

REPORT

project - 1

전자 - 임베디드 전공
2014146042
주성민

먼저 서버의 실행화면입니다.

```
st2014146042@602-d: ~/server
st2014146042@602-d:~/server$ ./server
Server-spcket() sockfd is OK..
Server-bind() is OK...
listen() is OK...

accept() is OK...

=====
User Information
ID: user1, PW: passwd1
=====
Log-in success!! [user1] ***

accept() is OK...

PW error
=====
accept() is OK...

=====
User Information
ID: user2, PW: passwd2
=====
Log-in success!! [user2] ***
```

동작 시나리오는 220.149.128.100에서 서버를 열어주고 220.149.128.101, 220.149.128.102 두 클라이언트에서 접속을 시도하여 로그인하는 상황입니다. 먼저 220.149.128.101 클라이언트가 접속 하여 user1 아이디로 로그인을 시도합니다. 아이디 비밀번호가 일치하여 로그인을 성공하고, 이어서 220.149.128.102 클라이언트가 접속을 시도합니다. 첫 번째 시도는 아이디와 비밀번호가 일치하지 않아 실패하게 되고 두 번째 시도에서 성공하여 로그인을 성공하게 됩니다. 다음은 두 클라이언트의 실행 화면입니다.

```
st2014146042@602-c: ~/server
st2014146042@602-c:~/server$ ./server
Clieit-socket() sockfd is OK..
Clinet-conneft() is OK..

=====
Hello! I'm an Embedded Os Server..
Please, LOG-IN!
=====

ID = user1
PW = passwd1
Log-in success!! [user1] ***

st2014146042@602-c:~/server$
```

-220.149.128.101 클라이언트의 실행화면입니다.

```
st2014146042@602-c: ~/server
st2014146042@602-c:~/server$ ./server
Client-socket() sockfd is OK..
Client-connect() is OK..

=====
Hello! I'm an Embedded Os Server..
Please, LOG-IN!
=====

ID = user2
PW = lala
PW error
st2014146042@602-c:~/server$ ./server
Client-socket() sockfd is OK..
Client-connect() is OK..

=====
Hello! I'm an Embedded Os Server..
Please, LOG-IN!
=====

ID = user2
PW = passwd2
Log-in success!! [user2] ***

st2014146042@602-c:~/server$
```

- 220.149.128.101 클라이언트의 실행화면입니다.

두 클라이언트가 동시에 접속하여 로그인을 대기 하고있는 순간도 구현해보았습니다.

```
st2014146042@602-d: ~/server
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** 시스템 을 다시 시작해 하 십 니 다 ***
Last login: Sat Nov 17 13:35:18 2018 from 114.204.73.147
st2014146042@602-d:~$ ls
foo  lab1  lab2  examples.desktop  file  homework_file.c  mult  server
st2014146042@602-d:~$ cd server
st2014146042@602-d:~/server$ ls
server  server.c
st2014146042@602-d:~/server$ ./server
Server-socket() sockfd is OK..
Server-bind() is OK...
listen() is OK...

accept() is OK...
accept() is OK...

void Init Timer(void)
```

