시스템 프로그래밍 실습

# Assignment3-3


**Class** : 금 1, 2 분반

**Professor** : 최상호 교수님

**Student ID** : 2020202031

**Name** : 김재현

# Introduction

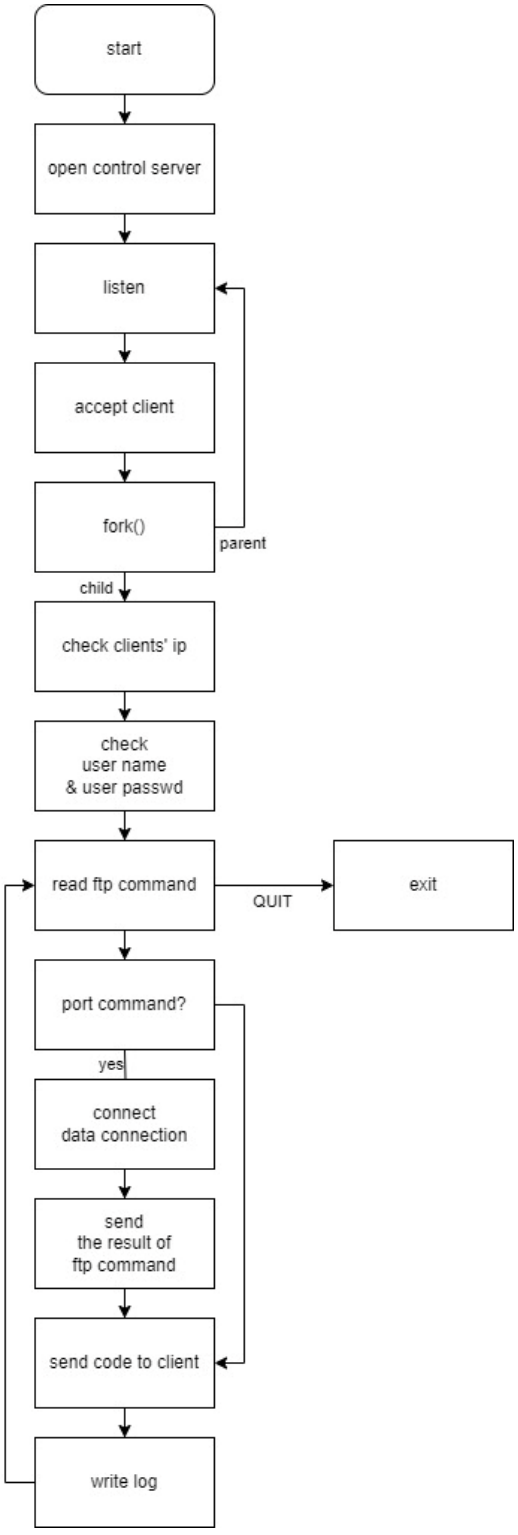
이번 과제는 Assignment #1, #2, #3 에서 구현한 것들의 총집합입니다. #2 에서 구현한 다중연결 시스템에 #3 에서 구현한 USER ip 차단, 로그인과 #1 에서 구현한 ftp command convert 와 송수신, 서버측 동작까지 전부 응용하여 server-client ftp model 을 구현합니다.

또한 server 에서 client 와의 소통을 logfile 에 기록합니다.

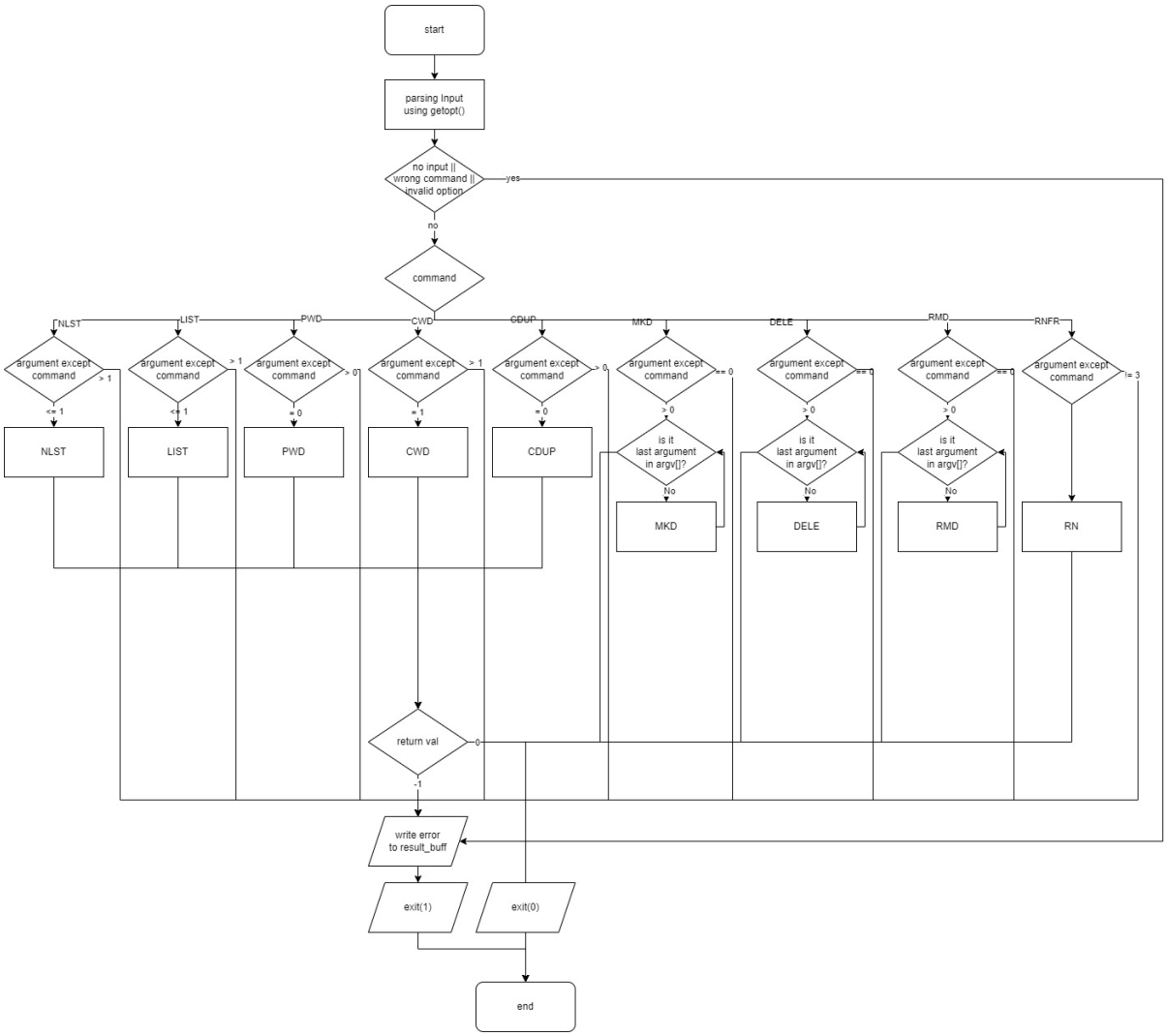
# Flow chart

server 측

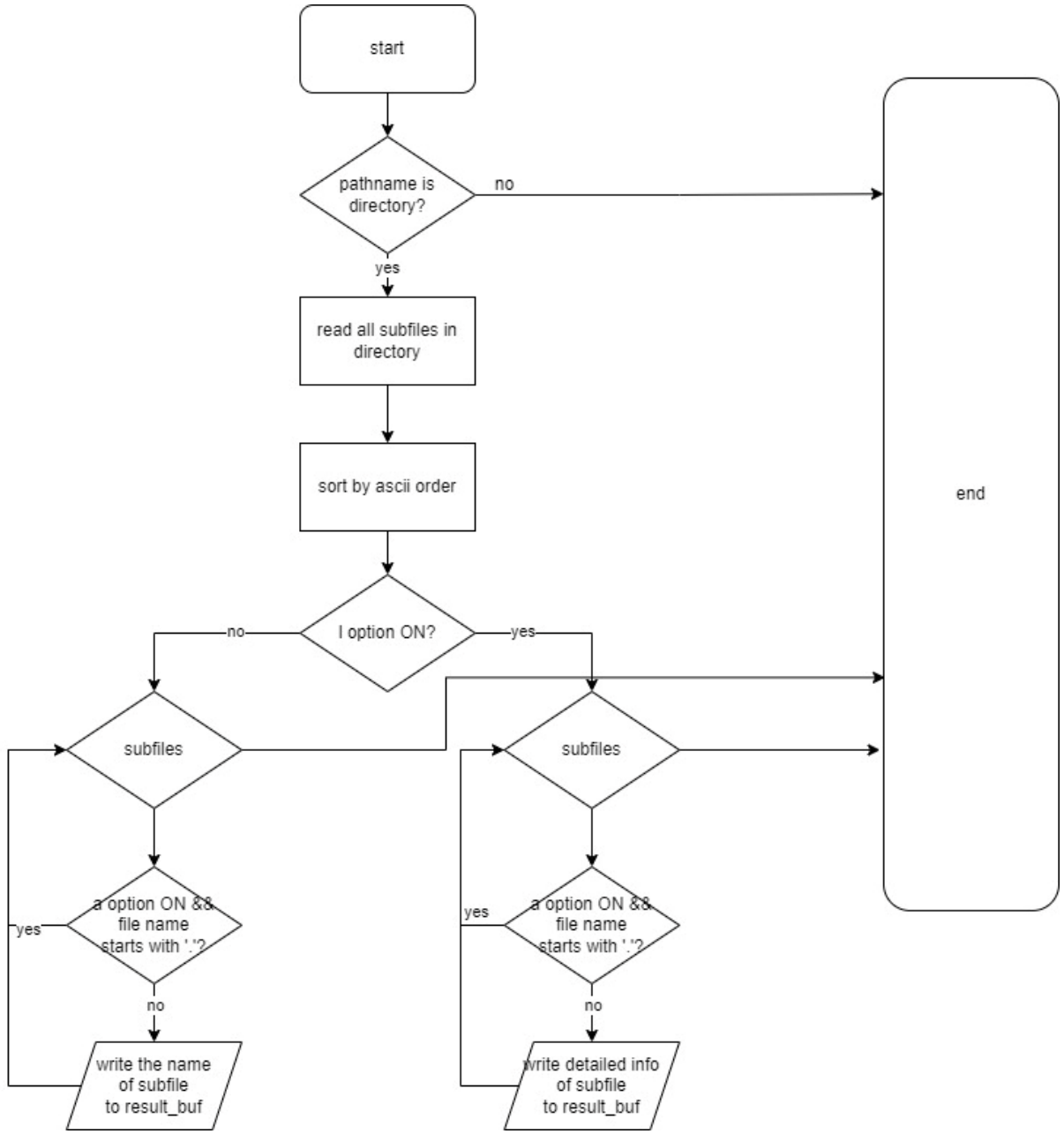
srv.c



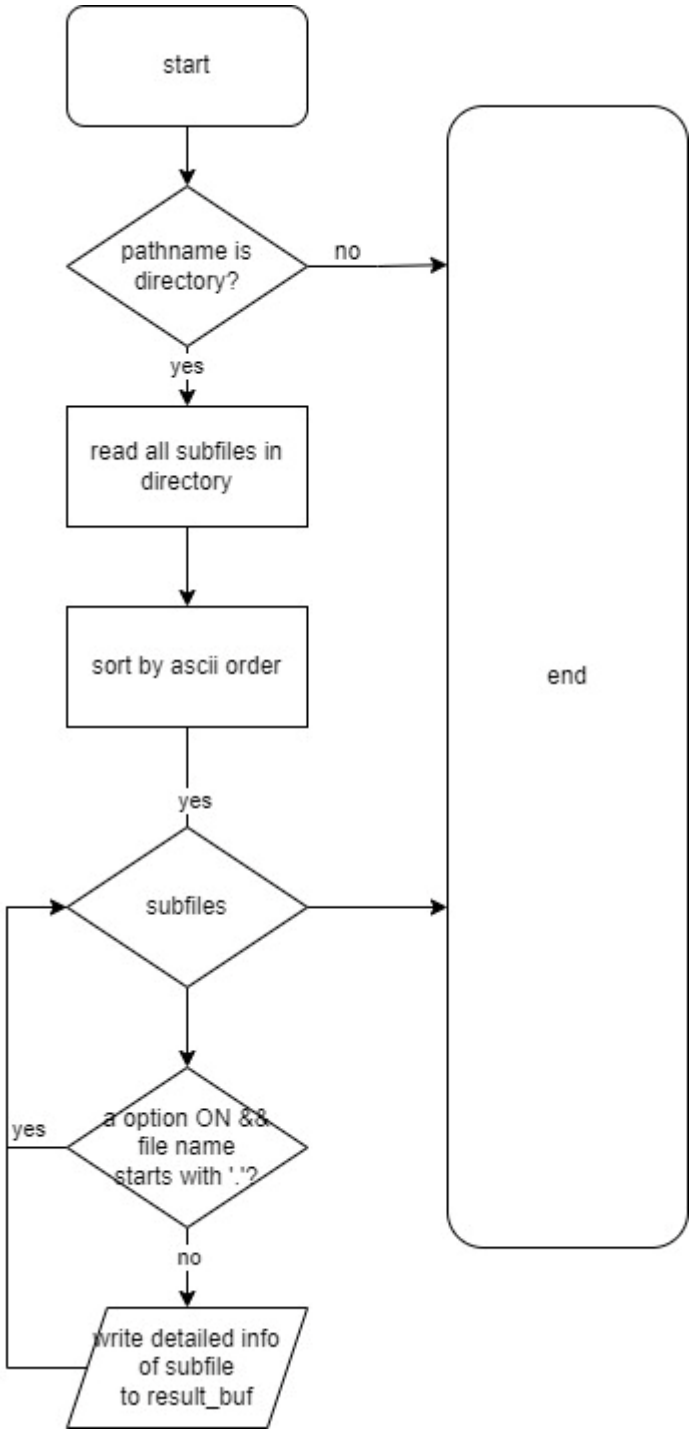
# cmd_process


start
parsing Input
using getopt()
no input ||
wrong command ||
invalid option
yes
no
command
NLST
LIST
PWD
CWD
CDUP
MKD
DELE
RMD
RNFR
argument except
command
> 1
<= 1
= 0
= 1
> 0
== 0
!= 3
is it
last argument
in argv[]?
No
NLST
LIST
PWD
CWD
CDUP
MKD
DELE
RMD
RN
return val
0
-1
write error
to result_buff
exit(1)
exit(0)
end

