

시스템 프로그래밍 실습

[FTP 3-1]

Class : D
Professor : 최상호 교수님
Student ID : 2020202092
Name : 강현민

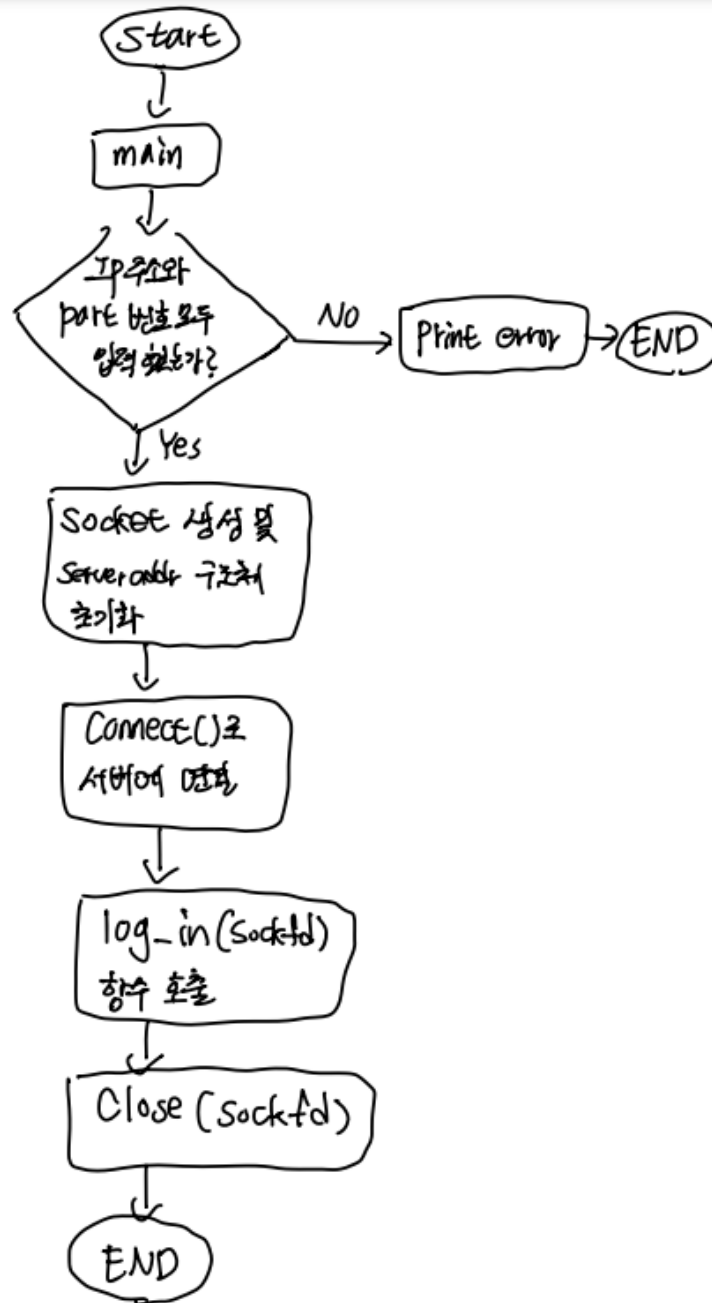
Introduction

이번 [FTP 3-1](#) 과제는 최종적으로 FTP 서버를 구현하는 것을 목적으로 하는 프로젝트의 세 번째 단계인 FTP service completion 단계의 첫번째 과정으로, 사용자 이름과 비밀번호 match 를 통한 사용자 인증, 클라이언트의 ip 주소 인증 등의 서비스 구현을 목적으로 한다. 먼저 클라이언트에서는 기존과 동일하게 ip 주소와 포트번호를 입력하여 접속하며, 이 때 access.txt 에 정의되지 않은 ip 주소로 접속을 시도할 경우 거절된다. 클라이언트가 서버에 성공적으로 연결된 경우에는 user ID 와 passwd 를 입력 해야하며, passwd.txt 에 정의되지 않은 사용자 아이디 혹은 비밀번호를 입력하는 경우 로그인에 실패하고, 3 번 실패한 경우 접속이 해제되도록 코드를 작성하여 구현해야 한다.

