

네트워크 프로그래밍

- FTP 서버 & 클라이언트 -



Korea University of Technology and Education

과 목 명	네트워크 프로그래밍
교 수 명	서희석 교수님
분 반	01
조 원 (학 번)	강인창(2011136004)
제 출 일	2017.06.16

목 차

I. 서론	1
II. 본론	1
1. 서버와 클라이언트 실행	1
2. ls 명령어 실행	4
3. get 명령어 실행	6
4. put 명령어 실행	16
5. del 명령어 실행	19
6. ren 명령어 실행	24
7. bye 명령어 실행	27
8. 모든 파일 확장자 제어	29
III. 결론	33
IV. 부록	33
1. 고찰	33
2. 참고 자료	34

I. 서론

- 이번 과제의 목표는 파일을 주고받을 수 있도록 하는 FTP 서버와 클라이언트를 작성함으로써 통신을 실습하는 것이다. 초기 조건은 다음과 같다.

1. 클라이언트는 서버에 파일을 업로드할 수 있다(put 명령어).
 2. 클라이언트는 서버 내의 파일을 다운로드할 수 있다(get 명령어).
- + 서버에 파일을 업로드하는 디렉토리는 서버 프로세스가 실행되는 곳으로 하며, 클라이언트가 다운로드할 수 있는 파일은 전체 디렉토리를 대상으로 한다.
- + 클라이언트가 파일을 다운로드할 때, 퍼미션을 고려하여 다운로드 가능한 파일인지 아닌지를 확인한다.

조건 외에 임의로 추가한 기능은 다음과 같다.

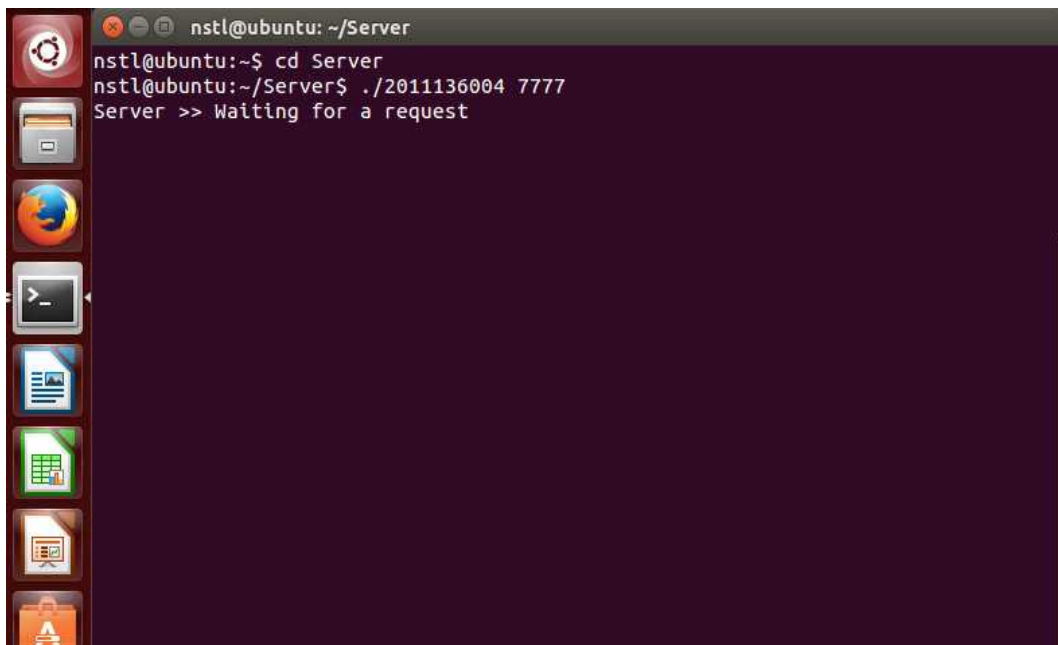
1. 클라이언트는 서버 내의 파일 리스트를 확인할 수 있다(ls 명령어).
2. 클라이언트는 서버 내의 파일을 삭제할 수 있다(del 명령어).
3. 클라이언트는 서버 내의 파일 이름을 변경할 수 있다(ren 명령어).
4. 클라이언트가 서버 접속을 끊고 퇴장한다(bye 명령어).

II. 본론

1. 서버와 클라이언트 실행

1.1. 서버 실행

- 서버가 작동하는 Server 디렉토리에 들어가 포트 번호를 입력하고 2011136004 파일을 실행한다.

A terminal window titled 'nsth@ubuntu: ~/Server' with a dark purple background. The terminal shows the following commands and output:

```
nsth@ubuntu:~$ cd Server
nsth@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
```

On the left side of the terminal window, there is a vertical dock with several application icons: a red circle with a white gear, a folder icon, a Firefox browser icon, a terminal icon, a document icon, a spreadsheet icon, a presentation icon, and a server rack icon.

1.1.1. 매개변수 검사

- 실행을 위한 매개변수가 적절히 입력되었는지 검사한다. 첫 번째 인자는 파일 이름이고 두 번째 인자는 포트 번호다.

```
//① 첫 번째 인자는 파일이름, 두 번째 인자는 포트 번호
if(argc != 2){
    printf("Usage : %s portWn", argv[0]);
    exit(1);
}
```

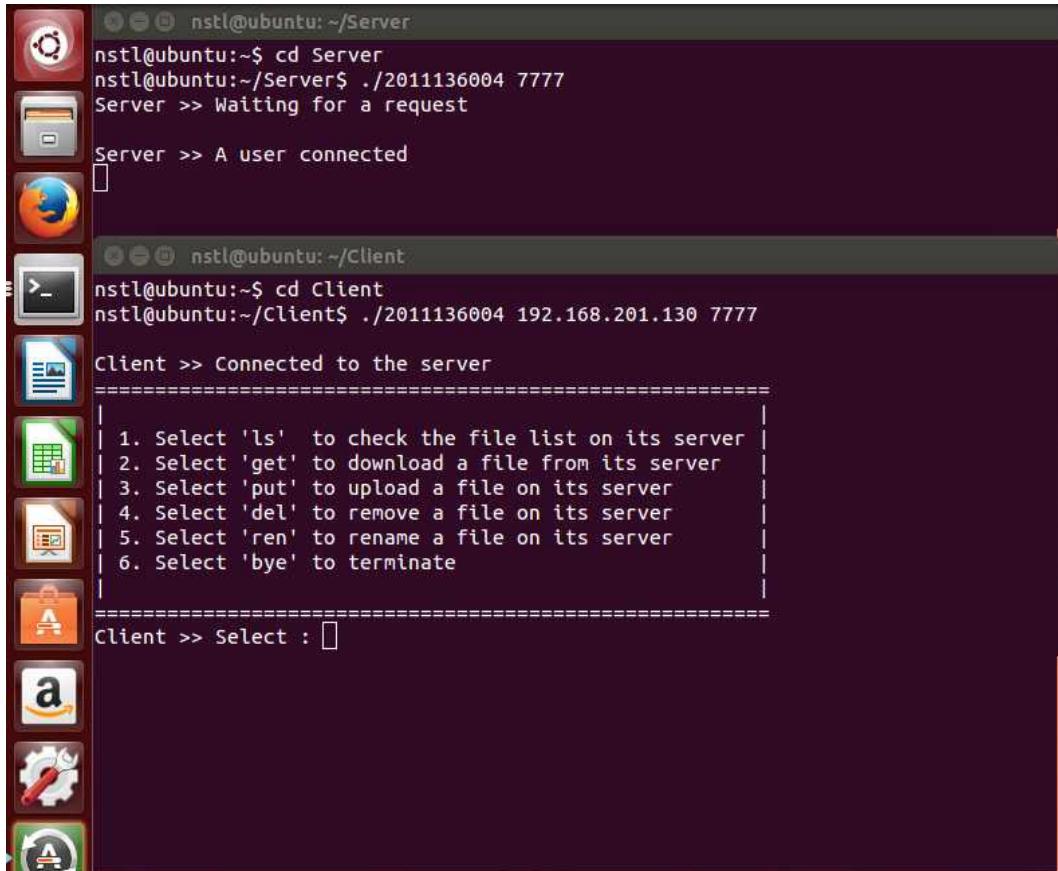
1.1.2. 연결 요청 수신

- 서버는 클라이언트의 연결 요청을 기다리며, 연결 요청이 있을 경우 이를 받아들인다.

```
//① TCP Socket Open
serv_sock = socket(PF_INET, SOCK_STREAM, 0);
if(serv_sock == -1) {
    errquit("Server >> Socket failed ");
}
memset(&serv_addr, 0, sizeof(serv_addr));
//② 연결 구조체 설정
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(atoi(argv[1]));
//③ Binding
if(bind(serv_sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1)
    errquit("Server >> Bind failed ");
while(1) {
    //④ Wating
    if(listen(serv_sock, 5) == -1) errquit("Server >> Listen failed ");
    printf("Server >> Waiting for a requestWnWn");
    cli_addr_len = sizeof(cli_addr);
    //⑤ Accepting
    cli_sock = accept(serv_sock, (struct sockaddr *)&cli_addr, &cli_addr_len);
    if(cli_sock == -1) errquit("Server >> Accept failed ");
    if(conn) printf("Server >> A user connectedWn");
    :
    :
}
```

1.2. 클라이언트 실행

- 클라이언트가 작동하는 Client 디렉토리에 들어가 서버의 IP와 포트 번호를 입력하고 2011136004 파일을 실행한다.



```
nstl@ubuntu: ~/Server
nstl@ubuntu:~$ cd Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
Server >> A user connected
[]

nstl@ubuntu: ~/Client
nstl@ubuntu:~$ cd Client
nstl@ubuntu:~/Client$ ./2011136004 192.168.201.130 7777
Client >> Connected to the server
=====
| 1. Select 'ls' to check the file list on its server |
| 2. Select 'get' to download a file from its server |
| 3. Select 'put' to upload a file on its server |
| 4. Select 'del' to remove a file on its server |
| 5. Select 'ren' to rename a file on its server |
| 6. Select 'bye' to terminate |
|=====
Client >> Select : []
```

1.2.1. 매개변수 검사

- 실행을 위한 매개변수가 적절히 입력되었는지 검사한다. 첫 번째 인자는 파일 이름이고 두 번째 인자는 서버의 IP, 마지막은 포트 번호다.

```
//① 파일 이름, 포트 번호, 클라이언트이름 순으로 인자가 들어간다.
if(argc != 3){
    printf("Usage : %s server_ip server_port\n", argv[0]);
    exit(0);
}
```

1.2.2. 연결 요청 송신

- 클라이언트는 tcp_connect() 메소드를 활용하여 서버로 연결 요청을 송신한다.

```
while(1) {
    //① TCP Socket Open
    cli_sock = socket(PF_INET, SOCK_STREAM, 0);
    if(cli_sock == -1) {
        errquit("Client >> Socket failed ");
    }
}
```

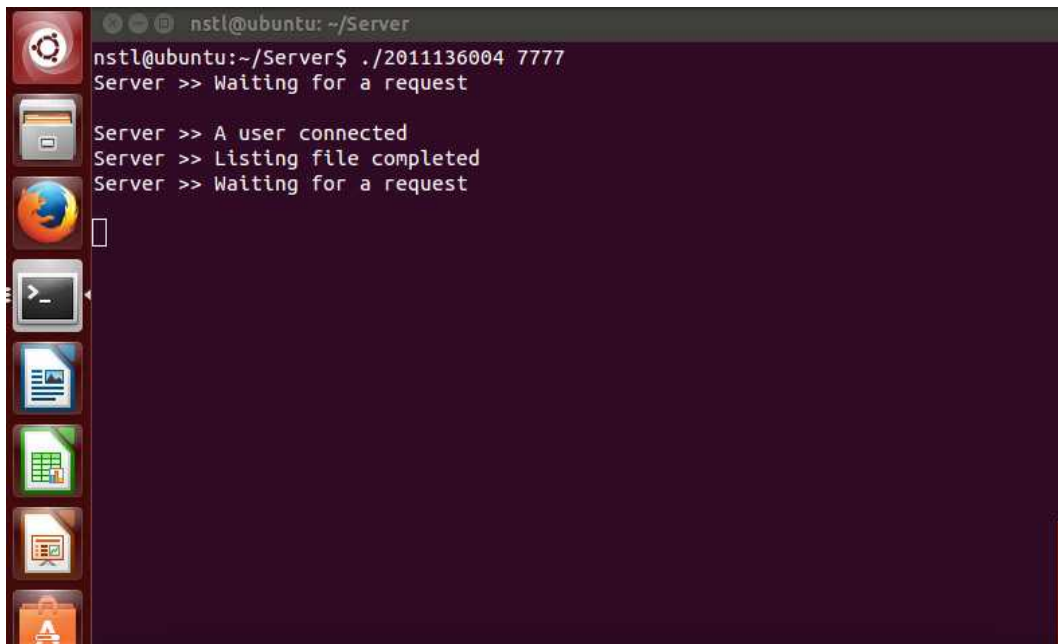
```

    }
    memset(&serv_addr, 0, sizeof(serv_addr));
    //② 연결 구조체 설정
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));
    //③ 연결 요청
    if(connect(cli_sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1)
        errquit("Client >> Connection failed ");
    if(conn) printf("\nClient >> Connected to the server\n");
    :
    :
}

```

2. ls 명령어 실행

2.1. 서버 화면



2.1.1. ls 명령어 수신 및 파일 리스트 송신

- 클라이언트로부터 'ls' 명령어를 수신할 경우, /home/nstl/Server 디렉토리 패스에 있는 모든 파일 이름을 저장하고 이를 클라이언트로 송신한다.

```

//① ls 문자열 판별
if(strstr(select, "ls") != NULL) {
    DIR *dir, struct dirent *ent, char buffer[1024] = "";
    //② /home/nstl/Server 패스에 있는 디렉토리에 있는 파일 이름들을 저장한다.
    if((dir = opendir ("/home/nstl/Server")) != NULL) {
        while((ent = readdir (dir)) != NULL) {
            strcat(buffer, ent->d_name);

