

REPORT - Final Project #3



2019년 2학기 임베디드운영체제

김수민 교수님

12팀

전자공학부 임베디드시스템전공 2014146012 박동훈

전자공학부 임베디드시스템전공 2017146036 이지현

Project #3는 Project #2에서 구현한 그룹채팅 시스템에 P2P (Peer-to-Peer) 파일전송 기능을 구현하는 것이 목표이다. 특이하게 Client간 대화를 하고 있는 상황에서 Server를 거치지 않고 한 Client가 파일을 요청하여 새로운 Client가 되면 다른 Client가 Server가 되어 요청한 파일을 기존 Server를 거치지 않고 바로 전송하게 된다. 자세히 말하자면, User1이 [FILE]이라는 특수한 메시지를 키보드로 입력해 Server로 전송하면 Server는 요청한 사용자의 P2P IP/포트번호와 함께 [FILE] 메시지를 User2로 보낸다. 메시지를 수신한 User2는 P2P 파일전송 Client가 되어 수신 받은 P2P IP/포트번호로 User1에 접속한다. 연결에 성공하면 User2는 소유한 파일리스트를 User1에 서버를 거치지 않고 전송한다. User1에 파일리스트가 출력되면 원하는 파일을 User2에 요청한다. User2는 요청한 파일을 User1에 전송하고 User1은 받은 파일을 저장함으로써 파일 전송에 성공하게 된다.

Server 코드는 로그인과 채팅 기능만 있으면 되기 때문에 Project #2에서 코드의 변화가 없다. Client가 Server의 역할도 하면서 파일을 전송, 저장해야 했기 때문에 많은 생각이 필요했다.

- Server 코드

```
st2017146036@602-d: ~/project/2019fall/base_code/s-c_model/Proj3
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 #include <errno.h>
10
11 #define SERV_IP "220.149.128.100"
12 #define SERV_PORT 4202
13 #define BACKLOG 10
14
15 #define INIT_MSG "=====\nHello! I'm Chatting Server\nPlease, LOG-IN!\n=====\n"
16 #define USER1_ID "user1"
17 #define USER1_PW "passwd1"
18 #define USER2_ID "user2"
19 #define USER2_PW "passwd2"
20
21 #define MAX SOCK 5
22
23 #define EXIT "[EXIT]"
24
25 int GetMax(int);
26
27 int client_s[MAX SOCK];
28 int num = 0;
29
```

```
30 int main(void)
31 {
32     int sockfd, new_fd;
33     struct sockaddr_in my_addr;
34     struct sockaddr_in their_addr;
35     unsigned sin_size;
36
37     int rcv_byte;
38     int val = 1;
39     char buf[512];
40     char id[20];
41     char pw[20];
42     char msg[512];
43
44     int maxfdp1;
45     int i, j, n;
46
47     fd_set read_fds;
48
49     sockfd = socket(AF_INET, SOCK_STREAM, 0);
50     if (sockfd == -1)
51     {
52         perror("Server-socket() error lol!");
53         exit(1);
54     }
55     else printf("Server-socket() sockfd is OK...\n");
56
57     my_addr.sin_family = AF_INET;
58     my_addr.sin_port = htons(SERV_PORT);
59     my_addr.sin_addr.s_addr = INADDR_ANY;
60     memset(&(my_addr.sin_zero), 0, 8);
61
62     if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) =
= -1)
63     {
64         perror("Server-bind() error lol!");
65         exit(1);
66     }
67     else printf("Server-bind() is OK...\n");
68
69     if (listen(sockfd, BACKLOG) == -1)
70     {
71         perror("listen() error lol!");
72         exit(1);
73     }
74     else printf("listen() is OK...\n\n");
```

```

76     maxfdp1 = sockfd + 1;
77
78     printf("=====\nHello! I'm Chatting Server...\nP
lease, LOG-IN!\n=====");
79
80     while(1)
81     {
82         msg[0] = '\0';
83         buf[0] = '\0';
84
85         FD_ZERO(&read_fds);
86         FD_SET(sockfd, &read_fds);
87         FD_SET(0, &read_fds);
88
89         for(i=0; i<num; i++) FD_SET(client_s[i], &read_fds);
90         maxfdp1 = GetMax(sockfd) + 1;
91
92         if (select(maxfdp1, &read_fds, (fd_set *)0, (fd_set *)0, NULL) == -1
93         )
94         {
95             printf("Server-select() error lol!");
96             exit(1);
97         }
98
99         if (FD_ISSET(sockfd, &read_fds))
100         {
101             sin_size = sizeof(struct sockaddr_in);
102             new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size);
103
104             if (new_fd == -1)
105             {
106                 printf("accept() error lol!");
107                 exit(1);
108             }
109             else printf("accept() is OK...\n\n");
110
111             send(new_fd, INIT_MSG, strlen(INIT_MSG) + 1, 0);
112             rcv_byte = recv(new_fd, id, sizeof(buf), 0);
113
114             if((strcmp(id, USER1_ID)) == 0)
115             {
116                 send(new_fd, "PW : ", sizeof(pw), 0);
117                 rcv_byte = recv(new_fd, pw, sizeof(buf), 0);
118
119                 if ((strcmp(pw, USER1_PW)) == 0)

