

Final Project #3

2014146034 이정욱

2014146019 석상우

목차

1. 동작설명
2. 코드 분석
3. 성능 수준

1. 동작 설명

1) [FILE]을 입력했을 때

(1) SERVER 화면 : 아직 user2의 메시지를 수신한다.(user1의 메시지는 수신하지 못함)

```
st2014146034@602-d: ~/tcp
st2014146034@602-d:~/tcp$ ./server
Server-socket() sockfd is OK...
Server-bind() is OK...
Server-listen() is OK...
=====
User Information
ID : user1, PW : passwd1
=====
Log-in success!! [user1] ***
-----Chatting Room-----
[user1] : <4181> , <220.149.128.101>
=====
User Information
ID : user2, PW : passwd2
=====
Log-in success!! [user2] ***
-----Chatting Room-----
[user2] : <4182> , <220.149.128.102>
[user2] : Hello~user1?
[user1] : hi~user2!
█
```

(2) user1 화면 : 채팅 중 client1이 먼저 [FILE]을 치면 P2P 하고싶은 ID를 입력 받는다.

```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] ***
-----Chatting Room-----
[user2] : Hello~user1?
hi~user2!
[FILE]
Please Enter ID : █
```

(3) user2 화면 : CLIENT2는 아직 채팅중이다.

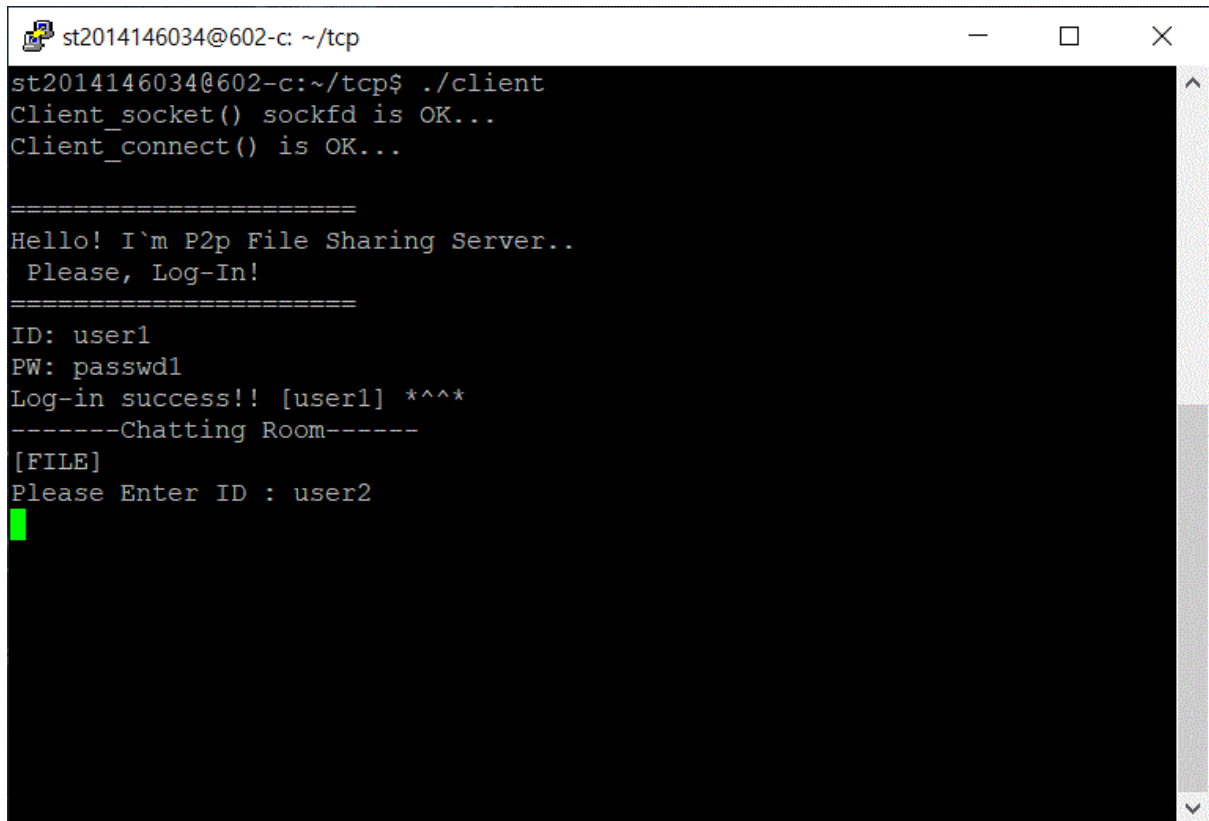
```
st2014146034@602-a: ~/tcp
st2014146034@602-a:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user2
PW: passwd2
Log-in success!! [user2] ***
-----Chatting Room-----
Hello~user1?
[user1] : hi~user2!
█
```

1. 동작 설명

2) P2P할 유저를 입력했을 때

(1) SERVER 화면 : SEVER는 이제 CLIENT1과 CLIENT2의 메시지를 수신하지 못한다.