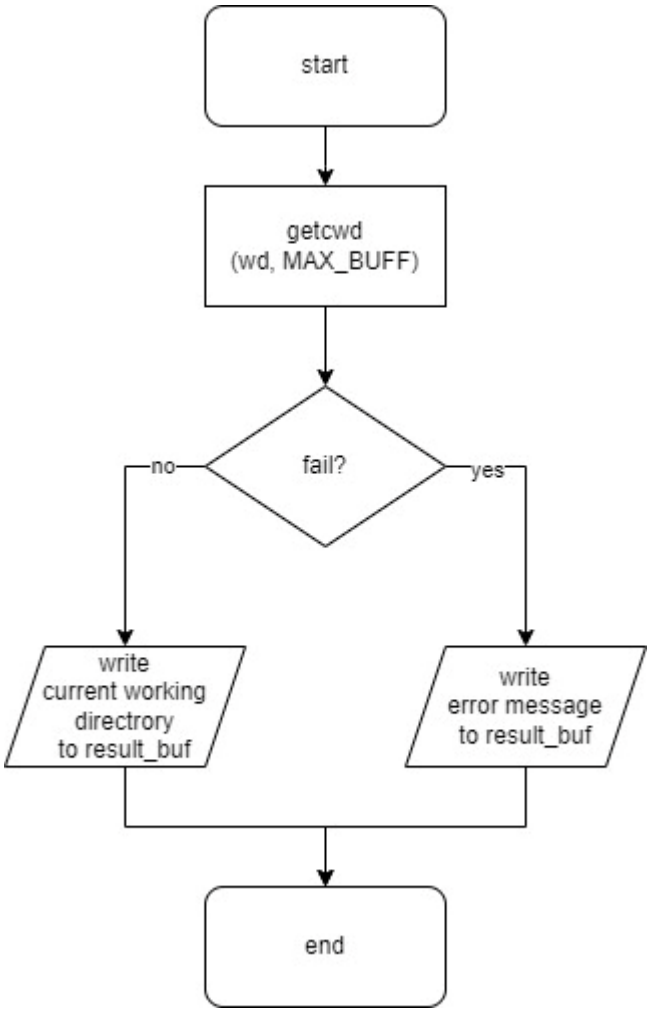
NLST


start
pathname is directory?
no
yes
read all subfiles in directory
sort by ascii order
l option ON?
no
yes
end
subfiles
subfiles
a option ON && file name starts with '.'?
a option ON && file name starts with '.'?
yes
yes
no
no
write the name of subfile to result_buf
write detailed info of subfile to result_buf

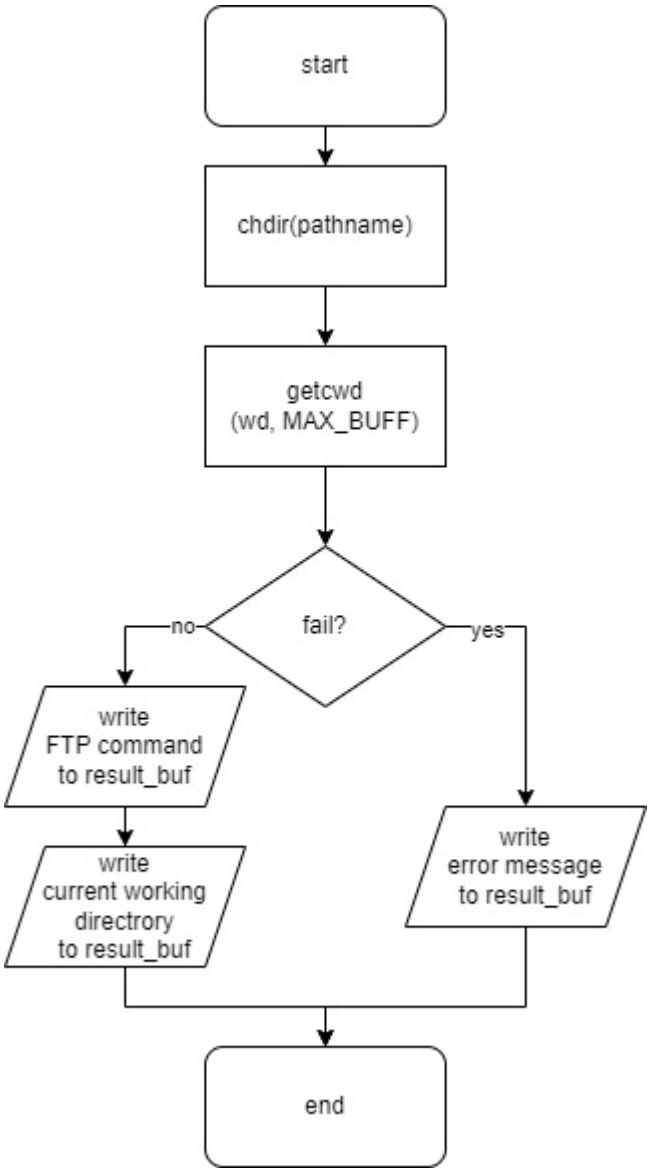
LIST


start
pathname is directory?
no
yes
read all subfiles in directory
sort by ascii order
yes
subfiles
end
yes
a option ON && file name starts with '.'?
no
write detailed info of subfile to result_buf

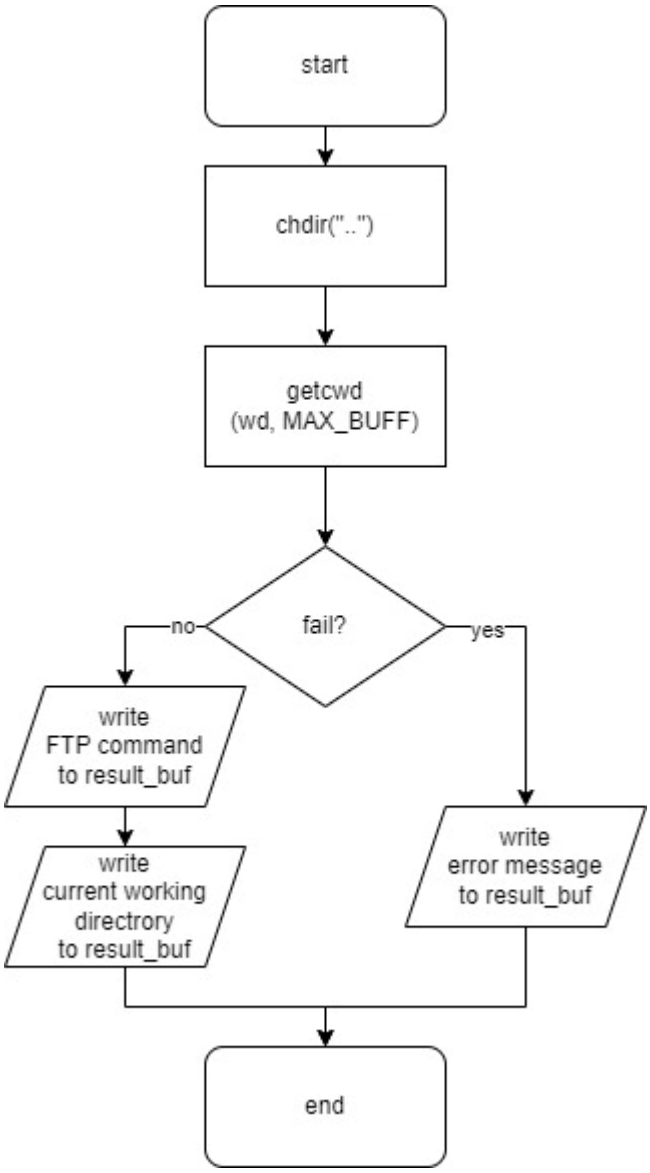
PWD


start
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
current working
directrory
to result_buf
write
error message
to result_buf
end

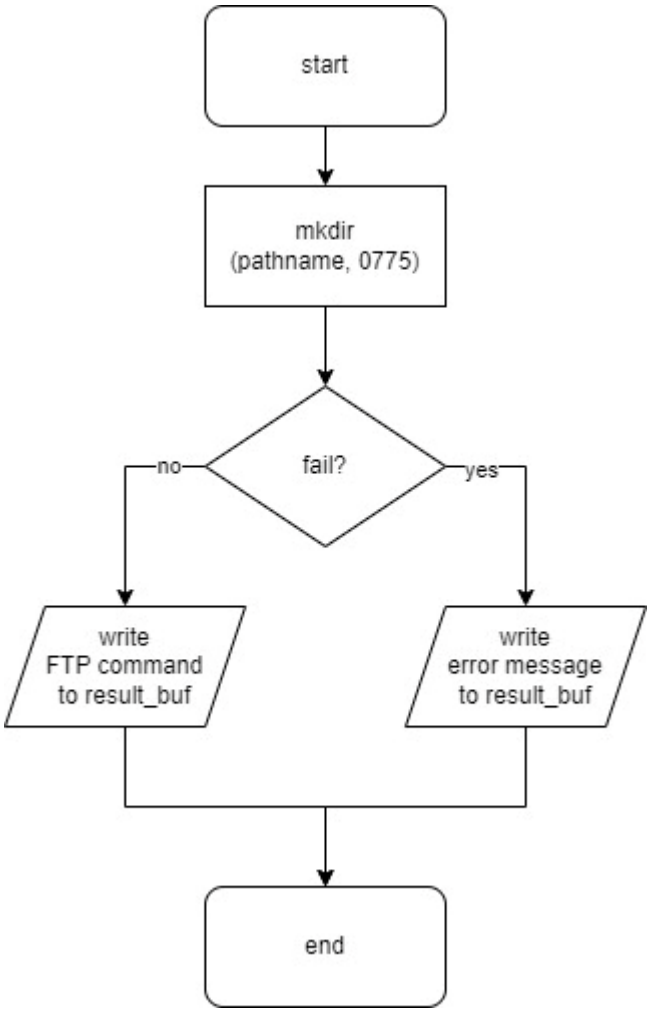
CWD


start
chdir(pathname)
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
FTP command
to result_buf
write
current working
directrory
to result_buf
write
error message
to result_buf
end

