시스템 프로그래밍 실습

# [Assignment #3-1: login]


**Class** : [금요일 1,2 교시]

**Professor** : [최상호 교수님]

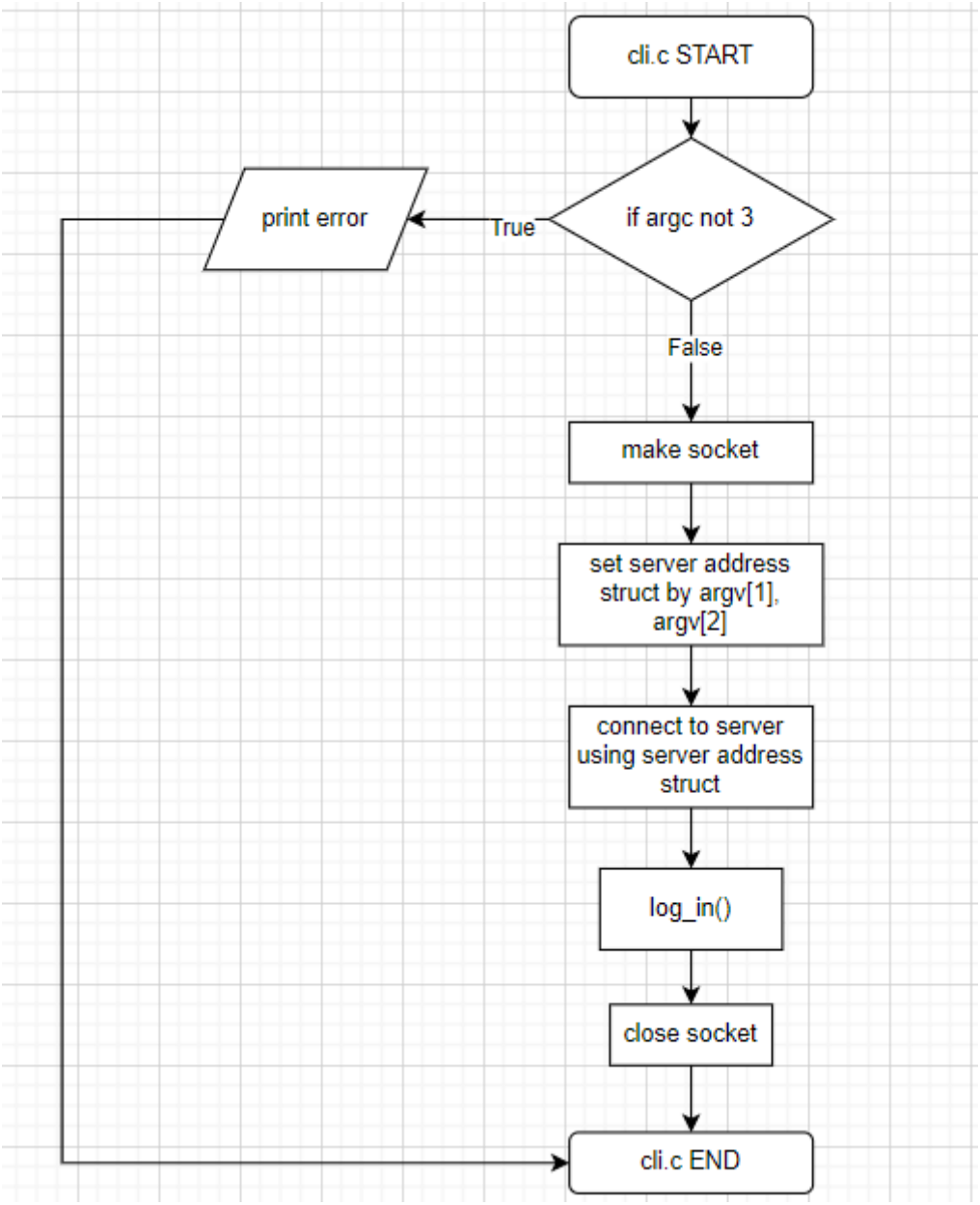
**Student ID** : [2020202034]