```

```

        strcat(buffer, ",");
    }
    closedir(dir);
    //③ 저장한 파일 이름들을 버퍼에 담아 클라이언트 송신한다.
    if(send(cli_sock, buffer, 1024, 0) < 0)
        errquit("Server >> Write failed ");
    printf("Server >> Listing file completed\n");
}
//④ 해당 패스에 있는 디렉토리를 얻을 수 없는 경우
else {
    printf("Server >> Listing files failed\n");
    continue;
}
}
}

```

2.2. 클라이언트 화면

```

nssl@ubuntu: ~/Server
nssl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> Listing file completed
Server >> Waiting for a request

nssl@ubuntu: ~/Client
Client >> Connected to the server

=====
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
=====

Client >> Select : ls
.. 2011136004.c . 2011136004 Server.bin Server.txt
=====

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
=====

Client >> Select :

```

2.2.1. ls 명령어 송신 및 파일 리스트 수신 후 출력

- 서버로 'ls' 명령어를 송신하고, 파일 리스트를 수신 받아 출력한다.

```

printf("Client >> Select : ");
scanf("%s", select);

```

//① 문자열을 소문자로 변형

```
for(i=0; i<3; i++) select[i] = tolower(select[i]);
```

//② 사용자의 입력을 서버로 전송

```
if(send(cli_sock, select, strlen(select), 0) < 0) errquit("Client >> Write failed ");
```

//③ ls 명령어일 경우

```
if((strncmp("ls", select, 2) == 0)) {
```

//④ 서버로부터 파일 이름이 저장된 버퍼를 수신 받아 이를 출력한다.

```
char exception[] = ","; char *token = NULL;
```

```
buf_len = recv(cli_sock, buf, 1024, 0); buf[buf_len] = 0;
```

```
token = strtok(buf, exception);
```

```
while( token != NULL ) {
```

```
    printf("%s ", token); token = strtok(NULL, exception);
```

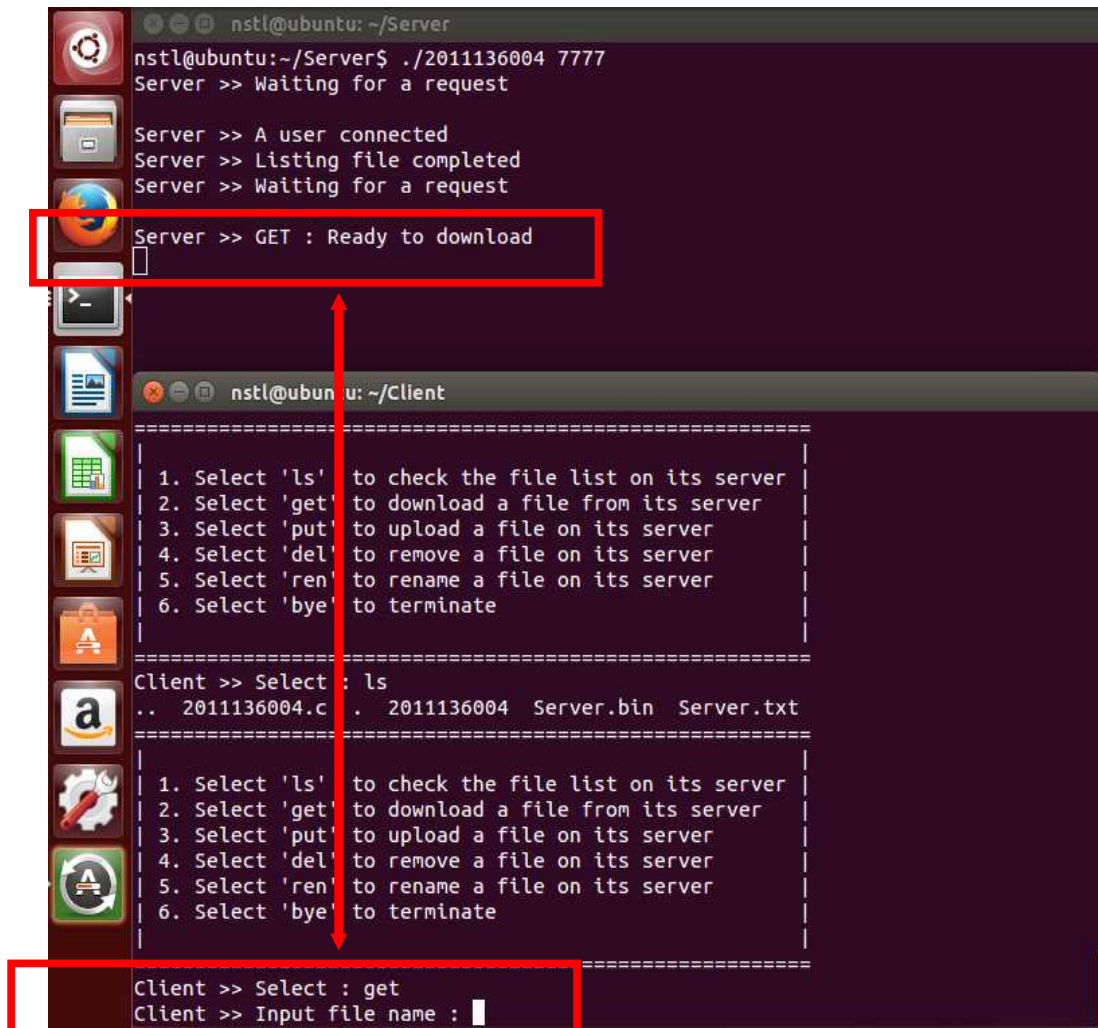
```
}
```

```
printf("\n");
```

```
}
```

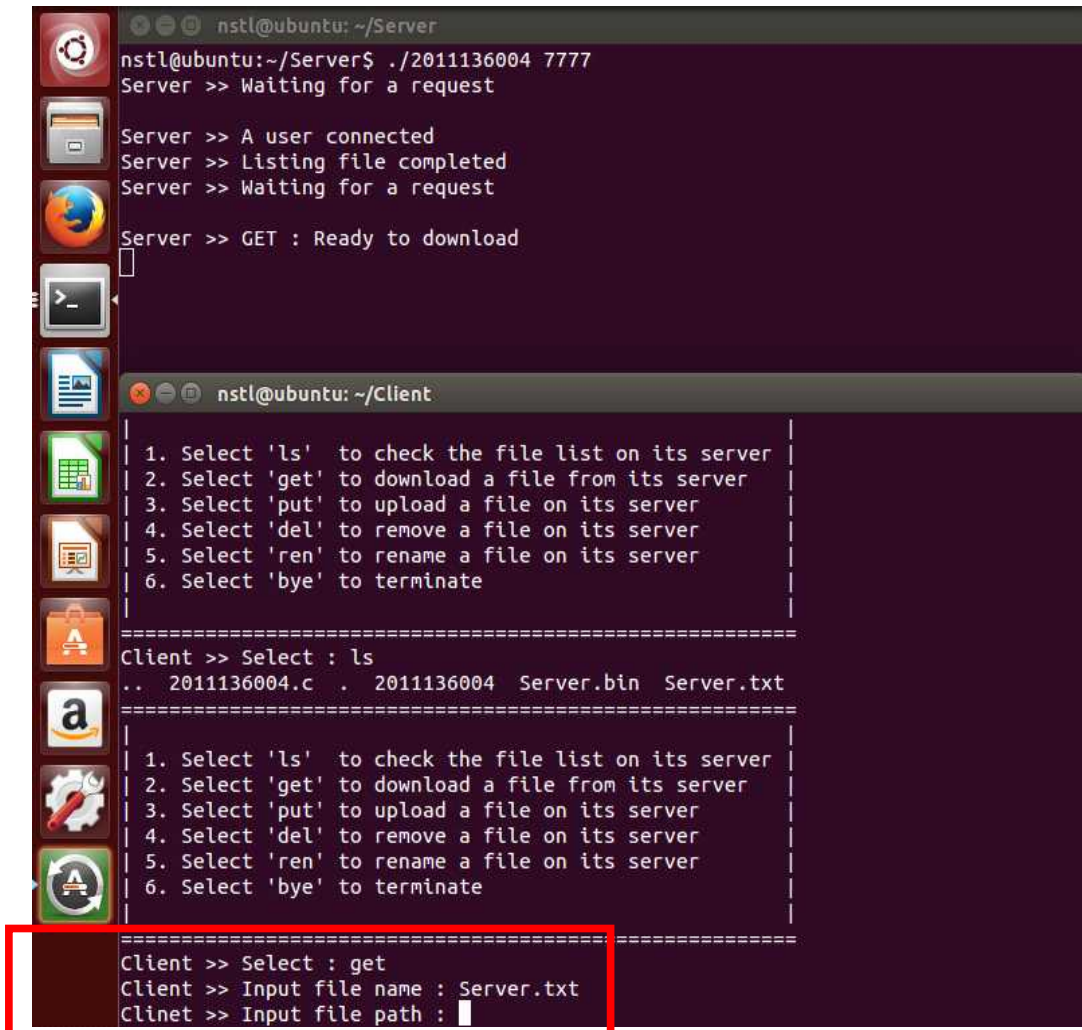
3. get 명령어 실행

3.1. 클라이언트가 get 명령어를 입력하였을 때



3.2. 클라이언트가 파일 이름을 입력하였을 때

- 클라이언트 폴더로 서버 디렉토리에 있는 Server.txt 파일을 다운로드하기 위해 해당 파일 이름을 입력한다.



```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> Listing file completed
Server >> Waiting for a request

Server >> GET : Ready to download
[ ]

nstl@ubuntu: ~/Client

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

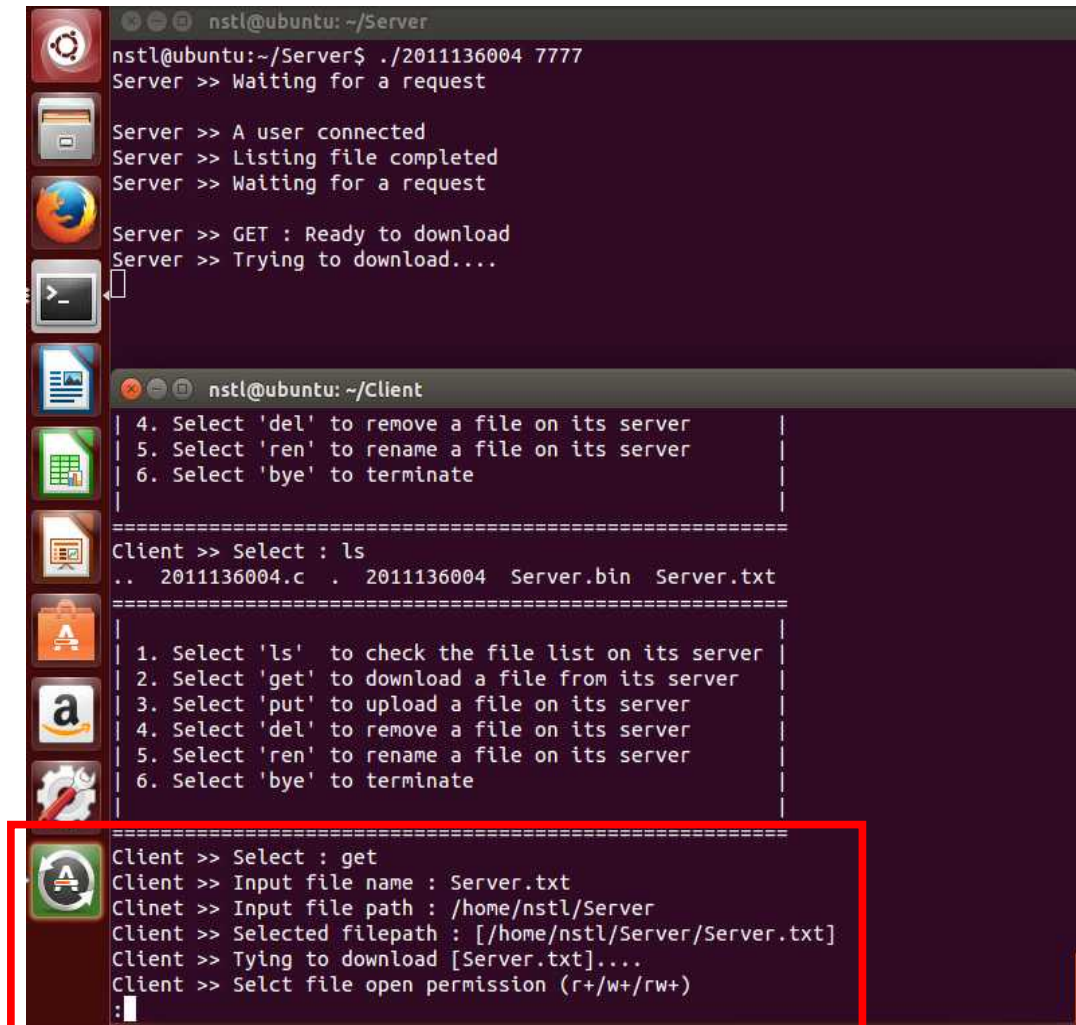
=====
Client >> Select : ls
.. 2011136004.c . 2011136004 Server.bin Server.txt
=====

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

=====
Client >> Select : get
Client >> Input file name : Server.txt
Client >> Input file path : [ ]
```