Flow chart

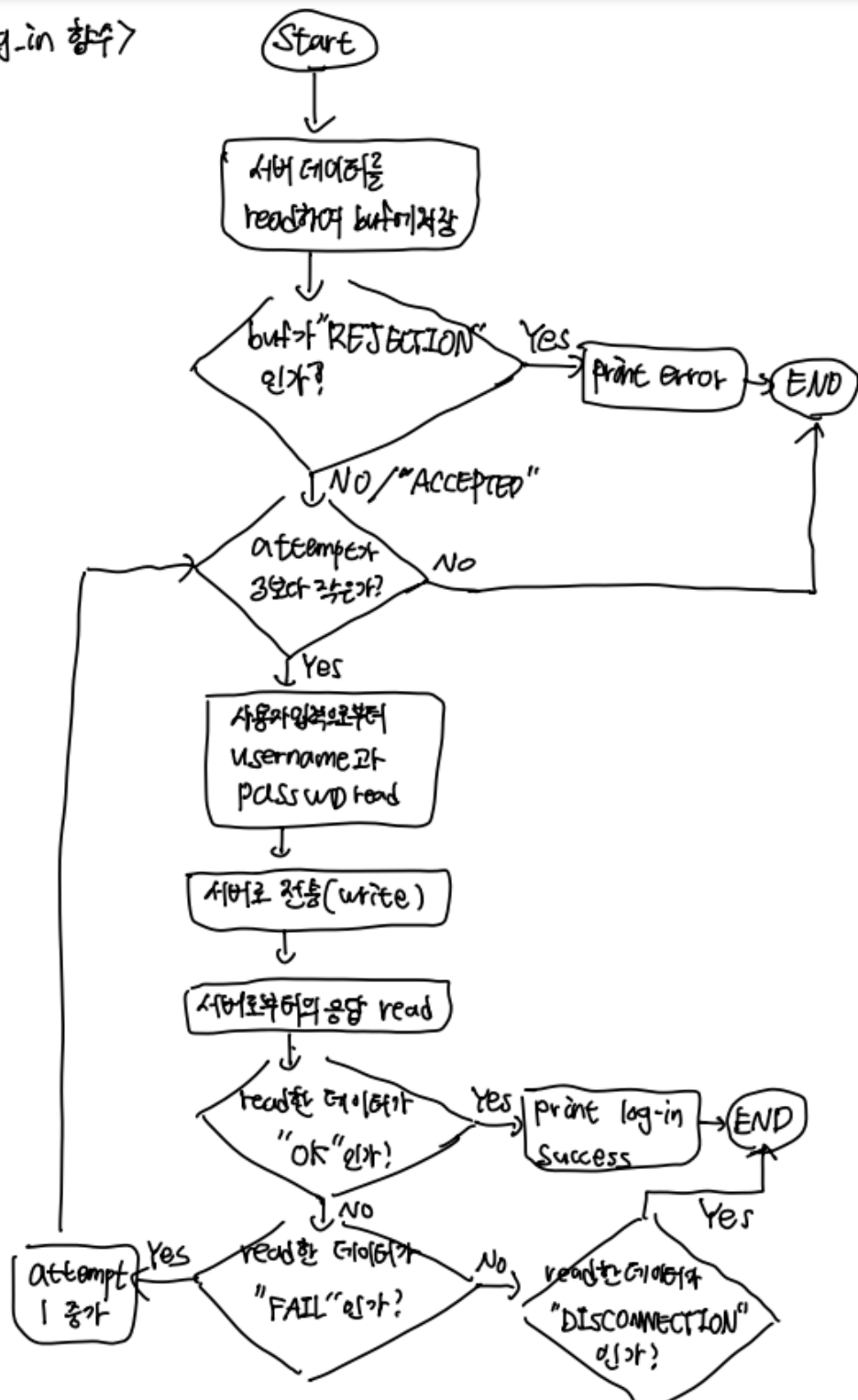
<cli.c>

<cli.c>



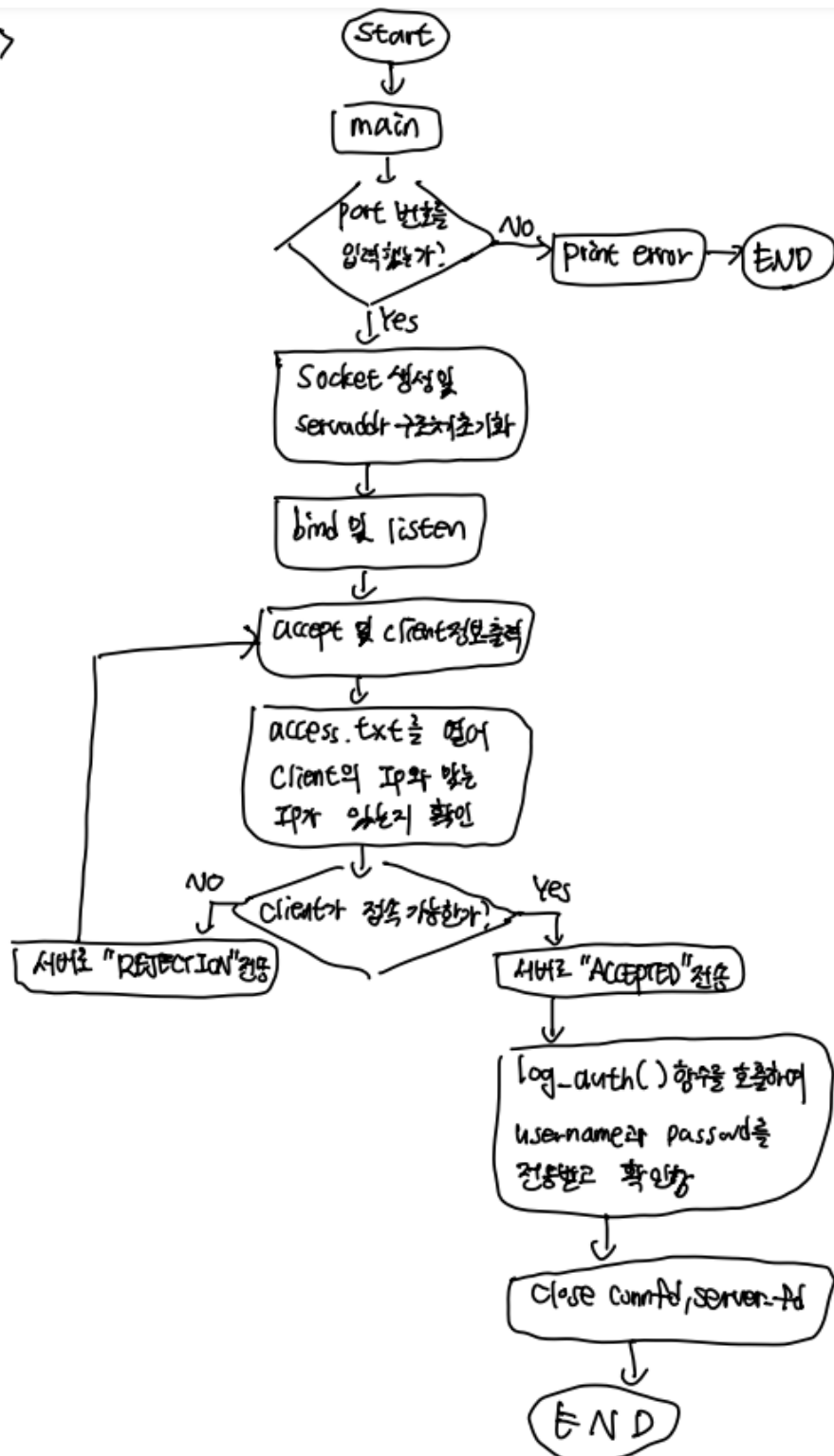
<cli.c: log_in 함수>

<cli.c: log_in 함수>



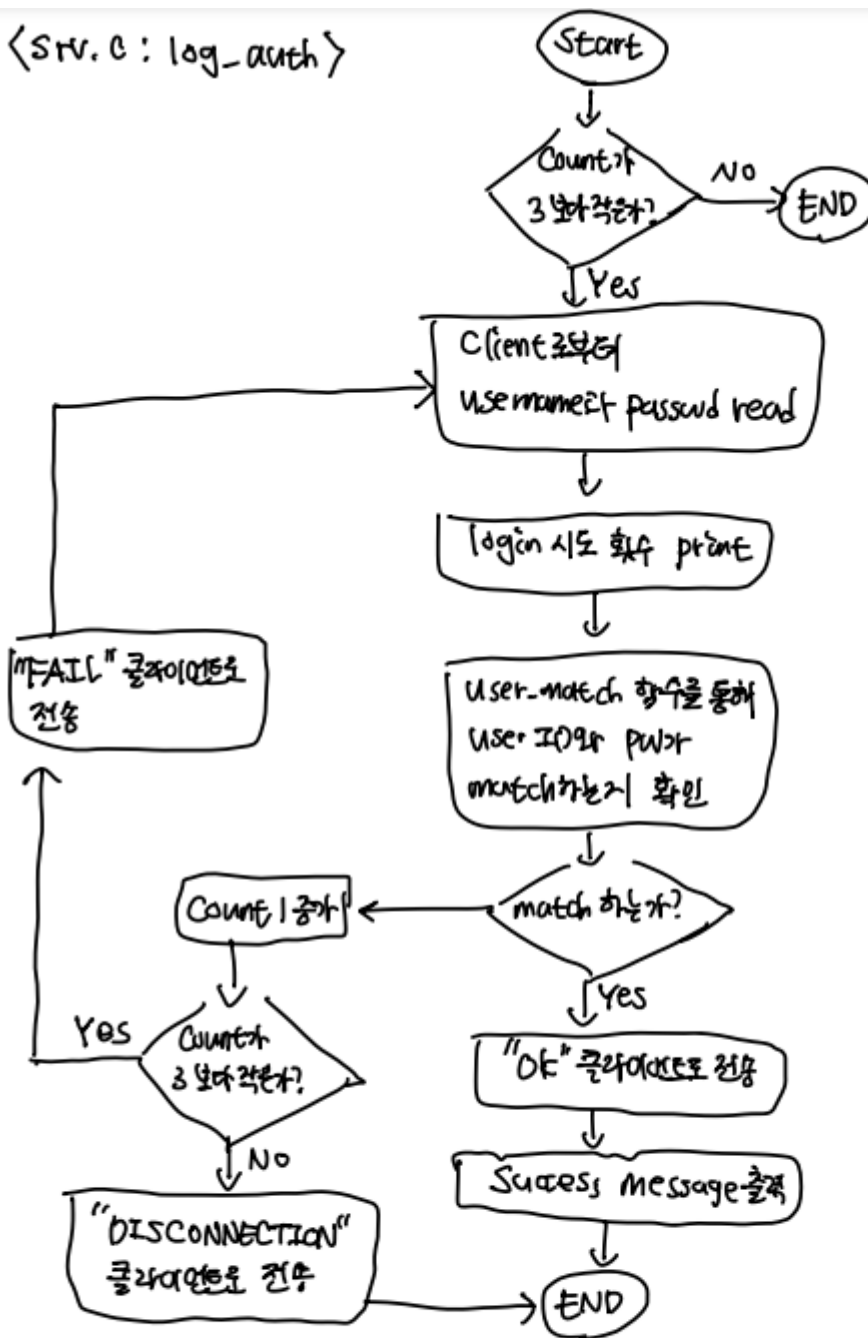
<srv.c>

<srv.c>



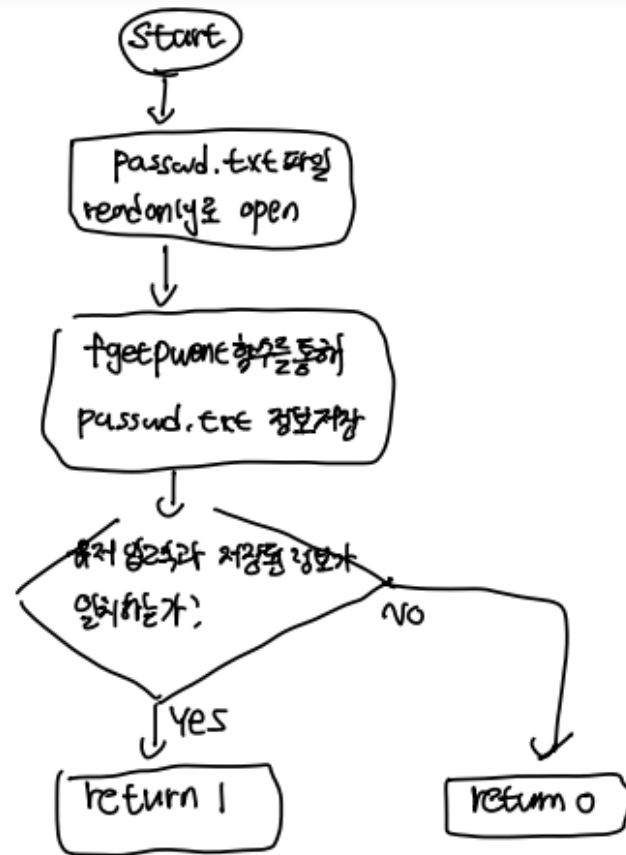
<srv.c : log_auth 함수>

<srv.c : log_auth>



<srv.c : user_match 함수>

<srv.c: user_match 함수>



Pseudo code

<cli.c>

void log_in (int sockfd);

int main(int argc, char* argv[]) {

 if more arguments needed

 print error message

 return -1;

 declare variables to connect with server

 socket(PF_INET, SOCK_STREAM, 0);

 store server information to struct

 connect(sockfd, struct sockaddr* &servaddr, sizeof(servaddr))

 log_in(sockfd);

 close socket fd

 return;

void log_in(int sockfd)

 declare buf to store username and passwd

 read from server

 if message from server is "ACCEPTED"

 for three time attempt

 prompt the user for their username

 store username into buf


```
        prompt the user for their password
        store passwd into buf
    read server's response to the login attempt
    if read data is "OK"
        print success message
        return;
    else if read data is "FAIL"
        print fail message
    else if read data is "DISCONNECTION"
        print connection closed;
        exit(1);
```

