시스템 프로그래밍 실습

[FTP_Assignment 3-1]

Class : **D**

Professor : **최상호 교수님**

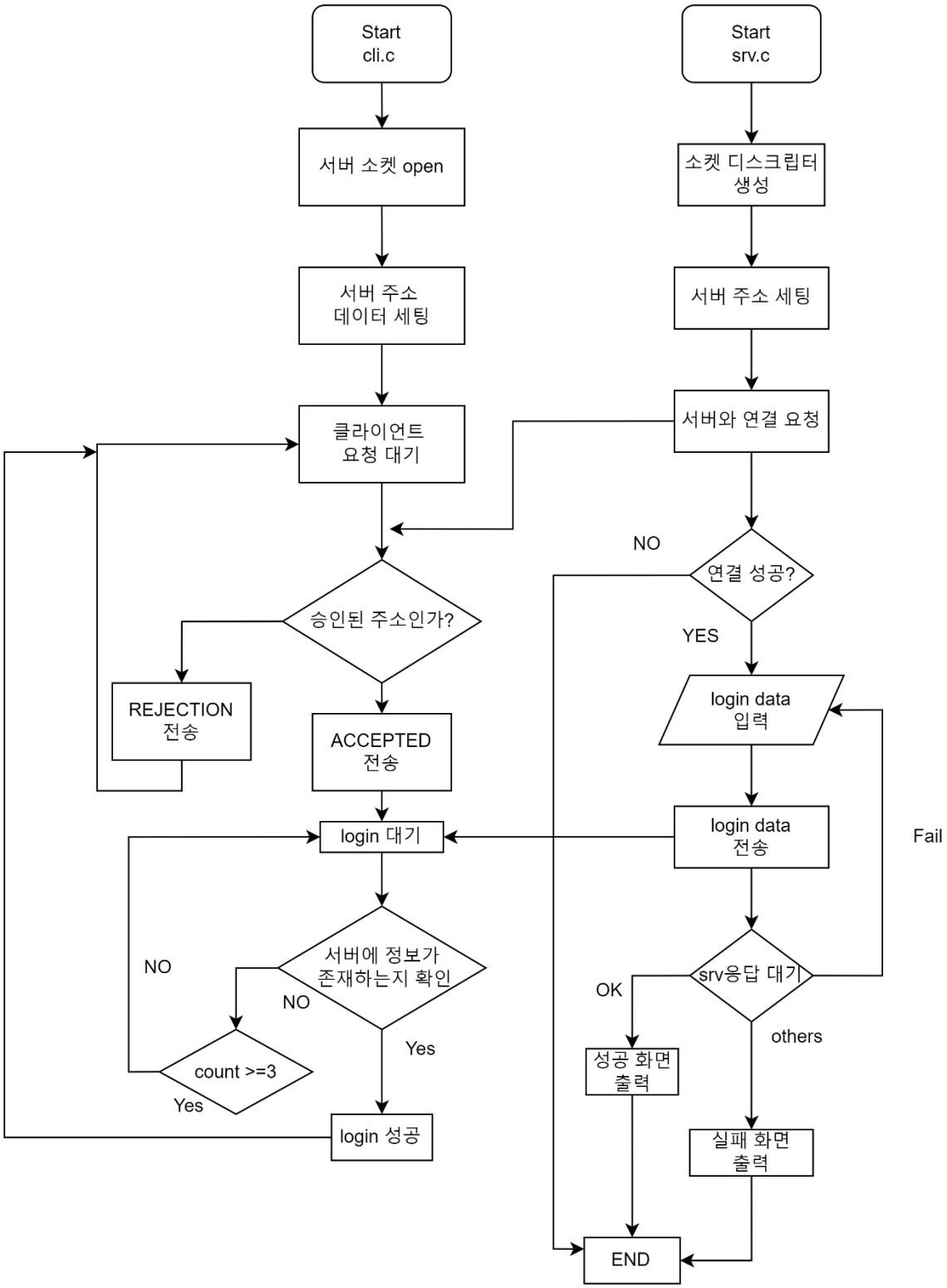
Student ID : **2020202037**

Name : **염정호**

Introduction

이번 과제에서는 로그인 기능을 지원하는 서버와 클라이언트를 구현하는 것을 목표로 한다. 서버를 실행하고 클라이언트가 서버에 연결 요청시 클라이언트의 ip 를 확인하고 access.txt 파일에 정의된 ip 의 경우 로그인 기능을 수행한다. 로그인 가능한 데이터는 사전에 txt 파일로 정의되어 있으며 3 회 이상 로그인에 실패할 경우 해당 클라이언트의 연결을 종료한다. 성공한 경우 성공 메시지를 출력한다.

