

시스템 프로그래밍 실습

[Assignment #2-1: basic socket]

Class : [금요일 1, 2 교시]

Professor : [최상호 교수님]

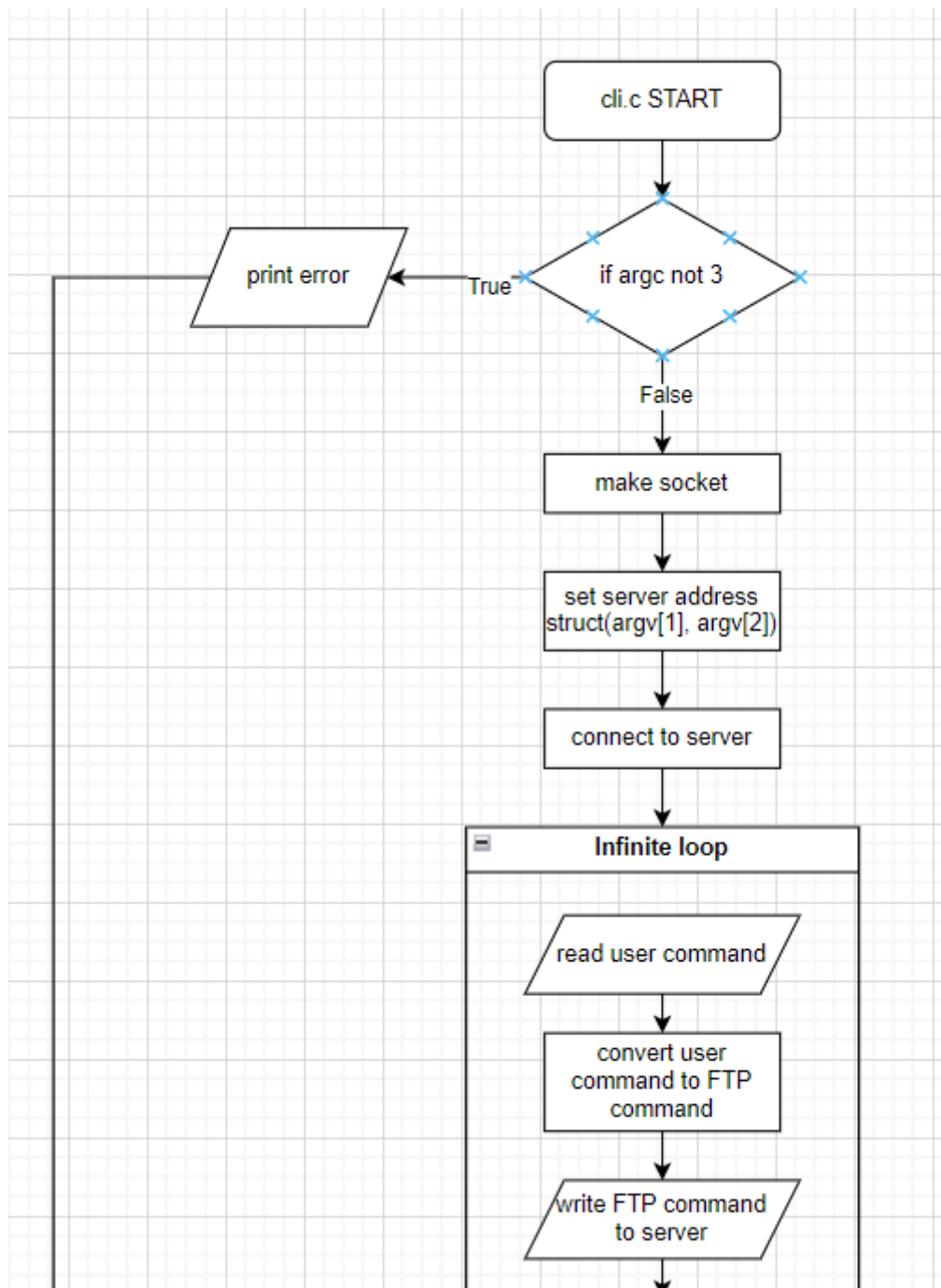
Student ID : [2020202034]