<SRV.C>

Int log_auth(int connfd);

Int user_match(char* user, char *passwd);

Void client_info(struct sockaddr_in *cliaddr)

Print client ip address and port number

Int main(int argc, char* argv[])

If more arguments are needed

Print error message

Return -1;

Socket(PF_INET, SOCK_STREAM, 0);

Store client information into struct

Bind(server_fd, struct sockaddr* &servaddr, sizeof(servaddr));

Listen (server_fd, 5);

While(1)

Print client information

fopen the access.txt file

fgets each line in access.txt file

declare access_granted = 0;

if line is "*. *.*.*

```
        access_granted = 1;

        break;

    else if line == client_ip

        access_granted = 1;

    else if last leteer of line is "*" and line == client_ip

        access_granted = 1;

    close access.txt

    if access_granted == 0

        write "REJECTION" to server

    else if access_granted == 1

        write "ACCEPTED" to server

    if(log_auth(connfd) == 0)

        print fail to login

        close

        continue;

    close connfd;

close server fd
```

```

int log(int connfd)

    int count = 0;

    while (count < 3)

        read username from client

        read passwd from client


        print the number of attempt to log-in

    if user_match(user, passwd) == 1)

        write "OK" to client

        print success message

        return 1;

    else

        count++;

        print fail message

        if (count < 3)

            write "FAIL" to client

        else

            write "DISCONNECTION" to client

            return 0;

    return 0;

```

```

int user_match(char *user, char *passwd)

    FILE *fp;

    fopen "passwd.txt" for readonly

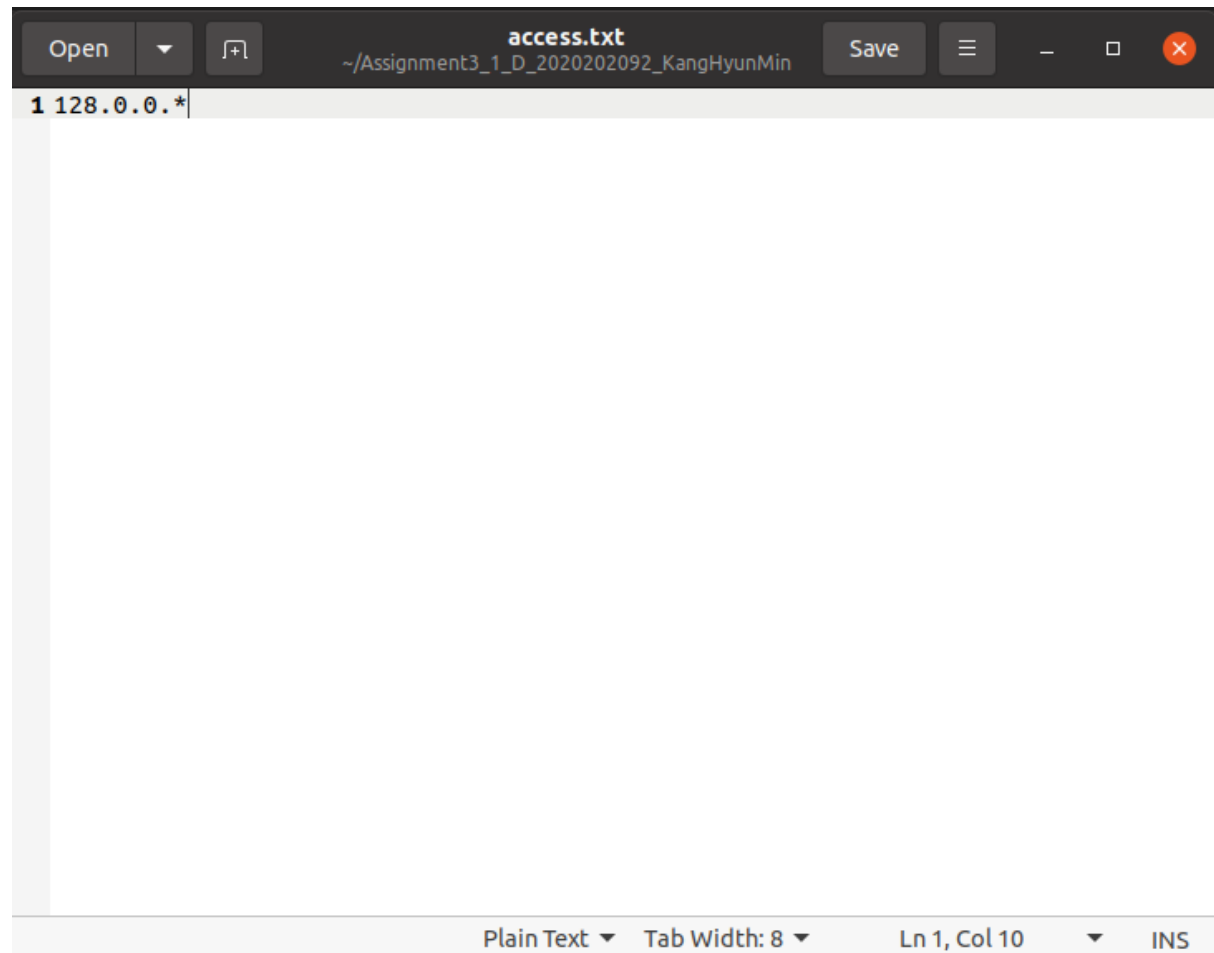
```

```
while (pw = fgetpwent(fp) != NULL)
    store inputtedusername into stored_user
    store inputted passwd into stored_passwd
if they match "passwd.txt" information
    fclose fp
    return -1;

fclose fp
return 0;
```