3.3. 클라이언트가 파일 경로를 입력하였을 때

- 클라이언트 입장에서 모든 경로에서 파일을 다운로드할 수 있어야 하므로 경로를 입력해준다.



```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

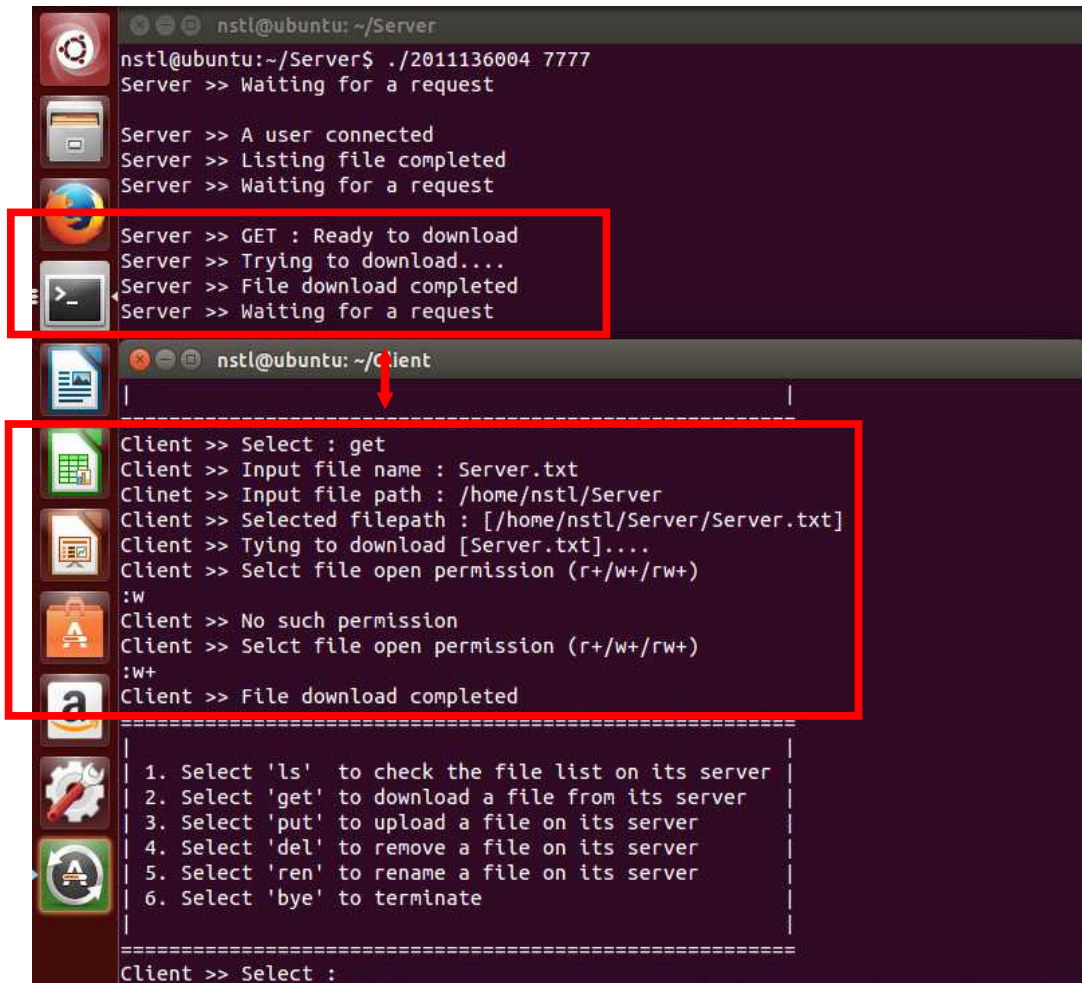
Server >> A user connected
Server >> Listing file completed
Server >> Waiting for a request

Server >> GET : Ready to download
Server >> Trying to download....

nstl@ubuntu: ~/Client
| 4. Select 'del' to remove a file on its server |
| 5. Select 'ren' to rename a file on its server |
| 6. Select 'bye' to terminate |
|=====|
Client >> Select : ls
.. 2011136004.c . 2011136004 Server.bin Server.txt
|=====|
| 1. Select 'ls' to check the file list on its server |
| 2. Select 'get' to download a file from its server |
| 3. Select 'put' to upload a file on its server |
| 4. Select 'del' to remove a file on its server |
| 5. Select 'ren' to rename a file on its server |
| 6. Select 'bye' to terminate |
|=====|
Client >> Select : get
Client >> Input file name : Server.txt
Client >> Input file path : /home/nstl/Server
Client >> Selected filepath : [/home/nstl/Server/Server.txt]
Client >> Tying to download [Server.txt]...
Client >> Selct file open permission (r+/w+/rw+)
:
```

3.4. 파일에 대한 접근 권한 설정

- 클라이언트가 서버로부터 파일을 다운로드 받을 때 이것을 읽기 전용(r+)으로 할 것인지, 쓰기 전용(w+)로 할 것인지, 읽기 쓰기 가능(rw+)로 할 것인지 선택할 수 있다.



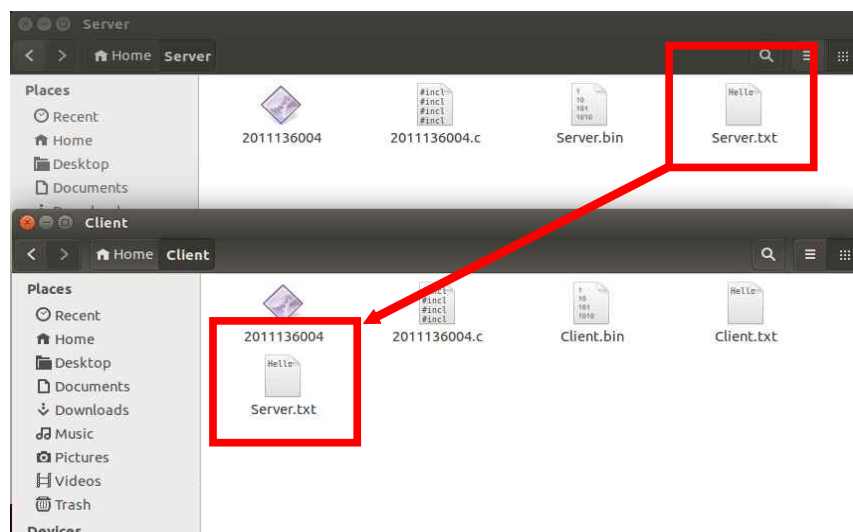
```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> Listing file completed
Server >> Waiting for a request

Server >> GET : Ready to download
Server >> Trying to download....
Server >> File download completed
Server >> Waiting for a request

nstl@ubuntu: ~/Client
Client >> Select : get
Client >> Input file name : Server.txt
Client >> Input file path : /home/nstl/Server
Client >> Selected filepath : [/home/nstl/Server/Server.txt]
Client >> Tying to download [Server.txt]....
Client >> Selct file open permission (r+/w+/rw+)
:w
Client >> No such permission
Client >> Selct file open permission (r+/w+/rw+)
:w+
Client >> File download completed

=====
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
=====
Client >> Select :
```



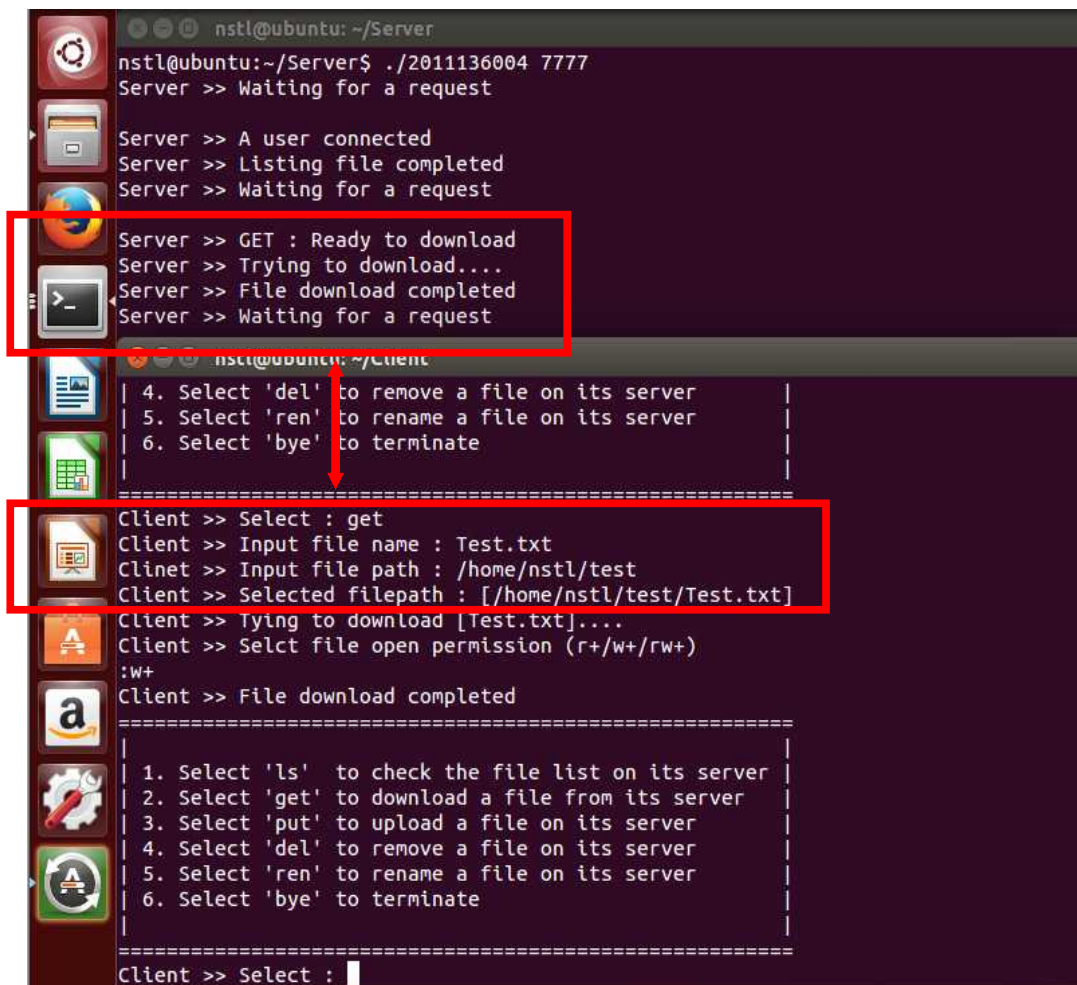
< Server 폴더의 Server.txt와 Client 폴더의 Server.txt >

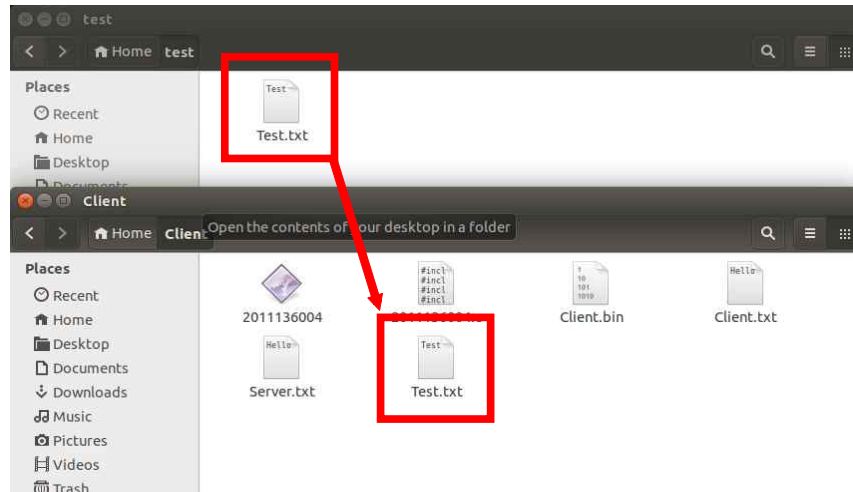


< Server와 Client 폴더의 Server.txt 내용 >

3.5. 클라이언트가 다른 경로에 있는 파일을 다운로드 받을 때

- 전체 경로에서 파일을 다운받을 수 있도록 한다. 실습을 위해 /home/nstl/Server 경로였던 이전과 달리, /home/nstl/test 디렉토리에 들어가 Test.txt 파일을 다운로드 한다.





< test 폴더의 Test.txt와 Client 폴더의 test.txt >



< Test와 Client 폴더의 Test.txt 내용 >

3.6. 파일에 대한 퍼미션 판별

3.6.1. 파일 정보 보기

- 터미널에서 'ls -al' 명령어를 입력하면, 해당 디렉토리에 있는 파일들의 파일 정보를 확인할 수 있다. 왼쪽부터 [파일 퍼미션, 링크 수, 소유자, 그룹, 용량, 생성일, 파일 명]을 의미한다.