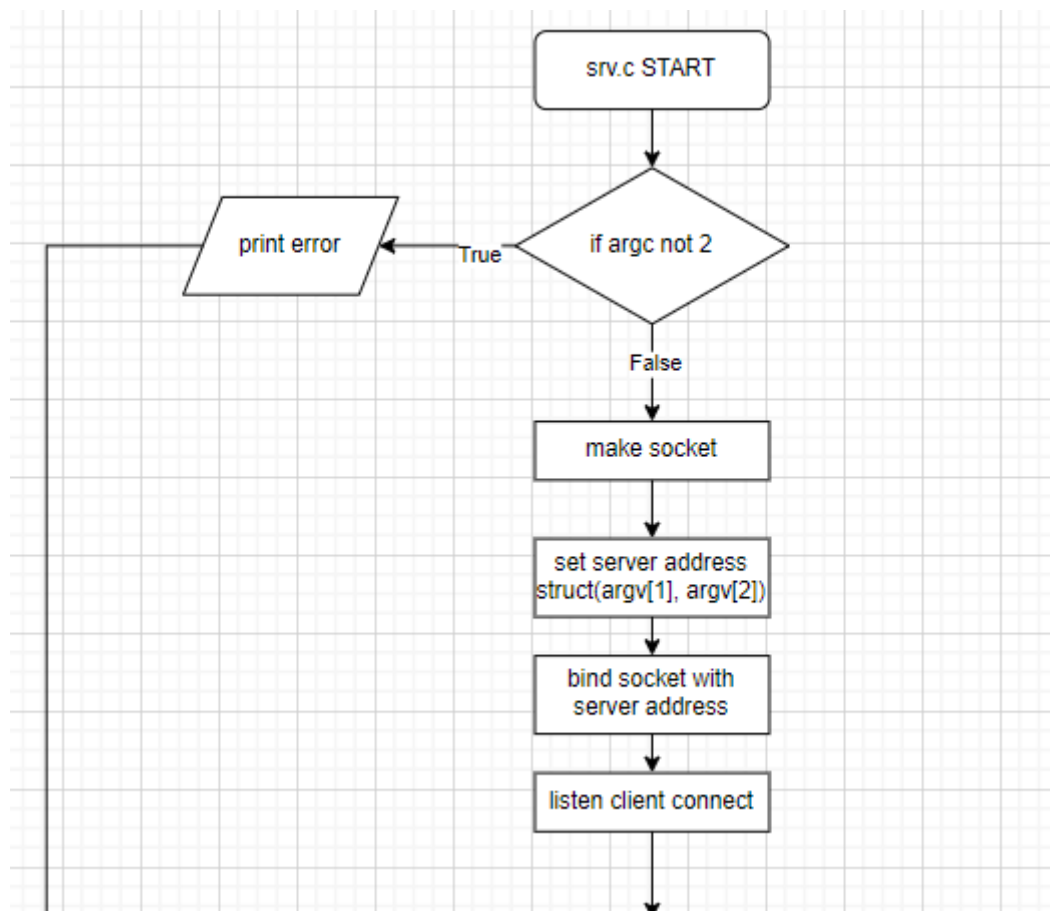
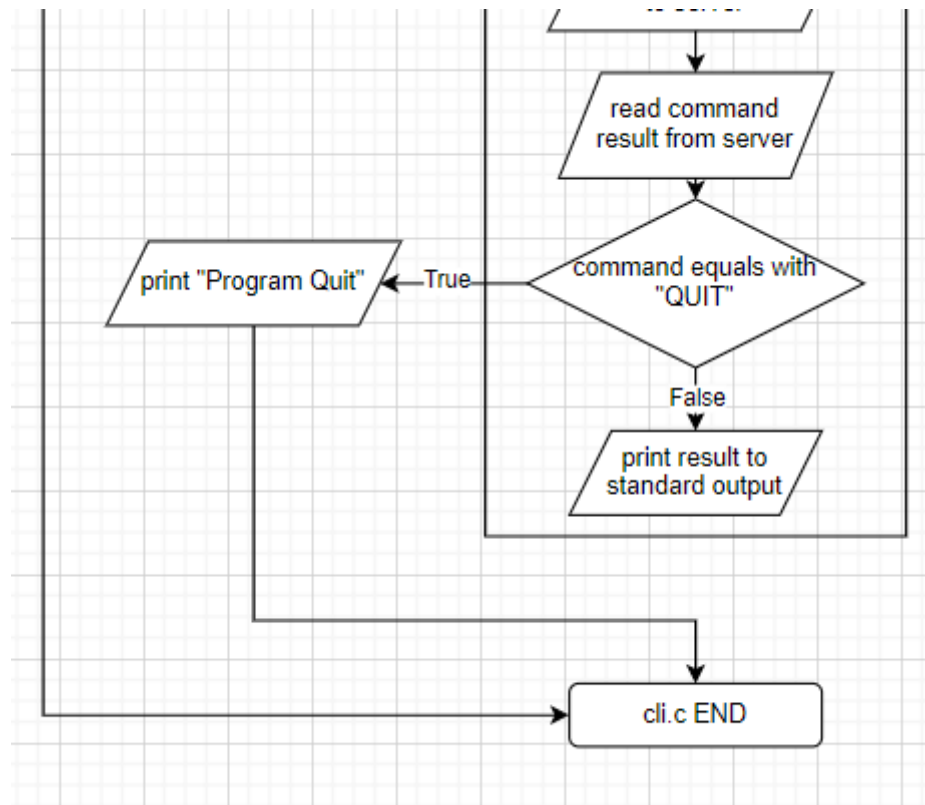
Name : [김태완]