결과화면

<접속 불가능한 IP 의 클라이언트인 경우>

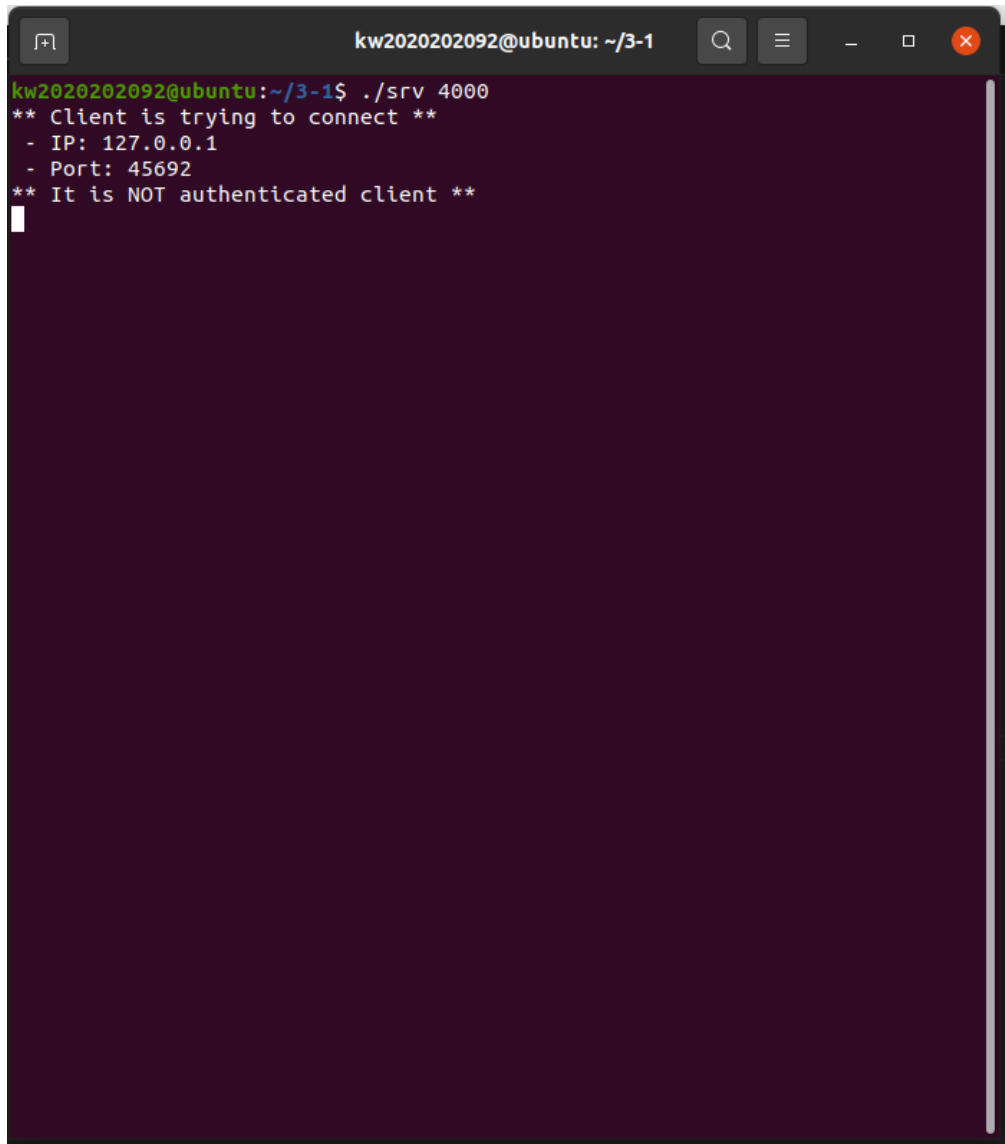


The screenshot shows a text editor window with a dark theme. The title bar at the top reads "access.txt" and the file path is "~/Assignment3_1_D_2020202092_KangHyunMin". The editor contains a single line of text: "1 128.0.0.*". The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 1, Col 10", and "INS".

```
1 128.0.0.*
```

```
Edit Selection View Go Run Terminal Help
kw2020202092@ubuntu: ~/3-1
kw2020202092@ubuntu:~/3-1$ ./cli 127.0.0.1 4000
** It is connected to Server **
** Connection refused **
kw2020202092@ubuntu:~/3-1$
```

(Client)