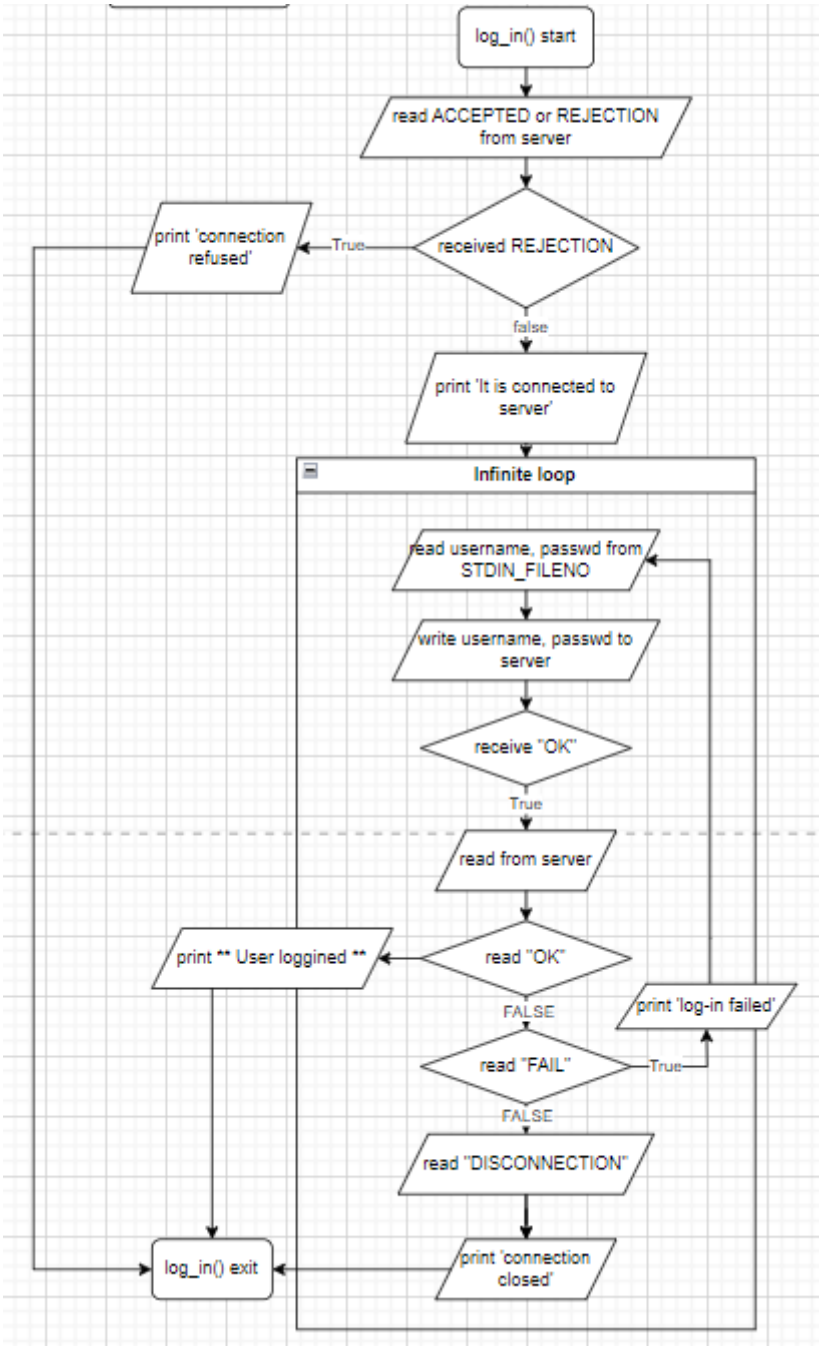
**Name** : [김태완]