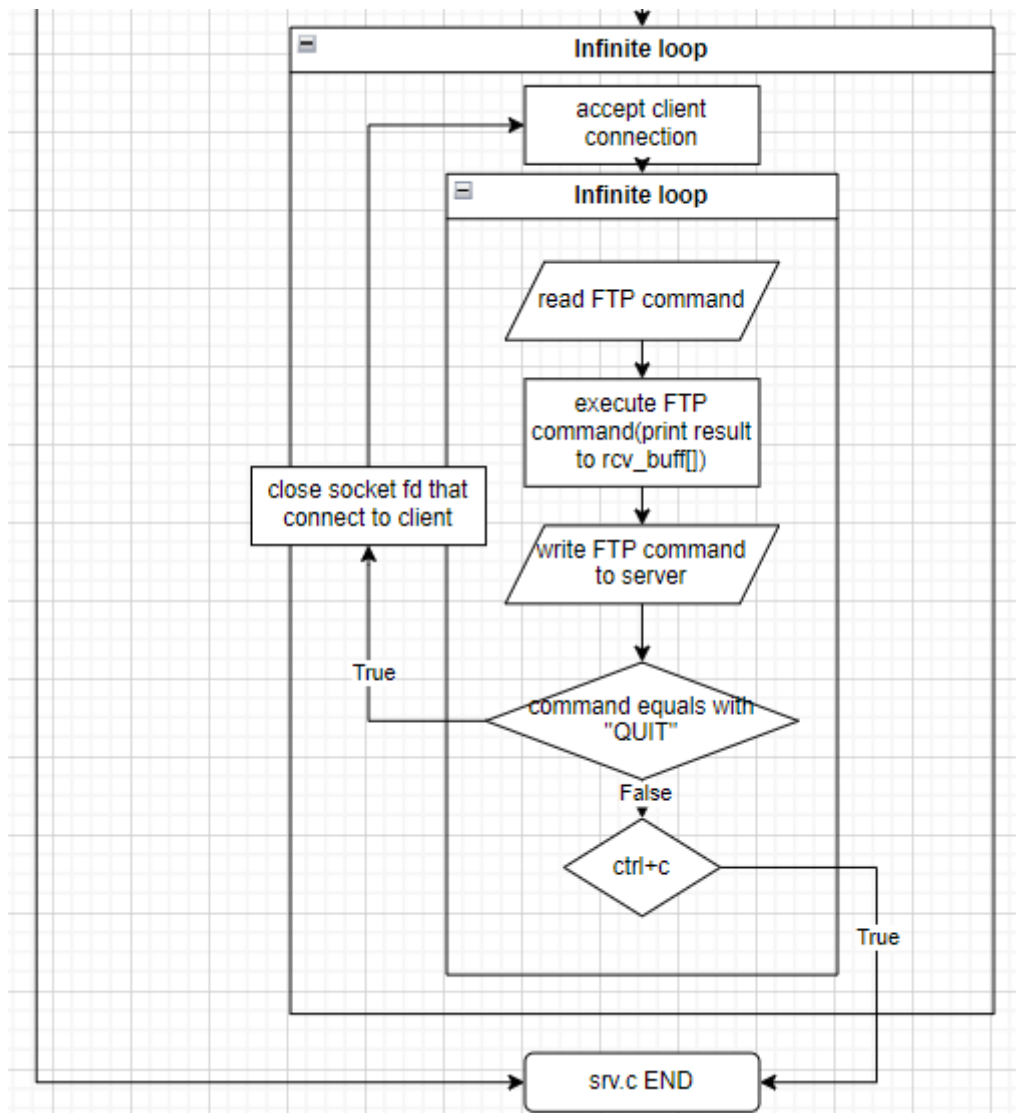
Introduction

이번 과제는 socket programming 을 간단한 client, server program 을 구현함으로써 실습해보는 과제입니다. application layer 에 있는 process 가 socket 을 통해 다른 process 와 데이터를 주고받는 방식을 이해하고, 이를 단계적으로 만들어주는 함수(socket(), bind(), listen(), connect(), accept() 등)들을 사용해보면서 익히는 과제입니다. 이는 후에 FTP server 를 만들 때, client 와 server 가 socket 을 이용해 데이터를 주고받을 때 이용될 것입니다.

Flow chart







Pseudo code

cli.c

Main(argc, argv)

sockfd, len, server_addr;

str = NULL

buff[], cmd_buff[], rcv_buff[];

if argc is not 3

print error and exit

sockfd = CreateSocket()

if sockfd is invalid

PrintErrorAndExit("can't create socket")

server_addr = SetServerAddress(argv[1], argv[2])

if ConnectToServer(sockfd, server_addr) is failed

PrintErrorAndExit("can't connect")

ResetBuffers(buff, cmd_buff, rcv_buff)

PrintPrompt("> ")

while True

n = ReadFromUser(buff)

if n is less than 0

PrintErrorAndExit("read() error!!")

NULLTerminateBuffer(buff, n)

ConvertUserCommandToFTPCommand(buff, cmd_buff)

if WriteToServer(sockfd, cmd_buff) is failed

PrintErrorAndExit("write() error!!")

```
n = ReadFromServer(sockfd, rcv_buff)
```

```
if n is less than 0
```

```
    PrintErrorAndExit("read() error")
```

```
NULLTerminateBuffer(rcv_buff, n)
```

```
if rcv_buff is "QUIT"
```

```
    PrintMessage("Program quit!!")
```

```
    CloseSocket(sockfd)
```

```
    Exit(0)
```

```
PrintResult(rcv_buff)
```

```
ResetBuffers(buff, cmd_buff, rcv_buff)
```

```
PrintPrompt("> ")
```

```
CloseSocket(sockfd)
```

```
close Main(argc, argv)
```

```
ConvertUserCommandToFTPCommand(buff, cmd_buff)
```

```
if buff starts with "ls":
```

```
    put cmd_buff "NLST"
```

```
elseif buff equals with "quit"
```

```
    put cmd_buff "QUIT"
```

```
else
```

```
    put cmd_buff buff's first word
```

```
    put buff's options & arguments to cmd_buff  
close ConvertUserCommandToFTPCommand(buff, cmd_buff)
```

