;)
    {
        y = read(fd, buf, 50);
        if(y == 0) break;
        write(s2, buf, y);
    }
    close(fd);
}

else if(strcmp(buf, "put") == 0)
{
    //file 생성
    int fd = open(fileName, O_WRONLY | O_CREAT
    | O_TRUNC, 00777);

    //입력 내용 요청
    strcpy(buf, "Send ");
    strncat(buf, fileName, sizeof(buf)
    - strlen(buf) - 1);
    printf("buf: %s\n", buf);
    printf("%d\n", strlen(buf));
    write(s2, buf, strlen(buf));
}

```

Get 명령어의 경우 입력받은 메시지와 동일한 파일을 오픈한 후 클라이언트에게 내용을 보낸다.

```

        //내용 읽고 파일에 입력
        y = read(s2, buf, 50);
        write(fd, buf, y);
        close(fd);
        write(s2, "finish", strlen("finish"));
    }

    else if(strcmp(buf, "bye") == 0)
    {
        close(s1);
        close(s2);    // disconnect the connection
        exit(0);
    }

    else printf("invalid value\n");
}

close(s1);
close(s2);    // disconnect the connection
exit(0);
}
else
{
    close(s2);
}
}
close(s1);    // close the original socket
}

```

161,0-1 Bot

Put의 경우 입력 받은 파일명을 생성한다. 그리고 다시 입력받은 내용을 그대로 파일에 작성한다.

Bye의 경우 모든 소켓을 닫고 exit한다.



[결과]

```
kyumin@DESKTOP-NUDFAPK ~  
$ ftpcli  
Hi, I am the client  
socket opened successfully. socket num is 3  
client => server: hello  
server => client: ok  
  
client => server: put f1  
server => client: Send f1  
  
client => server: hello my name is kyumin  
server => client: finish  
  
client => server: get f1  
server => client: hello my name is kyumin  
  
client => server: bye  
  
kyumin@DESKTOP-NUDFAPK ~  
$
```

```
kyumin@DESKTOP-NUDFAPK ~/d2  
$ put f1  
buf : put, FN : f1  
buf: Send f1  
7  
get f1  
buf : get, FN : f1  
get  
bye  
|
```

Put, get, bye 명령어가 정상 작동하는 것을 볼 수 있다.

Ls는 아쉽게 구현하지 못했고, 시간이 없어 과제를 제출하겠다. 다음 시간까지 구현해서 제출할 예정이다.

6) Write a client in your PC as follows and let it talk to the server program in the lab server. To compile the client program:

- Make an empty c++ project and copy the code given below.
- Adjust the server IP and port number
- Select
  - “project->project property->manifest tools->input and output->include manifest” and set “No”
- add ws2\_32.lib in project>project property>link>input>additional dependencies
- Select build->Solution Build
- You should see “Success 1” at the bottom of the compile window.
- Sometimes, the platform of the program (the one shown in the main interface screen: release or debug) is different from the platform in properties configuration. They should match.)
- If you have an error related with "pre-compiled header", go to
  - “project->project property->c/c++>pre-compiled header > pre-compiled header>” and set “No”

(\* If you use MAC, download cli.c from the lab server and use it as your local client program. You may have to include "unistd.h" to avoid compile errors.)

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include "winsock2.h"
```

```
#include "ws2tcpip.h"
```

```
#include "stdio.h"
```

```
#define SERVER_PORT 9924 // server port number
```

```
#define BUF_SIZE 4096 // block transfer size
```

```
#define QUEUE_SIZE 10
```

```
#define IPAddress "165.246.38.152" // server IP address
```

```
int main()
```

```
{
```

```
    WORD
```

```
    wVersionRequested;
```

```
    WSADATA
```

```
    wsaData;
```

```
    SOCKADDR_IN
```

```
    target; //Socket address information
```

```
    SOCKET
```

```
    s;
```

```
    int
```

```
    err;
```

```
    int
```

```
    bytesSent;
```

```
    char
```

```
    buf[100];
```

```

//--- INITIALIZATION -----
wVersionRequested = MAKEWORD( 1, 1 );
err = WSASStartup( wVersionRequested, &wsaData );

if ( err != 0 ) {
    printf("WSAStartup error %ld", WSAGetLastError() );
    WSACleanup();
    return false;
}
//-----

//---- Build address structure to bind to socket.-----
target.sin_family = AF_INET; // address family Internet
target.sin_port = htons (SERVER_PORT); //Port to connect on
inet_pton(AF_INET, IPAddress, &(target.sin_addr.s_addr)); // target IP
//-----

// ---- create SOCKET-----
s = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP); //Create socket
if (s == INVALID_SOCKET)
{
    printf("socket error %ld", WSAGetLastError() );
    WSACleanup();
    return false; //Couldn't create the socket
}
//-----

//---- try CONNECT -----
if (connect(s, (SOCKADDR *)&target, sizeof(target)) == SOCKET_ERROR)
{
    printf("connect error %ld", WSAGetLastError() );
    WSACleanup();
    return false; //Couldn't connect
}
//-----

//---- SEND bytes -----
printf("enter a string to send to server\n");
gets_s(buf, 99);
bytesSent = send( s, buf, strlen(buf), 0 ); // use "send" in windows
printf( "Bytes Sent: %ld\n", bytesSent );

// now receive
int n;

```

```
n=recv(s, buf, 50, 0); // read max 50 bytes
buf[n]=0; // make a string
printf("received: %s\n", buf);

//-----
closesocket( s );
WSACleanup();

return 0;
}
```

7) Write an ftp client in your PC and let it talk to the ftp server you made in problem 5). Use this client to download a file from the lab server and save under the same file name.