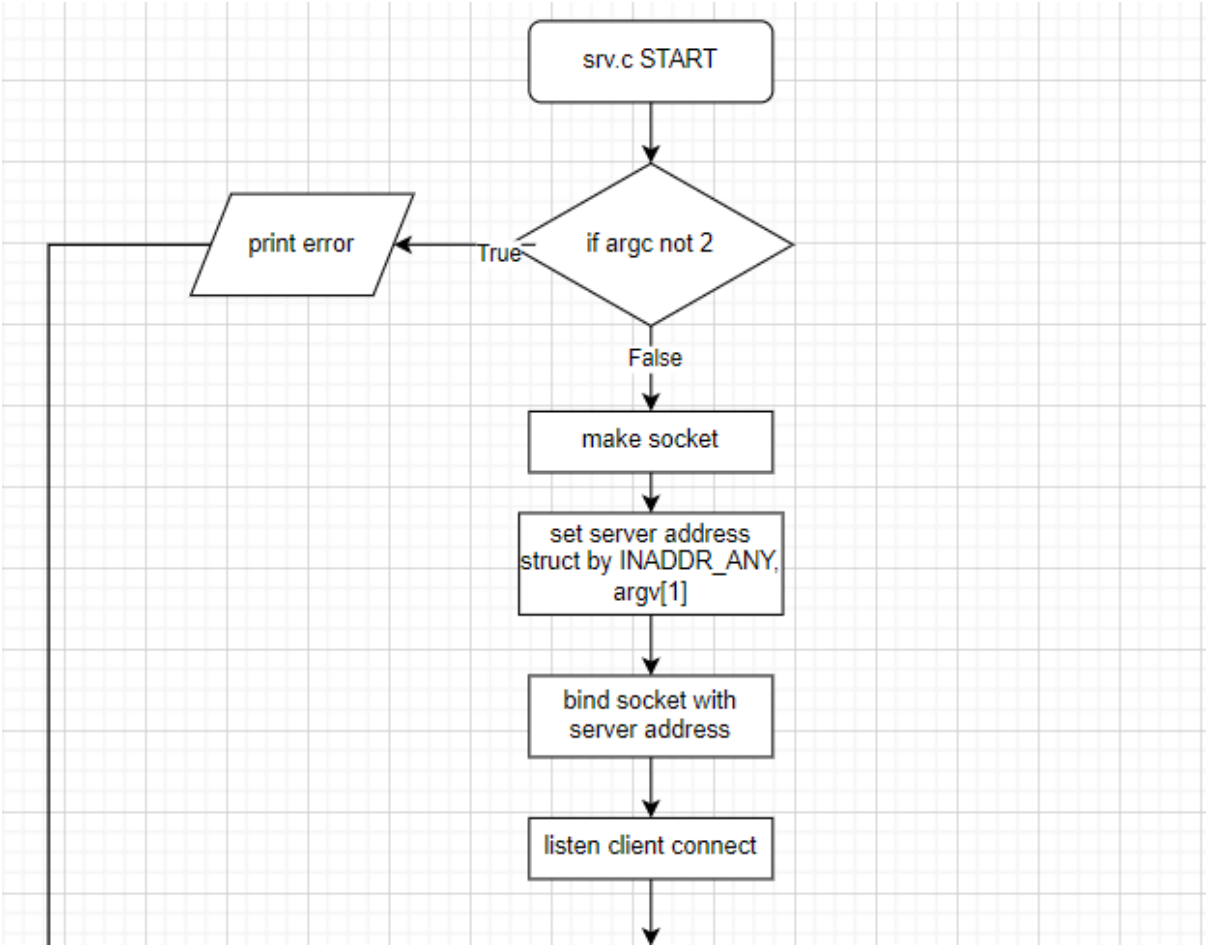
# Introduction

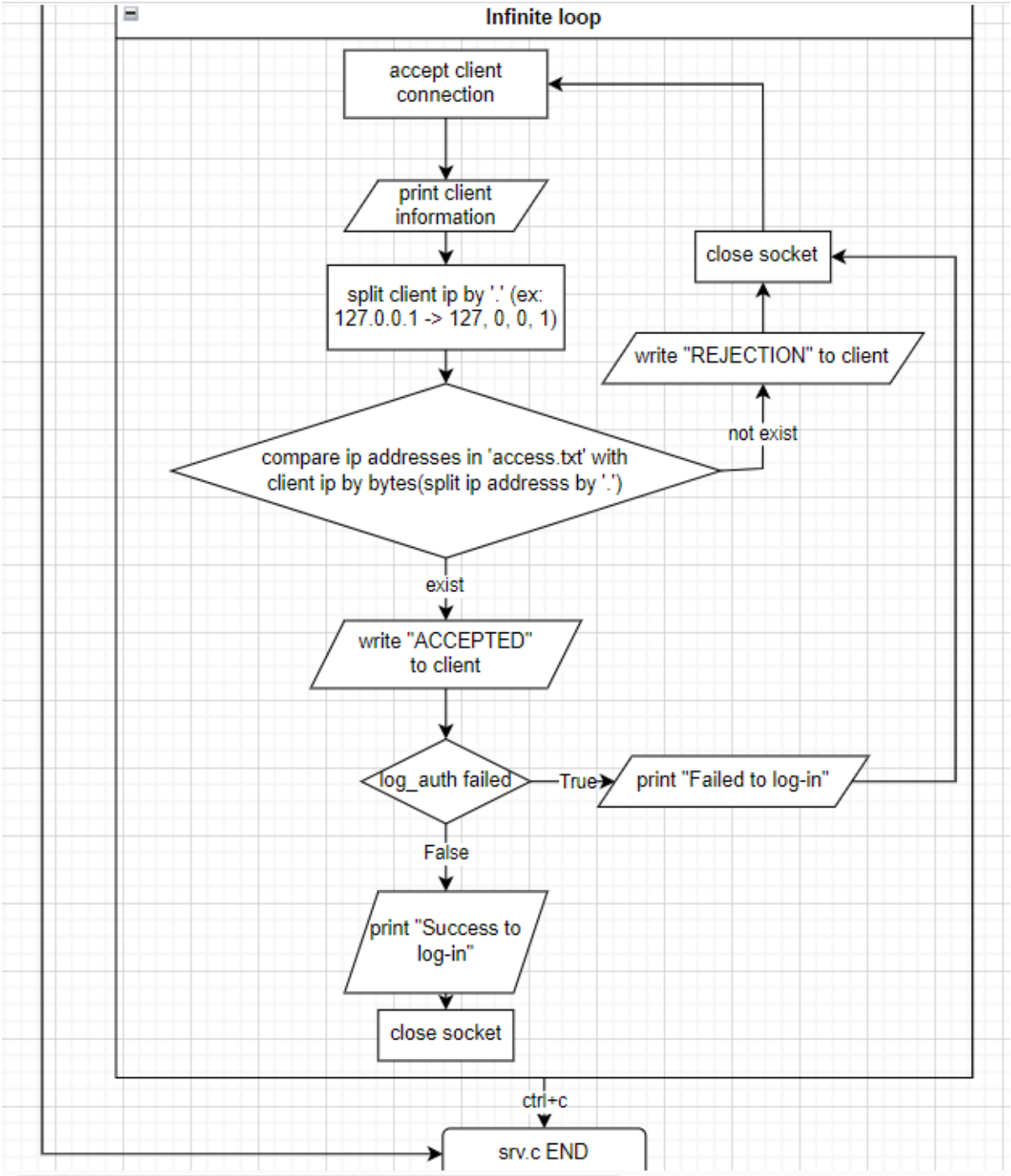
이번 과제는 Assignment #2 에서 진행한 socket 통신을 이용해 client, server 를 만들고, 이에 추가로 access.txt, passwd 를 이용해 ip accept & reject 와 login system 을 구현하는 내용입니다. 기존의 개념에 추가적으로 high level I/O 를 이용하여 access.txt, passwd 를 open, read 해야 합니다. 현재는 하나의 서버, 하나의 클라이언트에서만 로그인 시스템을 구현하는 것이지만, 추가적으로 다수의 클라이언트에서 연결 요청을 수행해도 login 을 수행하는 시스템을 만들 수 있을 것입니다.