Flow chart


Start
cli.c
서버 소켓 open
서버 주소
데이터 세팅
클라이언트
요청 대기
승인된 주소인가?
REJECTION
전송
ACCEPTED
전송
login 대기
서버에 정보가
존재하는지 확인
NO
NO
count >=3
Yes
Yes
login 성공
Start
srv.c
소켓 디스크립터
생성
서버 주소 세팅
서버와 연결 요청
NO
연결 성공?
YES
login data
입력
login data
전송
Fail
srv응답 대기
OK
others
성공 화면
출력
실패 화면
출력
END

Pseudo code

cli.c

```
int main(int argc, char *argv[])
{
    int sockfd, n, p_pid;
    struct sockaddr_in servaddr;

    sockfd = socket(소켓 디스크립터 생성


    memset(&servaddr, 0, sizeof(servaddr));          //
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = ip 주소 설정
    servaddr.sin_port = htons(포트 호 설정)


    connect(서버에게 연결 요청

    log_in(sockfd);

    // Close the socket
    close(sockfd);
    return 0;
}

void log_in(int sockfd)
{
    int n;                                          // Variable to store number of bytes read
    char user[MAX_BUF], *passwd, buf[MAX_BUF]; // Buffers for user input and server messages
    memset(buf, 0, MAX_BUF);                        // Zero out the buffer

    // Read initial response from the server
    if (n = read(서버 연결 결과 확인)
    {
        print_data("server closed socket₩n");
    }
    if (연결 실패시)
    {
        print_data("** Connection refused **₩n");
        close(sockfd);
        return;
```

```
}
else if (연결 성공시)
{
    print_data("** It is connected to Server **₩n"); // Connection accepted by the server
}
else
{
    print_data("something wrong option₩n"); // Unexpected response from the server
    return;
}

memset(입력 버퍼 초기화);

for (;;)
{
    memset(user, 0, MAX_BUF);

    print_data("Input ID :");
    read(아이디 입력)
    write(아이디 전송);

    passwd = getpass(비밀번호 입력)

    write(서버로 전송); // Send password to the server


    n = read(sockfd, buf, MAX_BUF);

    buf[n] = '₩0'; // Null-terminate the buffer
    if (로그인 성공시)
    {
        print_data("** User '");          // Print success message
        user[strlen(user) - 1] = '₩0'; // Remove newline character from user input
        print_data(user);
        print_data("' logged in **₩n");
        close(sockfd); // Close the socket
        return;
    }
    else if (로그인 실패시)
    {
        print_data("** Log-in failed **₩n"); // Print failure message
        memset(buf, 0, MAX_BUF);                // Zero out the buffer for next use
    }
```

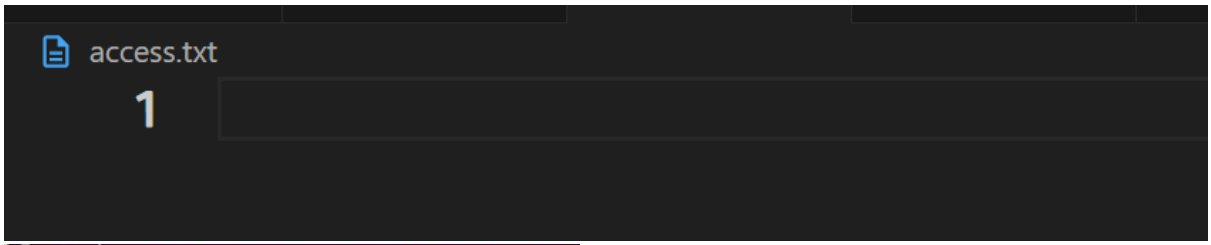
```
        else // 3 번 실패한 경우
        {
            print_data("** Connection closed **₩n"); // Print connection closed message
            return;
        }
    }
}
srv.c
int main(int argc, char *argv[])
{
    int listenfd, connfd;
    struct sockaddr_in servaddr, cliaddr;
    FILE *fp_checkIP; // FILE stream to check client's IP

    char check_ip[20], *client_ip;
    int i = 0, j = 0;
    char stri[10];

    listenfd = socket(소켓 생성); // Get server file descriptor

    // Initialize server address structure
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(주소 설정); // Get server address
    servaddr.sin_port = htons(port 설정);       // Get port number

    // Bind the socket to the server address
    bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

    // Listen for incoming connections
    listen(소켓 연결 대기);
    for (;;)
    {
        int len = sizeof(cliaddr);
        connfd = accept(연결 요청 확인시);

        // Print client connection details
        printf("**Client is trying to connect **₩n");
        printf("- IP:    %s₩n", client_ip);
        printf("Port:    %d₩n", cliaddr.sin_port);

        // Open the access file to check client's IP
        fp_checkIP = fopen(ip 주소 파일 오픈)
```