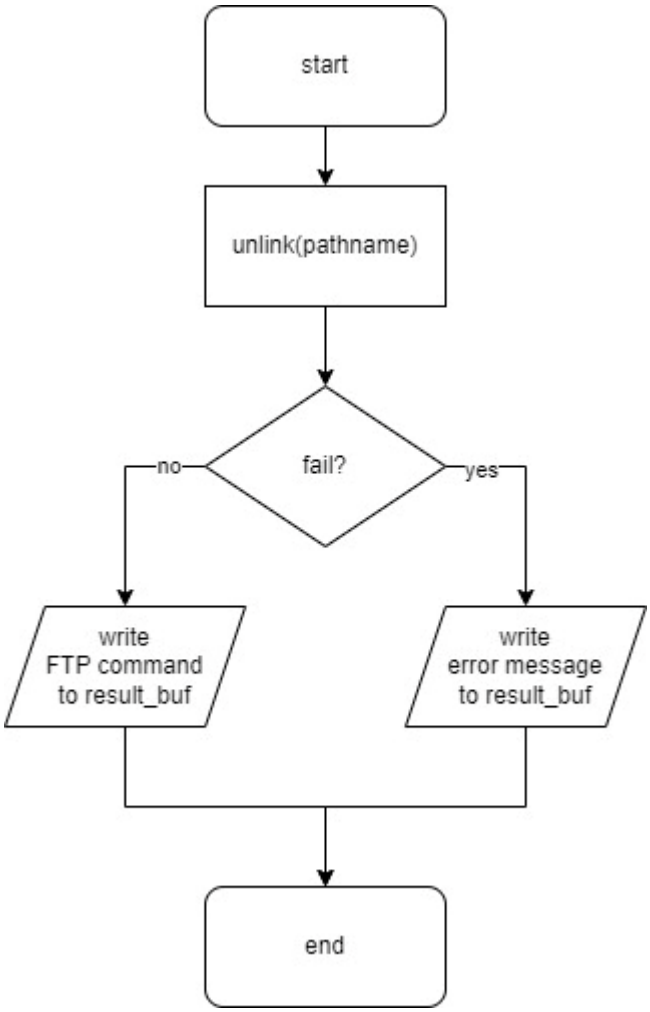
CDUP


start
chdir("..")
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
FTP command
to result_buf
write
current working
directrory
to result_buf
write
error message
to result_buf
end

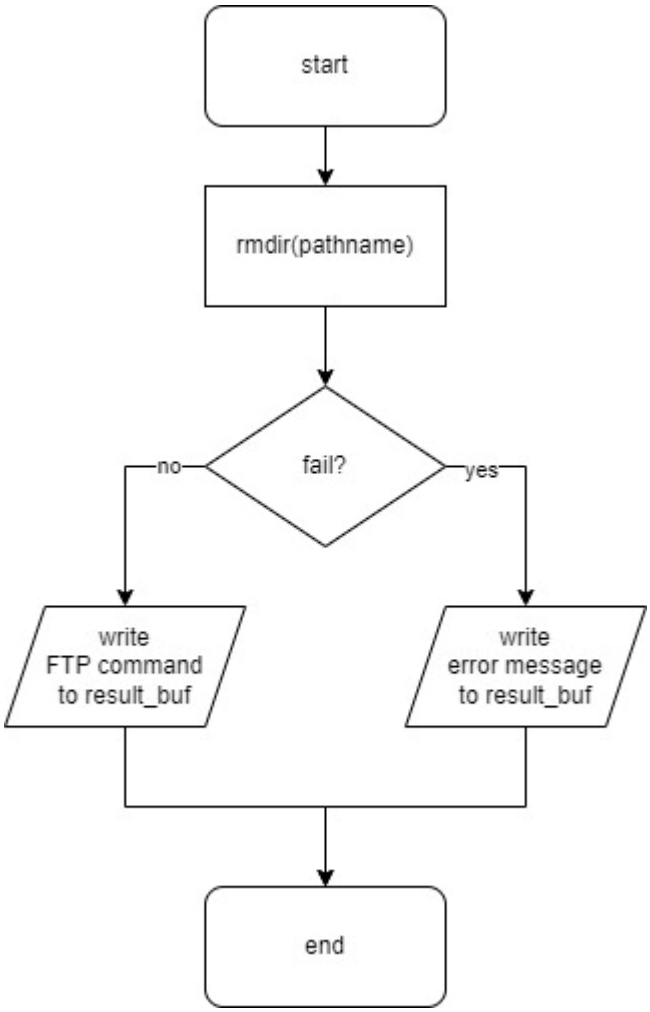
MKD


start
mkdir
(pathname, 0775)
fail?
no
yes
write
FTP command
to result_buf
write
error message
to result_buf
end

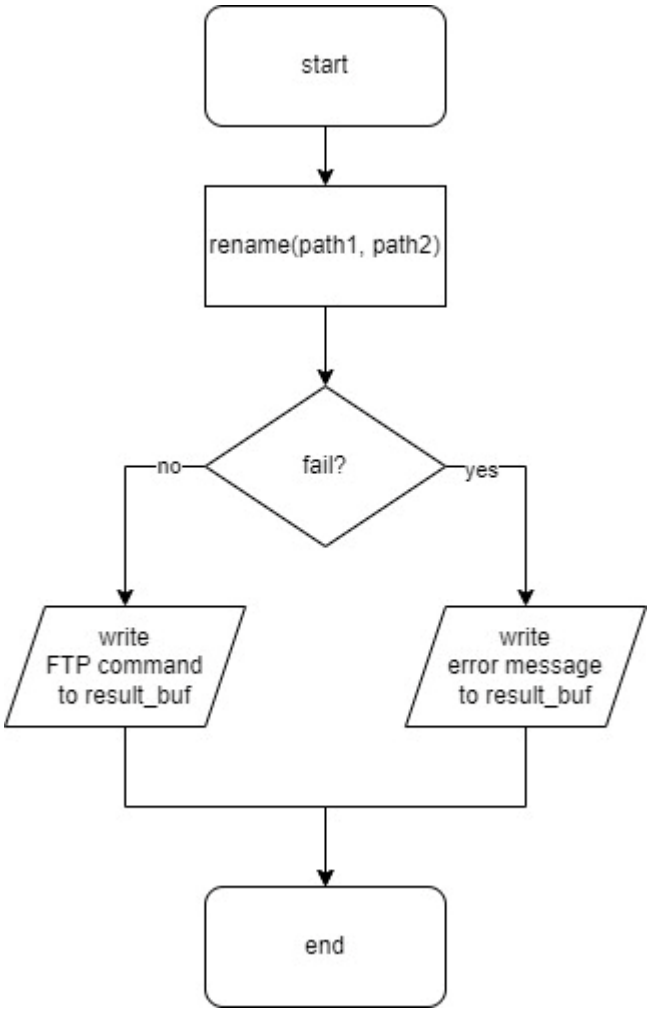
DELE


start
unlink(pathname)
fail?
no
yes
write
FTP command
to result_buf
write
error message
to result_buf
end

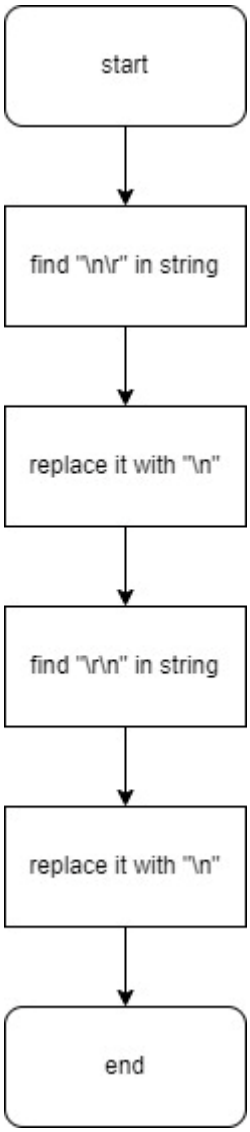
RMD


start
rmdir(pathname)
fail?
no
yes
write
FTP command
to result_buf
write
error message
to result_buf
end

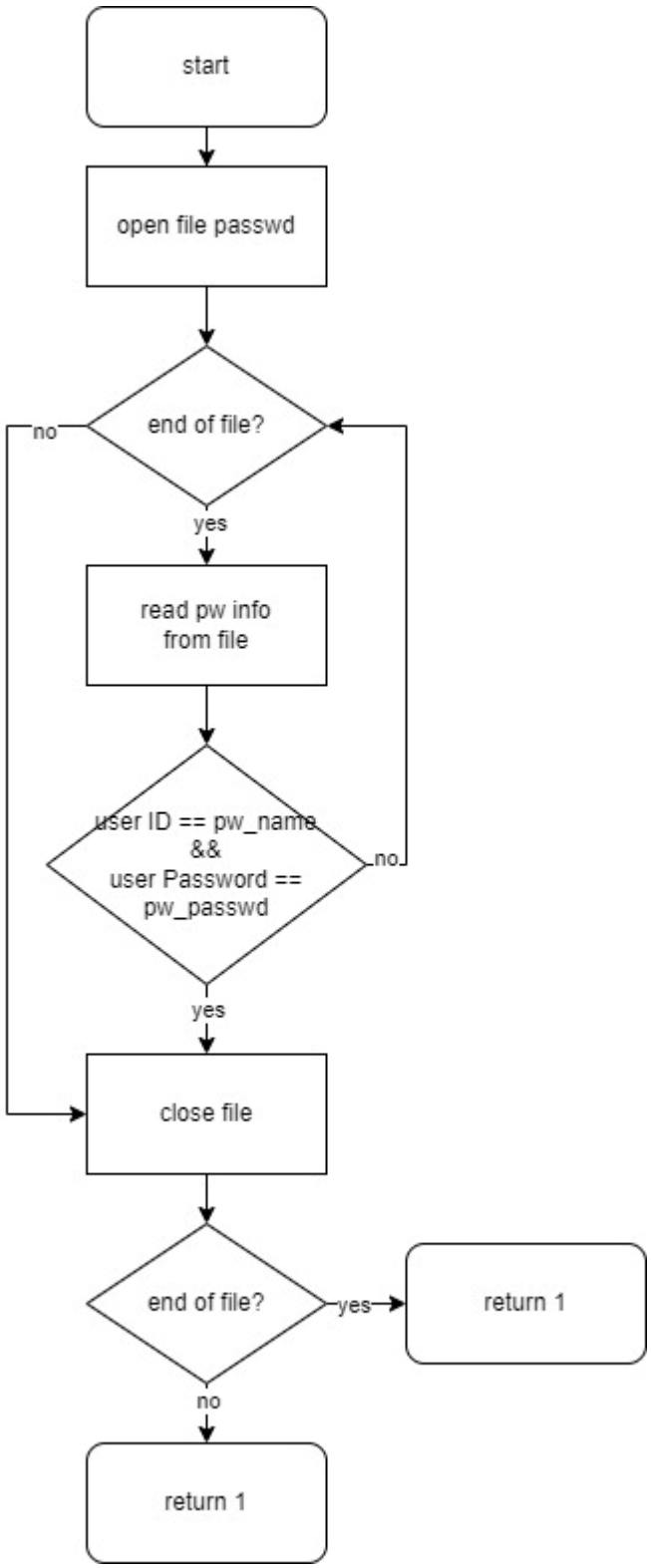
RN


start
rename(path1, path2)
fail?
no
yes
write
FTP command
to result_buf
write
error message
to result_buf
end

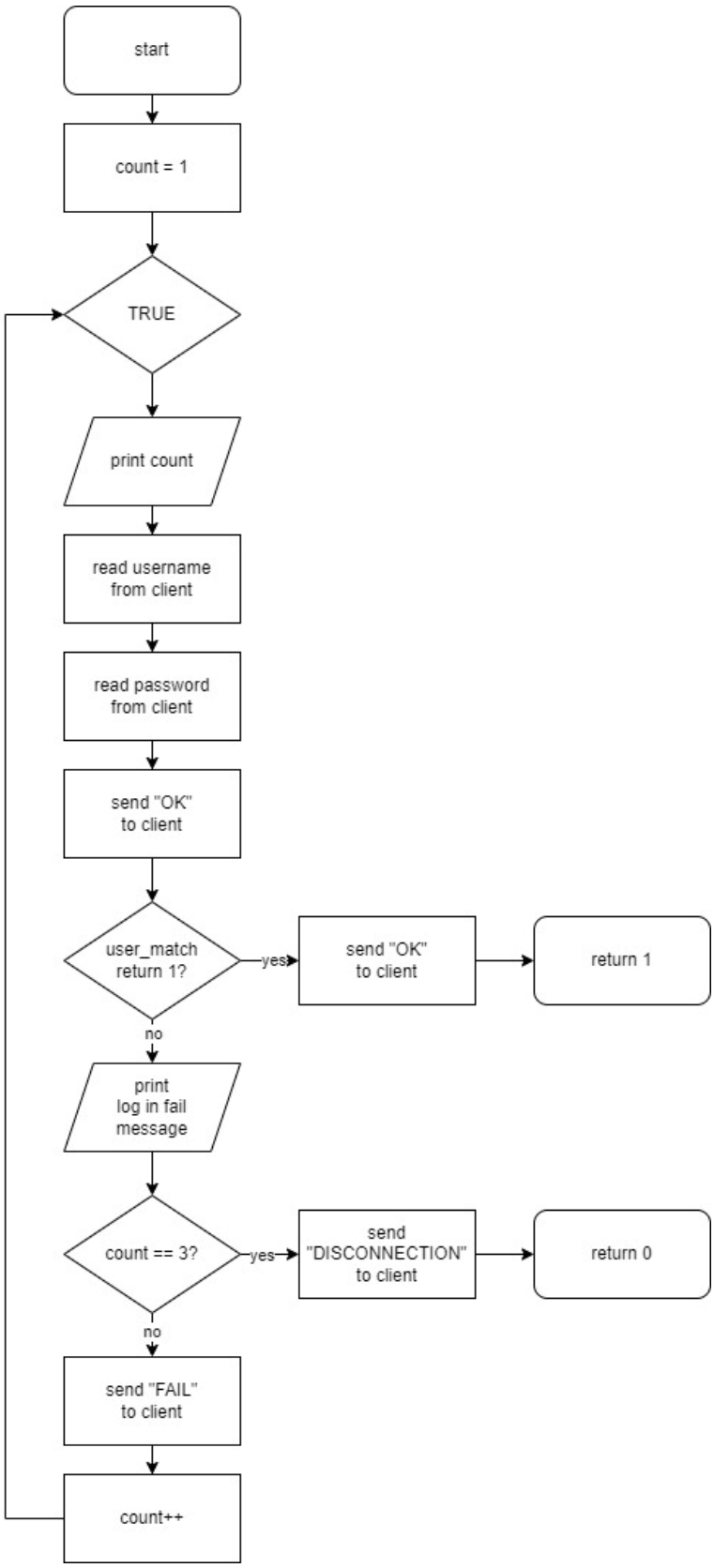
remove_r


start
find "\n\r" in string
replace it with "\n"
find "\r\n" in string
replace it with "\n"
end