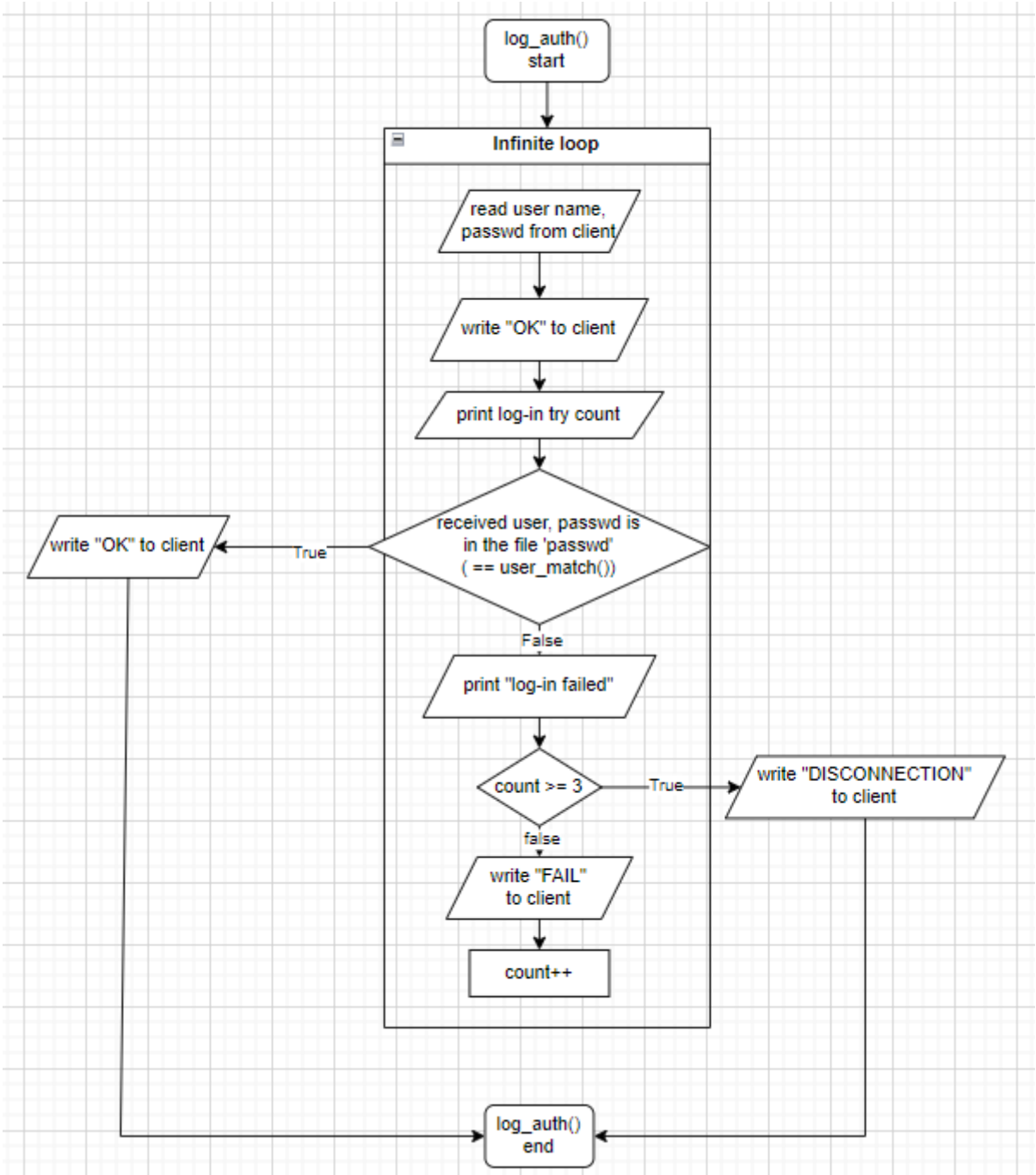
# Flow chart




log_in() start
read ACCEPTED or REJECTION from server
received REJECTION
True
print 'connection refused'
false
print 'It is connected to server'
Infinite loop
read username, passwd from STDIN_FILENO
write username, passwd to server
receive "OK"
True
read from server
read "OK"
print ** User loggined **
FALSE
print 'log-in failed'
read "FAIL"
True
FALSE
read "DISCONNECTION"
print 'connection closed'
log_in() exit


srv.c START
if argc not 2
True
print error
False
make socket
set server address struct by INADDR_ANY, argv[1]
bind socket with server address
listen client connect


