

시스템 프로그래밍 실습

[Assignment3-1]

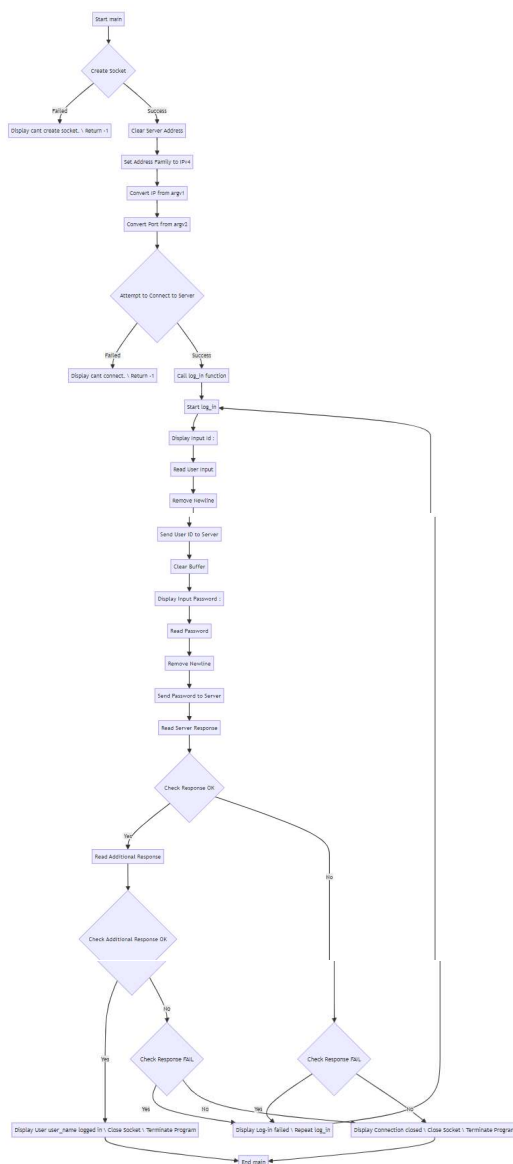
Class : D 반(실습 2 금 56)
Professor : 최상호 교수님
Student ID : 2022202104
Name : 김유찬

Introduction

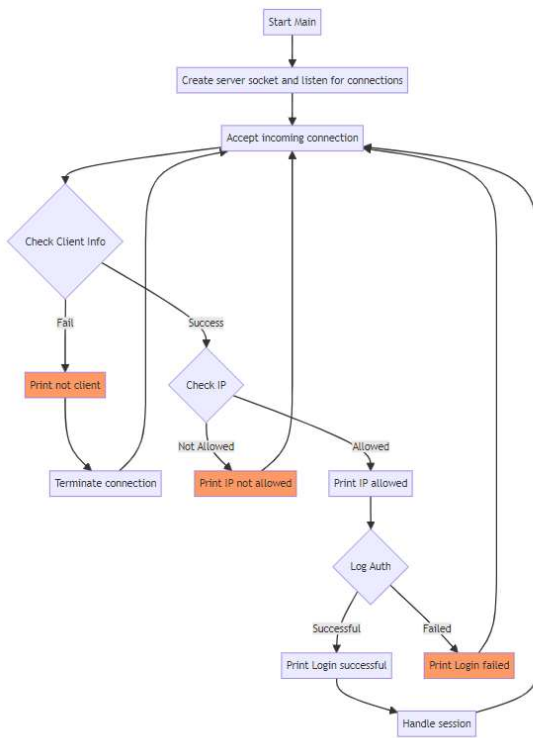
지금까지 어떤 client 의 IP 라고 해도 받을 수 있게 server 에 접속하게 만들었다. 원하는 IP 만 허용되게 만들고 싶을 수 있다. 또한 client 쪽에서 id 와 password 을 입력해서 server 에 있는 passwd 에 저장된 아이디와 비번과 비교해서 있으면 로그인 되고 안 되면 다시 client 한테 id 와 password 을 입력을 받을 수 있게 한다. 즉 client 와 server 간의 Authorization 을 만들어 보는 시간을 갖을 것이다.

Flow chart

1) client



2) server



Pseudo code

1) client

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAX_BUF 1024
#define CONT_PORT 20001

void log_in(int sockfd) {
    repeat {
        clear buffer

        display "Input Id : "
        read user input into buffer
        remove newline character from buffer if present
        send buffer to server

        clear buffer

        display "Input Password : "
        read password into buffer
        remove newline character from buffer if present
        send buffer to server

        read server response into buffer
        if (buffer equals "OK") {
            read additional server response into buffer
            if (buffer equals "OK") {
                display "*** User '[user_name]' logged in ***"
                close socket
                terminate program
            } else if (buffer equals "FAIL") {
                display "*** Log-in failed ***"
            } else {
            }
        }
    }
```