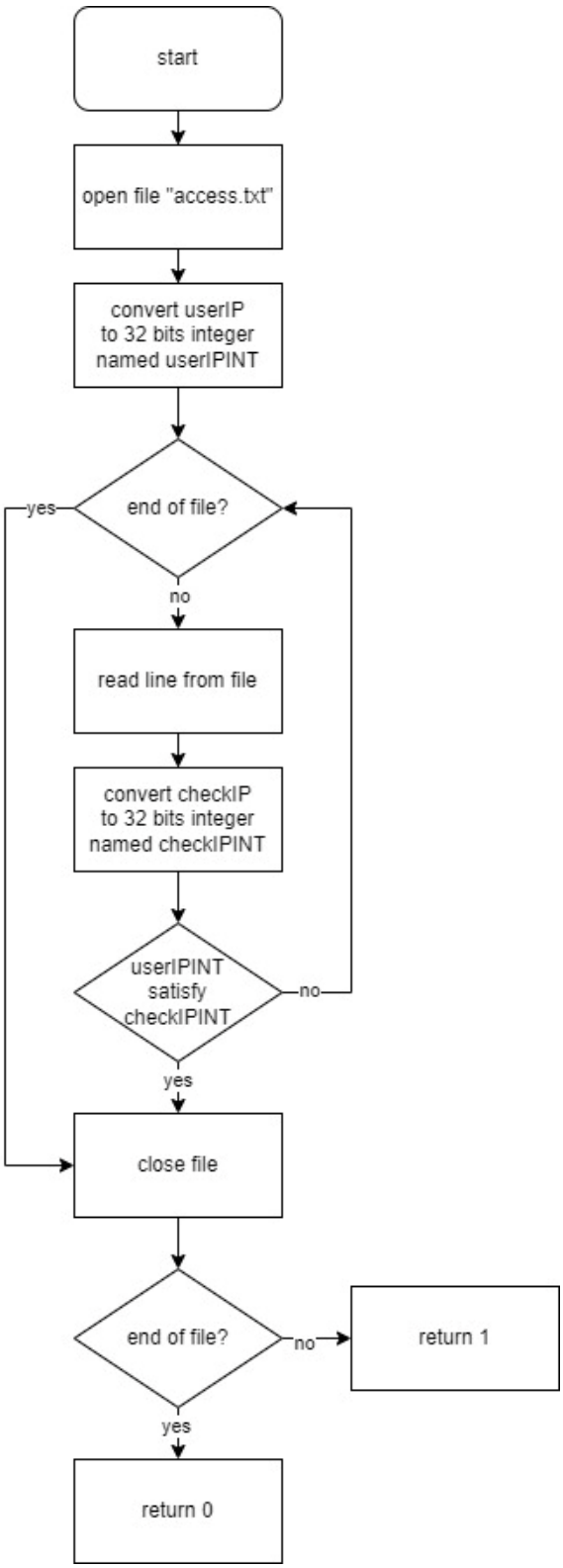
user_match


start
open file passwd
end of file?
no
yes
read pw info
from file
user ID == pw_name
&&
user Password ==
pw_passwd
no
yes
close file
end of file?
yes
return 1
no
return 1

log_auth

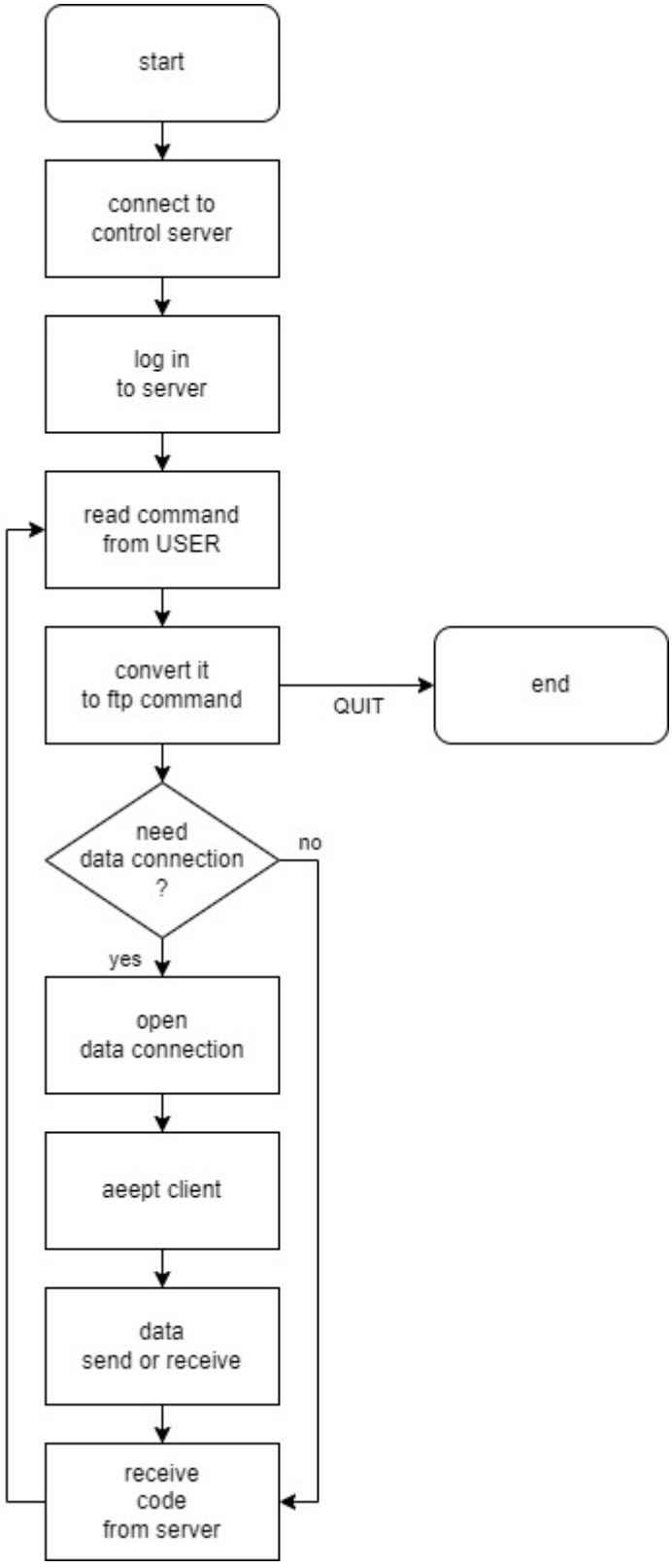

start
count = 1
TRUE
print count
read username from client
read password from client
send "OK" to client
user_match return 1?
yes
send "OK" to client
return 1
no
print log in fail message
count == 3?
yes
send "DISCONNECTION" to client
return 0
no
send "FAIL" to client
count++

check_ip

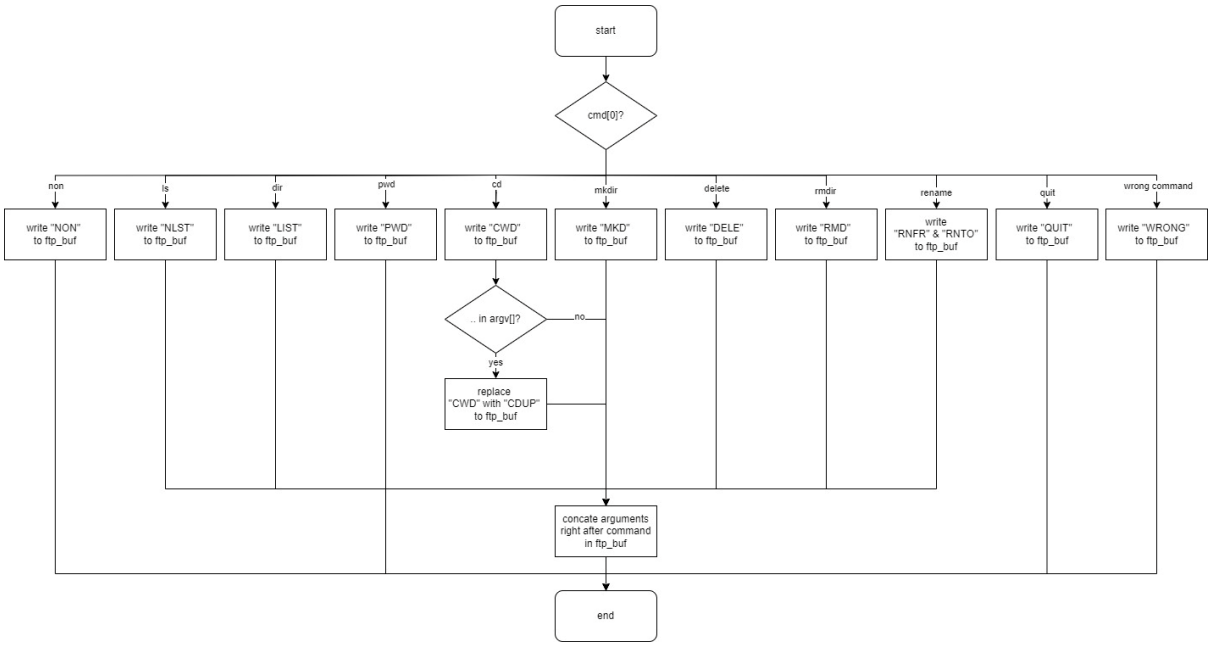

start
open file "access.txt"
convert userIP to 32 bits integer named userIPINT
end of file?
yes
no
read line from file
convert checkIP to 32 bits integer named checkIPINT
userIPINT satisfy checkIPINT
no
yes
close file
end of file?
no
return 1
yes
return 0

client 측

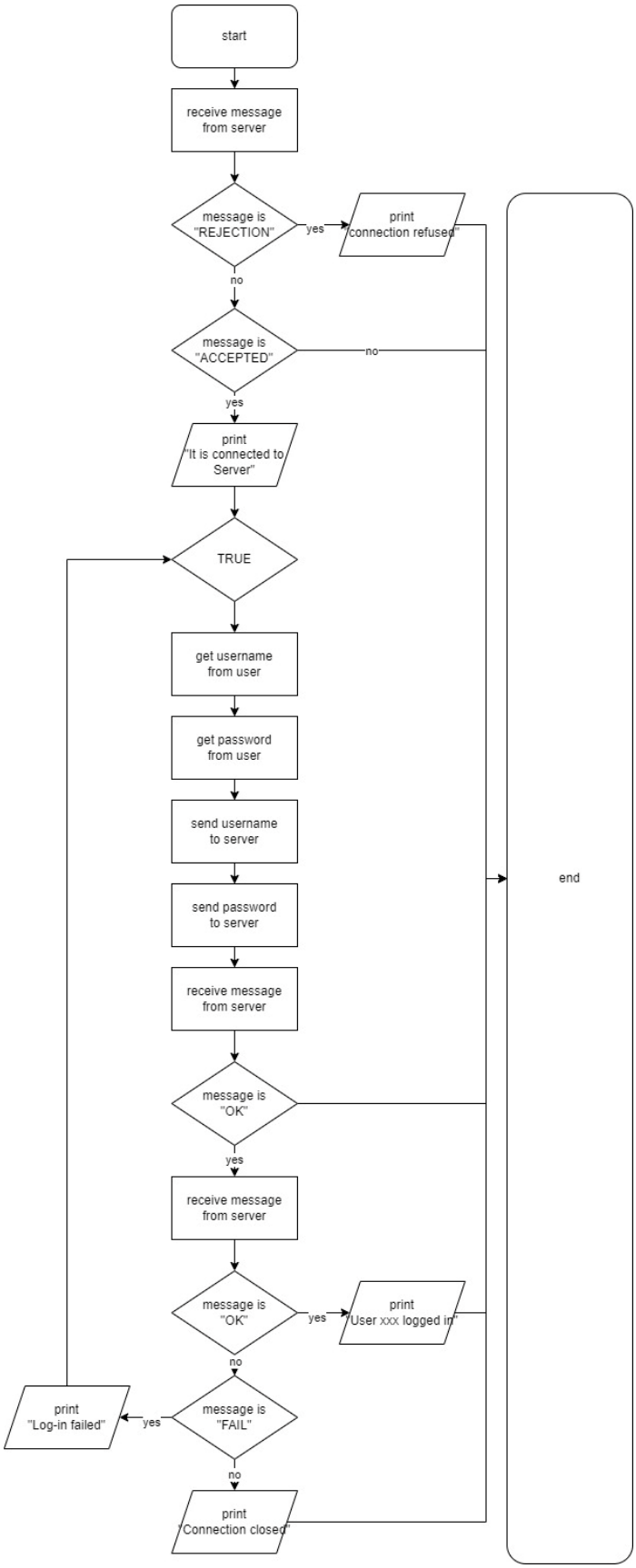
cli.c



conv_cmd


start
cmd[0]?
non
ls
dir
pwd
cd
mkdir
delete
rmdir
rename
quit
wrong command
write "NON" to ftp_buf
write "NLST" to ftp_buf
write "LIST" to ftp_buf
write "PWD" to ftp_buf
write "CWD" to ftp_buf
write "MKD" to ftp_buf
write "DELE" to ftp_buf
write "RMD" to ftp_buf
write "RNFR" & "RNTO" to ftp_buf
write "QUIT" to ftp_buf
write "WRONG" to ftp_buf
.. in argv[]?
no
yes
replace "CWD" with "CDUP" to ftp_buf
concate arguments right after command in ftp_buf
end