Infinite loop
accept client connection
print client information
split client ip by '.' (ex: 127.0.0.1 -> 127, 0, 0, 1)
compare ip addresses in 'access.txt' with client ip by bytes(split ip addresss by '.')
not exist
write "REJECTION" to client
close socket
exist
write "ACCEPTED" to client
log_auth failed
True
print "Failed to log-in"
False
print "Success to log-in"
close socket
ctrl+c
srv.c END




# Pseudo code

## cli.c

main:

if number of arguments != 3:

print error message

```
        exit

    create socket
    if socket creation failed:
        print error message
        exit

    set server address structure
    connect to server
    if connection failed:
        print error message
        exit

    call log_in function with socket descriptor
    close connection
    return

log_in:
    read from server
    if read failed:
        print error message
        exit
    if received "REJECTION":
        print "Connection refused"
```

```
        close connection
        exit
else:
        print "It is connected to Server"

loop:
        print "Input ID : "
        read ID from user
        if read failed:
                exit
        send ID to server
        if send failed:
                exit

        store ID in user variable
        read password from user
        send password to server
        if send failed:
                exit

        read result from server
        if read failed:
                exit
        if received "OK":
```

```
        read response from server
        if read failed:
            exit
        if received "OK":
            print "User '[ID]' logged in"
            break loop
        else if received "FAIL":
            print "Log-in failed"
        else:
            print "Connection closed"
            break loop
```

## srv.c

```
main:
    if number of arguments != 2:
        print error message
        exit

    create socket for server
    if socket creation failed:
        print error message
        exit

    set server address structure
    bind server address to server socket
```

```
if binding failed:
    print error message
    exit

listen for incoming connections with queue size 5

loop:
    accept client connection
    if accept failed:
        print error message
        exit
    print "Client is connected"
    print client IP and port

    open "access.txt" file
    if file open failed:
        exit

    store client IP address
    split client IP address by '.'

    loop:
        read line from "access.txt"
        if line does not match client IP address:
```

```
                continue
            else:
                break loop

        close "access.txt" file
        if no matching IP address found:
            print "It is NOT authenticated client"
            send "REJECTION" to client
            close connection
            continue outer loop
        else:
            send "ACCEPTED" to client

        if log_auth function returns 0 (login failed):
            print "Fail to log-in"
            close connection
            continue outer loop
        else:
            print "Success to log-in"
            close connection

log_auth:
    loop:
        read user name and password from client
```

```
            if read failed:
                return 0

            send "OK" to client (read successful)
            print "User is trying to log-in (count/3)"

            call user_match function with user name and password
            if user_match returns 1 (success):
                send "OK" to client
                break loop
            else if user_match returns 0 (failure):
                print "Log-in failed"
                if login attempts >= 3:
                    send "DISCONNECTION" to client
                    return 0
                else:
                    send "FAIL" to client
                    increment login attempt count

        return 1 (success)

user_match:
        open "passwd" file
        loop:
```