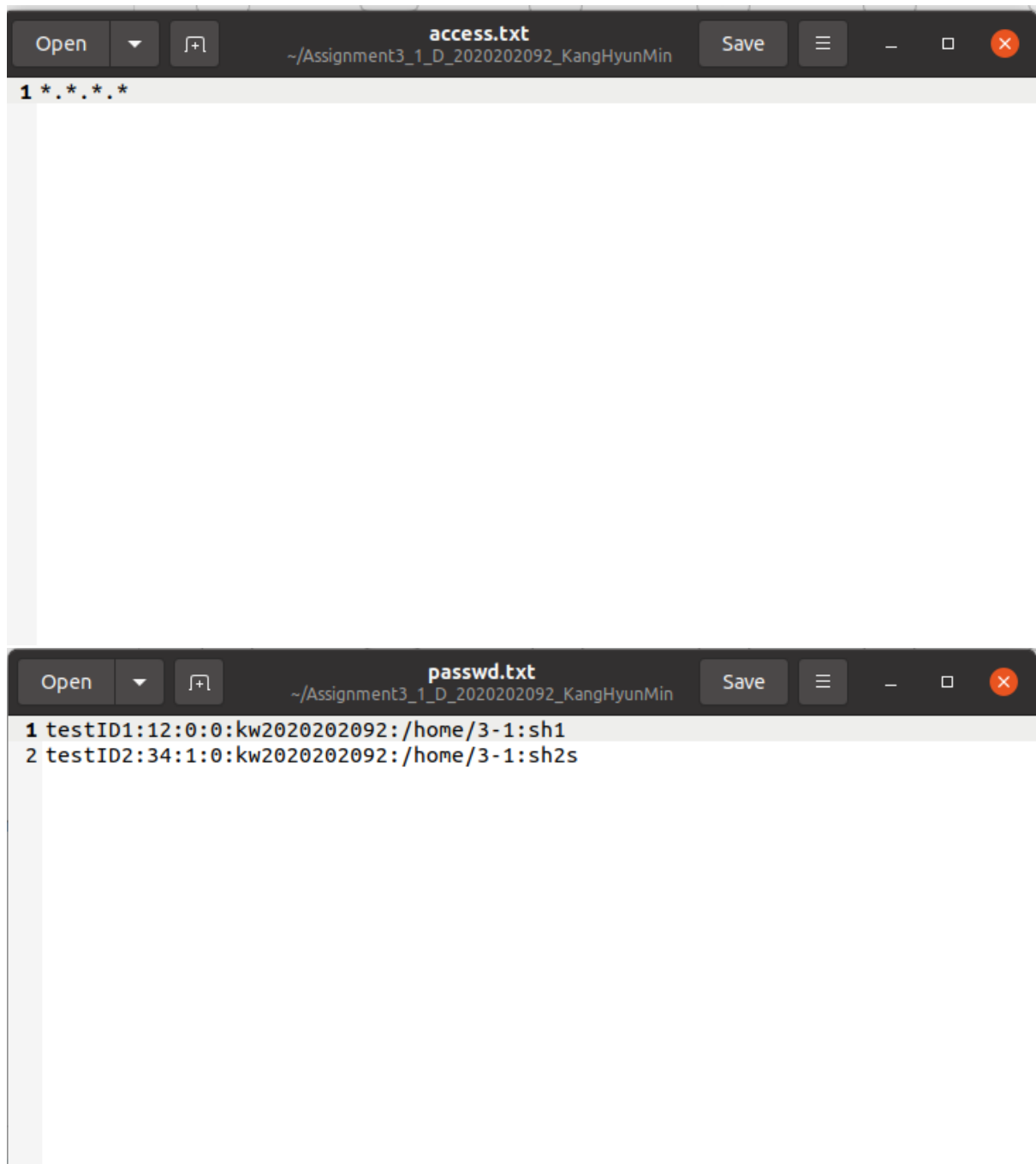


```
kw2020202092@ubuntu: ~/3-1
kw2020202092@ubuntu:~/3-1$ ./srv 4000
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 45692
** It is NOT authenticated client **
```

(Server)

위와 같이 access.txt 파일에 128.0.0.1 만 작성되어 있는 경우에 127.0.0.1 IP 주소를 갖는 클라이언트가 서버에 연결할 경우, 서버에 연결 요청한 클라이언트의 IP 주소와 port 번호가 출력된 후 위와 같은 에러메세지가 출력되며, 클라이언트 또한 서버와 연결된 후 Connection refused 에러메세지가 출력된 후 클라이언트의 연결이 제거되는 것을 확인할 수 있다.