log_in


start
receive message from server
message is "REJECTION"
yes
print "connection refused"
no
message is "ACCEPTED"
no
yes
print "It is connected to Server"
TRUE
get username from user
get password from user
send username to server
send password to server
receive message from server
message is "OK"
yes
receive message from server
message is "OK"
yes
print "User xxx logged in"
no
message is "FAIL"
yes
print "Log-in failed"
no
print "Connection closed"
end

# Pseudo code

srv.c

main

```
START

DEFINE variables and buffers
HANDLE SIGINT signal with sh_int function
OPEN logfile for writing

IF number of command line arguments is not 2
    PRINT error message
    EXIT

OPEN control server socket

INITIALIZE buffers

BIND control server socket to given port

WRITE log for server start

WHILE true
    CLEAR buffers

    ACCEPT incoming control connection
    FORK child process

    IF child process
        IF client IP is not allowed
            SEND "431 This client can't access. Close the session." to client
            PRINT the message
            CLOSE control connection
            CONTINUE loop

        OPEN "motd" file
        READ contents into buffer
        GET current time
        FORMAT time string
        SEND welcome message with formatted time to client
        PRINT the welcome message

        IF log_auth fails (authentication failure)
            CLOSE control connection
            CONTINUE loop
```

```
        WHILE true
            CLEAR buffers

            READ command from client
            WRITE command to log

            IF command is "QUIT"
                SEND "221 Goodbye." to client
                PRINT the message
                WRITE to log
                EXIT child process

            ELSE IF command is "PORT"
                CONVERT PORT command to IP and port number
                OPEN data server socket
                CONNECT data server socket to client IP and port
                IF connection fails
                    SEND "550 Failed to access." to client
                    PRINT the message
                    WRITE to log
                    CONTINUE inner loop
                ELSE
                    SEND "200 Port command performed successfully." to client
                    PRINT the message
                    WRITE to log

                READ FTP command from client
                WRITE to log

                IF command is NLST or LIST
                    SEND "150 Opening data connection for directory list." to
client
                    PRINT the message
                    WRITE to log

                    PROCESS the command and get result
                    IF result is error
                        SEND "not exist" to client
                        SEND "550 Failed transmission." to client
                        PRINT the message
                    ELSE
                        SEND "exist" to client
                        SEND result data via data connection
                        SEND "226 Complete transmission." to client
                        PRINT the message
```

```
                WRITE to log

            ELSE IF command is RETR
                SEND appropriate message for ASCII or BINARY mode to client
                PRINT the message
                WRITE to log

                PROCESS the command and get result
                IF result is error
                    SEND "not exist" to client
                    SEND "550 Failed transmission." to client
                    PRINT the message
                ELSE
                    SEND "exist" to client
                    SEND file name to client
                    SEND file data via data connection
                    SEND "226 Complete transmission." to client
                    PRINT the message

                WRITE to log

            ELSE IF command is STOR
                SEND appropriate message for ASCII or BINARY mode to client
                PRINT the message
                WRITE to log

                READ response from client
                IF response is "not exist"
                    CONTINUE inner loop

                PROCESS the command and get result
                IF result is error
                    SEND "exist" to client (to receive file data)
                    SEND "550 Failed transmission." to client
                    PRINT the message
                ELSE
                    SEND "226 Complete transmission." to client
                    PRINT the message

                WRITE to log

            CLOSE data server socket

        ELSE
            PROCESS the command and get result
            SEND result to client
            PRINT the result
```

```
                WRITE to log

        CLOSE control connection
        EXIT child process

CLOSE logfile

END
```

```
int cmd_process(const char *buff, char *result_buff)
{
    parsing buf using getopt();

    if (input not fit in ftp command form)
        write error message to result_buf;
    else
    {
        if (command is "NLST")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (NLST < 0)
                write an error message to result_buf and return -1;
        }
        else if (command is "LIST")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (LIST < 0)
                print an error message and return -1;
        }
        else if (command is "PWD")
        {
            if (an argument is provided)
                write an error message to result_buf and return 0;
            if (PWD < 0)
                return -1;
        }
        else if (command is "CWD")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (CWD < 0)
                return -1;
```

```
        }
        else if (command is "CDUP")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (CDUP < 0)
                return -1;
        }
        else if (command is "MKD")
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;

            for (argv[])
                MKD;
        }
        else if (command is "DELE")
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;
            for (argv[])
                DELE;
        }
        else if (command is RMD)
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;
            for (argv[])
                RMD;
        }
        else if (command is RNFR and RNTO)
        {
            if (the number of arguments != 2)
                write an error message to result_buf and return 0;
            if (filename already exists)
                write an error message to result_buf and return 0;
            RN;
        }
    }

    return 0;
}
```

```
int NLST(char *result_buff, const char *pathname, int opflag)
{
    if (pathname is not directory)
        return -1;

    read all subfiles in directory named pathname;
    sort subfiles by ascii order;

    if (l option ON)
    {
        while (subfiles)
        {
            if (a option off && filename starts with '.')
                continue;
            else
                write detailed information of subfile to result_buf;


        }
    }
    else // l option OFF
    {
        while (subfiles)
        {
            if (a option off && filename starts with '.')
                continue;
            else
                write name of subfile to result_buf;
        }
    }
}
```

```
int LIST(char *result_buff, const char *pathname)
{
    if (pathname is not directory)
        return -1;

    read all subfiles in directory named pathname;
    sort subfiles by ascii order;

    while (subfiles)
    {
        if (a option off && filename starts with '.')
            continue;
        else
            write detailed information of subfile to result_buf;
    }
}
```

```
int PWD(char *result_buff)
{
    char wd[MAX_BUFF];

    if (getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write current working directory to result_buf;
        return 0;
    }
}
```

```
int CWD(char *result_buff, const char *pathname)
{
    char wd[MAX_BUFF];

    if (chdir(pathname) < 0 || getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write FTP command to result_buf;
        write current working directory to result_buf;
        return 0;
    }
}
```

```
int CDUP(char *result_buff)
{
    char wd[MAX_BUFF];

    if (chdir("..") < 0 || getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write FTP command to result_buf;
        write current working directory to result_buf;
        return 0;
    }
}
```

```
int MKD(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (mkdir(pathname, 0775) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int DELE(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (unlink(pathname) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int RMD(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (rmdir(pathname) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int RN(char *result_buff, const char *pathname1, const char *pathname2)
{



    if (rename(pathname1, pathname2) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
FUNCTION check_ip(userIP)
    OPEN file "access.txt" for reading
    COPY userIP to str_uip

    PARSE str_uip into an array uip[] by splitting on "." (dot)

    WHILE not end of file
        READ a line from "access.txt" into str_cip
        REMOVE newline character from str_cip

        i = 0
        PARSE str_cip into tokens by splitting on "."
```

```
        FOR each token
            IF token is "*" OR token matches uip[i]
                INCREMENT i
            ELSE
                BREAK loop
        IF i equals 4 (all tokens matched)
            BREAK outer loop

    CLOSE file "access.txt"

    IF i equals 4
        RETURN 1 (IP is allowed)
    ELSE
        RETURN 0 (IP is not allowed)
END FUNCTION
```

```
FUNCTION log_auth(connfd)
    count = 1

    LOOP
        READ username from client
        EXTRACT username from received data

        IF userID_match(username) is 1 (username is valid)
            SEND "331 Password is required for username." to client
            PRINT the message

        ELSE IF userID_match(username) is 0 (username is invalid)
            WRITE "LOG_FAIL" to log

            IF count is greater than or equal to 3 (maximum attempts reached)
                SEND "530 Failed to log-in." to client
                PRINT the message
                RETURN 0 (authentication failed)

            SEND "430 Invalid username or password." to client
            PRINT the message
            INCREMENT count
            CONTINUE loop

        READ password from client
        EXTRACT password from received data

        IF user_match(username, password) is 1 (password is valid)
            SEND "230 User username logged in." to client
            PRINT the message
```

```
            BREAK loop

        ELSE IF user_match(username, password) is 0 (password is invalid)
            WRITE "LOG_FAIL" to log

            IF count is greater than or equal to 3 (maximum attempts reached)
                SEND "530 Failed to log-in." to client
                PRINT the message
                RETURN 0 (authentication failed)

            SEND "430 Invalid username or password." to client
            PRINT the message
            INCREMENT count
            CONTINUE loop

    WRITE "LOG_IN" to log
    RETURN 1 (authentication successful)
END FUNCTION
```

```
FUNCTION user_match(user, passwd)
    OPEN file "passwd" for reading

    WHILE not end of file
        READ next user record from file into pw

        IF pw->pw_name matches user AND pw->pw_passwd matches passwd
            BREAK loop

    CLOSE file "passwd"

    IF pw is not NULL (a matching record was found)
        RETURN 1 (authentication successful)
    ELSE
        RETURN 0 (authentication failed)
END FUNCTION
```

cli.c

main

```
START

IF the number of command line arguments is not 3
    PRINT error message
    EXIT

OPEN control connection socket
CONNECT to FTP server using the given IP address and port

IF connection fails
    PRINT error message
    EXIT

IF log in fails
    EXIT

WHILE true
    CLEAR buffers
    PRINT prompt
    READ user command from STDIN

    CONVERT user command to FTP command

    IF user didn't input anything
        PRINT "Non Command!" error
        CONTINUE
    ELSE IF invalid command
        PRINT "Invalid Command!" error
        CONTINUE
    ELSE IF command is "QUIT"
        SEND "QUIT" command to server
        RECEIVE and PRINT response from server
        BREAK loop
    ELSE IF command needs data connection (NLST, LIST, RETR, STOR)
        OPEN data connection socket
        BIND to a random port between 10001 and 60000
        SEND PORT command to server
        ACCEPT incoming data connection
        RECEIVE and PRINT response from server

        IF response indicates failure
            CLOSE data connection sockets
            CONTINUE
```

```
        SEND command to server
        RECEIVE and PRINT response from server

        IF command is NLST or LIST
            RECEIVE data from server via data connection
            PRINT received data to STDOUT
            CLOSE data connection sockets
            RECEIVE and PRINT final response from server
            IF transmission failed
                CONTINUE
            PRINT number of bytes received
        ELSE IF command is RETR
            RECEIVE data from server via data connection
            WRITE received data to a file
            CLOSE data connection sockets
            RECEIVE and PRINT final response from server
            IF transmission failed
                CONTINUE
            PRINT number of bytes received
        ELSE IF command is STOR
            IF file doesn't exist
                SEND "not exist" to server
                CONTINUE
            SEND "exist" to server
            RECEIVE response from server
            IF server indicates file doesn't exist
                OPEN file
                READ file contents
                SEND file contents via data connection
            CLOSE data connection sockets
            RECEIVE and PRINT final response from server
            IF transmission failed
                CONTINUE
            PRINT number of bytes sent
    ELSE
        SEND command to server
        RECEIVE and PRINT response from server

CLOSE control connection socket

END
```

```
conv_cmd
```

```
{
    getopt(cmd_buf)
    if( the number of input arguments is 0)
        Copy the string "NON" to ftp_buf.
    else if (first input argument is "ls")
        Copy the string "NLST" to ftp_buf.
    else if (first input argument is "dir")
        Copy the string "LIST" to ftp_buf.
    else if (first input argument is "pwd")
        Copy the string "PWD" to ftp_buf.


    else if (first input argument is "cd")
        Copy the string "CWD" to ftp_buf.

    If additional argument is ".."
        Copy the string "CDUP" to ftp_buf.
    else
        append additional argument to ftp_buf.
    else if (first input argument is "mkdir")
        Copy the string "MKD" to ftp_buf.
    else if (first input argument is "delete")
        Copy the string "DELE" to ftp_buf.
    else if (first input argument is "rmdir")
        Copy the string "RMD" to ftp_buf.
    else if (first input argument is "rename")
        Copy the string "RNFR" and the second argument to ftp_buf.
        Copy the string "RNTO" and the third argument to ftp_buf.
    else if (first input argument is "quit")
        Copy the string "QUIT" to ftp_buf.
    else (incorrect command entered)
        Copy the string "WRONG" to ftp_buf.

    If there are additional arguments:
        Append a space to ftp_buf.
        Append the additional argument to ftp_buf.
}
```