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~$ cd Server
nstl@ubuntu:~/Server$ ls -al
total 48
drwxrwxr-x  2 nstl nstl  4096 Jun 15 19:47 .
drwxr-xr-x 21 nstl nstl  4096 Jun 15 20:41 ..
-rwxrwxr-x  1 nstl nstl 18353 Jun 15 19:46 2011136004
-rw-rw-r--  1 nstl nstl 11519 Jun 15 19:46 2011136004.c
-rw-----  1 nstl nstl   12 Jun  9 23:52 Server.bin
-rw-r--r--  1 nstl nstl    7 Jun  9 21:45 Server.txt
nstl@ubuntu:~/Server$
```

3.6.2. 파일 퍼미션

- 파일 퍼미션은 위의 사진에서 볼 수 있듯이, -rwxrwxr-x 와 같이 된 문자열을 의미한다. 왼쪽부터 ①로 해서 가장 끝을 ⑩으로 둔다면, 다음과 같은 의미를 지닌다.

- ① : 디렉토리인지 파일인지를 판별한다. 디렉토리일 경우 'd', 아니면 '-'
- ② ~ ④ : 소유자에 대한 퍼미션을 지정한다. [② : r / ③ : w / ④ : x]
- ⑤ ~ ⑦ : 소유자 그룹에 대한 퍼미션을 지정한다. [⑤ : r / ⑥ : w / ⑦ : x]
- ⑧ ~ ⑩ : 공개 외부 사용자 대한 퍼미션을 지정한다. [⑤ : r / ⑥ : w / ⑦ : x]

3.6.3. 파일 퍼미션을 고려한 제한사항

- 클라이언트가 파일 퍼미션 권한에 위반되어 다운로드 받을 수 없는 상황은 다음과 같다. 여기서 *는 r/w/x 아무거나 상관없다고 본다.
- 소유자가 root 이면서 파일 퍼미션이 *****-* 일 경우, 클라이언트는 해당 파일을 다운로드할 수 없다. 즉, 소유자가 root일 때, 공개 외부 사용자의 읽기 권한이 없을 경우 접근이 제한된다.
- 소유자가 nstl 이면서 파일 퍼미션이 *-***** 일 경우, 클라이언트는 해당 파일을 다운로드할 수 없다. 즉 소유자가 공개 외부 사용자(nstl)일 때, 자기 자신에 대한 읽기 권한이 없을 경우 접근이 제한된다.

3.7. 파일 퍼미션에 따른 다운로드 제한

3.7.1. 파일 퍼미션 변경

- 현재 소유자가 nstl(공개 외부 사용자)이므로 소유자 부문에 읽기 권한이 없다면 해당 파일을 다운로드할 수 없을 것이다. 따라서 '-rw-r--r--'이었던 Server.txt를 '--wxrw-rw-'로 파일 퍼미션을 바꾸어 준다.

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~$ cd Server
nstl@ubuntu:~/Server$ ls -al
total 48
drwxrwxr-x 2 nstl nstl 4096 Jun 15 19:47 .
drwxr-xr-x 21 nstl nstl 4096 Jun 15 20:41 ..
-rwxrwxr-x 1 nstl nstl 18353 Jun 15 19:46 2011136004
-rw-rw-r-- 1 nstl nstl 11519 Jun 15 19:46 2011136004.c
-rw----- 1 nstl nstl 12 Jun 9 23:52 Server.bin
-rw-r--r-- 1 nstl nstl 7 Jun 9 21:45 Server.txt
nstl@ubuntu:~/Server$
```

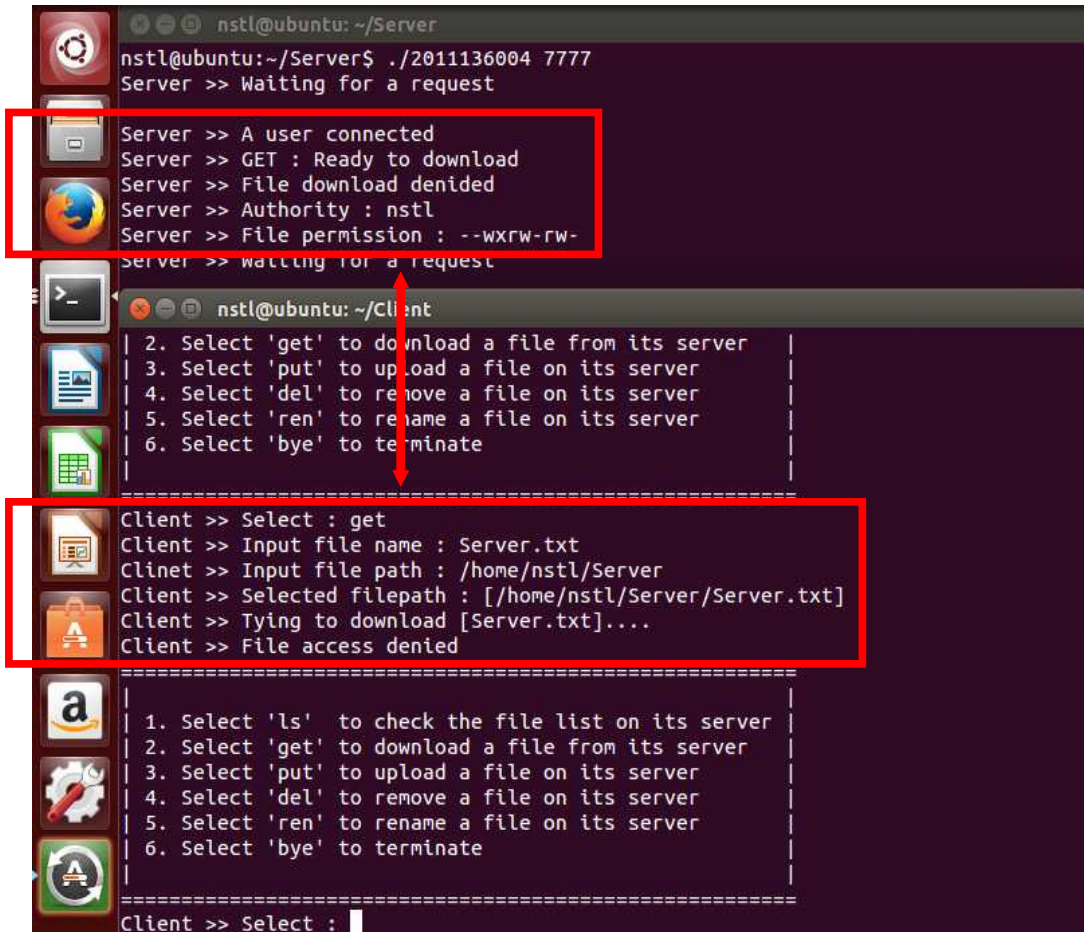
< Server.txt 파일 퍼미션 변경 전 '-rw-----' >

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ chmod 366 Server.txt
nstl@ubuntu:~/Server$ ls -al
total 48
drwxrwxr-x 2 nstl nstl 4096 Jun 15 19:47 .
drwxr-xr-x 21 nstl nstl 4096 Jun 15 20:41 ..
-rwxrwxr-x 1 nstl nstl 18353 Jun 15 19:46 2011136004
-rw-rw-r-- 1 nstl nstl 11519 Jun 15 19:46 2011136004.c
-rw----- 1 nstl nstl 12 Jun 9 23:52 Server.bin
--wxrw-rw- 1 nstl nstl 7 Jun 9 21:45 Server.txt
nstl@ubuntu:~/Server$
```

< Server.txt 파일 퍼미션 변경 후 '--wxrw-rw-' >

3.7.2. 파일 다운로드 제한

- 파일 퍼미션에 따라 파일 다운로드가 제한된다.



```
nsth@ubuntu: ~/Server
nsth@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> GET : Ready to download
Server >> File download denied
Server >> Authority : nsth
Server >> File permission : --wxrw-rw-
Server >> waiting for a request

nsth@ubuntu: ~/Client
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : get
Client >> Input file name : Server.txt
Client >> Input file path : /home/nsth/Server
Client >> Selected filepath : [/home/nsth/Server/Server.txt]
Client >> Trying to download [Server.txt]....
Client >> File access denied

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select :
```

3.8. get 명령어 코드 설명

3.8.1. 서버 코드

```
//① 클라이언트가 get 명령어를 사용했을 경우
else if(strstr(select, "get") != NULL) {
    char flag[3], char permission[10];
    int flag_len, i;
    printf("Server >> GET : Ready to download\n");
    //② 파일 경로와 파일 이름이 합쳐진 문자열을 수신한다.
    file_path_len = recv(cli_sock, filepath, 100, 0);
    filepath[file_path_len] = 0;
    //③ 해당 파일이 없을 경우를 처리한다.
    if(stat(filepath, &file_stat) < 0) {
        char *nak = "NAK";
        if(send(cli_sock, nak, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> No such file or directory\n");
        continue;
    }
}
```

//④ 해당 파일이 있을 경우 파일 퍼미션을 검사한다.

```
struct passwd *pw = getpwuid(file_stat.st_uid);
struct group *gr = getgrgid(file_stat.st_gid);
if(S_ISDIR(file_stat.st_mode)) permission[0] = 'd';
else permission[0] = '-';
if(file_stat.st_mode & S_IRUSR) permission[1] = 'r';
else permission[1] = '-';
if(file_stat.st_mode & S_IWUSR) permission[2] = 'w';
else permission[2] = '-';
if(file_stat.st_mode & S_IXUSR) permission[3] = 'x';
else permission[3] = '-';
if(file_stat.st_mode & S_IRGRP) permission[4] = 'r';
else permission[4] = '-';
if(file_stat.st_mode & S_IWGRP) permission[5] = 'w';
else permission[5] = '-';
if(file_stat.st_mode & S_IXGRP) permission[6] = 'x';
else permission[6] = '-';
if(file_stat.st_mode & S_IROTH) permission[7] = 'r';
else permission[7] = '-';
if(file_stat.st_mode & S_IWOTH) permission[8] = 'w';
else permission[8] = '-';
if(file_stat.st_mode & S_IXOTH) permission[9] = 'x';
else permission[9] = '-';
```

//⑤ 앞서 정의한 파일 퍼미션 조건에 따라 소유주가 root일 때 공개 사용자의 읽기 권한이 없거나, 소유주가 공개 사용자일 때 소유주의 읽기 권한이 없을 경우 다운로드를 제한한다.

```
if( (strstr(pw->pw_name, "root") != NULL)
    && (permission[7] == '-') || (strstr(pw->pw_name, "nsl") != NULL)
    && (permission[1] == '-') ) {
    char *ban = "BAN";
    if(send(cli_sock, ban, 3, 0) < 0) errquit("Server >> Write failed");
    printf("Server >> File download denied\n");
    printf("Server >> Authority : %s\n", pw->pw_name);
    printf("Server >> File permission : ");
    for(i=0;i<10;i++) { printf("%c", permission[i]); if(i==9) printf("\n"); }
    continue;
}
```

//⑥ 파일이 존재하고, 다운 가능한 파일일 경우, 해당 파일을 다운로드한다.

```
strcpy(buf, filepath);
fd = open(filepath, O_RDONLY);
if(fd == -1) {
    char *nak = "NAK";
    if(send(cli_sock, nak, 3, 0) < 0) errquit("Server >> Write failed");
    printf("Server >> No such file or directory\n");
    continue;
}
```



```

else {
    char *ack = "ACK";
    if(send(cli_sock, ack, 3, 0) < 0) errquit("Server >> Write failed");
    printf("Server >> Trying to download...\n");
    flag_len = recv(cli_sock, flag, 3, 0);
    flag[flag_len] = 0;
    if(strstr(flag, "ACK") != NULL) {
        while((file_len = read(fd, buf, 1)) == 1) write(cli_sock, buf, 1);
        printf("Server >> File download completed\n");
    }
}
}
}

```

3.8.2. 클라이언트 코드

```

printf("Client >> Input file name : "); scanf("%s", filename);
printf("Client >> Input file path : "); scanf("%s", filepath);
//① 파일 경로와 파일 이름을 합쳐 저장한다.
sprintf(filepath, "%s/%s", filepath, filename);
printf("Client >> Selected filepath : [%s]\n", filepath);
file_len = strlen(filename) + 1;
strcpy(buf, filename);
//② 파일 경로와 파일 이름이 합쳐진 filepath를 송신한다.
if(send(cli_sock, filepath, 100, 0) < 0) errquit("Client >> Write failed");
flag_len = recv(cli_sock, flag, 3, 0);
flag[flag_len] = 0;
printf("Client >> Trying to download [%s]...\n", filename);
//③ 서버에 파일이 있음을 확인하면
if(strstr(flag, "ACK") != NULL) {
    while(1) {
        fflush(stdin);
        char permit[3];
        //④ 클라이언트가 원하는 파일 접근 권한을 설정한다.
        while(1) {
            fflush(stdin);
            char permit[3];
            printf("Client >> Select file open permission (r+/w+/rw+)\n");
            scanf("%s", permit);
            if(strcmp("r+", permit) == 0) { //⑤ 읽기 전용
                fd = open(filename, O_RDONLY | O_CREAT, S_IRWXU); break;
            }
            else if(strcmp("w+", permit) == 0) { //⑥ 쓰기 전용
                fd = open(filename, O_WRONLY | O_CREAT, S_IRWXU); break;
            }
            else if(strcmp("rw+", permit) == 0) { //⑦ 읽고 쓰기
                fd = open(filename, O_RDWR | O_CREAT, S_IRWXU); break;
            }
        }
    }
}

```

```

    }
    else {
        printf("Client >> No such permission\n"); continue;
    }
}

//⑧ 해당 파일을 읽어와 저장한다.
char *ack = "ACK";
if(send(cli_sock, ack, 3, 0) < 0) errquit("Client >> Write failed");
while((file_len = read(cli_sock, buf, 1)) == 1) write(fd, buf, 1);
printf("Client >> File download completed\n");
}

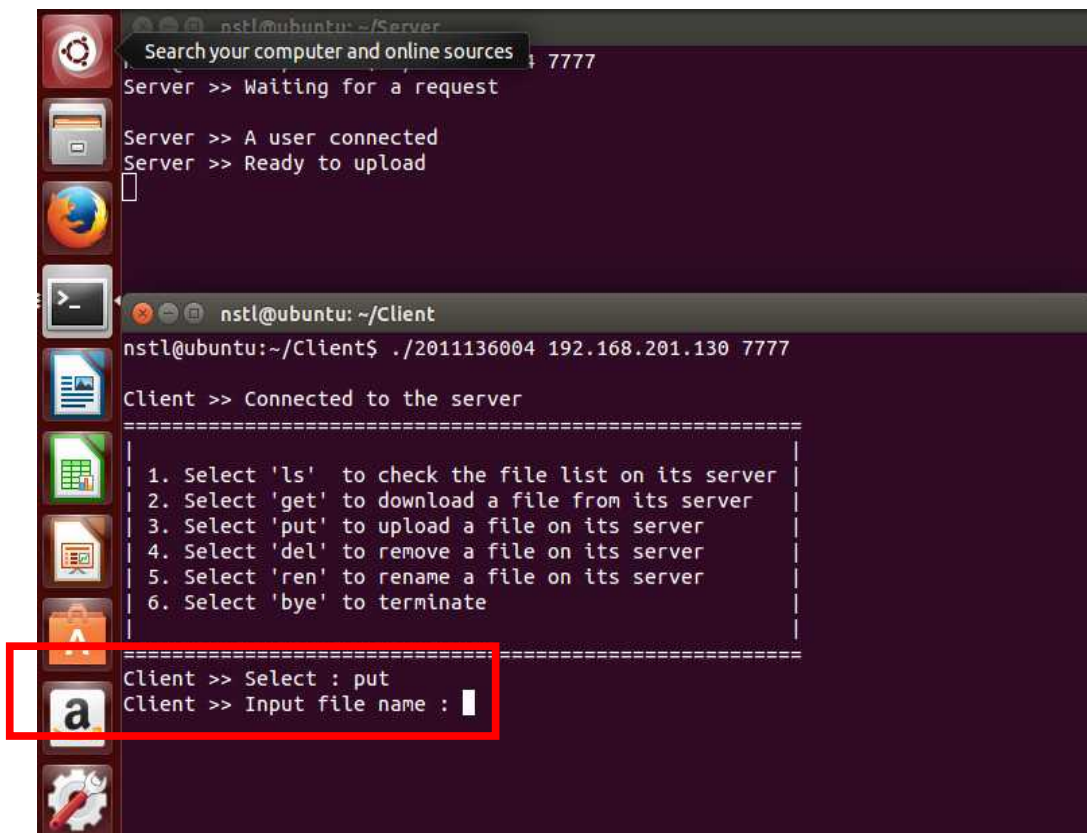
//⑨ 해당 파일이 없을 경우 처리한다.
if(strstr(flag, "NAK") != NULL) {
    printf("Client >> No such file or directory\n"); continue;
}