Srv.C

```
Main(argc, argv)
```

```
    srvaddr, cliaddr
```

```
    serverfd, connfd = InvalidSocketDescriptor
```

```
    clilen = 0
```

```
    str = NULL
```

```
    buff, result_buff = AllocateBuffers()
```

```
if argc is not 2
```

```
    print "Format: ./srv [port num]\n"
```

```
    exit(1)
```

```
serverfd = CreateSocket()
```

```
if serverfd is invalid
```

```
    print "Server: Can't open stream socket\n"
```

```
    exit(1)
```

```
srvaddr = SetServerAddress(argv[1])
```

```
if BindSocketToAddress(serverfd, srvaddr) is failed
```

```
    print "Server: Can't bind local address\n"
```

```
    exit(1)
```



```
Listen(serverfd, 10)
```

```
while True
```

```
    clilen = sizeof(cliaddr)
```

```
    connfd = AcceptClientConnection(serverfd, cliaddr, clilen)
```

```
    if PrintClientInfo(cliaddr) is failed
```

```
        print "client_info() err!!\n"
```

```
    ResetBuffers(buff, result_buff)
```

```
    while True
```

```
        n = ReadFromClient(connfd, buff)
```

```
        if n is less than 0
```

```
            print "read() error\n"
```

```
            break
```

```
    NullTerminateBuffer(buff, n)
```

```
    ProcessCommand(buff, result_buff)
```

```
    if WriteToClient(connfd, result_buff) is failed
```

```
        print "write() err\n"
```

break

if result_buff is "QUIT"

break

ResetBuffers(buff, result_buff)

CloseClientSocket(connfd)

CloseServerSocket(serverfd)

close Main(argc, argv)

ProcessCommand(buff, result_buff)

if buff equals with "QUIT"

write buff to result_buff and standard output

else if buff starts with "NLST"

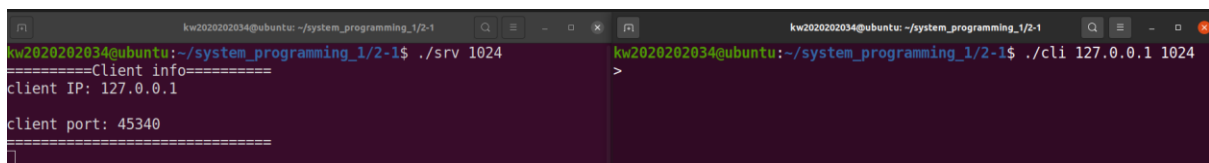
execute NLST() and write result to result_buff and write buff to standard output

else

write "wrong command" to result_buff and standard output

close ProcessCommand(buff, result_buff)

결과화면

The image shows two terminal windows side-by-side. The left window is the server process, titled 'kw2020202034@ubuntu: ~/system_programming_1/2-1'. It shows the command './srv 1024' being executed, followed by the output '====Client info====', 'client IP: 127.0.0.1', and 'client port: 45340'. The right window is the client process, titled 'kw2020202034@ubuntu: ~/system_programming_1/2-1'. It shows the command './cli 127.0.0.1 1024' being executed, followed by a prompt '>'.

server 를 port number 1024 로 주고 실행한 뒤, client 에 server address, port number 를 주어 접속하면 server 에 client 의 information(주소, port number)이 뜨게 됩니다.

```

=====Client info=====
client IP: 127.0.0.1
client port: 45340
=====
NLST
NLST -l
[]

> ls
Makefile
cli
cli.c
empty/
srv
srv.c
> ls -l
-rw-rw-r-- 1 kw2020202034 kw2020202034 98 Apr 29 13:29 Makefile
-rwxrwxr-x 1 kw2020202034 kw2020202034 17432 Apr 30 18:46 cli
-rw-rw-r-- 1 kw2020202034 kw2020202034 6286 Apr 30 18:45 cli.c
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
>

```

client 에서 ls, ls -l 명령어를 실행하니 server 에서는 FTP 명령어로 변환된 NLST, NLST -l 가 출력이 되었고, client 에서는 ls 와 ls -l 의 결과(ls 는 단순 파일명 출력, ls -l 은 파일의 상세 정보도 출력)를 보여줄 수 있습니다.