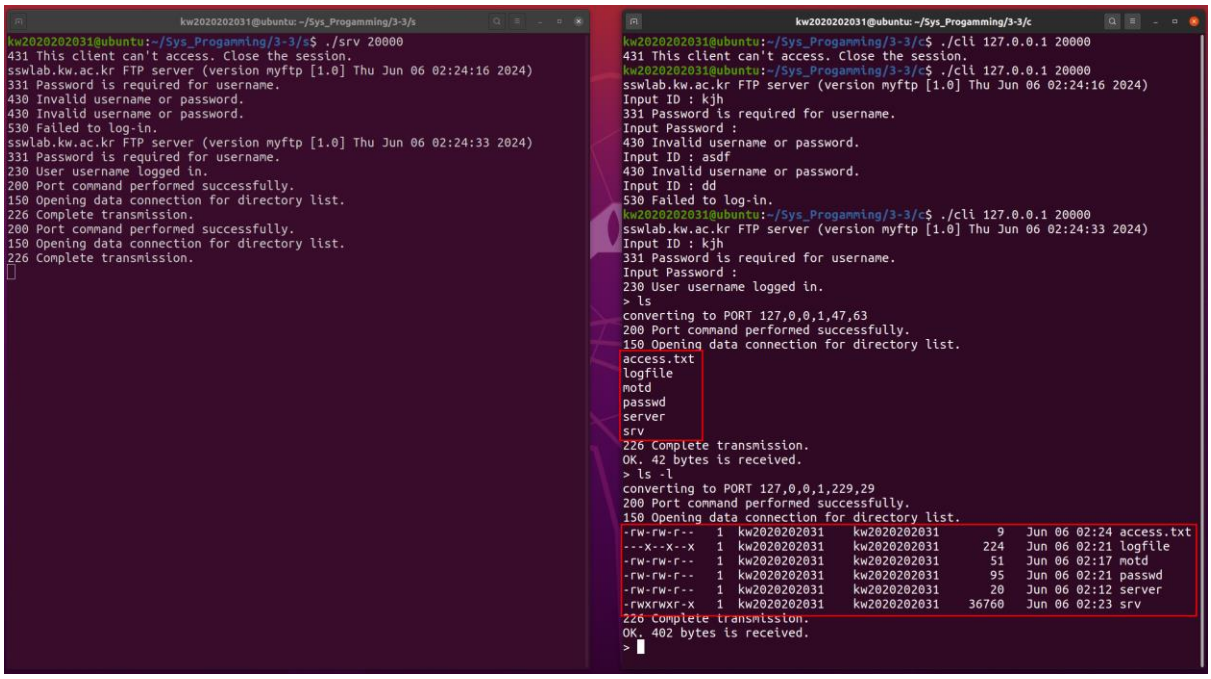
```
FUNCTION log_in(server_fd)
    READ response from server and PRINT it
    IF response indicates client is not allowed
        RETURN 0 (login failed)

    LOOP
        PRINT "Input ID : "
        READ username from user input
```

```
        SEND "USER <username>" command to server
        READ response from server and PRINT it
        IF response indicates invalid username or password
            CONTINUE loop
        ELSE IF response indicates login failed
            RETURN 0 (login failed)

        PRINT "Input Password : " (without echoing the password)
        READ password from user input
        SEND "PASS <password>" command to server
        READ response from server and PRINT it
        IF response indicates successful login
            BREAK loop
        ELSE IF response indicates invalid username or password
            CONTINUE loop
        ELSE (response indicates login failed)
            RETURN 0 (login failed after three attempts)

    RETURN 1 (login successful)
END FUNCTION
```

```
FUNCTION convert_addr_to_str(ip_addr, port)
    DECLARE cmd_port AS character string
    DECLARE j AS integer, initialized to 0

    CONVERT ip_addr to host byte order
    CONVERT port to host byte order

    ALLOCATE memory for cmd_port with size 30

    APPEND "PORT " to cmd_port
    j = j + length of "PORT "

    FOR i FROM 3 DOWN TO 0
        CONVERT (ip_addr & (0xFF << (8 * i))) >> 8 * i to string
        APPEND the string to cmd_port, followed by a comma
        j = j + length of the appended string + 1

    CONVERT (port & 0xFF00) >> 8 to string
    APPEND the string to cmd_port, followed by a comma
    j = j + length of the appended string + 1

    CONVERT port & 0x00FF to string
    APPEND the string to cmd_port
    j = j + length of the appended string
```

```
    RETURN cmd_port
END FUNCTION
```

# 결과화면



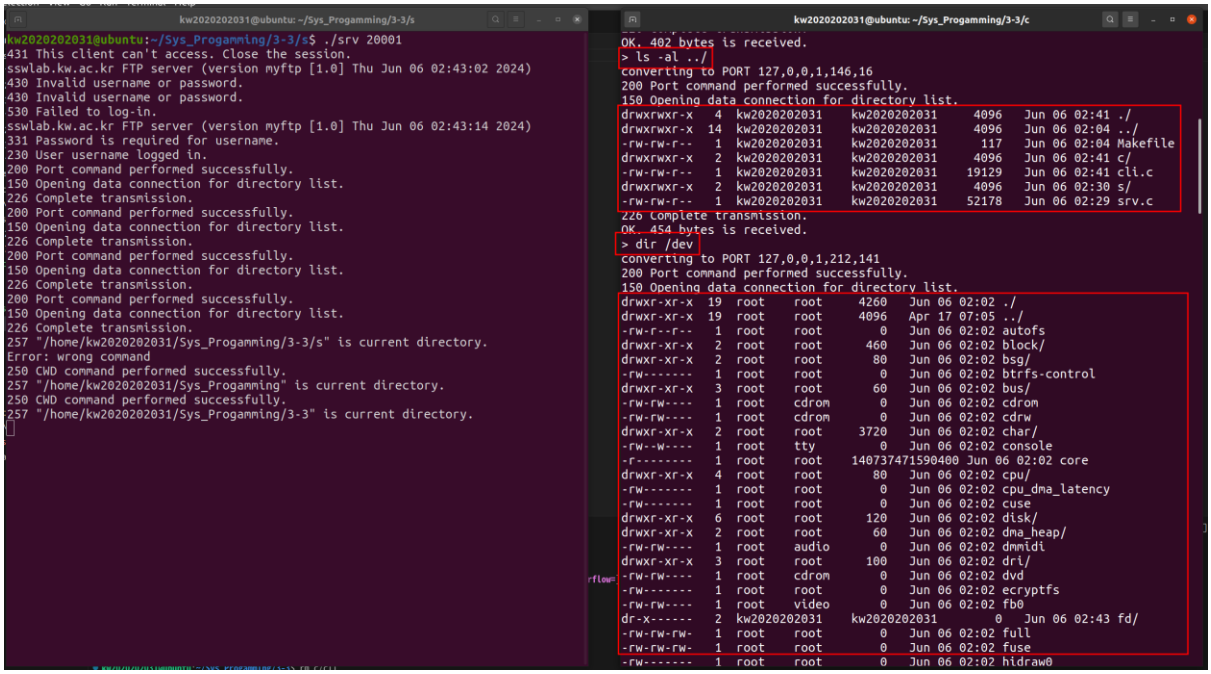

access.txt 에 client 의 ip 가 없다면 431 error 가 발생합니다.

로그인에 3 번 실패하면 530 error 가 발생합니다.

로그인에 성공하면 server client 가 통신을 시작합니다

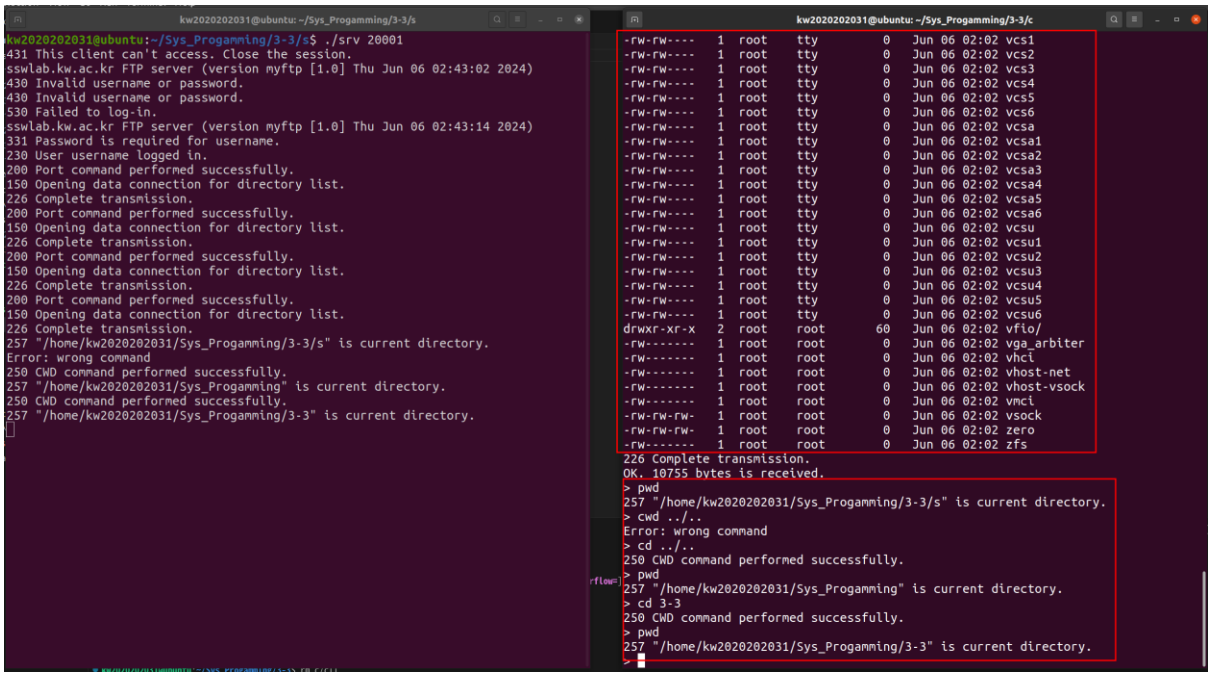
client 가 server 에 접속하면 server 와 같은 디렉토리에서 시작합니다.

ls 명령어를 입력하자 server 의 디렉토리인 s 의 file 들이 나열됩니다.



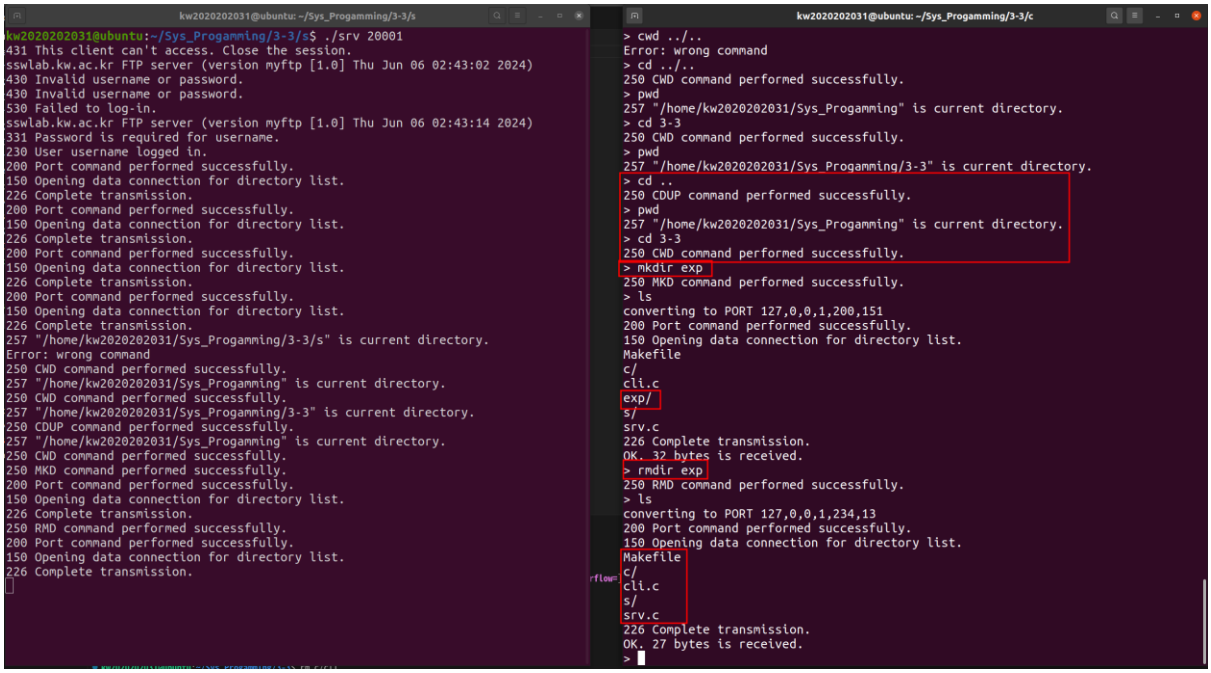

ls 의 옵션에 따라 출력형식이 다르게 나오는 것을 확인할 수 있습니다.

dir 명령어는 ls -al 과 동일합니다.