//⑩ 파일 퍼미션에 제한될 경우 처리한다.
if(strstr(flag, "BAN") != NULL) {
    printf("Client >> File access denied\n"); continue;
}
}
}

```

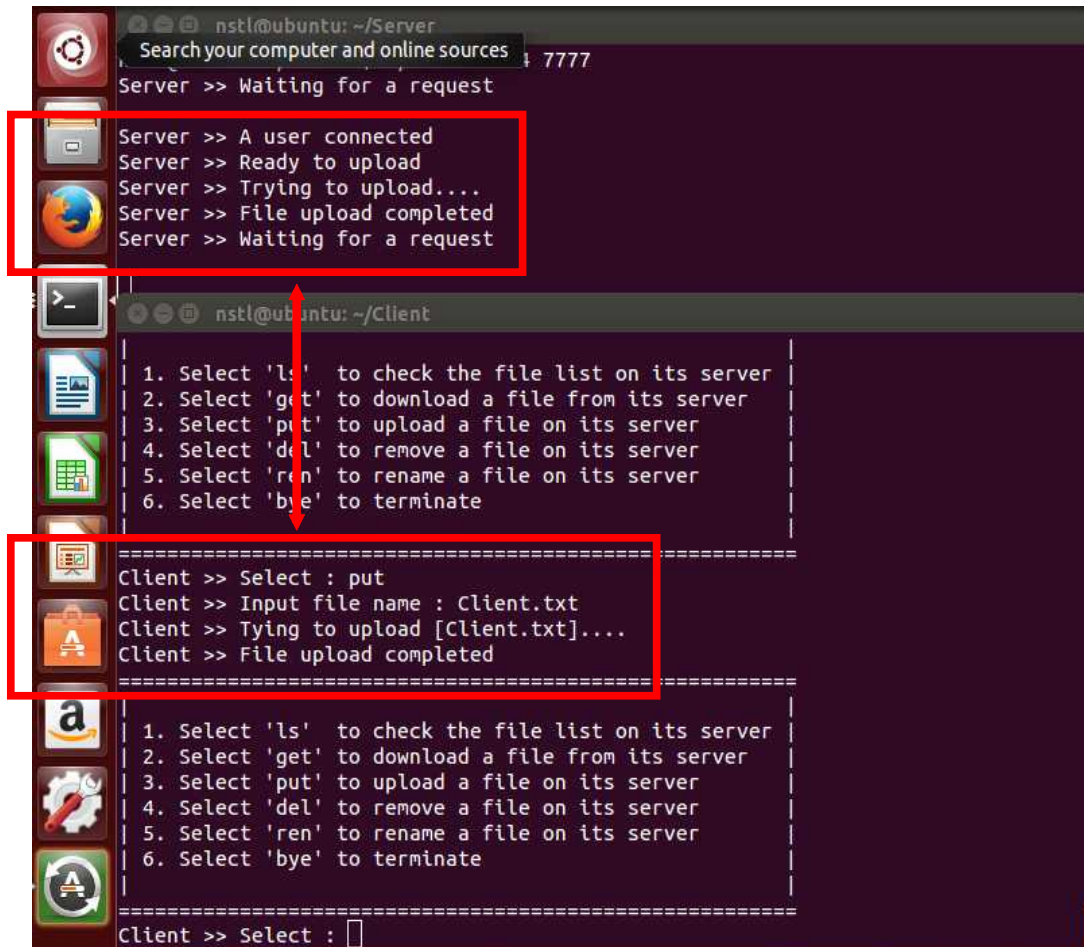
4. put 명령어 실행

4.1. 클라이언트가 put 명령어를 입력하였을 때



4.2. 클라이언트가 파일 이름을 입력하였을 때

- 서버 폴더로 클라이언트 디렉토리에 있는 Client.txt 파일을 업로드 하기 위해 해당 파일 이름을 입력한다.



```
nstl@ubuntu: ~/Server
Search your computer and online sources 7777
Server >> Waiting for a request

Server >> A user connected
Server >> Ready to upload
Server >> Trying to upload....
Server >> File upload completed
Server >> Waiting for a request

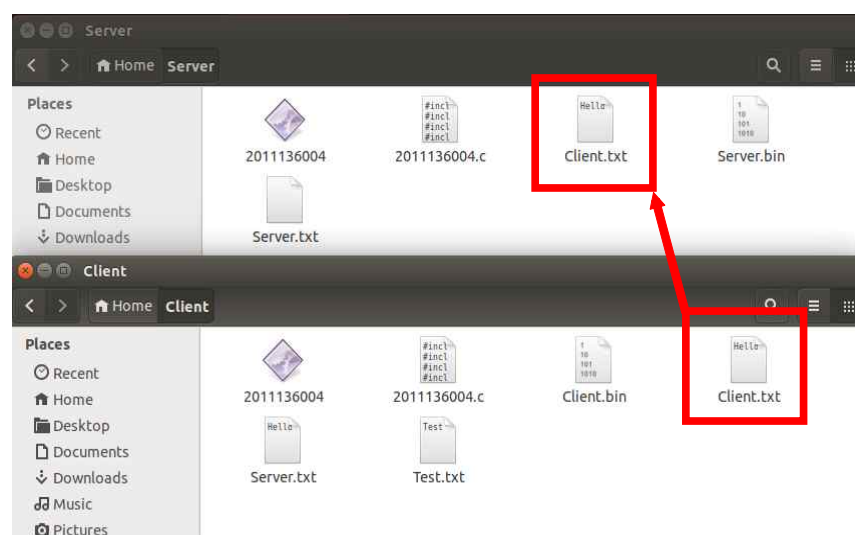
nstl@ubuntu: ~/Client

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : put
Client >> Input file name : Client.txt
Client >> Tying to upload [Client.txt]....
Client >> File upload completed

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : 
```



< Client 폴더의 Client.txt와 Server 폴더의 Client.txt >



< Client와 Server 폴더의 Client.txt의 내용 >

4.3. put 명령어 코드 설명

4.3.1. 서버 코드

```
//① 클라이언트가 put 명령어를 입력하였을 경우
else if(strstr(select, "put") != NULL) {
    printf("Server >> Ready to upload\n");
    //② 클라이언트로부터 파일 이름을 수신한다.
    file_len = recv(cli_sock, filename, 30, 0);
    filename[file_len] = 0;
    //③ 클라이언트가 해당 이름을 파일을 가지고 있지 않을 경우
    if(strstr(filename, "NAK") != NULL) {
        printf("Server >> No such file or directory\n"); continue;
    }
    //④ 해당 파일의 내용을 읽어온다.
    fd = open(filename, O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
    if(fd == -1) {
        errquit("Server >> File open failed "); continue;
    }
    else {
        printf("Server >> Trying to upload....\n");
        while((file_len = read(cli_sock, buf, 1)) == 1) write(fd, buf, 1);
        printf("Server >> File upload completed\n");
    }
}
```

4.3.2. 클라이언트 코드

```
else if((strcmp("put", select, 3) == 0)) {
    printf("Client >> Input file name : "); scanf("%s", filename);
    printf("Client >> Tying to upload [%s]....\n", filename);
    file_len = strlen(filename) + 1;
    strcpy(buf, filename);
    fd = open(filename, O_RDONLY);
    //① put 하려는 파일이 없을 경우
```

```

if(fd == -1) {
    //② nak 플래그 송신
    char *nak = "NAK";
    printf("Client >> No such file or directory\n");
    if(send(cli_sock, nak, 30, 0) < 0) errquit("Client >> Write failed ");
    continue;
}
//③ 해당 이름을 가진 파일이 있을 경우 파일 이름을 송신하고 파일 전송
else {
    if(send(cli_sock, filename, 30, 0) < 0) errquit("Client >> Write failed");
    while((file_len = read(fd, buf, 1)) == 1) write(cli_sock, buf, 1);
    printf("Client >> File upload completed\n");
}
}

```

5. del 명령어 실행

5.1. 클라이언트가 del 명령어를 입력하였을 때

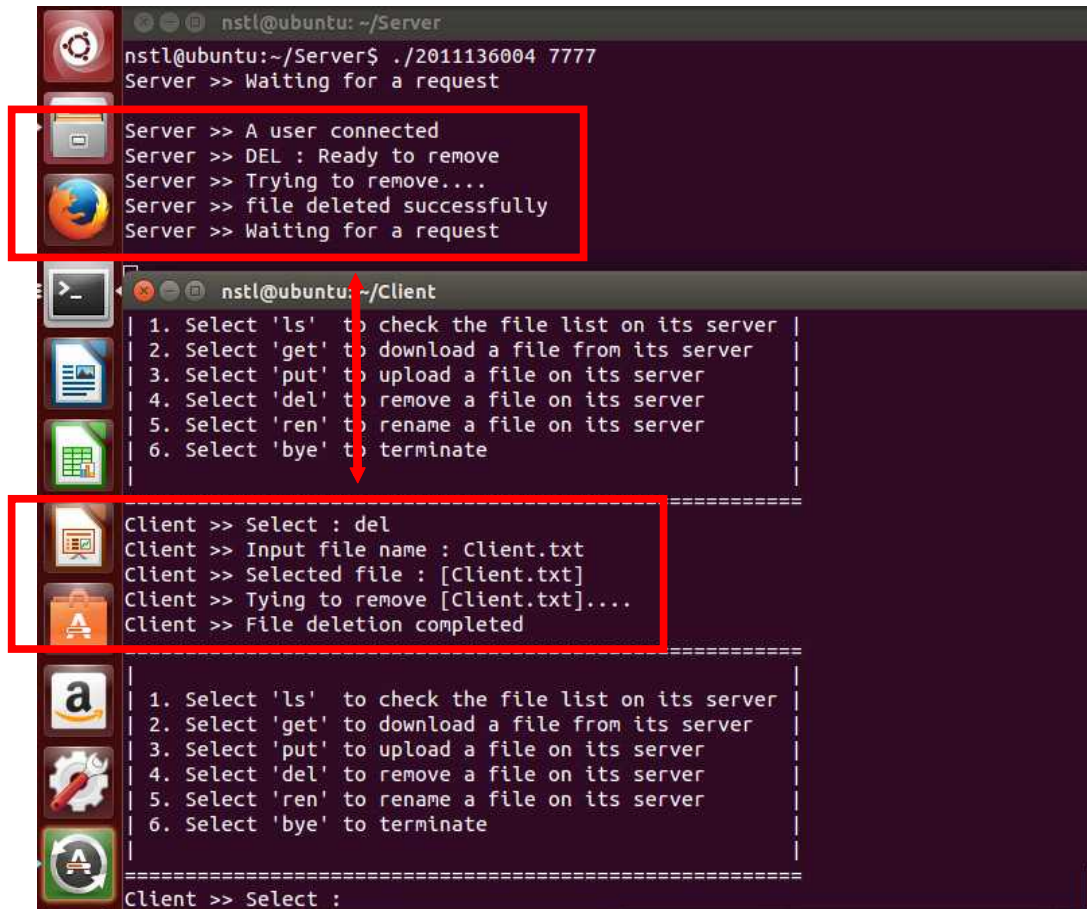
```

nssl@ubuntu: ~/Server
nssl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
Server >> A user connected
Server >> Ready to upload
Server >> Trying to upload....
Server >> File upload completed
Server >> Waiting for a request
Server >> DEL : Ready to remove

nssl@ubuntu: ~/Client
1. Select 's' to check the file list on its server |
2. Select 'et' to download a file from its server |
3. Select 'ut' to upload a file on its server      |
4. Select 'el' to remove a file on its server      |
5. Select 'en' to rename a file on its server      |
6. Select 'ye' to terminate                       |
=====
Client >> Select : put
Client >> Input file name : Client.txt
Client >> Tying to upload [Client.txt]....
Client >> File upload completed
=====
1. Select 's' to check the file list on its server |
2. Select 'et' to download a file from its server |
3. Select 'ut' to upload a file on its server      |
4. Select 'el' to remove a file on its server      |
5. Select 'en' to rename a file on its server      |
6. Select 'ye' to terminate                       |
=====
Client >> Select : del
Client >> Input file name :

```


5.2. 클라이언트가 삭제하려는 파일 이름을 입력하였을 때



```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> DEL : Ready to remove
Server >> Trying to remove....
Server >> file deleted successfully
Server >> Waiting for a request

nstl@ubuntu: ~/Client
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : del
Client >> Input file name : Client.txt
Client >> Selected file : [Client.txt]
Client >> Tying to remove [Client.txt]....
Client >> File deletion completed