```

```
119         {
120             send(new_fd, "Log-in success! - Welcome to GTalk\n=====  
===== Chatting Room =====\n(Write [EXIT] if you want  
to exit. / Write [FILE] if you want to download the file.)\n", sizeof(msg),  
0);
121             printf("Log-in success! [user1] - Welcome to GTalk\n=====  
===== Chatting Room =====\n");
122         }
123     else
124     {
125         send(new_fd, "PW error!\n=====\n", sizeof(msg), 0);
126         printf("PW error!\n=====\n");
127     }
128 }
129
130 else if((strcmp(id, USER2_ID)) == 0)
131 {
132     send(new_fd, "PW : ", sizeof(pw), 0);
133     rcv_byte = recv(new_fd, pw, sizeof(buf), 0);
134
135     if ((strcmp(pw, USER2_PW)) == 0)
136     {
137         send(new_fd, "Log-in success! - Welcome to GTalk\n=====  
===== Chatting Room =====\n(Write [EXIT] if you want  
to exit. / Write [FILE] if you want to download the file.)\n", sizeof(msg),  
0);
138         printf("Log-in success! [user2] - Welcome to GTalk\n=====  
===== Chatting Room =====\n");
139     }
140     else
141     {
142         send(new_fd, "PW error!\n=====\n", sizeof(msg), 0);
143         printf("PW error!\n=====\n");
144     }
145 }
146
147 else
148 {
149     send(new_fd, "ID error!\n=====\n", sizeof(msg), 0);
150     printf("ID error!\n=====\n");
151 }
152
153 client_s[num] = new_fd;
```

```
157     getsockname(new_fd, (struct sockaddr *)&their_addr, &sin_size);
158
159     num++;
160 }
161
162 for(i=0; i<num; i++)
163 {
164     if (FD_ISSET(client_s[i], &read_fds))
165     {
166         if ((n = recv(client_s[i], buf, sizeof(buf), 0)) <= 0)
167         {
168             close(client_s[i]);
169
170             if(i != num - 1)
171             {
172                 client_s[i] = client_s[num - 1];
173             }
174
175             num--;
176
177             continue;
178         }
179
180         buf[n] = '\0';
181
182         if (strstr(buf, EXIT) != NULL)
183         {
184             close(client_s[i]);
185
186             if (i != num - 1)
187             {
188                 client_s[i] = client_s[num - 1];
189             }
190
191             num--;
192         }
193
194         for(j = 0; j < num; j++)
195         {
196             if (i != j)
197                 send(client_s[j], buf, n, 0);
198         }
199
200         printf("%s", buf);
201     }
202 }
```



```

203     }
204 }
205
206 int GetMax(int i)
207 {
208     int max = i;
209     int j;
210
211     for(j = 0; j<num; j++)
212     {
213         if(client_s[j] > max) max = client_s[j];
214     }
215
216     return max;
217 }

```

217,1 바 닥

- Client1 코드 (220.149.128.101)

이제 Client끼리 파일을 요청해 전송해야 하므로 [FILE]이라는 특수한 메시지를 입력하면 새로운 소켓을 생성하도록 한다. 그래서 요청한 Client는 새로운 Client가 되고 요청받은 Client는 새로운 Server가 된다.

원래 Server나 새로운 Server가 된 Client로부터 메시지를 전송하고 받는 부분(채팅)에 각각 파일 요청과 전송에 대한 코드를 추가했다. 220.149.128.101에 할당된 Client와 220.149.128.102에 할당된 Client와 코드가 비슷하므로 101에 할당된 Client의 코드를 먼저 보자.

```

st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
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 #include <errno.h>
10
11 #define SERV_IP "220.149.128.100"
12 #define SERV_PORT 4202
13 #define P2P_PORT 4201
14
15 #define EXIT "[EXIT]"
16 #define P2PFILE "[FILE]"
17
18 #define recvfile1 "[user1] : [FILE]\n"
19 #define recvfile2 "[user2] : [FILE]\n"
20
21 #define MAX SOCK 5

```

파일전송을 할 때 새로운 Server가 되는 Client가 존재하므로 파일전송을 위한 새로운 포트

번호 P2P_PORT를 선언해 주었다. 16열에는 P2P 파일 송수신을 위한 P2P 파일전송 서버를 생성하기 위해 특수한 문자 [FILE]을 지정해 주었다. 또한, 파일 요청을 받은 Server가 된 Client가 User2일 때는 18열의 문자를, User1일 때는 19열의 문자를 받으면 파일 리스트를 보낼 수 있도록 정의했다.

st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3

```
23 int main()
24 {
25     int sockfd;
26     struct sockaddr_in dest_addr;
27
28     int check;
29     int rcv_byte;
30     char buf[512];
31     char id[20];
32     char pw[20];
33     char msg[512];
34     char line[512];
35
36     int maxfdpl;
37
38     fd_set read_fds;
39
40     sockfd = socket(AF_INET, SOCK_STREAM, 0);
41     if (sockfd == -1)
42     {
43         perror("Client-socket() error lol! \n");
44         exit(1);
45     }
46     else printf("Client-socket() sockfd is OK...\n");
47
48     dest_addr.sin_family = AF_INET;
49     dest_addr.sin_port = htons(SERV_PORT);
50     dest_addr.sin_addr.s_addr = inet_addr(SERV_IP);
51     memset(&(dest_addr.sin_zero), 0, 8);
52
53     if(connect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct sockaddr
54 )) == -1)
55     {
56         perror("Client-connect() error lol! \n");
57         exit(1);
58     }
59     else printf("Client-connect() is OK...\n\n");
60
61     rcv_byte = recv(sockfd, buf, sizeof(buf), 0);
62     printf("%s\n", buf);
63
64     printf("ID : ");
65     scanf("%s", id);
66     buf[0] = '\0';
67     while(getchar() != '\n');
68     send(sockfd, id, strlen(id)+1, 0);
```

68,0-1

8%


```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
69     rcv_byte = recv(sockfd, buf, sizeof(buf), 0);
70     if((check = strcmp(buf, "PW : ")) == 0)
71     {
72         printf("%s", buf);
73         buf[0] = '\0';
74         scanf("%[^\\n]", pw);
75         while(getchar() != '\\n');
76         send(sockfd, pw, 20, 0);
77
78         rcv_byte = recv(sockfd, msg, sizeof(buf), 0);
79         printf("%s\\n", msg);
80     }
81
82     else
83     {
84         printf("%s\\n", buf);
85     }
86
87     maxfdp1 = sockfd + 1;
88
89     while (1)
90     {
91         FD_ZERO(&read_fds);
92         FD_SET(0, &read_fds);
93         FD_SET(sockfd, &read_fds);
94
95         if (select(maxfdp1, &read_fds, (fd_set *)0, (fd_set *)0, (struct tim
eval *)0) < 0)
96         {
97             printf("Select error lol! \\n");
98             exit(0);
99         }
100
101         if (FD_ISSET(0, &read_fds))
102         {
103             if(fgets(msg, sizeof(buf), stdin))
104             {
105                 sprintf(line, "[%s] : %s", id, msg);
106
107                 if (send(sockfd, line, strlen(line), 0) == -1)
108                 {
109                     perror("Error : Write error on socket. \\n");
110                     exit(1);
111                 }
112
113                 if (strstr(msg, EXIT) != NULL)
114                 {
```

```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
115             printf("Good bye! See you next time!\\n");
116             close(sockfd);
117             exit(1);
118         }
```

여기까진 Project #2의 코드와 동일하다.

```

st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
120         if (strstr(msg, P2PFILE) != NULL)
121         {
122             sleep(1);
123
124             int PP_newfd;
125             struct sockaddr_in sock_addr;
126             FILE *fp2;
127
128             char buffer[512];
129             char select1[10];

```

이제 파일을 요청하는 부분이다.