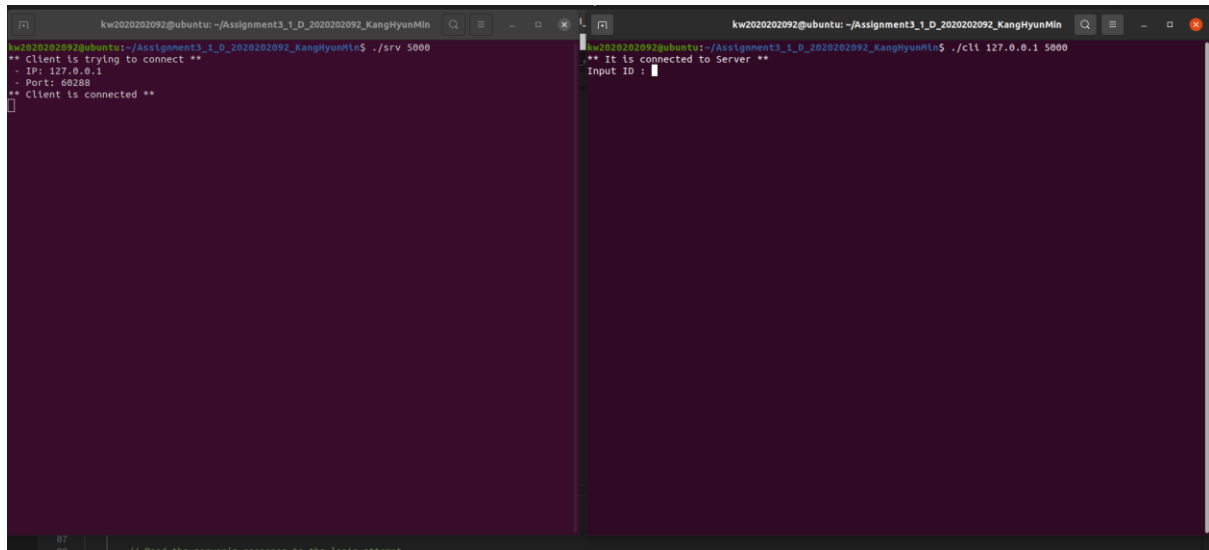
<연결 후 로그인 시도를 세 번 실패한 경우>



The image displays two screenshots of a text editor window. The top screenshot shows a file named 'access.txt' with the content '1 *.*.*.*'. The bottom screenshot shows a file named 'passwd.txt' with two lines of user data: '1 testID1:12:0:0:kw2020202092:/home/3-1:sh1' and '2 testID2:34:1:0:kw2020202092:/home/3-1:sh2s'. Both windows have a dark theme and standard window controls.

```
access.txt
1 *.*.*.*

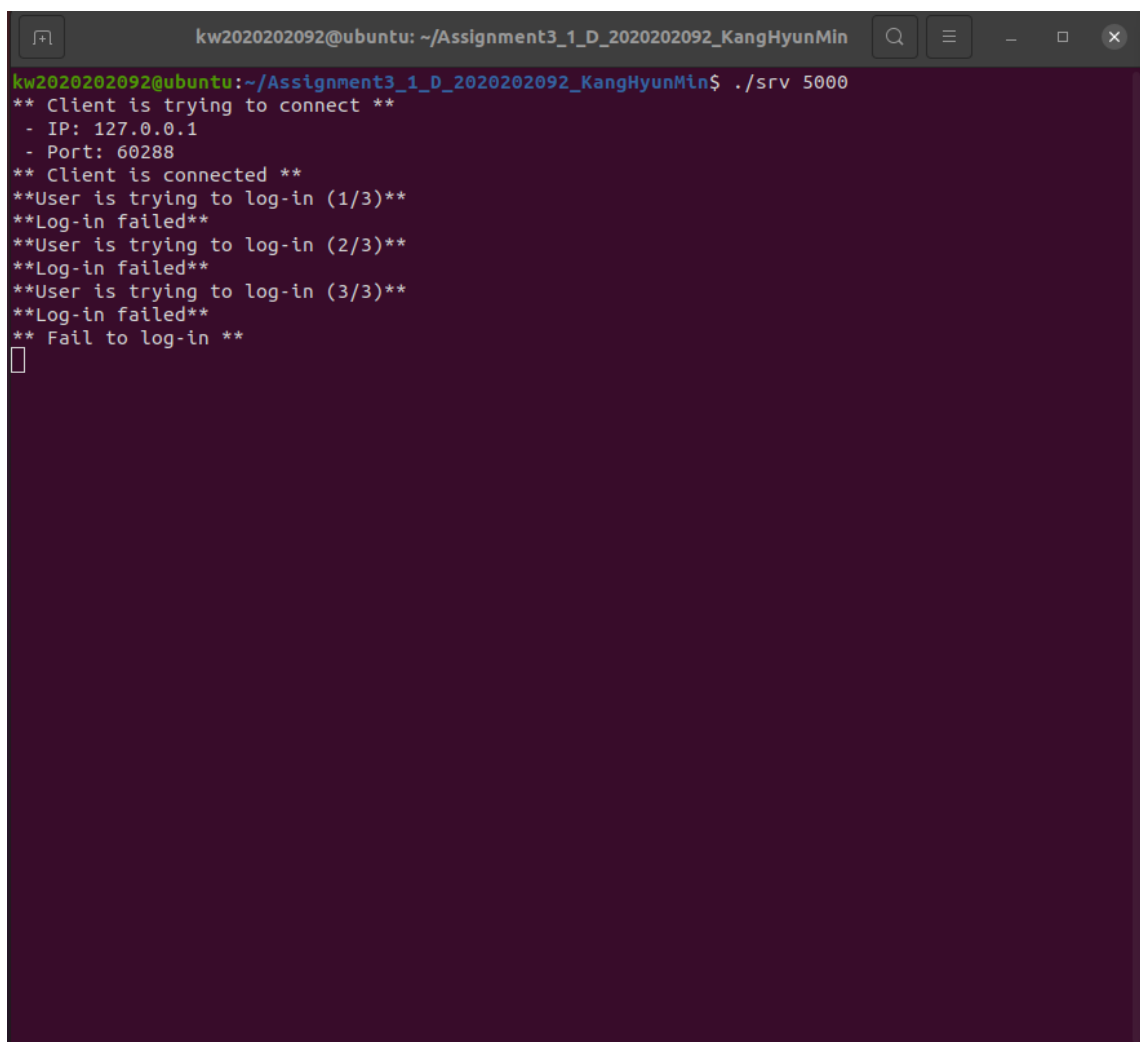
passwd.txt
1 testID1:12:0:0:kw2020202092:/home/3-1:sh1
2 testID2:34:1:0:kw2020202092:/home/3-1:sh2s
```



```
kw2020202092@ubuntu: ~/Assignment3_1_D_2020202092_KangHyunMin
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./srv 5000
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 60288
** Client is connected **

kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
** It is connected to Server **
Input ID : 
```

위와 같이 access.txt 에 *.*.*이 작성된 경우에는 모든 IP 주소에 대해 연결을 허용하므로 위와 같이 클라이언트가 서버에 정상적으로 연결되는 것을 확인할 수 있다.



```
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./srv 5000
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 60288
** Client is connected **
**User is trying to log-in (1/3)**
**Log-in failed**
**User is trying to log-in (2/3)**
**Log-in failed**
**User is trying to log-in (3/3)**
**Log-in failed**
** Fail to log-in **
```