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select :
```



< Client.txt 파일이 삭제된 Server 디렉토리 >

5.3. 접근 권한이 제한된 파일을 삭제하려고 할 때

- 다운로드일 경우, 소유주가 root일 때 공개 사용자의 읽기 권한이 없을 때 또는 소유주가 공개 사용자일 때 소유주의 읽기 권한이 없을 때 다운로드가 제한되었다. 반면에, 삭제일 경우, 소유주가 root일 때 공개 사용자의 쓰기 권한이 없거나 소유주가 공개 사용자일 때 소유주의 쓰기 권한이 없을 때 삭제가 제한된다. 즉, *가 r/w/x 아무거나 상관이 없다고 할 때, 조건은 다음과 같다.
- 소유자가 root 이면서 파일 퍼미션이 *-*-*-*-*-* 일 경우
- 소유자가 nstl 이면서 파일 퍼미션이 **-*-*-*-* 일 경우

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ chmod 577 Server.txt
nstl@ubuntu:~/Server$ ls -al
total 48
drwxrwxr-x  2 nstl nstl  4096 Jun 15 23:56 .
drwxr-xr-x 21 nstl nstl  4096 Jun 15 20:41 ..
-rwxrwxr-x  1 nstl nstl 18406 Jun 15 23:54 2011136004
-rw-rw-r--  1 nstl nstl 11596 Jun 15 23:53 2011136004.c
-rw-rw-r--  1 nstl nstl  12 Jun  9 23:52 Server.bin
-r-xrwxrwx  1 nstl nstl   7 Jun  9 21:45 Server.txt
nstl@ubuntu:~/Server$
```

< 서버 디렉토리의 Server.txt의 파일 퍼미션 '-r-xrwxrwx' >

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
Server >> A user connected
Server >> Listing file completed
Server >> Waiting for a request
Server >> DEL : Ready to remove
Server >> Selected file is protected from being removed
Server >> Authority : nstl
Server >> File permission : -r-xrwxrwx
Server >> Waiting for a request

nstl@ubuntu: ~/Client
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
Client >> Select : del
Client >> Input file name : Server.txt
Client >> Selected file : [Server.txt]
Client >> Trying to remove [Server.txt]....
Client >> File access denied

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
```

5.4. del 명령어 코드 설명

5.4.1. 서버 코드

```
//① 클라이언트가 del 명령어를 입력하였을 경우
else if(strstr(select, "del") != NULL) {
    char permission[10];
    printf("Server >> DEL : Ready to remove\n");
    file_len = recv(cli_sock, filename, 30, 0);
    filename[file_len] = 0;
    //② 해당 파일이 서버 디렉토리에 있는지 확인한다.
    if(stat(filename, &file_stat) < 0) {
        char *nak = "NAK";
        if(send(cli_sock, nak, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> No such file or directory\n");
        continue;
    }
    //③ 해당 파일의 퍼미션을 확인한다.
    struct passwd *pw = getpwuid(file_stat.st_uid);
    struct group *gr = getgrgid(file_stat.st_gid);
    if(S_ISDIR(file_stat.st_mode)) permission[0] = 'd';
    else permission[0] = '-';
    if(file_stat.st_mode & S_IRUSR) permission[1] = 'r';
    else permission[1] = '-';
    if(file_stat.st_mode & S_IWUSR) permission[2] = 'w';
    else permission[2] = '-';
    if(file_stat.st_mode & S_IXUSR) permission[3] = 'x';
    else permission[3] = '-';
    if(file_stat.st_mode & S_IRGRP) permission[4] = 'r';
    else permission[4] = '-';
    if(file_stat.st_mode & S_IWGRP) permission[5] = 'w';
    else permission[5] = '-';
    if(file_stat.st_mode & S_IXGRP) permission[6] = 'x';
    else permission[6] = '-';
    if(file_stat.st_mode & S_IROTH) permission[7] = 'r';
    else permission[7] = '-';
    if(file_stat.st_mode & S_IWOTH) permission[8] = 'w';
    else permission[8] = '-';
    if(file_stat.st_mode & S_IXOTH) permission[9] = 'x';
    else permission[9] = '-';
    //④ 앞서 언급한 것 처럼 소유주가 root일 때 공개 사용자의 쓰기 권한이 없거나,
    //소유주가 공개 사용자일 때 소유주의 쓰기 권한이 없을 경우 파일 삭제를 제한한다.
    if( (strstr(pw->pw_name, "root") != NULL)
        && (permission[8] == '-') || (strstr(pw->pw_name, "nsth") != NULL)
        && (permission[2] == '-') ) {
        char *ban = "BAN";
```



```

        if(send(cli_sock, ban, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> Selected file is protected from being removed\n");
        printf("Server >> Authority : %s\n", pw->pw_name);
        printf("Server >> Access permission : %s\n", permission);
        continue;
    }
    //⑤ 해당 이름을 가진 파일이 있을 경우 해당 파일을 삭제한다.
    fd = open(filename, O_RDONLY);
    if(fd == -1) {
        char *nak = "NAK";
        if(send(cli_sock, nak, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> No such file or directory\n");
        continue;
    }
    else {
        int result;
        char *ack = "ACK";
        if(send(cli_sock, ack, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> Trying to remove...\n");
        //⑤ c에서 제공해주는 파일 삭제 메소드 remove() 사용
        result = remove(filename);
        if(result == 0) printf("Server >> file deleted successfully\n");
        else errquit("Server >> file deletion failed ");
    }
}

```

5.4.2. 클라이언트 코드

```

else if((strcmp("del", select, 3) == 0)) {
    printf("Client >> Input file name : "); scanf("%s", filename);
    printf("Client >> Selected file : [%s]\n", filename);
    file_len = strlen(filename) + 1;
    //① 삭제하려는 파일 이름을 전송한다.
    if(send(cli_sock, filename, 30, 0) < 0) errquit("Client >> Write failed");
    flag_len = recv(cli_sock, flag, 3, 0);
    flag[flag_len] = 0;
    printf("Client >> Tying to remove [%s]...\n", filename);
    //② 서버로부터 ack 메시지를 받을 경우, 완료 메시지를 출력한다.
    if(strstr(flag, "ACK") != NULL) printf("Client >> File deletion completed\n");
    //② 서버에 해당 파일이 없을 경우 처리한다.
    if(strstr(flag, "NAK") != NULL) {
        printf("Client >> No such file or directory\n"); continue;
    }
    //③ 해당 파일의 삭제 권한이 없을 경우 처리한다.
    if(strstr(flag, "BAN") != NULL) {
        printf("Client >> File access denied\n"); continue;
    }
}

```

```
}  
}
```

6. ren 명령어 실행

6.1. 클라이언트가 ren 명령어를 입력하였을 때

```
nstl@ubuntu: ~/Server  
nstl@ubuntu:~/Server$ ./2011136004 7777  
Server >> Waiting for a request  
Server >> A user connected  
Server >> REN : Ready to rename  
  
nstl@ubuntu: ~/Client  
nstl@ubuntu:~/Client$ ./2011136004 192.168.201.130 7777  
Client >> Connected to the server  
=====
```

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

```
=====
```

```
Client >> Select : ren  
Client >> Input file name : 
```

6.2. 클라이언트가 이름을 변경하려는 파일을 입력하였을 때

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

Server >> A user connected
Server >> REN : Ready to rename
[]

nstl@ubuntu: ~/Client
nstl@ubuntu:~/Client$ ./2011136004 192.168.201.130 7777
Client >> Connected to the server

=====
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
=====

Client >> Select : ren
Client >> Input file name : Server.txt
Client >> Input new file name :
```

6.3. 클라이언트가 변경할 이름을 입력하였을 때

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

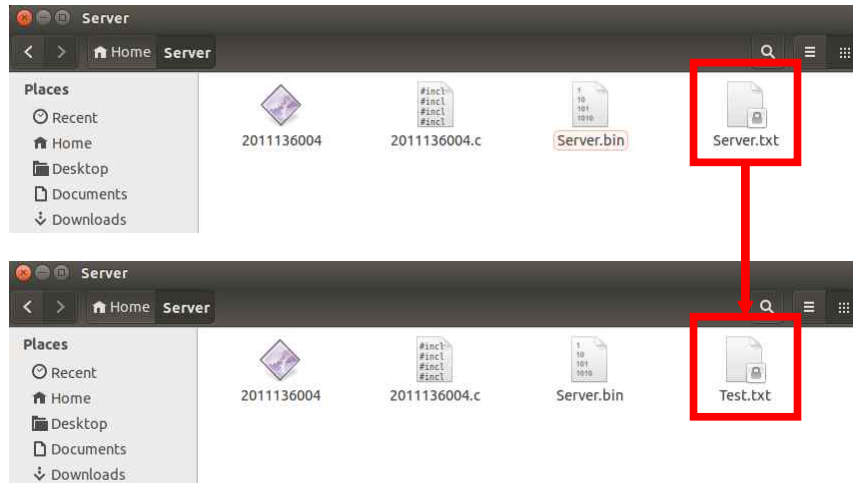
Server >> A user connected
Server >> REN : Ready to rename
Server >> Trying to rename....
Server >> file name changed successfully
Server >> Waiting for a request

nstl@ubuntu: ~/Client
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : ren
Client >> Input file name : Server.txt
Client >> Input new file name : Test.txt
Client >> Trying to change [Server.txt] to [Test.txt]....
Client >> File name changed successfully

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select :
```



< Server 디렉토리 내 Sever.txt의 이름을 Test.txt로 변경 >

6.4. ren 명령어 코드 설명

6.4.1. 서버 코드

```
//① 클라이언트가 ren 명령어를 입력하였을 경우
else if(strstr(select, "ren") != NULL) {
    char new_filename[30];
    int new_file_len;
    printf("Server >> REN : Ready to rename\n");
    //② 기존 파일 이름을 수신한다.
    file_len = recv(cli_sock, filename, 30, 0);
    filename[file_len] = 0;
    //③ 변경할 파일 이름을 수신한다.
    new_file_len = recv(cli_sock, new_filename, 30, 0);
    new_filename[new_file_len] = 0;
    fd = open(filename, O_RDONLY);
    //④ 파일이 존재하지 않을 경우 처리한다.
    if(fd == -1) {
        char *nak = "NAK";
        if(send(cli_sock, nak, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> No such file or directory\n");
        continue;
    }
    //⑤ 파일이 존재할 경우 c에서 제공하는 rename()메소드 활용하여 파일 이름을 변경한다.
    else {
        int result;
        char *ack = "ACK";
        if(send(cli_sock, ack, 3, 0) < 0) errquit("Server >> Write failed");
        printf("Server >> Trying to rename...\n");
        result = rename(filename, new_filename);
    }
}
```

```

        if(result == 0) printf("Server >> file name changed successfully\n");
        else errquit("Server >> file rename failed ");
    }
}

```

6.4.2. 클라이언트 코드

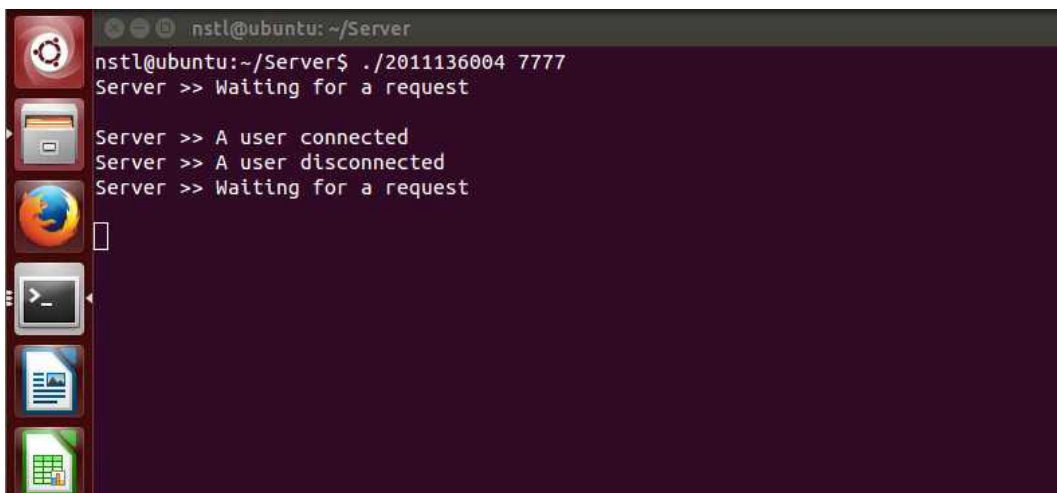
```

else if((strcmp("ren", select, 3) == 0)) {
    char new_filename[30];
    int new_file_len;
    printf("Client >> Input file name : "); scanf("%s", filename);
    file_len = strlen(filename) + 1;
    //① 기존 파일 이름을 전송한다.
    if(send(cli_sock, filename, 30, 0) < 0) errquit("Client >> Write failed");
    printf("Client >> Input new file name : "); scanf("%s", new_filename);
    new_file_len = strlen(new_filename) + 1;
    //② 새로운 파일 이름을 전송한다.
    if(send(cli_sock, new_filename, 30, 0) < 0) errquit("Client >> Write failed");
    printf("Client >> Trying to change [%s] to [%s]...\n", filename, new_filename);
    flag_len = recv(cli_sock, flag, 3, 0);
    flag[flag_len] = 0;
    //③ 파일 이름을 성공적으로 변경한 경우 처리한다.
    if(strstr(flag, "ACK") != NULL)
        printf("Client >> File name changed successfully\n");
    //④ 해당 파일이 존재하지 않을 경우 처리한다.
    if(strstr(flag, "NAK") != NULL) {
        printf("Client >> No such file or directory\n"); continue;
    }
}
}

```

7. bye 명령어 실행

7.1. 서버 화면



```

nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
Server >> A user connected
Server >> A user disconnected
Server >> Waiting for a request

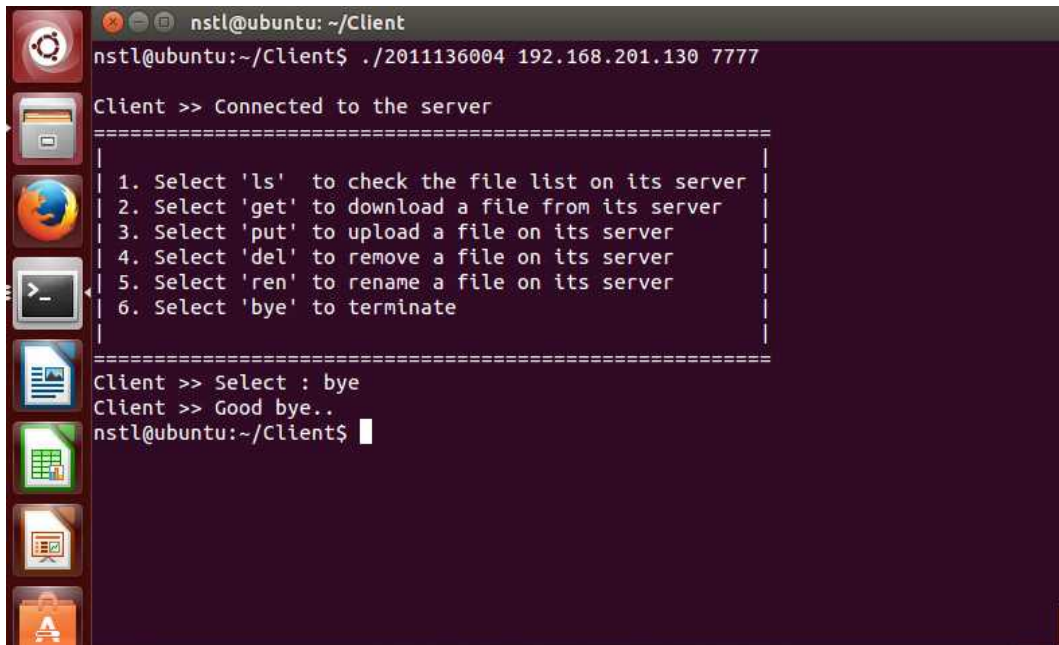
```

7.1.1. 유저 퇴장 메시지 출력

- 클라이언트로부터 bye 명령어를 입력받았을 경우, 유저가 퇴장했다는 메시지를 서버 화면에 출력한다.