[FILE]로 선언된 P2PFILE과 키보드로부터 입력한 문자열을 비교한다. NULL이 아니면 파일을 요청하는 Client가 새로운 Client가 되므로 새로운 Client 소켓을 생성하고 새로운 버퍼와 파일 리스트의 숫자를 선택하는 크기 10의 배열을 선언해 주었다. 이 배열은 select 함수와 헷갈릴 수 있기 때문에 1을 붙여 새롭게 지정했다. 122열은 파일 요청을 받아 Server가 되는 Client가 [FILE] 문자열을 받은 후 소켓을 생성하는 시간을 기다려 주기 위해 1초 정도 쉬게 하였다.

```

131         PP_newfd = socket(PF_INET, SOCK_STREAM, 0);
132         if (PP_newfd == -1)
133         {
134             perror("P2P-socket() error lol!");
135             exit(1);
136         }
137
138         memset(&sock_addr, 0, sizeof(sock_addr));
139         sock_addr.sin_family = AF_INET;
140         sock_addr.sin_addr.s_addr = inet_addr("220.149.128.102")
;
141         sock_addr.sin_port = htons(P2P_PORT);
142
143         if (connect(PP_newfd, (struct sockaddr*)&sock_addr, size
of(sock_addr)) == -1)
144         {
145             perror("P2P-connect() error lol!");
146             exit(1);

```

이전과 같이 새로운 소켓이 잘 생성되었는지 잘 연결되었는지 확인하는 과정이다.

이제 파일 전송 시 Server의 IP와 포트번호가 아닌 새로운 Server가 된 Client와 통신해야 하므로 새로운 Server의 IP 주소와 파일 전송을 위한 새로운 포트 번호를 적어 주었다.

```

149         int buf3;
150
151         printf("1.a2.txt\n2.b2.txt\n3.c2.txt\n4.d2.txt\n");
152         printf("Select file number. :");
153         scanf("%s", select1);
154
155         send(PP_newfd, select1, strlen(select1) + 1, 0);
156
157         buf3 = recv(PP_newfd, buffer, strlen(buffer) + 1, 0);
158         buffer[buf3] = '\0';

```

buffer에 새로운 Server로부터 받은 내용을 저장하고 그 값을 다시 대입해 줄 새로운 버퍼 변수 buf3을 선언했다. 나중에 0으로 초기화하여 버퍼의 중복을 피하기 위한 용도이다. [FILE] 메시지를 입력 후 파일 리스트를 화면에 띄운다. 그 후 원하는 파일의 숫자를 scanf로 입력한다. 이 숫자는 select1 배열에 해당하는 숫자이다. select1에 저장된 숫자를 새로운 Server로 보낸다. Server로부터 받은 내용을 저장한다.

```

st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
160         if ((strcmp(select1, "1") == 0)) {
161             fp2 = fopen("a2.txt", "a");
162         }
163
164         else if ((strcmp(select1, "2") == 0)) {
165             fp2 = fopen("b2.txt", "a");
166         }
167
168         else if ((strcmp(select1, "3") == 0)) {
169             fp2 = fopen("c2.txt", "a");
170         }
171
172         else if ((strcmp(select1, "4") == 0)) {
173             fp2 = fopen("d2.txt", "a");
174         }
175
176         else
177         {
178             printf("Wrong number!");
179             exit(1);
180         }
181
182         fprintf(fp2, "%s", buffer);
183         printf("Download complete!\n");
184
185         fclose(fp2);
186         close(PP_newfd);
187     }
188 }
189 }

```

string compare 함수를 통해 select1에 저장된 숫자가 동일한 부분을 찾는다. 같다면 0을 출력할 것이다. 각자 맞는 숫자를 비교하여 0이 출력되면 그 숫자에 맞는 파일을 연다. 만약 1,2,3,4 이외의 숫자를 입력하면 잘못된 숫자라는 메시지를 띄운 후 프로그램을 종료한다. 182열의 fprintf 함수를 통해 새로운 Server가 보낸 파일의 내용을 fp2에

저장한다. 저장한 후 다운로드가 완료되었다는 메시지를 출력시킨다.
그 다음 파일과 Client의 소켓을 닫아준다.

```
191     if (FD_ISSET(sockfd, &read_fds))
192     {
193         int buf1;
194
195         buf1 = recv(sockfd, msg, sizeof(buf), 0);
196         msg[buf1] = '\0';
197         printf("%s", msg);
```

Project #2에서와 똑같이 Server로부터 온 메시지를 받는 부분이다. 즉 한 Client가 채팅 메시지를 전송하면 Server로 거쳐 다른 Client로 전송되는 것이다.

```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
199     if ((strstr(msg, recvfile1) != NULL) || (strstr(msg, recvfile2)
    != NULL))
200     {
201         int P_sockfd;
202         int P_newfd;
203         int P_size;
204
205         struct sockaddr_in sock_addr;
206         struct sockaddr_in new_addr;
207
208         FILE *fp3;
209
210         char text[512];
211         char buffer2[512];
212         char select2[10];
```

이제 [FILE]이라는 파일 요청 문자열을 받은 Client는 새로운 Server가 된다. strstr 함수를 통해 Client로부터 받은 입력이 “[user1] : [FILE]\n”, 또는 “[user2] : [FILE]\n”인지 확인하여 아닐 경우 NULL을 반환한다.

파일 요청을 받는 Client라 Server가 되므로 새로운 Server 소켓과 Client 소켓을 정의했다. 그 외로 새로운 버퍼 배열 변수와 파일 리스트 숫자를 선택할 배열 변수, 파일 구조체 fp3에 담긴 파일 내용을 열 배열 변수 text를 선언해주었다.

```

st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
214     P_sockfd = socket(PF_INET, SOCK_STREAM, 0);
215     if (P_sockfd == -1)
216     {
217         perror("P2P-socket() error lol!");
218         exit(1);
219     }
220
221     memset(&sock_addr, 0, sizeof(sock_addr));
222     sock_addr.sin_family = AF_INET;
223     sock_addr.sin_addr.s_addr = htonl(INADDR_ANY);
224     sock_addr.sin_port = htons(P2P_PORT);
225
226     if (bind(P_sockfd, (struct sockaddr*)&sock_addr, sizeof(sock
227 _addr)) == -1)
228     {
229         perror("P2P-bind() error lol!");
230         exit(1);
231     }
232
233     if (listen(P_sockfd, 10) == -1)
234     {
235         perror("P2P-listen error lol!");
236         exit(1);
237     }
238
239     memset(buffer2, 0, 512);
240     P_size = sizeof(new_addr);
241     P_newfd = accept(P_sockfd, (struct sockaddr*)&new_addr, &P_s
242 ize);
243
244     if(P_newfd == -1)
245     {
246         perror("accept() error lol!");
247         exit(1);
248     }