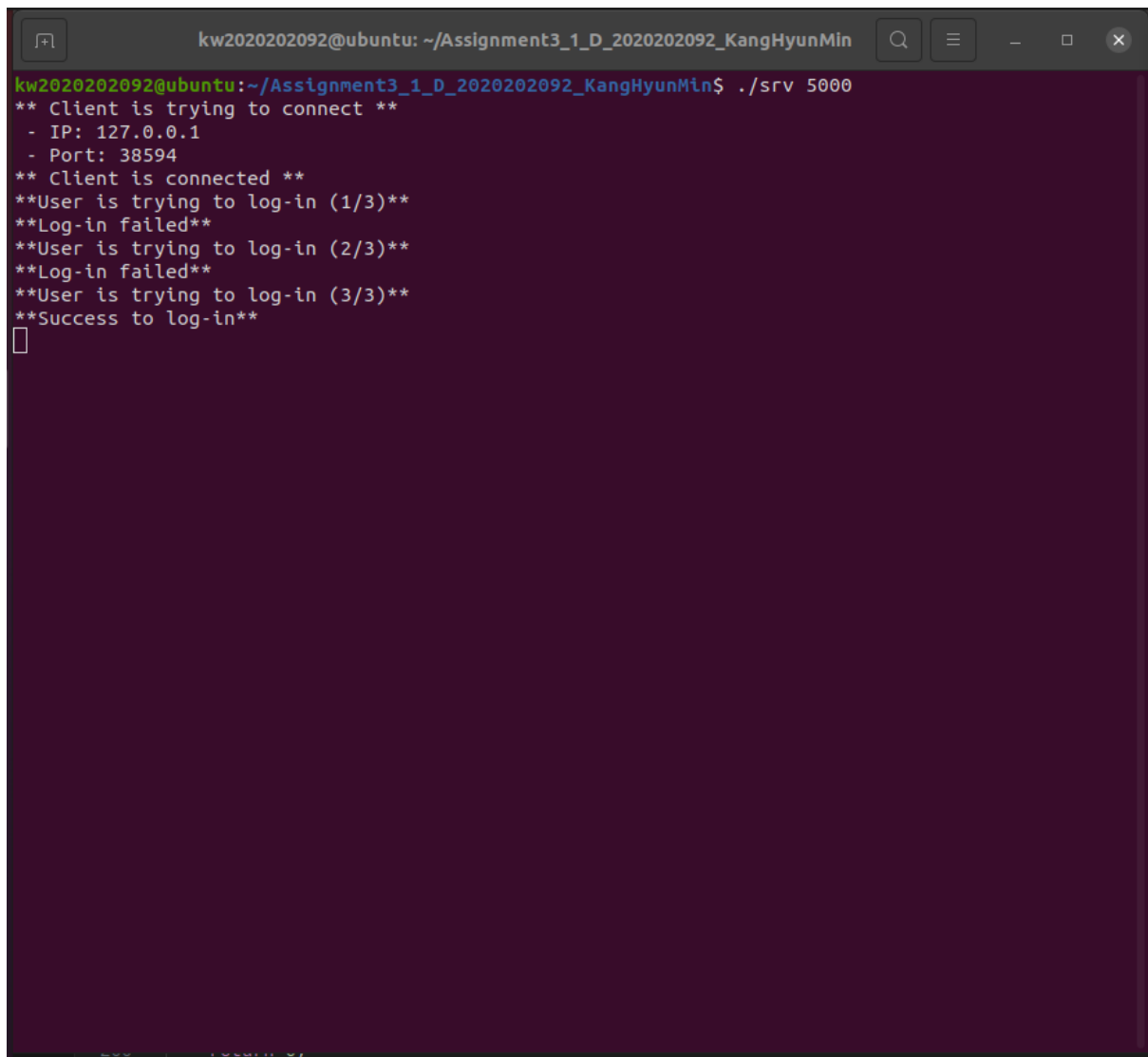
(Server)

```
kw2020202092@ubuntu: ~/Assignment3_1_D_2020202092_KangHyunMin
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
** It is connected to Server **
Input ID : testID1
Input Password : 11
** Log-in failed **
Input ID : testID2
Input Password : 12
** Log-in failed **
Input ID : testID1
Input Password : 12345
** Connection closed **
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$
```

(Client)

위와 같이 passwd.txt 파일에 정의되지 않은 userID 혹은 passwd 를 이용해 로그인을 시도할 경우 서버에는 로그인 시도 횟수와 실패 메시지가, 클라이언트에는 실패 메시지만 출력되는 것을 확인할 수 있다. 또한, 세번의 로그인 시도를 실패할 경우 서버와 클라이언트에 에러 메시지가 출력되며 클라이언트의 연결이 종료되는 것을 확인할 수 있다.

<성공적으로 로그인한 경우>

A terminal window with a dark purple background. The title bar shows the user 'kw2020202092' and the directory '~/Assignment3_1_D_2020202092_KangHyunMin'. The prompt is 'kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin\$'. The user has entered './srv 5000'. The output shows a client connection from 127.0.0.1 on port 38594. The user attempts to log in three times, with the first two failing and the third succeeding. The terminal ends with a cursor on a new line.

```
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./srv 5000
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 38594
** Client is connected **
**User is trying to log-in (1/3)**
**Log-in failed**
**User is trying to log-in (2/3)**
**Log-in failed**
**User is trying to log-in (3/3)**
**Success to log-in**
█
```

(Server)

```
kw2020202092@ubuntu: ~/Assignment3_1_D_2020202092_KangHyunMin
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
** It is connected to Server **
Input ID : testID1
Input Password : 123
** Log-in failed **
Input ID : testID2
Input Password : 12
** Log-in failed **
Input ID : testID1
Input Password : 12
** User 'testID1' logged in **
kw2020202092@ubuntu:~/Assignment3_1_D_2020202092_KangHyunMin$
```

(Client)

Passwd.txt 및 access.txt 는 이전 캡처화면과 동일하게 진행했으며, 먼저 첫 로그인 시도와 두번째 로그인 시도는 이전과 동일하게 실패했을 때 세번째 로그인 시도에서 userID 와 password 를 올바르게 입력한 경우, 위와 같이 서버와 클라이언트에 성공 메시지가 출력되며 클라이언트가 종료되는 것을 확인할 수 있다.

고찰

이번 [FTP 3-1](#) 과제를 통해 네트워크 프로그래밍의 기초 개념과 소켓 프로그래밍의 중요성을 다시 한번 깊이 이해하게 되었다. 이전 과제들과 마찬가지로 클라이언트와 서버 간의 통신을 구현하면서 IP 주소와 포트 번호의 역할, 소켓의 생성과 연결 과정 등을 실습을 통해 체득할 수 있었다. 또한, 사용자 인증 기능을 추가하며 보안의 중요성을 깨달았고, 파일 입출력을 통한 데이터 처리 방법도 익혔다. 하지만, 코드 작성 중 오류 처리와 예외 상황 대응이 미흡했던 점은 개선이 필요하다고 느꼈다. 앞으로 더 효율적이고 안정적인 네트워크 프로그램을 작성하기 위해 코드 최적화와 보안 강화에 대해서 공부해야겠다는 생각이 들었다.

Reference