```
//① 유저 퇴장을 공고
else if(strstr(select, "bye") != NULL) {
    printf("Server >> A user disconnected\n");
}
```

7.2. 클라이언트 화면



7.2.1. 서버로의 접속 종료

- 클라이언트가 bye 명령어를 입력할 경우, 퇴장 메시지를 출력하고 서버로의 접속을 끝낸다.

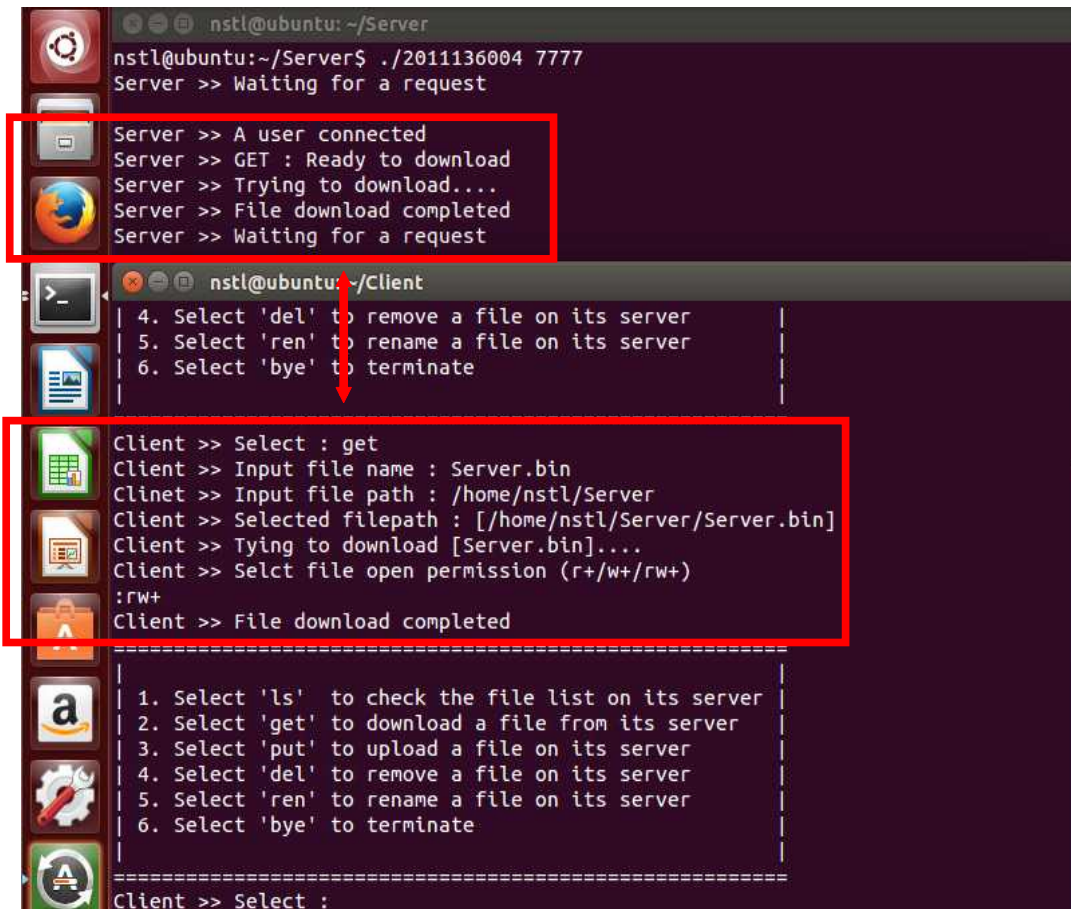
```
//① 퇴장을 공고하고 소켓을 닫음
else if(strcmp("bye", select, 3) == 0) {
    printf("Client >> Good bye..\n");
    close(fd);
    close(cli_sock);
    exit(0);
}
```


8. 모든 파일 확장자 제어

- 앞에서 언급했던 모든 명령어들을 .txt 파일만을 활용하여 실습하였다. 하지만, 해당 FTP 서버와 클라이언트는 .txt파일 외에도 모든 파일 확장자를 제어할 수 있다.

8.1. bin 확장자

8.1.1. get



```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request

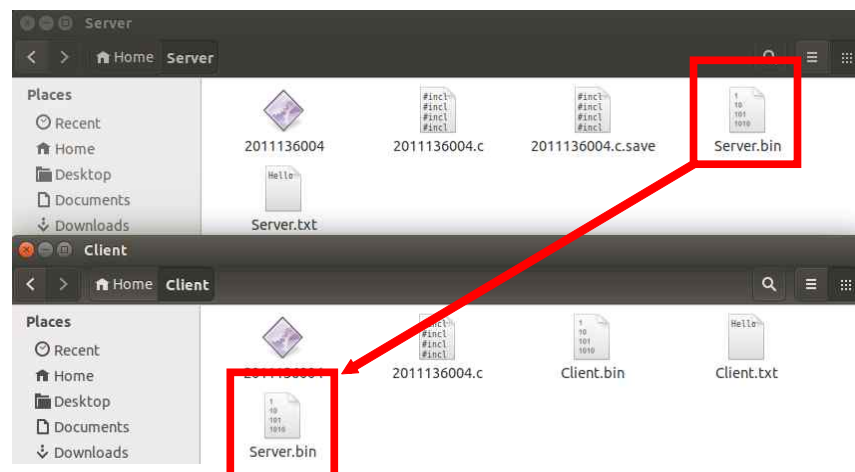
Server >> A user connected
Server >> GET : Ready to download
Server >> Trying to download....
Server >> File download completed
Server >> Waiting for a request

nstl@ubuntu: ~/Client
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select : get
Client >> Input file name : Server.bin
Client >> Input file path : /home/nstl/Server
Client >> Selected filepath : [/home/nstl/Server/Server.bin]
Client >> Tying to download [Server.bin]....
Client >> Selct file open permission (r+/w+/rw+)
:r+w
Client >> File download completed

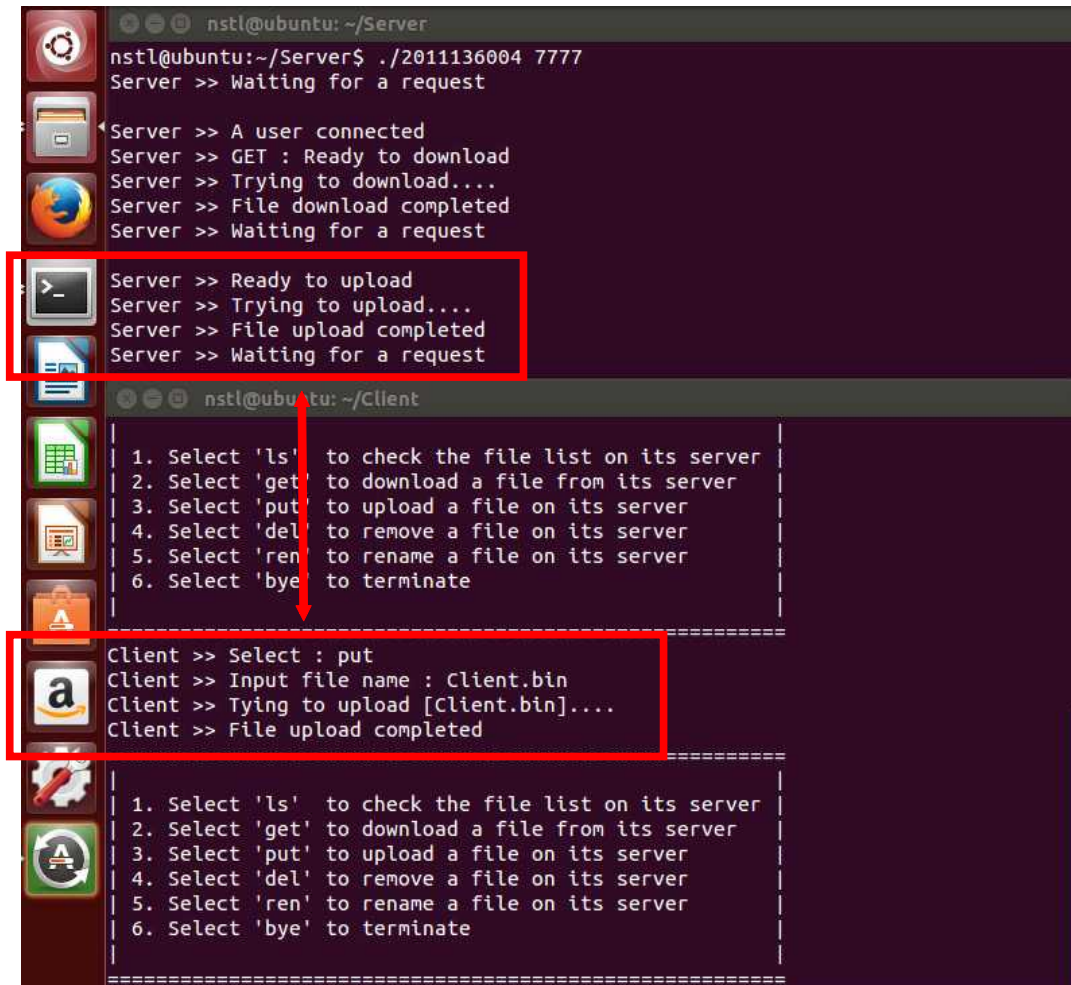
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

Client >> Select :
```



< Server 폴더의 Server.bin과 Client 폴더의 Server.bin >

8.1.2. put

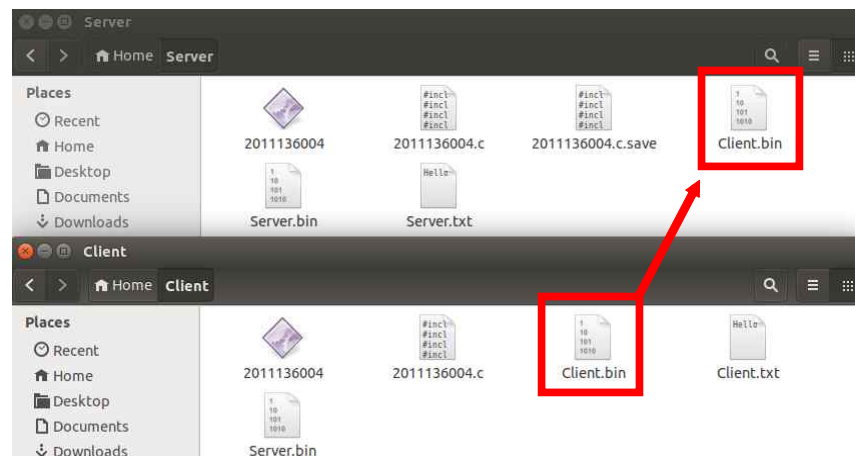


The image shows two terminal windows. The top window is the 'Server' terminal, and the bottom window is the 'Client' terminal. The 'Server' terminal shows the process of waiting for a request, receiving a GET request, downloading a file, and then being ready to upload. The 'Client' terminal shows a menu of options, selecting 'put', entering the filename 'Client.bin', and completing the upload. A red box highlights the upload completion in the Server terminal, and another red box highlights the selection of 'put' and the filename entry in the Client terminal. A red arrow points from the Client terminal's 'File upload completed' message to the Server terminal's 'File upload completed' message.