이때, empty directory 를 생성하였는데, 이는 read 권한이 없어 ls 명령어의 인자로 주었을 때 에러가 날 것입니다.

```

NLST -l ..
[]

-rw-rw-r-- 1 kw2020202034 kw2020202034 98 Apr 29 13:29 Makefile
-rwxrwxr-x 1 kw2020202034 kw2020202034 17432 Apr 30 18:46 cli
-rw-rw-r-- 1 kw2020202034 kw2020202034 6286 Apr 30 18:45 cli.c
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
> ls -l empty
Error: cannot access
> ls -l ..
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 12 09:38 1-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 09 11:36 1-2/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 17 21:38 1-3/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 30 18:46 2-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 17 21:41 Assignment
t 1 3 C 2020202034 김태완 /
-rw-rw-r-- 1 kw2020202034 kw2020202034 1868062 Apr 17 21:42 Assignment_1_3_C_2020202034 김태완.tar.gz
-rw-rw-r-- 1 kw2020202034 kw2020202034 2222 Apr 08 20:06 header.sh
>

```

상위 디렉토리(ls -l ..)의 정보를 출력하였을 때도, 잘 출력됨을 볼 수 있습니다.

```

NLST -l empty
[]

> ls -l
-rw-rw-r-- 1 kw2020202034 kw2020202034 98 Apr 29 13:29 Makefile
-rwxrwxr-x 1 kw2020202034 kw2020202034 17432 Apr 30 18:46 cli
-rw-rw-r-- 1 kw2020202034 kw2020202034 6286 Apr 30 18:45 cli.c
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
> ls -l empty
Error: cannot access
>

```

ls -l 의 인자로 empty 를 주니 server 에서는 client 로부터 받은 FTP 명령어가 출력되고, client 에는 오류문이 출력됨을 볼 수 있습니다.

```

kw2020202034@ubuntu:~/system_programming_1/2-1$ ./srv 1024
=====Client info=====
client IP: 127.0.0.1
client port: 43728
=====
NLST -e
NLST -a ..
NLST -l ./not_exist_path
[]

kw2020202034@ubuntu:~/system_programming_1/2-1$ ./cli 127.0.0.1 1024
> ls -e
Error: invalid option
> ls -a ..
Error: too many arguments
> ls -l ./not_exist_path
Error: No such file or directory
>

```

ls 명령어에 잘못된 옵션, 인자 2 개, 존재하지 않는 path 의 에러 케이스를 주었을 때도 server 에는 변환된 FTP 명령어가 나오고, client 에는 해당하는 error 문이 출력되는 것을 볼 수 있습니다.

```

WRONG COMMAND
[
-rwxrwxr-x 1 kw2020202034 kw2020202034 17432 Apr 30 18:46 cli.c
-rw-rw-r-- 1 kw2020202034 kw2020202034 6286 Apr 30 18:45 cli.c
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
> ls -l empty
Error : cannot access
> ls -l ..
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 12 09:38 1-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 09 11:36 1-2/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 17 21:38 1-3/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 30 18:46 2-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 17 21:41 Assignment
t_1_3_C_2020202034 김태완 /
-rw-rw-r-- 1 kw2020202034 kw2020202034 1868062 Apr 17 21:42 Assignment_1_3_C_2020202034 김태완.tar.gz
-rw-rw-r-- 1 kw2020202034 kw2020202034 2222 Apr 08 20:06 header.sh
> wrong
wrong command
>