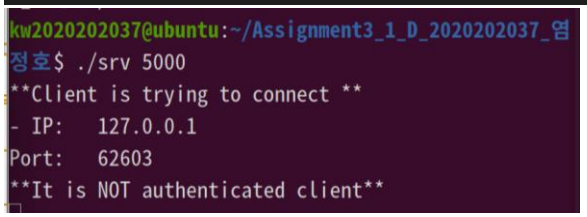
```
        if (fp_checkIP == NULL)
        {
            printf("**file ₩"access.txt₩" doesn't exist**");
            return 0;
        }

        // Check if the client IP matches any entry in the access file
        while (파일 내에 클라이언트 ip 존재 여부 확인) != NULL)
        {

            while ((check_ip[i] != '₩0') && (client_ip[j] != '₩0'))
            {

            if (해당 ip 가 존재하는 경우))
            {
                i = 100;
                break;
            }
        if (존재시)
        {
            write(클라이언트 한테 accepted 전송);
            printf("** Client is connected **₩n");
        }
        else//존재하지 않는 ip
        {
            // If client IP is not matched, reject connection
            write(클라이언트에게 "REJECTION" 전송);
            printf("**It is NOT authenticated client**₩n");
            continue;
        }

        // Authenticate user log-in
        if (log_auth(connfd) ==로그인 성공)
        { // if 3 times fail (ok : 1, fail : 0)
            printf(로그인 실패 메시지 출력)
        }
    }
}

int log_auth(int connfd)
{
    char user[MAX_BUF], passwd[MAX_BUF];
```

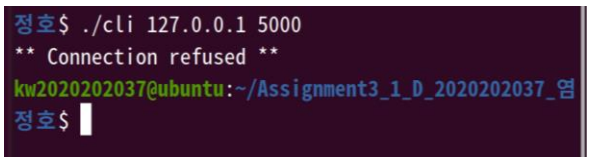
```
    int n, count = 1;
    char count_s[2];
    while (1)
    {

        read(ID 입력받음);
        read(pw 입력);

        // Print log-in attempt information
        print_data("** User is trying to log-in (");

        print_data(로그인 시도 횟수 출력");

        // Match user credentials
        if ((존재하는 유저인 경우)
        {
            write(cli 에게 ok 전송);
            print_data("** Success to log-in **₩n");
            break;
        }
        else if (존재하지 않을 경우)
        {
            print_data("** Log-in failed **₩n");
            if (count >= 3)
            {
                write(클라이언트 종료를 위한 문자열 전송);
                return 0;
            }
            write(cli 에게 fail 전송);
            count++;
            continue;
        }
    }
    return 1;
}

int user_match(char *user, char *passwd)
{
    FILE *fp;
    struct passwd *pw;

    char line[200];
    char *ptr;
```

```
    fp = fopen(로그인 파일 오픈)

    if (fp == NULL)
    {
        perror("Error opening file");
        return 0;
    }

    while ((파일의 끝까지 탐색)
    {

        // Compare the provided user ID and password with the stored values
        if (strcmp(ID 확인) == 0)
        {
            if (password 확인) == 0)
            {
                fclose(fp); // Close the file before returning
                return 1;    // Return 1 if both match
            }
        }
    }
    fclose(fp);
    return 0; // Return 0 if no match is found
}

void print_data(const char *print) // Function to print execution results
{
    int length = strlen(print);
    write(1, print, length);
    return;
}


void print_data(const char *print) // Function to print execution results
{
    int length = strlen(print); // Get the length of the string
    write(1, print, length);     // Write the string to standard output
    return;
}
```