```

Server에서 소켓을 생성하는 과정과 똑같다. 소켓을 생성하고 bind, listen 등을 체크해준다.

```

248     int buf2;
249     buf2 = recv(P_newfd, select2, strlen(select2) + 1, 0);
250     select2[buf2] = '\0';
251
252     printf("Received number: %s\n", select2);
253     memset(buffer2, 0, 512);

```

select2를 받을 변수를 선언해 주었고, 새로운 Client가 보낸 파일 리스트에서 선택한 숫자가 무엇인지 변수 select2에 저장해주었다. 받은 숫자가 무엇인지 알려주도록 출력한다.


```

255         if ((strcmp(select2, "1") == 0)) {
256             fp3 = fopen("a1.txt", "rt");
257         }
258
259         else if ((strcmp(select2, "2") == 0)) {
260             fp3 = fopen("b1.txt", "rt");
261         }
262
263         else if ((strcmp(select2, "3") == 0)) {
264             fp3 = fopen("c1.txt", "rt");
265         }
266
267         else if ((strcmp(select2, "4") == 0)) {
268             fp3 = fopen("d1.txt", "rt");
269         }
270
271         else
272         {
273             printf("Wrong number!");
274             exit(1);
275         }
276
277         if (fp3 == NULL)
278         {
279             perror("file-open() error lol!");
280             exit(1);
281         }
282
283         fgets(text, sizeof(text), fp3);
284         puts(text);
285         strcpy(buffer2, text);
286         send(P_newfd, buffer2, strlen(buffer2) + 1, 0);
287
288         fclose(fp3);
289
290         close(P_newfd);
291         close(P_sockfd);
292     }
293 }
294 }
295 }

```

249, 1-4

바 닥

위의 새로운 Client와 마찬가지로 string compare 함수를 통해 숫자가 몇 번인지 구분하고 각 숫자에 맞는 파일을 오픈한다. 만일 1~4 외에 다른 문자가 입력되면 잘못 입력했다는 말과 함께 프로그램을 종료한다.

보낸 파일을 열 수 있는 기능을 넣었는데, 277열에서 파일을 열 때 오류가 생긴 경우 프로그램을 종료하는 조건문이다.

fgets 파일 함수를 통해 파일 구조체 fp3에 있는 파일 내용을 가져오고 puts 함수를 통해 화면에 출력한다. 그런 다음 buffer2 변수에 파일 내용을 복사한 후 Client로 전송한다. 전송이 완료되었으므로 Client와 Server 소켓을 닫아준 후 Client와 Server였던 Client 모두 다시 채팅으로 복귀한다.

- Client2 코드 (220.149.128.102)

Client1과 파일 리스트나 통신할 P2P IP 주소는 다르지만 코드는 동일하다.

st2017146036@602-a: ~/project/2019fall/base_code/s-c_model/Proj3

```
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 #include <errno.h>
10
11 #define SERV_IP "220.149.128.100"
12 #define SERV_PORT 4202
13 #define P2P_PORT 4201
14
15 #define EXIT "[EXIT]"
16 #define P2PFILE "[FILE]"
17
18 #define rcvfile1 "[user1] : [FILE]\n"
19 #define rcvfile2 "[user2] : [FILE]\n"
20
21 #define MAX_SOCKET 5
22
23 int main()
24 {
25     int sockfd;
26     struct sockaddr_in dest_addr;
27
28     int check;
29     int rcv_byte;
30     char buf[512];
31     char id[20];
32     char pw[20];
33     char msg[512];
34     char line[512];
35
36     int maxfdp1;
37
38     fd_set read_fds;
39
40     sockfd = socket(AF_INET, SOCK_STREAM, 0);
41     if (sockfd == -1)
42     {
43         perror("Client-socket() error lol! \n");
44         exit(1);
45     }
46     else printf("Client-socket() sockfd is OK...\n");
```

```
48     dest_addr.sin_family = AF_INET;
49     dest_addr.sin_port = htons(SERV_PORT);
50     dest_addr.sin_addr.s_addr = inet_addr(SERV_IP);
51     memset(&(dest_addr.sin_zero), 0, 8);
52
53     if(connect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct sockaddr
54 )) == -1)
55     {
56         perror("Client-connect() error lol! \n");
57         exit(1);
58     }
59     else printf("Client-connect() is OK...\n\n");
60
61     rcv_byte = recv(sockfd, buf, sizeof(buf), 0);
62     printf("%s\n", buf);
63
64     printf("ID : ");
65     scanf("%s", id);
66     buf[0] = '\0';
67     while(getchar() != '\n');
68     send(sockfd, id, strlen(id)+1, 0);
69
70     rcv_byte = recv(sockfd, buf, sizeof(buf), 0);
71     if((check = strcmp(buf, "PW : ")) == 0)
72     {
73         printf("%s", buf);
74         buf[0] = '\0';
75         scanf("%s", pw);
76         while(getchar() != '\n');
77         send(sockfd, pw, 20, 0);
78
79         rcv_byte = recv(sockfd, msg, sizeof(buf), 0);
80         printf("%s\n", msg);
81     }
82     else
83     {
84         printf("%s\n", buf);
85     }
86
87     maxfdp1 = sockfd + 1;
88
89     while (1)
90     {
91         FD_ZERO(&read_fds);
92         FD_SET(0, &read_fds);
93         FD_SET(sockfd, &read_fds);
```