```

client 에서 ls, quit 외의 잘못된 명령어를 주었을 때는 server 에는 WRONG COMMAND 가 출력되고, client 에서는 wrong command 가 출력됩니다.

```

QUIT
[
-rw-rw-r-- 1 kw2020202034 kw2020202034 6286 Apr 30 18:45 cli.c
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
> ls -l empty
Error : cannot access
> ls -l ..
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 12 09:38 1-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 09 11:36 1-2/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 17 21:38 1-3/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 30 18:46 2-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 17 21:41 Assignment
t_1_3_C_2020202034 김태완 /
-rw-rw-r-- 1 kw2020202034 kw2020202034 1868062 Apr 17 21:42 Assignment_1_3_C_2020202034 김태완.tar.gz
-rw-rw-r-- 1 kw2020202034 kw2020202034 2222 Apr 08 20:06 header.sh
> wrong
wrong command
> quit
Program quit!!
kw2020202034@ubuntu:~/system_programming_1/2-1$

```

client 에 quit 명령어를 입력하니 server 에는 FTP 명령어로 변환된 QUIT 이 전달되어 출력되었고, client 는 이를 다시 받아 프로그램이 종료됩니다. 이때, server 는 해당 client 와 연결된 소켓이 해제됩니다.

```

=====Client info=====
client IP: 127.0.0.1
client port: 34962
=====
[
d----- 2 kw2020202034 kw2020202034 4096 Apr 30 16:36 empty/
-rwxrwxr-x 1 kw2020202034 kw2020202034 22544 Apr 30 18:46 srv
-rw-rw-r-- 1 kw2020202034 kw2020202034 14746 Apr 30 18:46 srv.c
> ls -l empty
Error : cannot access
> ls -l ..
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 12 09:38 1-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 09 11:36 1-2/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 17 21:38 1-3/
drwxrwxr-x 3 kw2020202034 kw2020202034 4096 Apr 30 18:46 2-1/
drwxrwxr-x 2 kw2020202034 kw2020202034 4096 Apr 17 21:41 Assignment
t_1_3_C_2020202034 김태완 /
-rw-rw-r-- 1 kw2020202034 kw2020202034 1868062 Apr 17 21:42 Assignment_1_3_C_2020202034 김태완.tar.gz
-rw-rw-r-- 1 kw2020202034 kw2020202034 2222 Apr 08 20:06 header.sh
> wrong
wrong command
> quit
Program quit!!
kw2020202034@ubuntu:~/system_programming_1/2-1$ ./cli 127.0.0.1 1024
>

```

client 를 다시 server 와 연결하면, 위와 같이 연결되었음을 볼 수 있는데, 이때 지금 실행한 client process 는 이전에 중단된 process 와 다르기 때문에 port number 가 다를 수 있습니다.

```
QUIT
=====Client info=====
client IP: 127.0.0.1

client port: 34962
=====
NLST /dev
[]

kw2020202034@ubuntu:~/system_programming_1/2-1$ ./cli 127.0.0.1 1024
> ls /dev
autofs
block/
bsg/
btrfs-control
bus/
cdrom
cdrw
char/
console
core
cpu/
cpu_dma_latency
cuse
disk/
dma_heap/
dmmidi
dri/
dvd
ecryptfs
fb0
fd/
```

다시 연결한 후 명령어를 입력해도 여전히 잘 나옴을 볼 수 있습니다.

고찰

socket 통신을 시작하는 부분은 강의 자료를 참고해 구현하여 크게 어렵진 않았지만, 강의 자료의 Big size data 전송을 구현하는데 처음엔 어려움이 있었습니다. 강의 자료에서는 Big size data 를 보낸 후, server 에서 연결을 끊어 read 가 -1 이 되어 while(read())를 탈출할 수 있었지만, 현 과제에서는 예를 들어 NLST -i 의 결과를 다 읽어도 socket 이 끊기지 않고 다음 명령어를 받기 위해 대기하기 때문에 client 에서 다 읽었음을 판단할 수 없었습니다. 그래서 Big size data 전송을 구현하지 않고 recv_buff, send_buff 의 사이즈를 크게 주어 partial read 가 일어나지 않도록 하였습니다.