결과화면

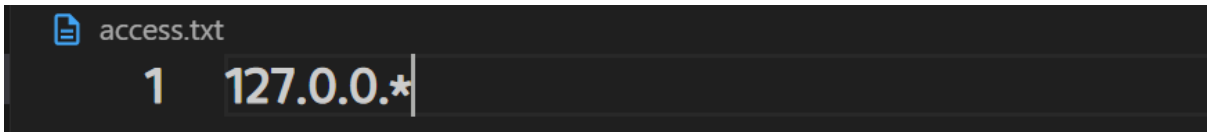


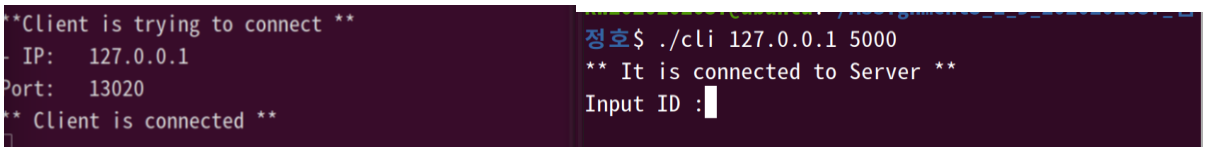




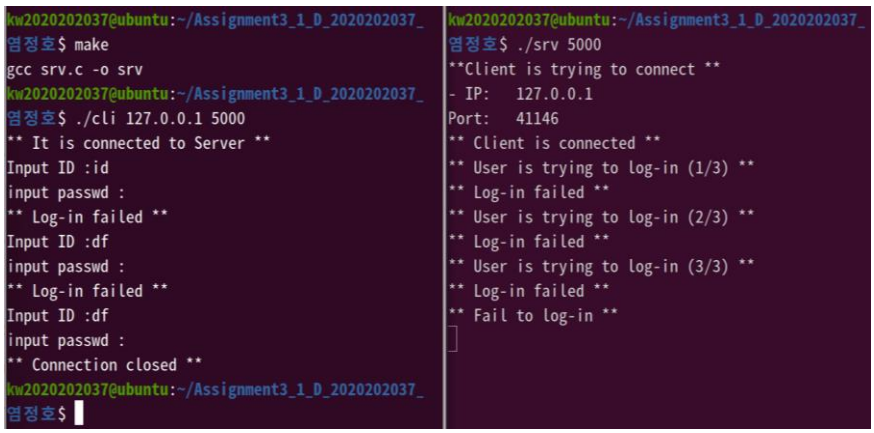
잘못된 ip 입력시

클라이언트 – 서버 연결 성공시

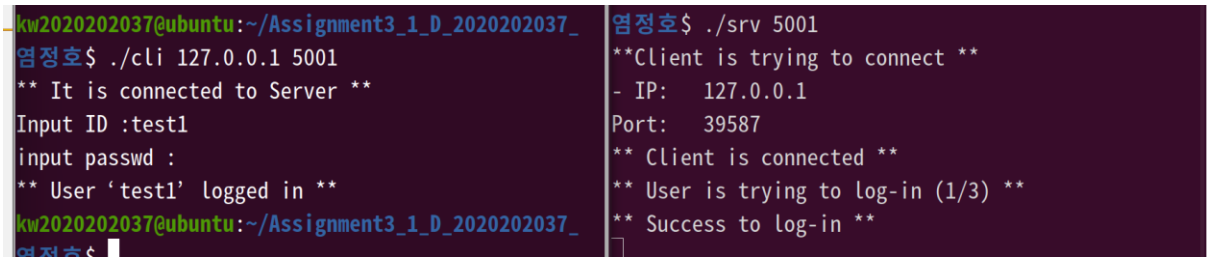




로그인 3 회 실패시



로그인 성공시



고찰

해당 과제를 수행하면서 서버의 ip 주소를 출력하는 과정에서 출력 오류가 발생하였다. 기존에 입력 받은 값을 출력하고 반복문으로 들어가야 했는데 출력 없이 반복문을 수행하며 해당 프로세스를 종료 할 경우 값이 출력되었다. 해당 문제에 대해 찾아보니 wirte 와 printf 의 혼용으로 발생한 문제임을 확인 할 수 있었다. wirte 의 경우 버퍼의 내용을 그대로 출력하며 printf 는 개행문자 또는 널 문자를 만났을 때 내용을 출력한다. 즉 wirte 에서 입력한 내용에 개행 문자 또는 널 문자가 없을 경우/ 혹은 지워졌을 경우 printf 는 동작하지 않는다. 이에 대한 해결 방안으로 printf 사용 후 버퍼를 비워 종료문자 없이 출력을 진행하거나 write 로 출력을 통일하는 방안으로 해결 가능했다.

Reference