st2017146036@602-a: ~/project/2019fall/base_code/s-c_model/Proj3

```
95     if (select(maxfdp1, &read_fds, (fd_set *)0, (fd_set *)0, (struct tim
eval *)0) < 0)
96     {
97         printf("Select error lol! \n");
98         exit(0);
99     }
100
101     if (FD_ISSET(0, &read_fds))
102     {
103         if(fgets(msg, sizeof(buf), stdin))
104         {
105             sprintf(line, "[%s] : %s", id, msg);
106
107             if (send(sockfd, line, strlen(line), 0) == -1)
108             {
109                 perror("Error : Write error on socket. \n");
110                 exit(1);
111             }
112
113             if (strstr(msg, EXIT) != NULL)
114             {
115                 printf("Good bye! See you next time!\n");
116                 close(sockfd);
117                 exit(1);
118             }
119
120             if (strstr(msg, P2PFILE) != NULL)
121             {
122                 sleep(1);
123
124                 int PP_newfd;
125                 struct sockaddr_in sock_addr;
126                 FILE *fp2;
127
128                 char buffer[512];
129                 char select1[10];
130
131                 PP_newfd = socket(PF_INET, SOCK_STREAM, 0);
132                 if (PP_newfd == -1)
133                 {
134                     perror("P2P-socket() error lol!");
135                     exit(1);
136                 }
137
138                 memset(&sock_addr, 0, sizeof(sock_addr));
139                 sock_addr.sin_family = AF_INET;
```

```
140         sock_addr.sin_addr.s_addr = inet_addr("220.149.128.101");
141     ;
142         sock_addr.sin_port = htons(P2P_PORT);
143     if (connect(PP_newfd, (struct sockaddr*)&sock_addr, sizeof(sock_addr)) == -1)
144     {
145         perror("P2P-connect() error lol!");
146         exit(1);
147     }
148
149     int buf3;
150
151     printf("1.a1.txt\n2.b1.txt\n3.c1.txt\n4.d1.txt\n");
152     printf("Select file number. :");
153     scanf("%s", select1);
154
155     send(PP_newfd, select1, strlen(select1) + 1, 0);
156
157     buf3 = recv(PP_newfd, buffer, strlen(buffer) + 1, 0);
158     buffer[buf3] = '\0';
159
160     if ((strcmp(select1, "1") == 0)) {
161         fp2 = fopen("a1.txt", "a");
162     }
163
164     else if ((strcmp(select1, "2") == 0)) {
165         fp2 = fopen("b1.txt", "a");
166     }
167
168     else if ((strcmp(select1, "3") == 0)) {
169         fp2 = fopen("c1.txt", "a");
170     }
171
172     else if ((strcmp(select1, "4") == 0)) {
173         fp2 = fopen("d1.txt", "a");
174     }
175
176     else
177     {
178         printf("Wrong number!");
179         exit(1);
180     }
181
182     fprintf(fp2, "%s", buffer);
183     printf("Download complete!\n");
184
```

```
185         fclose(fp2);
186         close(PP_newfd);
187     }
188 }
189 }
190
191 if (FD_ISSET(sockfd, &read_fds))
192 {
193     int buf1;
194
195     buf1 = recv(sockfd, msg, sizeof(buf), 0);
196     msg[buf1] = '\0';
197     printf("%s", msg);
198
199     if ((strstr(msg, recvfile1) != NULL) || (strstr(msg, recvfile2)
200 != NULL))
201     {
202         int P_sockfd;
203         int P_newfd;
204         int P_size;
205
206         struct sockaddr_in sock_addr;
207         struct sockaddr_in new_addr;
208
209         FILE *fp3;
210
211         char text[512];
212         char buffer2[512];
213         char select2[10];
214
215         P_sockfd = socket(PF_INET, SOCK_STREAM, 0);
216         if (P_sockfd == -1)
217         {
218             perror("P2P-socket() error lol!");
219             exit(1);
220         }
221
222         memset(&sock_addr, 0, sizeof(sock_addr));
223         sock_addr.sin_family = AF_INET;
224         sock_addr.sin_addr.s_addr = htonl(INADDR_ANY);
225         sock_addr.sin_port = htons(P2P_PORT);
226
227         if (bind(P_sockfd, (struct sockaddr*)&sock_addr, sizeof(sock
228 _addr)) == -1)
229         {
230             perror("P2P-bind() error lol!");
231             exit(1);
232         }
233     }
234 }
```



```
230     }
231
232     if (listen(P_sockfd, 10) == -1)
233     {
234         perror("P2P-listen error lol!");
235         exit(1);
236     }
237
238     memset(buffer2, 0, 512);
239     P_size = sizeof(new_addr);
240     P_newfd = accept(P_sockfd, (struct sockaddr*)&new_addr, &P_s
ize);
241
242     if(P_newfd == -1)
243     {
244         perror("accept() error lol!");
245         exit(1);
246     }
247
248     int buf2;
249     buf2 = recv(P_newfd, select2, strlen(select2) + 1, 0);
250     select2[buf2] = '\0';
251
252     printf("Received number: %s\n", select2);
253     memset(buffer2, 0, 512);
254
255     if ((strcmp(select2, "1") == 0)) {
256         fp3 = fopen("a2.txt", "rt");
257     }
258
259     else if ((strcmp(select2, "2") == 0)) {
260         fp3 = fopen("b2.txt", "rt");
261     }
262
263     else if ((strcmp(select2, "3") == 0)) {
264         fp3 = fopen("c2.txt", "rt");
265     }
266
267     else if ((strcmp(select2, "4") == 0)) {
268         fp3 = fopen("d2.txt", "rt");
269     }
270
271     else
272     {
273         printf("Wrong number!");
274         exit(1);
275     }
```



```

283         fgets(text, sizeof(text), fp3);
284         puts(text);
285         strcpy(buffer2, text);
286         send(P_newfd, buffer2, strlen(buffer2) + 1, 0);
287
288         fclose(fp3);
289
290         close(P_newfd);
291         close(P_sockfd);
292     }
293 }
294 }
295 █

```

295,1 바 닷

- 실행화면

로그인과 채팅 기능은 Project #2에서 자세히 설명했으므로 전체적인 화면을 캡처하였다.

1. 로그인 화면

- Server 화면

```

st2017146036@602-d:~/project/2019fall/base_code/s-c_model/Proj3$ ./server_login
Server-socket() sockfd is OK...
Server-bind() is OK...
listen() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====
accept() is OK...

Log-in success! [user1] - Welcome to GTalk
===== Chatting Room =====
accept() is OK...

Log-in success! [user2] - Welcome to GTalk
===== Chatting Room =====
█

```

User1과 User2가 로그인에 성공했고 채팅 방을 나타낸 화면이다.

- User1 화면

```
st2017146036@602-c:~/project/2019fall/base_code/s-c_model/Proj3$ ./user_login
Client-socket() sockfd is OK...
Client-connect() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====

ID : user1
PW : passwd1
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)
```

User1이 로그인에 성공한 후 채팅 방에 입장했다.

- User2 화면

```
st2017146036@602-a:~/project/2019fall/base_code/s-c_model/Proj3$ ./user_login
Client-socket() sockfd is OK...
Client-connect() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====

ID : user2
PW : passwd2
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)
```

User2도 로그인에 성공한 후 채팅 방에 입장했다.