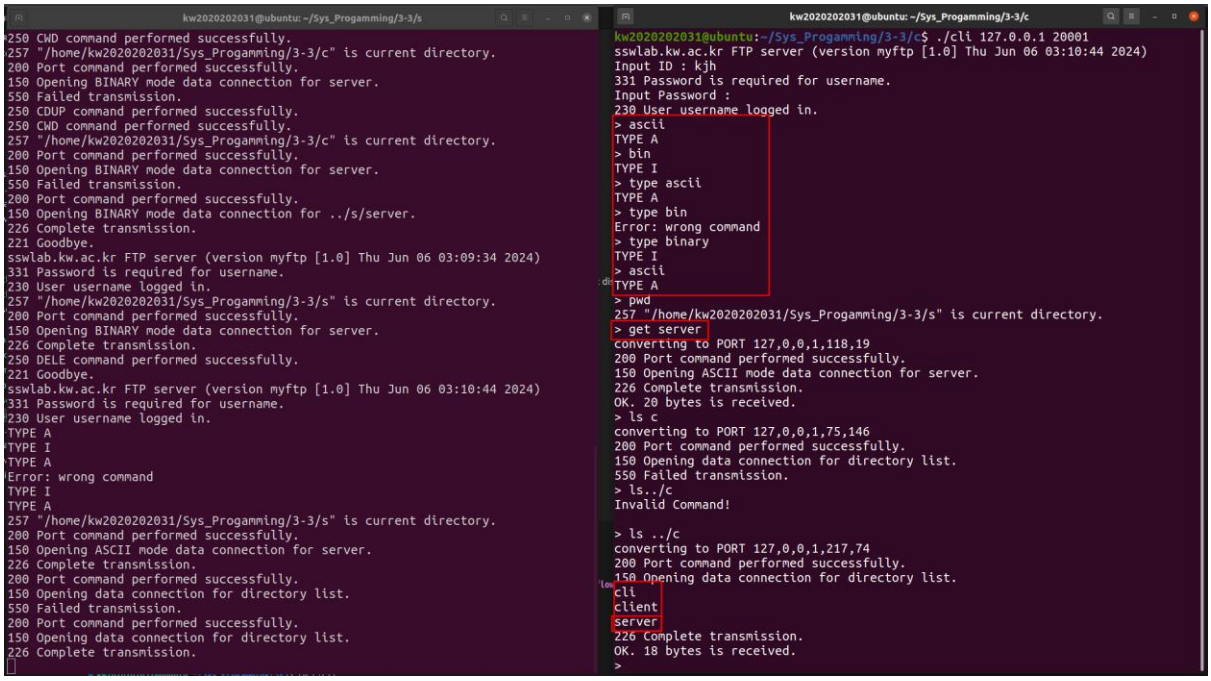



/dev 는 하위파일들이 많기 때문에 dir /dev 를 실행한 결과 10755 bytes 가 출력되는 것을 확인할 수 있습니다.

cd, pwd 도 잘 수행되는 것을 확인할 수 있습니다.

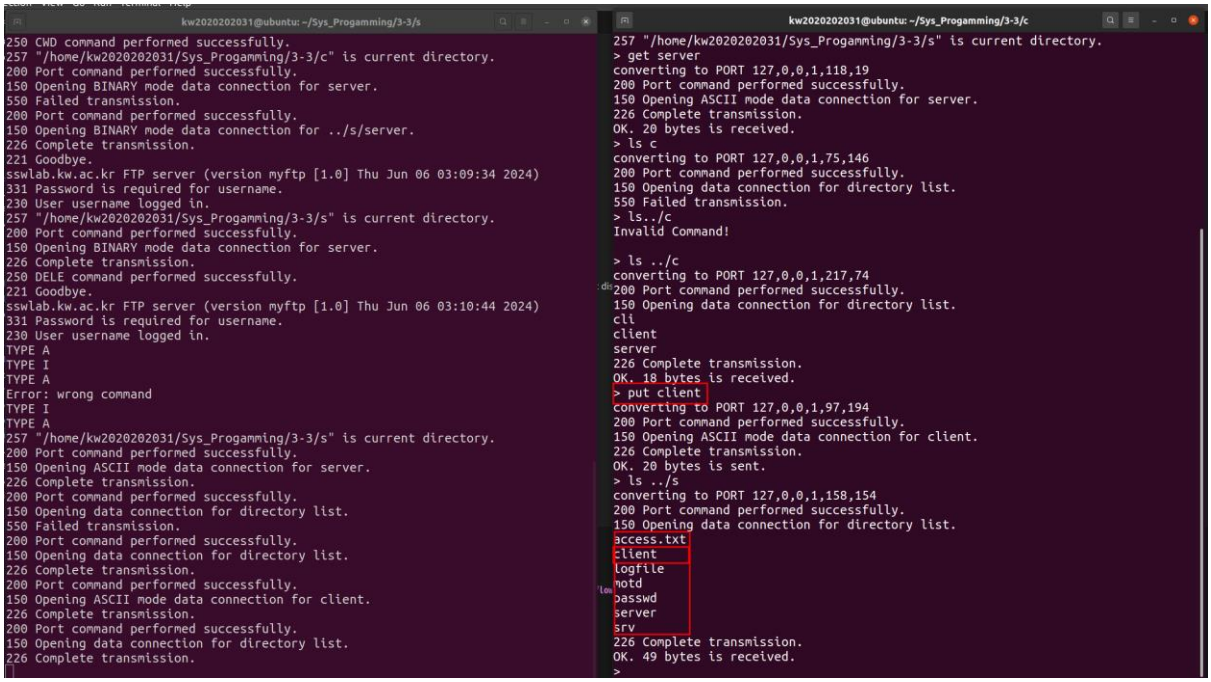



mkdir 과 rmdir 을 통해 디렉토리를 생성하고 삭제할 수 있습니다.

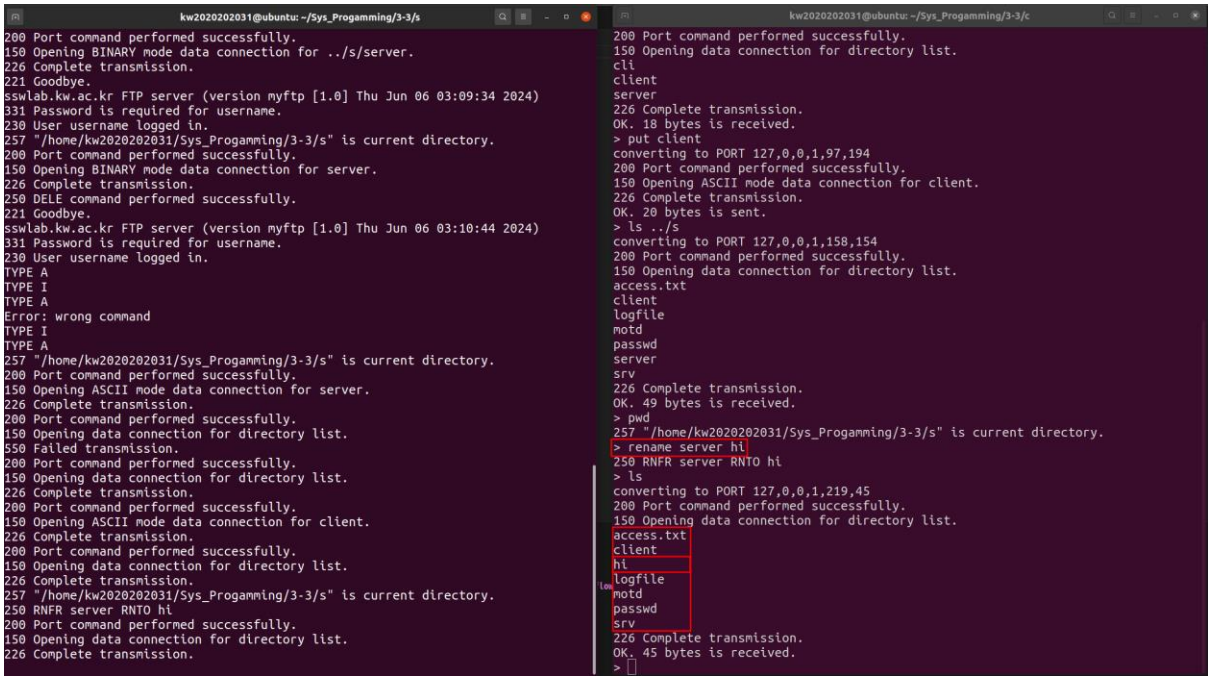



type 명령어를 통해 type 을 바꿀 수 있습니다.

get 명령어를 입력하자 ascii 모드를 통해 server 라는 파일을 client 의 디렉토리에 server 라는 파일을 get 하는 것을 확인할 수 있습니다.



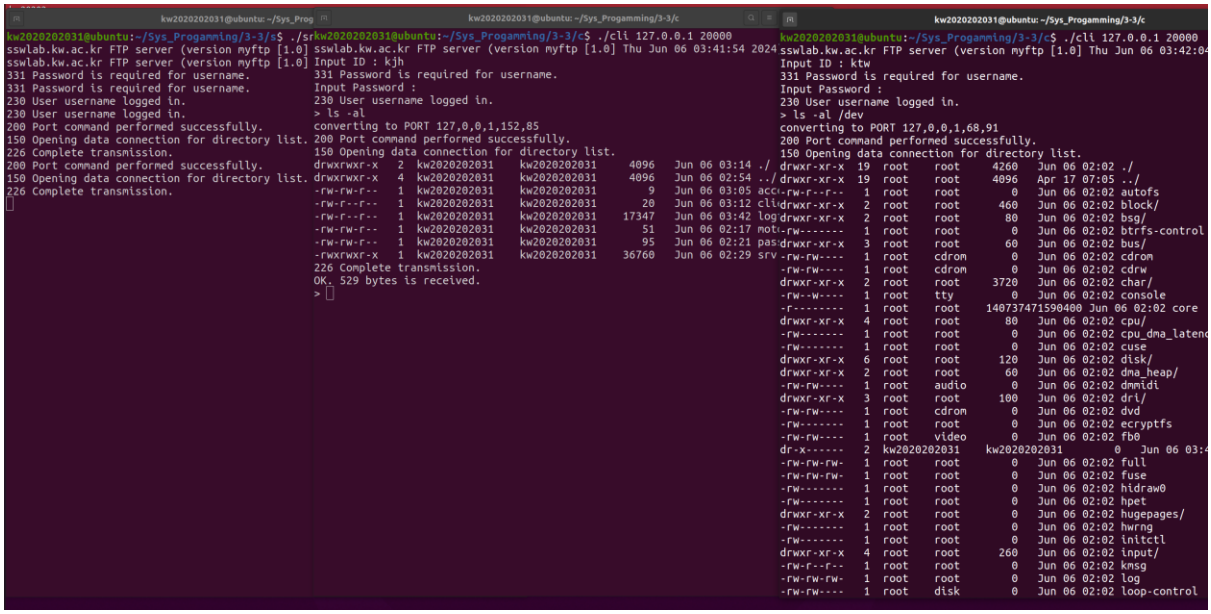

put 명령어를 통해 client 라는 파일을 server 에 저장하는 것을 확인할 수 있습니다.