```

        display "*** Connection closed ***"
        close sockfd
        terminate program
    }
} until false
}

int main() {
    sockfd = create_socket(ipv4, stream)
    if (sockfd < 0) {
        display "can't create socket."
        return -1
    }

    clear servaddr

    servaddr.family = ipv4
    servaddr.ip = convert_ip(argv[1])
    servaddr.port = convert_port(argv[2])

    if (connect_socket_to_server_fails) {
        display "can't connect."
        return -1
    }

    call_log_in(sockfd)

    close sockfd
    return 0
}

```

2) server

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <sys/types.h>
4  #include <sys/socket.h>
5  #include <netinet/in.h>
6  #include <unistd.h>
7  #include <arpa/inet.h>
8  #include <stdlib.h>
9  #include <dirent.h>
10 #include <pwd.h>
11 #include <grp.h>
12 #include <unistd.h>
13 #include <sys/stat.h>
14 #include <time.h>
15
16 int main() {
17     create_server_socket_and_listen_for_connections
18     for_each_incoming_connection {
19         if (client_info_fails) {
20             print "not client"
21             terminate_connection
22         }
23
24         if (not_allowed_ip) {
25             print "IP not allowed"
26             continue_to_next_iteration
27         } else {
28             print "IP allowed"
29         }
30
31         if (log_auth_successful) {
32             print "Login successful"
33         } else {
34             print "Login failed"
35             continue_to_next_iteration

```

```

36     }
37 }
38 }
39
40 int client_info(pointer to client address) {
41     print "client trying to connect"
42     if (family is not IPv4) {
43         return failure
44     }
45     print IP and port
46     return success
47 }
48
49 int user_match(username, password) {
50     open and read password file
51     if (file not open) {
52         return failure
53     }
54     for each entry in password file {
55         if (username and password match) {
56             return success
57         }
58     }
59     return failure
60 }
61
62 int log_auth(connection file descriptor) {
63     for up to 3 attempts {
64         read username and password from connection
65         if (user_match is successful) {
66             send "OK"
67             return success
68         } else {
69             send "FAIL"
70             if (3 attempts exceeded) {
71                 send "FAILFAIL"
72                 return failure
73             }
74         }
75     }
76     return success
77 }

```

결과화면

The screenshot shows a development environment with several windows and a terminal. The 'access.txt' window is empty. The 'passwd' window shows a list of users: test1:12:0:SPLab1:/home/1:sh1, test2:34:1:0:SPLab2:/home/2:sh2, and test3:56:2:0:SPLab3:/home/3:sh3. The terminal window shows the output of the server program. It displays the IP address 127.0.0.1 and port 40000, and states that the client is trying to connect but is not authenticated. The terminal also shows the command './srv 40' being executed.

```

C: srv x C: disc x E: access.txt x
E: access.txt
1

E: passwd
1 test1:12:0:SPLab1:/home/1:sh1
2 test2:34:1:0:SPLab2:/home/2:sh2
3 test3:56:2:0:SPLab3:/home/3:sh3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_D_2022202104_김유환$ ./srv 40
001
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 40000
** It is NOT authenticated client **

kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_D_2022202104_김유환$ ./cli 1
27.0.0.1 40001
** Connection refused **
kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_D_2022202104_김유환$

```

- access.txt 에 어떤 ip 도 허용되지 않기 때문에 client 가 server 에 접근하지 못하고 있다.

```
C srv.c C cli.c F access.txt F passwd
1 *.*.*.*
1 test1:12:0:0:SPLab1:/home/1:sh1
2 test2:34:1:0:SPLab2:/home/2:sh2
3 test3:56:2:0:SPLab3:/home/3:sh3

kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$ ./srv
40000
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 44770
** Client is connected **
** User is trying to log-in (1/3) **
** Success to log-in **

kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$ ./cli 127
.0.0.1 40000
** Client is connected **
Input Id : test1
Input Password : 12
** User 'test1' logged in **
kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$
```

- access.txt 에 어떤 ip 도 허용하고 passwd 에 있는 회원정보와 일치한 아이디, 비밀번호를 적었기 때문에 로그인 성공 출력이 나왔다.

```
C srv.c C cli.c F access.txt F passwd
1 *.*.*.*
1 test1:12:0:0:SPLab1:/home/1:sh1
2 test2:34:1:0:SPLab2:/home/2:sh2
3 test3:56:2:0:SPLab3:/home/3:sh3

kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$ ./srv
40001
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 7857
** 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 **

kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$ ./cli 127
.0.0.1 40001
** Client is connected **
Input Id : test
Input Password : 1234
** Log-in failed **
Input Id : test1
Input Password : 34
** Log-in failed **
Input Id : test1
Input Password : 45
** Connection closed **
kw2022202104@ubuntu:~/sslab/system_program_kwn/Assignment3_1_0_2022202104_김유찬$
```

- access.txt 에 어떤 ip 도 허용했지만 passwd 에 있는 회원정보와 일치하지 않는 아이디, 비밀번호를 3 번이나 입력해서 client 쪽에서 server 와 끊겼다.

고찰

Client 쪽에서 server 에 보낼 때 쓰는 buffer 을 항상 초기화 해야 한다는 것을 다시 한번 깨닫게 되었다. 허용된 IP 주소만 server 에 허용하게 만들려고 했을 때 와일드 카드 구현에서 시간이 걸렸다. 하지만 strtok 를 적절히 잘 이용하면 쉽게 해결하는 있는

부분이었다. 이번 시간을 통해 리눅스 passwd 파일을 어떻게 c 언어로 읽어서 Authorization 이 되는 알 수 있는 시간이 되었다.

Reference

시스템프로그래밍 실습 강의자료