2. 채팅 화면

- User1 화면

```
st2017146036@602-c:~/project/2019fall/base_code/s-c_model/Proj3$ ./user_login
Client-socket() sockfd is OK...
Client-connect() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====

ID : user1
PW : passwd1
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)

hello! I'm user1..
[user2] : Hi! Nice to meet you!
What's your name?
[user2] : My name is user2!
I want to download your file!
[user2] : Ok^^
```

- User2 화면

```
st2017146036@602-a:~/project/2019fall/base_code/s-c_model/Proj3$ ./user_login
Client-socket() sockfd is OK...
Client-connect() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====

ID : user2
PW : passwd2
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)

[user1] : hello! I'm user1..
Hi! Nice to meet you!
[user1] : What's your name?
My name is user2!
[user1] : I want to download your file!
Ok^^
```

- Server 화면

```
st2017146036@602-d: ~/project/2019fall/base_code/s-c_model/Proj3
st2017146036@602-d:~/project/2019fall/base_code/s-c_model/Proj3$ ./server_login
Server-socket() sockfd is OK...
Server-bind() is OK...
listen() is OK...

=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====
accept() is OK...

Log-in success! [user1] - Welcome to GTalk
===== Chatting Room =====
accept() is OK...

Log-in success! [user2] - Welcome to GTalk
===== Chatting Room =====
[user1] : hello! I'm user1..
[user2] : Hi! Nice to meet you!
[user1] : What's your name?
[user2] : My name is user2!
[user1] : I want to download your file!
[user2] : Ok^^
```

3. 파일 전송 화면

(1) User1의 파일 요청

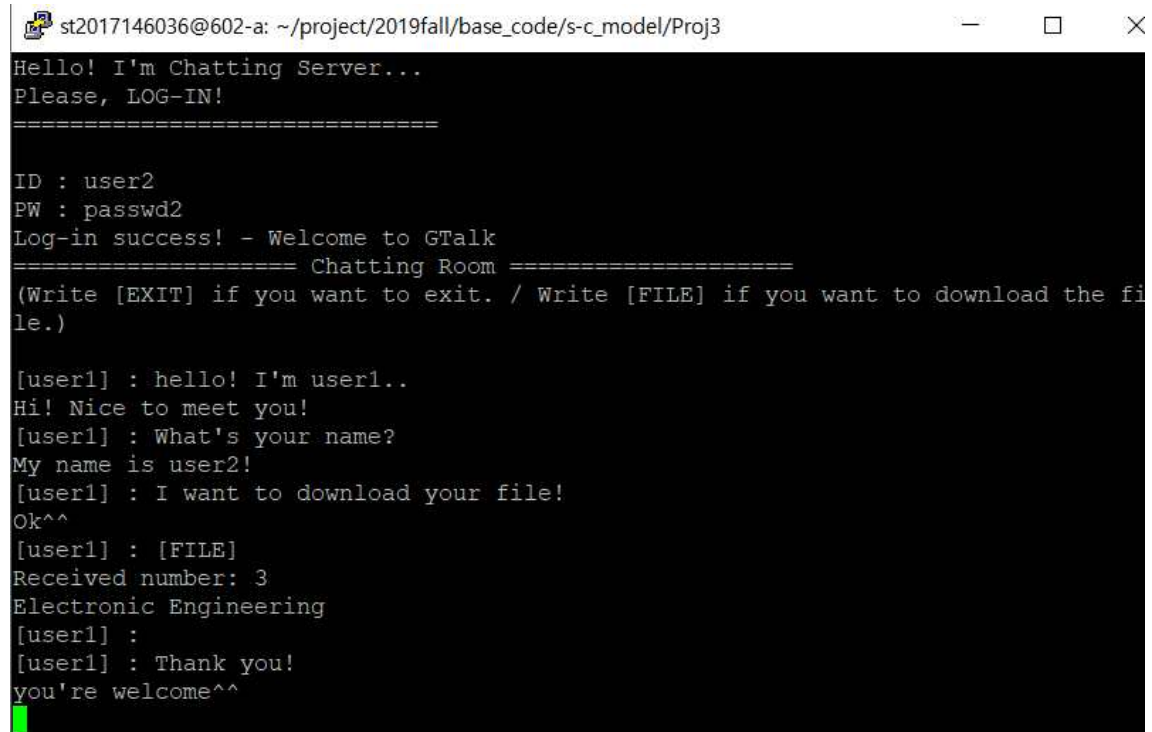
- User1 화면

```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
ID : user1
PW : passwd1
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)

hello! I'm user1..
[user2] : Hi! Nice to meet you!
What's your name?
[user2] : My name is user2!
I want to download your file!
[user2] : Ok^^
[FILE]
1.a2.txt
2.b2.txt
3.c2.txt
4.d2.txt
Select file number. :3
Download complete!
Thank you!
[user2] : you're welcome^^
```

[FILE]이라는 특수한 메시지를 입력하면 새로운 Server가 된 User2에서 새로운 Client인 User1으로 파일 리스트를 보내 User1에 출력이 된다. 원하는 파일 번호를 선택하면 다운이 완료되었다는 문구가 뜨고 다시 채팅으로 복귀한다.

- User2 화면

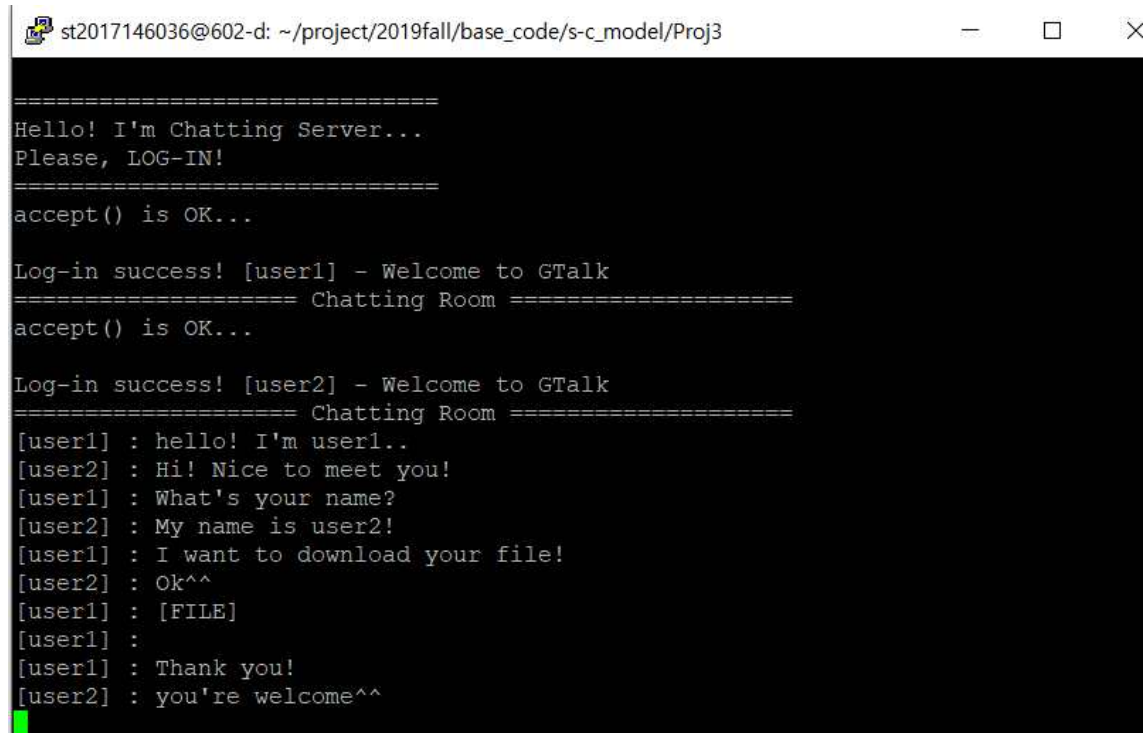


```
st2017146036@602-a: ~/project/2019fall/base_code/s-c_model/Proj3
Hello! I'm Chatting Server...
Please, LOG-IN!
=====
ID : user2
PW : passwd2
Log-in success! - Welcome to GTalk
===== Chatting Room =====
(Write [EXIT] if you want to exit. / Write [FILE] if you want to download the file.)

[user1] : hello! I'm user1..
Hi! Nice to meet you!
[user1] : What's your name?
My name is user2!
[user1] : I want to download your file!
Ok^^
[user1] : [FILE]
Received number: 3
Electronic Engineering
[user1] :
[user1] : Thank you!
you're welcome^^
```