```
nstl@ubuntu: ~/Server
nstl@ubuntu:~/Server$ ./2011136004 7777
Server >> Waiting for a request
Server >> A user connected
Server >> GET : Ready to download
Server >> Trying to download....
Server >> File download completed
Server >> Waiting for a request
Server >> Ready to upload
Server >> Trying to upload....
Server >> File upload completed
Server >> Waiting for a request

nstl@ubuntu: ~/Client
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
Client >> Select : put
Client >> Input file name : Client.bin
Client >> Tying to upload [Client.bin]....
Client >> File upload completed

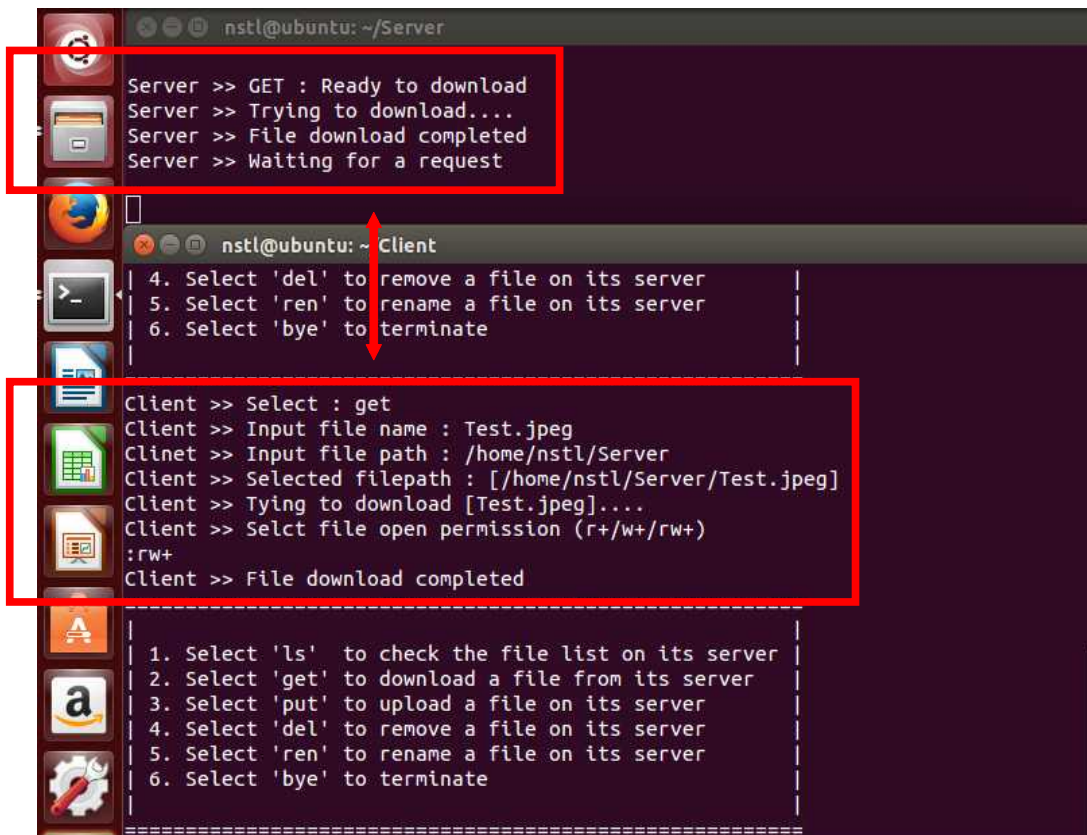
1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
```



< Server 폴더의 Client.bin과 Client 폴더의 Client.bin >

8.2. jpg 확장자

8.2.1. get

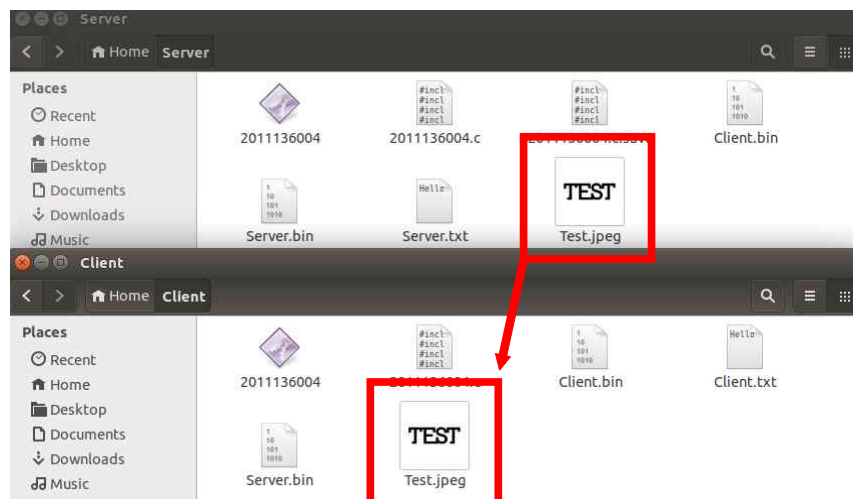


```
nstl@ubuntu: ~/Server
Server >> GET : Ready to download
Server >> Trying to download....
Server >> File download completed
Server >> Waiting for a request

nstl@ubuntu: ~ Client
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate

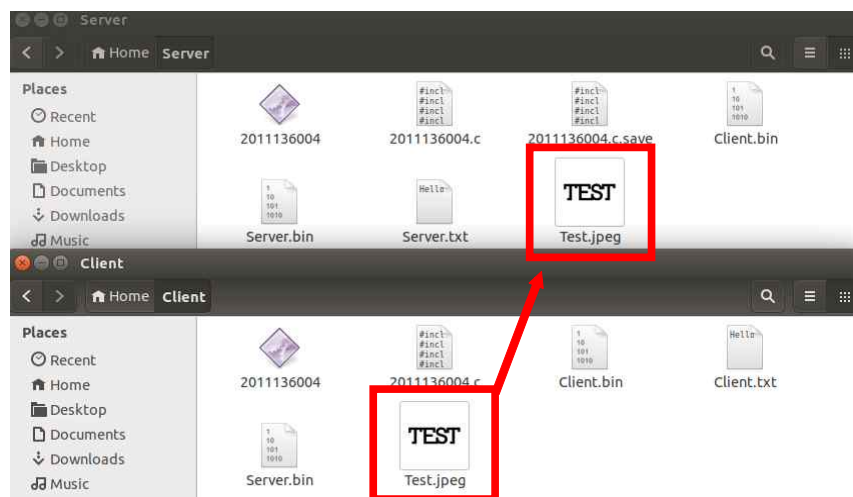
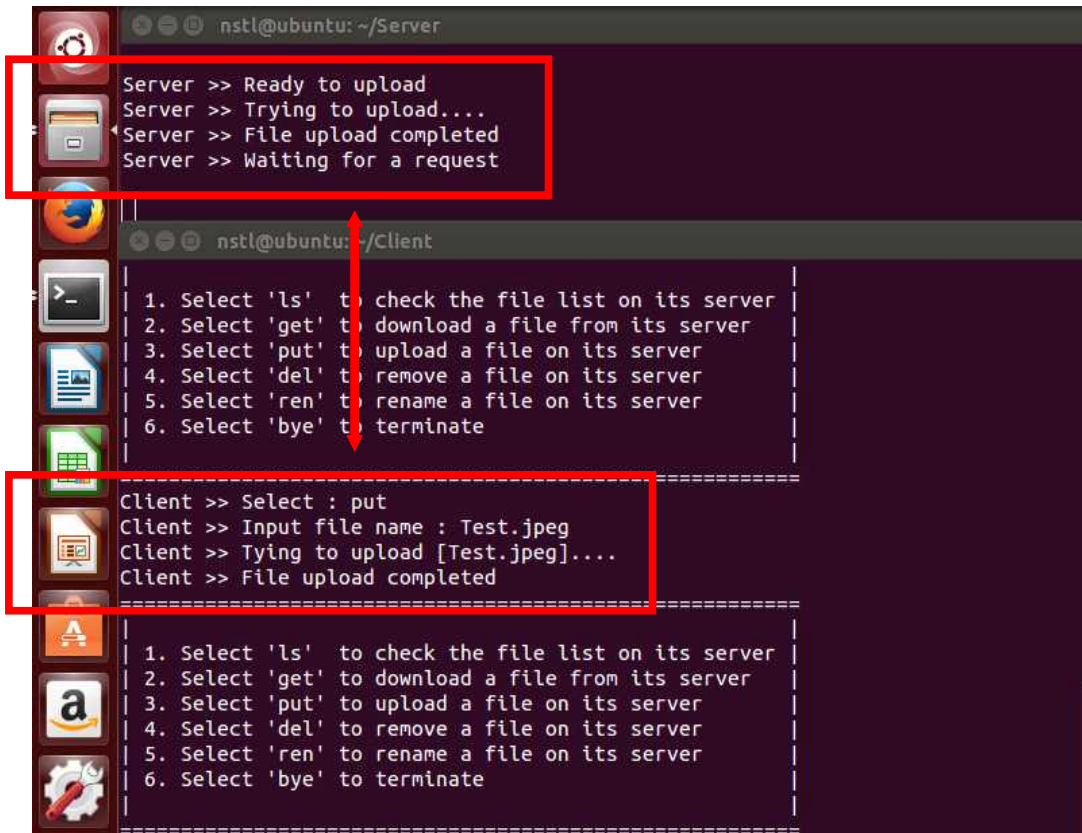
Client >> Select : get
Client >> Input file name : Test.jpeg
Client >> Input file path : /home/nstl/Server
Client >> Selected filepath : [/home/nstl/Server/Test.jpeg]
Client >> Tying to download [Test.jpeg]....
Client >> Selct file open permission (r+/w+/rw+)
:r+w+
Client >> File download completed

1. Select 'ls' to check the file list on its server
2. Select 'get' to download a file from its server
3. Select 'put' to upload a file on its server
4. Select 'del' to remove a file on its server
5. Select 'ren' to rename a file on its server
6. Select 'bye' to terminate
```



< Server 폴더의 Test.jpeg와 Client 폴더의 Test.jpeg >

8.2.2. put



< Server 폴더의 Test.jpeg와 Client 폴더의 Test.jpeg >

III. 결론

- C에서 제공해주는 소켓 라이브러리 함수를 활용하여 FTP 기반 서버와 클라이언트가 통신을 작성할 수 있었다. 서론에서 언급했던 조건을 모두 충족시킬 수 있도록 프로그래밍 하였고, 모두 정상 작동하는 것을 확인하였다. 조건 충족 내용은 다음과 같다.

1. 클라이언트가 자신의 디렉토리에 있는 파일을 서버 디렉토리에 업로드할 수 있도록 하였다(put 명령어). 그리고 이 때, 해당 파일이 존재하지 않으면 예외 처리하였다.
2. 클라이언트가 서버 디렉토리 뿐만 아니라 모든 경로에서 파일을 다운받을 수 있도록 하였다(get 명령어). 그리고 이 때, 해당 파일이 존재하지 않을 경우 예외 처리를 해주었고, 파일 퍼미션을 고려하여 다운로드 가능한 파일일 경우 다운로드하고 그렇지 않을 경우 예외 처리를 해주었다. 마지막으로, 파일을 클라이언트 디렉토리로 읽어올 때 읽기 전용, 쓰기 전용, 읽고 쓰기 등 파일 접근 제어를 할 수 있도록 하였다.
3. 클라이언트가 서버 디렉토리에 있는 파일의 리스트를 확인할 수 있도록 하였다(ls 명령어).
4. 클라이언트가 서버 디렉토리에 있는 파일을 삭제할 수 있도록 하였다(del 명령어). 그리고 이 때, 해당 파일이 존재하지 않을 경우 예외처리 하였다.
5. 클라이언트가 서버 디렉토리에 있는 파일 이름을 변경할 수 있도록 하였다(ren 명령어). 그리고 이 때, 해당 파일이 존재하지 않을 경우 예외처리 하였다.
6. 클라이언트가 서버 접속을 끊고 퇴장할 수 있도록 하였다(bye 명령어).

FTP 서버는 따로 접속을 종료하지 않는 이상 계속해서 실행되며 클라이언트는 언제든지 서버에 접속하여 위의 명령어들을 사용하여 파일을 컨트롤할 수 있다.

IV. 부록

1. 고찰

- 이번 기말고사 대체 과제의 목표는 FTP 기반 서버와 클라이언트를 작성하여 통신을 실습하는 것이었다. 따라서 FTP 클라이언트가 FTP 서버에 접속하여 파일을 다운로드 받을 수 있고(get), 업로드할 수 있도록 하였다(put). 또한, 서버 폴더의 파일 리스트를 출력 한다거나(ls) 파일을 삭제(del) 및 이름을 변경(ren)할 수 있도록 추가 구현 하였다. 마지막으로 클라이언트가 용무를 모두 마친 후 bye 명령어를 입력하여 서버로의 접속을 종료하면서 간단한 FTP 서버와 클라이언트 프로그램을 작성할 수 있었다.

과제를 수행하면서 가장 어려웠던 점은, 서버와 클라이언트 간 주고받는 플래그 메시지가 많았던 것이고, 파일이 없을 경우와 파일 퍼미션 등 고려해야 할 예외가 정말 많았다는 점이다. 예를 들어, 클라이언트가 get 명령어를 사용할 때, 우선 해당 파일이 있는지 없는지를 확인해야한다. 그리고 파일 퍼미션을 고려하여 접근이 제한되면 다운로드 받을 수 없도록 해야 한다. 이 과정을 모두 거쳐야 다운로드 기능을 정확히 수행할 수 있다. 이렇듯, 여러 예외를 처리해주는 데 많은 시간을 쏟아 부었다.

하지만, 이러한 문제에 대하여 전공 서적을 참고하고 검색을 통해 방법을 찾아 적용하여 해결할 수 있었다. 한 학기 동안 네트워크 프로그래밍을 수강하면서 많은 것을 배웠다. 특히, 서버와 클라이언트 간 TCP 통신을 이루기 위해 사용하는 bind(), listen(),

accept(), socket(), connect() 등 다양한 C언어의 소켓 라이브러리 함수에 대해 완벽히 이해할 수 있었다. 또한, TCP에 국한되지 않고 UDP도 사용해보고 시그널, 프로세스 등 다양하게 학습하고 실습할 수 있어서 좋은 기회가 되었다.

컴퓨터공학 학생으로서 이 경험은 학교 밖으로 나가 취업 후에도 분명 큰 힘이 될 것이라고 다시 한 번 생각하게 되었다.

2. 참고 자료

참고한 내용	사이트 주소 또는 서적
C 소켓 라이브러리 함수	http://forum.falinux.com/zbxe/index.php?mid=C_LIB&page=2
	http://rotapple.tistory.com/6
소켓 프로그래밍 이해	김화중(2004).컴퓨터 네트워크 프로그래밍. 홍릉과학 출판사
ls 명령어 수행을 위한 C에서 파일 리스트 가져오기	https://stackoverflow.com/questions/612097/how-can-i-get-the-list-of-files-in-a-directory-using-c-or-c
파일 퍼미션 관련	https://www.conory.com/note_linux/19194
	https://stackoverflow.com/questions/10323060/printing-file-permissions-like-ls-l-using-stat2-in-c
파일 삭제 메소드	https://www.tutorialspoint.com/c_standard_library/c_function_remove.htm
파일 이름 변경 메소드	https://www.tutorialspoint.com/c_standard_library/c_function_rename.htm