read next entry from "passwd" file

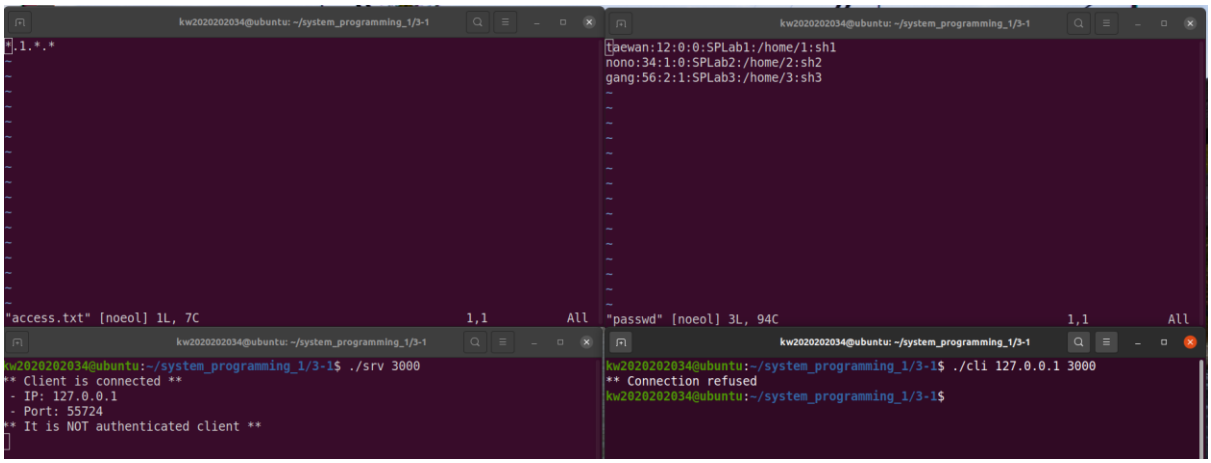
if user name and password match the entry:

return 1 (success)

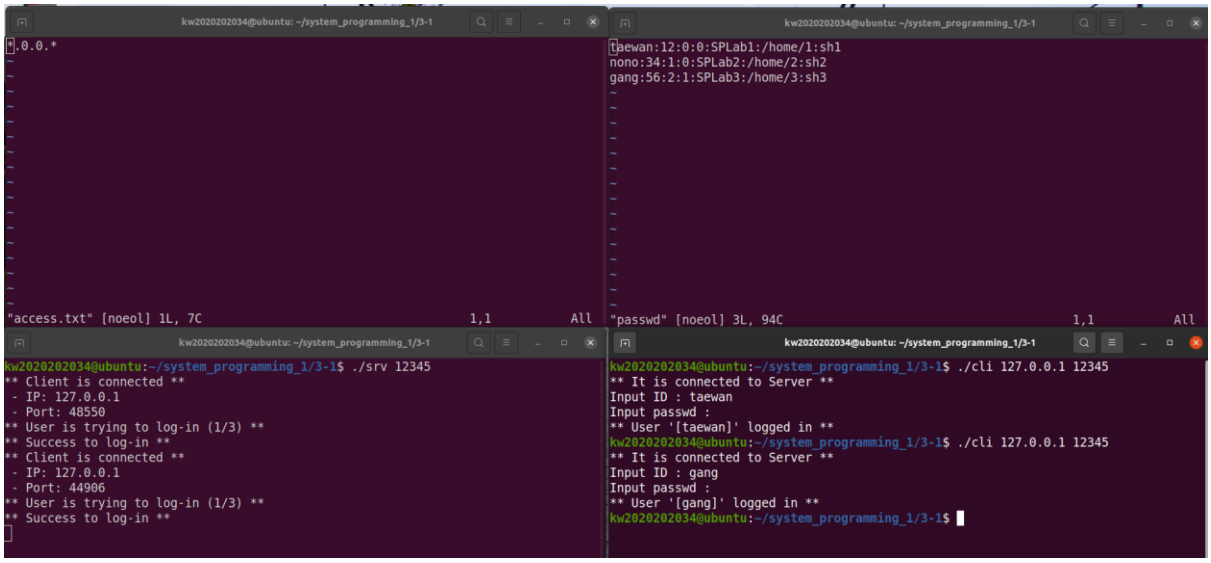
close "passwd" file

return 0 (failure)