다음은 소스코드 캡처 입니다. 먼저 서버의 소스코드입니다.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<unistd.h>
5 #include<sys/types.h>
6 #include<sys/socket.h>
7 #include<netinet/in.h>
8 #include<arpa/inet.h>
9
10 #define SERV_IP "220.149.128.100"
11 #define SERV_PORT 4370
12 #define BACKLOG 10
13
14 #define INIT_MSG "=====\nHello! I'm an Embedded Os Server..\nPlease, LOG-IN!\n=====\n"
15 #define USER1_ID "user1"
16 #define USER1_PW "passwd1"
17 #define USER2_ID "user2"
18 #define USER2_PW "passwd2"
19
20 int main(void)
21 {
22     int sockfd, new_fd;
23     struct sockaddr_in their_addr;
24     struct sockaddr_in my_addr;
25     unsigned int sin_size;
26     int count = 0;
27     int rcv_byte;
28     char buf[512];
29     char id[20];
30     char pw[20];
31     char msg[512];
32     int val = 1;
33     char msgg[40];
34     int check;
35     sockfd = socket(AF_INET, SOCK_STREAM, 0);
36     if(sockfd == -1){
37         perror("Server-socket() error lol!");
38         exit(1);
39     }
40     else printf("Server-socket() sockfd is OK..\n");
41
42     my_addr.sin_family = AF_INET;
43     my_addr.sin_port = htons(SERV_PORT);
44     my_addr.sin_addr.s_addr = INADDR_ANY;
45     memset(&(amp;my_addr.sin_zero), 0, 8);
46
47     if(setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (char*)&val, sizeof(val)) < 0)
48     {
49         perror("setsockopt");
50         close(sockfd);
51         return -1;
52     }
53
54     if(bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1)
55     {
56         perror("Server-bind() error lol!");
57         exit(1);
58     }
59     else printf("Server-bind() is OK..\n");
60
61     if(listen(sockfd, BACKLOG) == -1)
62     {
63         perror("listen() error lol!");
64         exit(1);
65     }
66     else printf("listen() is OK..\n\n");
67
68     while( 1){
69         sin_size = sizeof(struct sockaddr_in);
70         new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size);
71         count++;
72         pid_t pid;
73         pid = fork();
74         if(pid > 0){
75             close(new_fd);
76         }
77         else if(pid == 0){
78             close(sockfd);
79             printf("accept() is OK..\n\n");
80             send(new_fd, INIT_MSG, strlen(INIT_MSG) + 1, 0);
81             rcv_byte = recv(new_fd, id, sizeof(buf), 0);
82             if((check = strcmp(id, USER1_ID)) == 0)
83             {
84                 //printf("id good\n");
85                 send(new_fd, "PW = ", sizeof(pw), 0);
86                 rcv_byte = recv(new_fd, pw, sizeof(buf), 0);
87                 if((check = strcmp(pw, USER1_PW)) == 0)
88                 {
89                     send(new_fd, "Log-in success!! [user1] ^^^\n", sizeof(msgg), 0);
90                     printf("\n===== \nUser Information\nID: %s, PW: %s\n===== \n", USER1_ID, USER1_PW);
91                     printf("Log-in success!! [%s] ^^^\n", USER1_ID);
92                 }
93                 else
94                 {
95                     send(new_fd, "PW error", sizeof(msgg), 0);
96                     printf("PW error\n===== \n");
97                 }
98             }
99             else if((check = strcmp(id, USER2_ID)) == 0)
100             {
101                 send(new_fd, "PW = ", sizeof(pw), 0);
102                 rcv_byte = recv(new_fd, pw, sizeof(buf), 0);