User1으로부터 특수한 메시지를 받은 User2는 새로운 Server가 된다. 그리고 User1이 파일 번호를 선택하여 번호를 전송하면 User2에 출력된다. 그 후 보낸 파일의 내용이 출력된다. 다시 채팅으로 복귀한다.

- Server 화면



```
st2017146036@602-d: ~/project/2019fall/base_code/s-c_model/Proj3
=====
Hello! I'm Chatting Server...
Please, LOG-IN!
=====
accept() is OK...

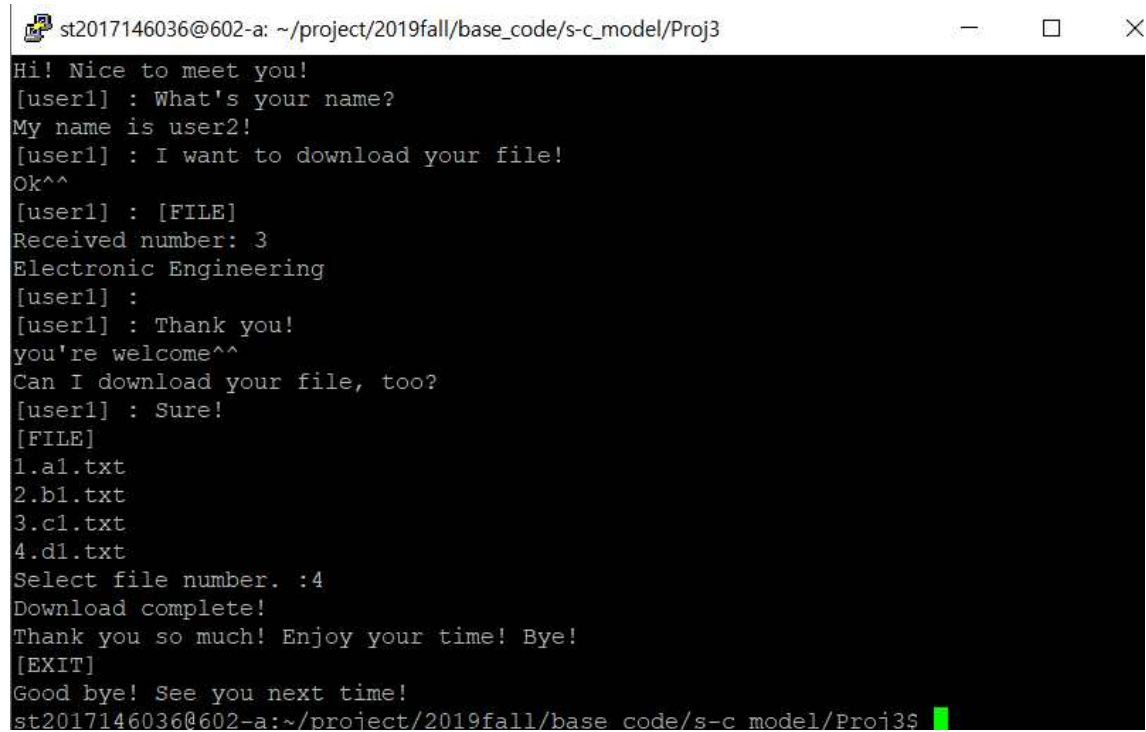
Log-in success! [user1] - Welcome to GTalk
===== Chatting Room =====
accept() is OK...

Log-in success! [user2] - Welcome to GTalk
===== Chatting Room =====
[user1] : hello! I'm user1..
[user2] : Hi! Nice to meet you!
[user1] : What's your name?
[user2] : My name is user2!
[user1] : I want to download your file!
[user2] : Ok^^
[user1] : [FILE]
[user1] :
[user1] : Thank you!
[user2] : you're welcome^^
```

Server 화면은 파일 리스트의 출력이나 파일 내용 출력이 아닌 그저 User1과 User2의 대화 내용만 보여준다.

(2) User2의 파일 요청

- User2 화면



```
st2017146036@602-a: ~/project/2019fall/base_code/s-c_model/Proj3
Hi! Nice to meet you!
[user1] : What's your name?
My name is user2!
[user1] : I want to download your file!
Ok^^
[user1] : [FILE]
Received number: 3
Electronic Engineering
[user1] :
[user1] : Thank you!
you're welcome^^
Can I download your file, too?
[user1] : Sure!
[FILE]
1.a1.txt
2.b1.txt
3.c1.txt
4.d1.txt
Select file number. :4
Download complete!
Thank you so much! Enjoy your time! Bye!
[EXIT]
Good bye! See you next time!
st2017146036@602-a:~/project/2019fall/base code/s-c model/Proj3$
```

이제 User1이 아닌 User2에서 특수 메시지 [FILE]을 전송해 파일을 요청해 보았다. 파일을 요청한 User2는 새로운 Client가 되었다. User1이 요청했을 때와 똑같이 [FILE] 메시지를 받은 User1은 파일 리스트를 전송해 준다. 원하는 파일 번호를 선택 후 다운이 완료된다. 채팅으로 복귀하여 채팅방에서 나가는 것까지 구현하였다.