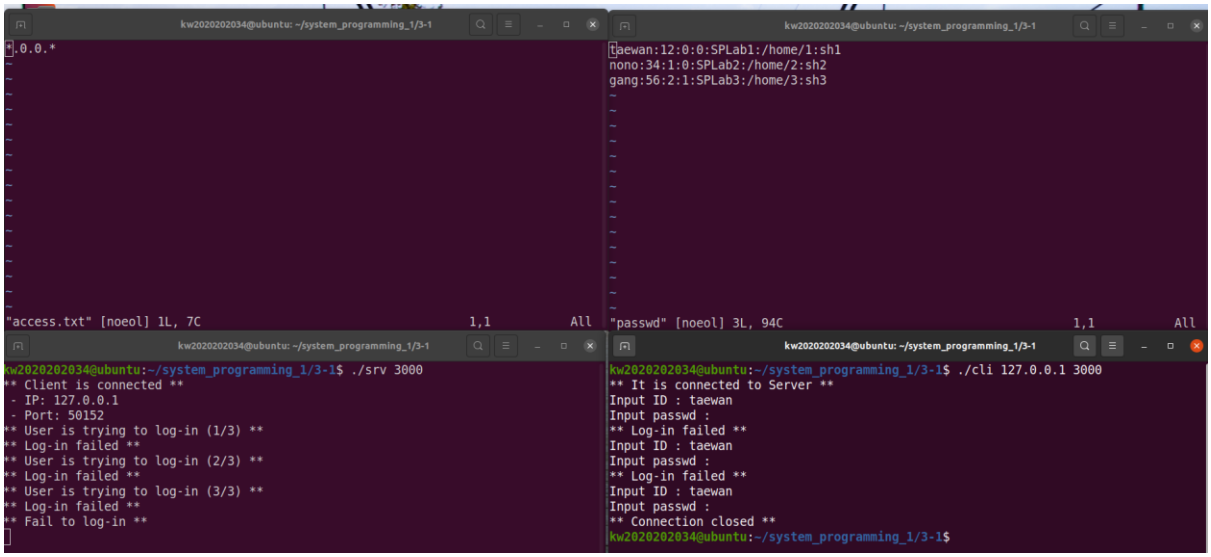
# 결과화면




접속을 시도하는 ip address 가 access.txt 에 존재하지 않아 접속이 불가한 경우를 테스트하였습니다. client 의 ip 는 127.0.0.1 로, wildcard 를 포함한 access.txt 의 *.1.*.*와 맞지 않습니다. 따라서 client 에서 서버로 접속 시에 서버에 클라이언트의 정보는 뜨지만, 접속은 허가되지 않아 client 는 즉시 종료되고, server 에는 auth 된 client 가 아니라는 문구가 뜨게 됩니다.



다음은 정상적으로 로그인이 된 경우를 테스트하였습니다. access.txt 에 client 의 ip address 127.0.0.1 과 일치하는 *.0.0.*가 작성되어 있으므로 접속이 성공적으로 이루어진 것을 볼 수 있고, passwd 파일에 존재하는 ID, passwd 를 입력하였을 때 로그인이 성공적으로 이루어진 것을 볼 수 있습니다(taewan, gang 2 user test). 그리고 사용자가 비밀번호를 입력할 때 이것이 출력에 보이지 않을 수 있도록 getpass()를 사용하였습니다.




다음은 3 번 로그인에 실패해 client 가 종료되는 경우를 테스트하였습니다. 존재하는 user name 'taewan'과 틀린 비밀번호를 3 번 연속 입력하니 각각의 경우에 서버에서는 로그인 시도 횟수, 로그인 실패 알림이 뜨고, 클라이언트에서는 로그인 실패 메시지가 뜨는 것을 볼 수 있습니다. 3 번째 로그인이 실패했을 때는 서버에는 'Fail to log-in'이, 클라이언트에는 연결이 종료되었음이 뜨고 종료됨을 볼 수 있습니다.

# 고찰

기본적으로 소켓 통신 부분은 기존 과제를 그대로 차용하여 크게 어렵지 않았고, client, server 간에 메시지를 주고 받는 것도 기존에 해오던 거라 크게 어렵지 않았습니다. 다만 fopen 으로 access.txt, passwd 를 열어 파싱해 ip 가 존재하는지(특히 wildcard 부분), ID, password 가 있는지(이 부분은 fgetpwent()로 쉽게 구현함)를 구현하는 것이 조금 생각할 거리가 있어 시간이 조금 소요되었습니다.