103                 if((check = strcmp(pw, USER2_PW)) == 0)
104                 {
105                     send(new_fd, "Log-in success!! [user2] ^^^\n", sizeof(msgg), 0);
106                     printf("\n===== \nUser Information\nID: %s, PW: %s\n===== \n", USER2_ID, USER2_PW);
107                     printf("Log-in success!! [%s] ^^^\n", USER2_ID);
108                 }
109                 else
110                 {
111                     send(new_fd, "PW error", sizeof(msgg), 0);
112                     printf("PW error\n===== \n");
113                 }
114             }
115             else
116             {
117                 send(new_fd, "ID error ", sizeof(msgg), 0);
118                 printf("ID error\n===== \n");
119             }
120         }
121         close(new_fd);
122     }
```

메시지를 송수신하기 위해 크기 40
의 배열을 만들어주었습니다.

while(1)로 부모프로세스는 계속해서 클
라이언트의 접속을 기다리게 됩니다.

fork를 사용하여 병행성을 구현합니다.

자식프로세스의 아이디 비밀번호를 확인하는
알고리즘의 시작입니다. 유저가 두명뿐이어서
간단한 비교알고리즘으로 구현하였습니다.

접속을 기다리는 부분부터는 while 과 fork를 사용하여 동시에 여러 클라이언트가 접속 가능하게 구현하였습니다. pid = 0일 경우는 자식 프로세스 pid > 0 일 경우 부모 프로세스로 동작하며, 한 클라이언트가 접속할 때마다 자식 프로세스와 부모 프로세스가 동시에 늘어나게 됩니다. 부모 프로세스에서는 자식 프로세서에서 사용하는 new_fd 소켓을 닫아주고 while문 최상단에서 다음 접속을 대기하게 됩니다. 자식 프로세서에서는 접속한 클라이언트와 new_fd를 통해 통신하여 아이디와 비밀번호를 입력하여 확인하는 알고리즘을 구현하였습니다. 이번 프로젝트에서는 아이디, 비밀번호가 두 명의 사용자 정보밖에 없었지만, 사용자가 많아진다면 비교하는 알고리즘이 너무 길어질 것 같아서 배열을 이용한 비교를 고려해보고 있습니다. 먼저 ID를 입력받고 데이터베이스에 존재하는 ID인지 확인하게 됩니다. 존재하지 않을 시 ID error라고 출력하고 존재하면 Password를 입력하게 됩니다. Password가 일치하지 않다면 Password error, 일치한다면 로그인을 성공하게 됩니다. 로그인 과정이 끝나서 로그인 성공 여부를 new_fd를 통해 전송하게 되고 자식 프로세스도 new_fd를 닫아주게 됩니다. 한 개의 프로세스가 부모와 자식 프로세스 두 개로 나누어지기 때문에 양쪽에서 new_fd를 정상적으로 닫아주어야 실행중인 프로세스가 쌓이지 않고 정상 종료할 수 있습니다. 문자열 비교를 구현하기 위해 strcmp함수를 사용했습니다.(일반적인 등호 비교 시 포인터값을 비교하게 되어 항상 거짓이 됩니다.)

이어서 클라이언트 소스코드입니다.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<unistd.h>
5 #include<sys/types.h>
6 #include<sys/socket.h>
7 #include<netinet/in.h>
8 #include<arpa/inet.h>
9
10 #define SERV_IP "220.149.128.100"
11 #define SERV_PORT 4370
12 int main()
13 {
14     int sockfd;
15     int check;
16     struct sockaddr_in dest_addr;
17
18     int rcv_byte;
19     char buf[512];
20     char id[20];
21     char pw[20];
22     char msg[40];
23     sockfd = socket(AF_INET, SOCK_STREAM, 0);
24     if(sockfd == -1)
25     {
26         perror("Client-socket() error LIL!");
27         exit(1);
28     }
29     else printf("Client-socket() sockfd is OK..\n");
30
31     dest_addr.sin_family = AF_INET;
32     dest_addr.sin_port = htons(SERV_PORT);
33     dest_addr.sin_addr.s_addr = inet_addr(SERV_IP);
34
35     memset(&(dest_addr.sin_zero), 0, 8);
36     if(connect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct sockaddr)) == -1)
37     {
38         perror("Client-connect() error lol");
39         exit(1);
40     }
41     else printf("Client-connect() is OK..\n\n");
```



기존의 소스코드에서 조금만 수정하면 project - 1을 성공적으로 수행 할 수 있었습니다. 기존의 login 메시지를 입력받은 후 클라이언트 실행화면에 아이디를 입력할 수 있게 “ID = ”를 출력하고 id배열에 입력받은 id를 저장합니다. 그 후 생성된 소켓을 통해 id를 전송하게 됩니다. 이어서 서버에서 메시지를 기다리는데, 등록된 사용자 ID라면 Password를 입력하라는 문구를 받게되고, 존재하지 않는 ID라면 “ID error” 문구를 받게됩니다. 비밀번호를 입력하는 문구를 받게 될 시 사용자가 비밀번호를 입력하여 서버에 전송하게 되고 최종적인 로그인 결과를 수신하게 됩니다.

서버와 클라이언트 사이에서 송수신의 크기를 마음대로 하게 되면 버퍼에 내용이 조금씩 남게 되는 현상을 겪어서 송신과 수신을 할 때 버퍼의 크기를 항상 고려하면서 과제를 수행 했습니다.