- User1 화면

```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
hello! I'm user1..
[user2] : Hi! Nice to meet you!
What's your name?
[user2] : My name is user2!
I want to download your file!
[user2] : Ok^^
[FILE]
1.a2.txt
2.b2.txt
3.c2.txt
4.d2.txt
Select file number. :3
Download complete!
Thank you!
[user2] : you're welcome^^
[user2] : Can I download your file, too?
Sure!
[user2] : [FILE]
Received number: 4
Embedded OS
[user2] :
[user2] : Thank you so much! Enjoy your time! Bye!
[user2] : [EXIT]
```

User2가 원하는 파일 번호를 받은 후 User1이 보낸 파일 내용을 출력하였다. 그런 다음 채팅으로 다시 복귀하였다.

- Server 화면

```
st2017146036@602-d: ~/project/2019fall/base_code/s-c_model/Proj3
Log-in success! [user1] - Welcome to GTalk
===== Chatting Room =====
accept() is OK...

Log-in success! [user2] - Welcome to GTalk
===== Chatting Room =====
[user1] : hello! I'm user1..
[user2] : Hi! Nice to meet you!
[user1] : What's your name?
[user2] : My name is user2!
[user1] : I want to download your file!
[user2] : Ok^^
[user1] : [FILE]
[user1] :
[user1] : Thank you!
[user2] : you're welcome^^
[user2] : Can I download your file, too?
[user1] : Sure!
[user2] : [FILE]
[user2] :
[user2] : Thank you so much! Enjoy your time! Bye!
[user2] : [EXIT]
```

4. 파일의 다운 확인

- User1

```
st2017146036@602-c: ~/project/2019fall/base_code/s-c_model/Proj3
I want to download your file!
[user2] : Ok^^
[FILE]
1.a2.txt
2.b2.txt
3.c2.txt
4.d2.txt
Select file number. :3
Download complete!
Thank you!
[user2] : you're welcome^^
[user2] : Can I download your file, too?
Sure!
[user2] : [FILE]
Received number: 4
Embedded OS
[user2] :
[user2] : Thank you so much! Enjoy your time! Bye!
[user2] : [EXIT]
[EXIT]
Good bye! See you next time!
st2017146036@602-c:~/project/2019fall/base_code/s-c_model/Proj3$ ls
a1.txt b1.txt c1.txt c2.txt client1.c d1.txt user_login
st2017146036@602-c:~/project/2019fall/base code/s-c model/Proj3$
```

User2로부터 받은 3번 파일인 c2.txt의 다운로드가 완료되어 있는 것을 확인할 수 있다.

- User2

```
st2017146036@602-a: ~/project/2019fall/base_code/s-c_model/Proj3
My name is user2!
[user1] : I want to download your file!
Ok^^
[user1] : [FILE]
Received number: 3
Electronic Engineering
[user1] :
[user1] : Thank you!
you're welcome^^
Can I download your file, too?
[user1] : Sure!
[FILE]
1.a1.txt
2.b1.txt
3.c1.txt
4.d1.txt
Select file number. :4
Download complete!
Thank you so much! Enjoy your time! Bye!
[EXIT]
Good bye! See you next time!
st2017146036@602-a:~/project/2019fall/base_code/s-c_model/Proj3$ ls
a2.txt b2.txt c2.txt client2.c d1.txt d2.txt user_login
st2017146036@602-a:~/project/2019fall/base_code/s-c_model/Proj3$
```

User2도 마찬가지로 요청한 파일 번호 4번의 d1.txt가 다운완료되어 있는 것을 확인할 수 있다.

- Project #3를 마치며...

처음에 Project #2에서 파일 전송만 추가하면 된다고 생각하여 걱정이 없었다. 하지만 Project #3에 대한 안내문과 어떻게 해야 하는지 자세히 공부하고 이해해 보다보니 단순한 과정이 아니라는 것을 알았다. 단순히 주된 Server와 Client 하나 둘이 아닌, Client가 '새로운' Client가 되고 또 다른 Client가 '새로운' Server가 되어야 했기 때문에 시간이 오래 걸릴 것을 예상했다. 텀프로젝트를 수행하는데 11월 24일부터 3주의 시간을 잡았다. 일주일동안 Project #2를 완성하고 또 다음 일주일동안 Project #3를 완성하는 것을 목표로 했는데 무려 2주 동안 Project #2를 구성하고 코드를 짜는 데 시간을 들였다. 이제 제출까지 일주일만 남았기 때문에 Project #3는 Project #2보다 고려해야 할 것이 많다고 생각하여 걱정이 앞섰다. 위의 말대로 Client가 새로운 Server, 새로운 Client가 되는 것을 생각해야 했는데 단순히 생각해보니 생각보다 어려운 것이 아니었다. 처음 Server와 Client를 생성했던 것처럼 Server일 때는 Server와 Client 소켓을 모두 생성해주고 Client일 때는 Client 소켓을 생성해주면 되는 것이었다. 그래서 본 Server, Client와 겹치지 않게 새롭게 모든 변수와 소켓을 선언해 주었고 recv와 send 함수, 파일 함수를 이용하여 파일을 전송하고 받는 것을 구현하는 데 성공했다. 아쉬웠던 점은 추가 기능으로 파일을 받은 쪽에서 다운 받은 파일의 내용을 열어 확인하게 하고 싶었으나 잘 구현되지 않았다. 그래서 파일을

전송하는 쪽에서 받은 파일 리스트 번호를 출력하고 그 번호에 해당된 파일의 내용을 출력하는 것으로 대신 구현했다. 한 가지 더는 파일을 요청하고 다운이 완료된 Client가 다시 채팅으로 복귀하여 첫 대화를 전송할 때 “[user1] : ”이 나온 후 다음 줄에 “[user1] : 메시지” 이렇게 중복이 되어 출력되었다. Project #2 때처럼 단순히 채팅만 하여 중복을 잡는 것은 어렵지 않았지만 Project #3의 파일 전송 기능을 구현한 후 정상적으로 어떻게 출력시키는지 생각하기 어려웠다.

이렇게 Client가 단순히 두 개일 때는 어떤 Client가 새로운 Client가 되고 새로운 Server가 되는지 쉽게 결정할 수 있지만, ‘카카오톡’의 단체 채팅처럼 20명, 아니 수백명이상이 있을 때는 감히 어떤 Client가 Server가 되고 어떻게 관리해야 할지 매우 어려운 과제가 될 것 같다.