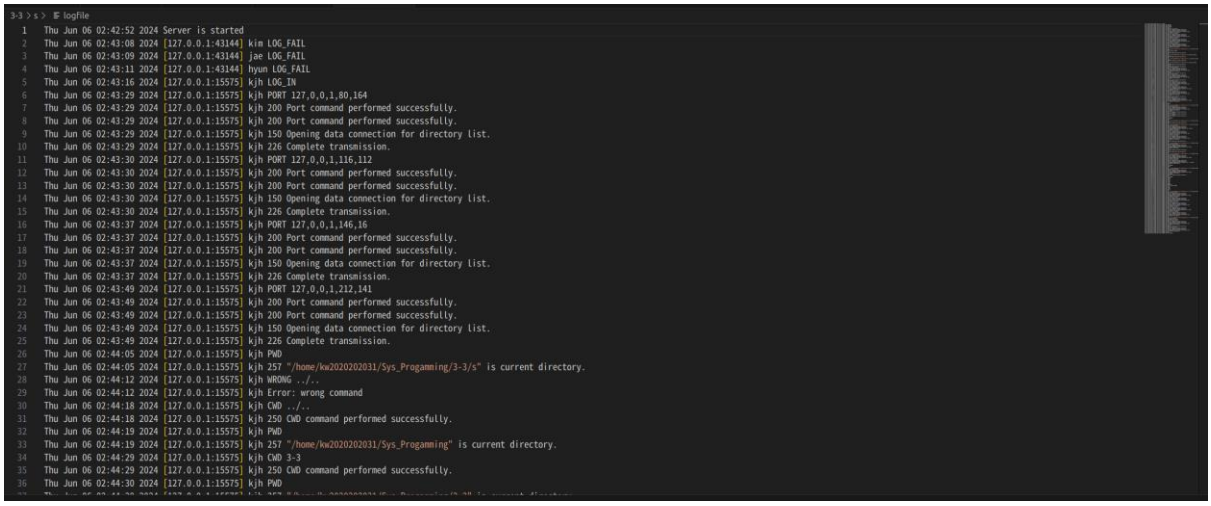

rename 명령어를 통해 server 측 파일의 이름을 바꾸는 것을 확인할 수 있습니다.

```
> delete hi
250 DELE command performed successfully.
> ls
converting to PORT 127,0,0,1,70,43
200 Port command performed successfully.
150 Opening data connection for directory list.
access.txt
client
logfile
motd
passwd
srv
226 Complete transmission.
OK. 42 bytes is received.
> quit
221 Goodbye.
kw2020202031@ubuntu:~/Sys_Progamming/3-3/c$
```

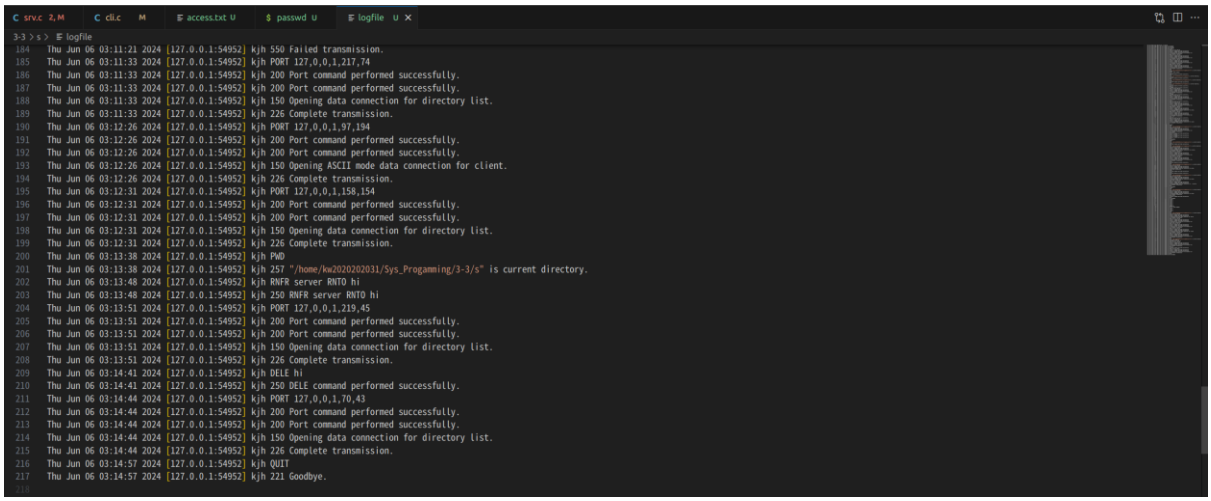
delete 명령어를 통해 파일을 삭제하는 것을 확인할 수 있습니다.



client 다중연결 또한 잘 수행되는 것을 확인할 수 있습니다.



```
3-3 > s > logfile
1  Thu Jun 06 02:42:52 2024 Server is started
2  Thu Jun 06 02:43:08 2024 [127.0.0.1:43144] kim LOG_FAIL
3  Thu Jun 06 02:43:09 2024 [127.0.0.1:43144] jae LOG_FAIL
4  Thu Jun 06 02:43:11 2024 [127.0.0.1:43144] hyun LOG_FAIL
5  Thu Jun 06 02:43:16 2024 [127.0.0.1:15575] kjh LOG_IN
6  Thu Jun 06 02:43:29 2024 [127.0.0.1:15575] kjh PORT 127,0,0,1,80,164
7  Thu Jun 06 02:43:29 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
8  Thu Jun 06 02:43:29 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
9  Thu Jun 06 02:43:29 2024 [127.0.0.1:15575] kjh 150 Opening data connection for directory list.
10 Thu Jun 06 02:43:29 2024 [127.0.0.1:15575] kjh 226 Complete transmission.
11 Thu Jun 06 02:43:30 2024 [127.0.0.1:15575] kjh PORT 127,0,0,1,116,112
12 Thu Jun 06 02:43:30 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
13 Thu Jun 06 02:43:30 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
14 Thu Jun 06 02:43:30 2024 [127.0.0.1:15575] kjh 150 Opening data connection for directory list.
15 Thu Jun 06 02:43:30 2024 [127.0.0.1:15575] kjh 226 Complete transmission.
16 Thu Jun 06 02:43:37 2024 [127.0.0.1:15575] kjh PORT 127,0,0,1,146,16
17 Thu Jun 06 02:43:37 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
18 Thu Jun 06 02:43:37 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
19 Thu Jun 06 02:43:37 2024 [127.0.0.1:15575] kjh 150 Opening data connection for directory list.
20 Thu Jun 06 02:43:37 2024 [127.0.0.1:15575] kjh 226 Complete transmission.
21 Thu Jun 06 02:43:49 2024 [127.0.0.1:15575] kjh PORT 127,0,0,1,212,141
22 Thu Jun 06 02:43:49 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
23 Thu Jun 06 02:43:49 2024 [127.0.0.1:15575] kjh 200 Port command performed successfully.
24 Thu Jun 06 02:43:49 2024 [127.0.0.1:15575] kjh 150 Opening data connection for directory list.
25 Thu Jun 06 02:43:49 2024 [127.0.0.1:15575] kjh 226 Complete transmission.
26 Thu Jun 06 02:44:05 2024 [127.0.0.1:15575] kjh PWD
27 Thu Jun 06 02:44:05 2024 [127.0.0.1:15575] kjh 257 "/home/kw2020202031/Sys_Progamming/3-3/s" is current directory.
28 Thu Jun 06 02:44:12 2024 [127.0.0.1:15575] kjh WRONG ../..
29 Thu Jun 06 02:44:12 2024 [127.0.0.1:15575] kjh Error: wrong command
30 Thu Jun 06 02:44:18 2024 [127.0.0.1:15575] kjh CWD ../..
31 Thu Jun 06 02:44:18 2024 [127.0.0.1:15575] kjh 250 CWD command performed successfully.
32 Thu Jun 06 02:44:19 2024 [127.0.0.1:15575] kjh PWD
33 Thu Jun 06 02:44:19 2024 [127.0.0.1:15575] kjh 257 "/home/kw2020202031/Sys_Progamming" is current directory.
34 Thu Jun 06 02:44:29 2024 [127.0.0.1:15575] kjh CWD 3-3
35 Thu Jun 06 02:44:29 2024 [127.0.0.1:15575] kjh 250 CWD command performed successfully.
36 Thu Jun 06 02:44:30 2024 [127.0.0.1:15575] kjh PWD
```



```
srv.c 2,M   cli.c M   access.txt U   passwd U   logfile U
3-3 > s > logfile
184 Thu Jun 06 03:11:21 2024 [127.0.0.1:54952] kjh 550 Failed transmission.
185 Thu Jun 06 03:11:33 2024 [127.0.0.1:54952] kjh PORT 127,0,0,1,217,74
186 Thu Jun 06 03:11:33 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
187 Thu Jun 06 03:11:33 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
188 Thu Jun 06 03:11:33 2024 [127.0.0.1:54952] kjh 150 Opening data connection for directory list.
189 Thu Jun 06 03:11:33 2024 [127.0.0.1:54952] kjh 226 Complete transmission.
190 Thu Jun 06 03:12:26 2024 [127.0.0.1:54952] kjh PORT 127,0,0,1,97,194
191 Thu Jun 06 03:12:26 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
192 Thu Jun 06 03:12:26 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
193 Thu Jun 06 03:12:26 2024 [127.0.0.1:54952] kjh 150 Opening ASCII mode data connection for client.
194 Thu Jun 06 03:12:26 2024 [127.0.0.1:54952] kjh 226 Complete transmission.
195 Thu Jun 06 03:12:31 2024 [127.0.0.1:54952] kjh PORT 127,0,0,1,158,154
196 Thu Jun 06 03:12:31 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
197 Thu Jun 06 03:12:31 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
198 Thu Jun 06 03:12:31 2024 [127.0.0.1:54952] kjh 150 Opening data connection for directory list.
199 Thu Jun 06 03:12:31 2024 [127.0.0.1:54952] kjh 226 Complete transmission.
200 Thu Jun 06 03:13:38 2024 [127.0.0.1:54952] kjh PWD
201 Thu Jun 06 03:13:38 2024 [127.0.0.1:54952] kjh 257 "/home/kw2020202031/Sys_Progamming/3-3/s" is current directory.
202 Thu Jun 06 03:13:48 2024 [127.0.0.1:54952] kjh RNFR server RNTO hi
203 Thu Jun 06 03:13:48 2024 [127.0.0.1:54952] kjh 250 RNFR server RNTO hi
204 Thu Jun 06 03:13:51 2024 [127.0.0.1:54952] kjh PORT 127,0,0,1,219,45
205 Thu Jun 06 03:13:51 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
206 Thu Jun 06 03:13:51 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
207 Thu Jun 06 03:13:51 2024 [127.0.0.1:54952] kjh 150 Opening data connection for directory list.
208 Thu Jun 06 03:13:51 2024 [127.0.0.1:54952] kjh 226 Complete transmission.
209 Thu Jun 06 03:14:41 2024 [127.0.0.1:54952] kjh DELE hi
210 Thu Jun 06 03:14:41 2024 [127.0.0.1:54952] kjh 250 DELE command performed successfully.
211 Thu Jun 06 03:14:44 2024 [127.0.0.1:54952] kjh PORT 127,0,0,1,70,43
212 Thu Jun 06 03:14:44 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
213 Thu Jun 06 03:14:44 2024 [127.0.0.1:54952] kjh 200 Port command performed successfully.
214 Thu Jun 06 03:14:44 2024 [127.0.0.1:54952] kjh 150 Opening data connection for directory list.
215 Thu Jun 06 03:14:44 2024 [127.0.0.1:54952] kjh 226 Complete transmission.
216 Thu Jun 06 03:14:57 2024 [127.0.0.1:54952] kjh QUIT
217 Thu Jun 06 03:14:57 2024 [127.0.0.1:54952] kjh 221 Goodbye.
```

server 와 client 간 통신내역이 logfile 에 기록되는 것을 확인할 수 있습니다.

# 고찰

assignment #1, #2, #3 을 수행하면서 나름 짜임새 있게 코드를 작성했다고 생각했는데 이번 과제에서 지금까지 작성했던 코드들을 총망라하여 합치려고 하다 보니, 코드들의 구조를 뒤집어 엎어야 하거나, 수정해야 하는 부분이 많았습니다. 이러한 과정을 겪으면서, 처음부터 전체적인 구조를 파악하고, flowchart 를 통해 논리구조를 기록해 놓는 것이 차후에 세부적인 코드들을 작성할 때 도움이 될 것 같다는 생각이 들었습니다.

그리고 하나의 함수가 너무 많은 내용을 포함하지 말아야 하고, 한 가지의 역할만을 수행하도록 해야, 차후에 함수를 수정하거나 다른 곳에 적응할 때 수월하게 할 수 있다는 점을 깨달았습니다.

# Reference

시스템프로그래밍실습 / 광운대학교 / 최상호 교수님 / 2024-1_SPLab_07_FTP3_3_v1
시스템프로그래밍실습 / 광운대학교 / 최상호 교수님 / 2024-1_SPLab_FTP_Assginment3_3_v2