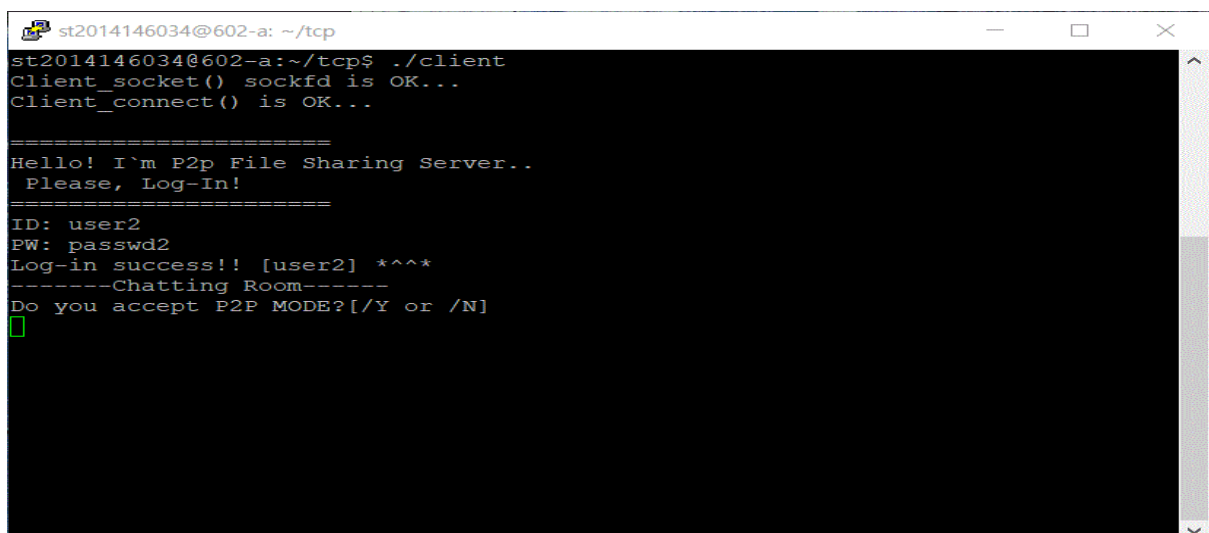
(2) user1 화면 : user1이 ID user2를 입력한다.



```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...

=====
Hello! I`m P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] *^^*
-----Chatting Room-----
[FILE]
Please Enter ID : user2
█
```

(3) user2 화면 : user2는 P2P 요청을 수락하거나 거절한다.[Y or /N]



```
st2014146034@602-a: ~/tcp
st2014146034@602-a:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...

=====
Hello! I`m P2p File Sharing Server..
Please, Log-In!
=====
ID: user2
PW: passwd2
Log-in success!! [user2] *^^*
-----Chatting Room-----
Do you accept P2P MODE?[/Y or /N]
█
```

1. 동작 설명

3-1) user2가 거절 했을 때

(1) SERVER 화면 : 다시 user1과 user2의 메시지를 수신한다.

```
st2014146034@602-d: ~/tcp
st2014146034@602-d:~/tcp$ ./server
Server-socket() sockfd is OK...
Server-bind() is OK...
Server-listen() is OK...
=====
User Information
ID : user1, PW : passwd1
=====
Log-in success!! [user1] ***
-----Chatting Room-----
=====
User Information
ID : user2, PW : passwd2
=====
Log-in success!! [user2] ***
-----Chatting Room-----
[user2] : nono!
[user1] : ok....
█
```

(2) user11 화면 : P2P요청이 거절되고 다시 채팅방으로 돌아간다.

```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] ***
-----Chatting Room-----
[FILE]
Please Enter ID : user2
[user2] : nono!
ok....
█
```

(3) user2 화면 : P2P요청 거절 후 다시 채팅방으로 돌아간다.

```
st2014146034@602-a: ~/tcp
st2014146034@602-a:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user2
PW: passwd2
Log-in success!! [user2] ***
-----Chatting Room-----
Do you accept P2P MODE?[/Y or /N]
/N
nono!
[user1] : ok....
█
```

1. 동작 설명

3-2) user2가 수락했을 때

(1) SERVER 화면 : user1과 user2의 메시지를 수신하지 못한다.

(2) user1 화면 : 수락시 user1은 P2P의 서버가 된다.

```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...

=====
Hello! I`m P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] *^^*
-----Chatting Room-----
[FILE]
Please Enter ID : user2
P2P-Server-socket() sockfd is OK...
P2P-Server-bind() is OK...
P2P-accept() is OK...

-----<P2P SERVER>-----
█
```

(3) user2 화면 : 수락시 user2는 P2P의 클라이언트가 되고 user1의 파일 리스트를 받는다.

```
st2014146034@602-a: ~/tcp
Client_connect() is OK...

=====
Hello! I`m P2p File Sharing Server..
Please, Log-In!
=====
ID: user2
PW: passwd2
Log-in success!! [user2] *^^*
-----Chatting Room-----
Do you accept P2P MODE?[/Y or /N]
/Y
[FILE]
4181
220.149.128.101
P2P-Client-socket() sockfd is OK...
P2P-Client-connect() is OK...

-----<P2P CLIENT>-----
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
█
```

1. 동작 설명

4) user2가 FILE LIST 2번(c.txt) 요청했을 때

(1) SERVER 화면 : user1과 user2의 메시지를 수신하지 못한다.

(2) user1 화면 : User2로부터 파일번호를 받으면 user1은 파일을 보내주고 전송 완료시

전송 완료 메시지가 출력된다.

```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...

=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] *^^*
-----Chatting Room-----
[FILE]
Please Enter ID : user2
P2P-Server-socket() sockfd is OK...
P2P-Server-bind() is OK...
P2P-accept() is OK...

-----<P2P SERVER>-----
Success P2P file tranfer!!
█
```

(3) user2 화면 : 2번 파일을 입력하면 file을 받았다는 확인 메시지와 함께 다시 파일 리스트를 보내준다.

```
st2014146034@602-a: ~/tcp
PW: passwd2
Log-in success!! [user2] *^^*
-----Chatting Room-----
Do you accept P2P MODE?[/Y or /N]
/Y
[FILE]
4181
220.149.128.101
P2P-Client-socket() sockfd is OK...
P2P-Client-connect() is OK...

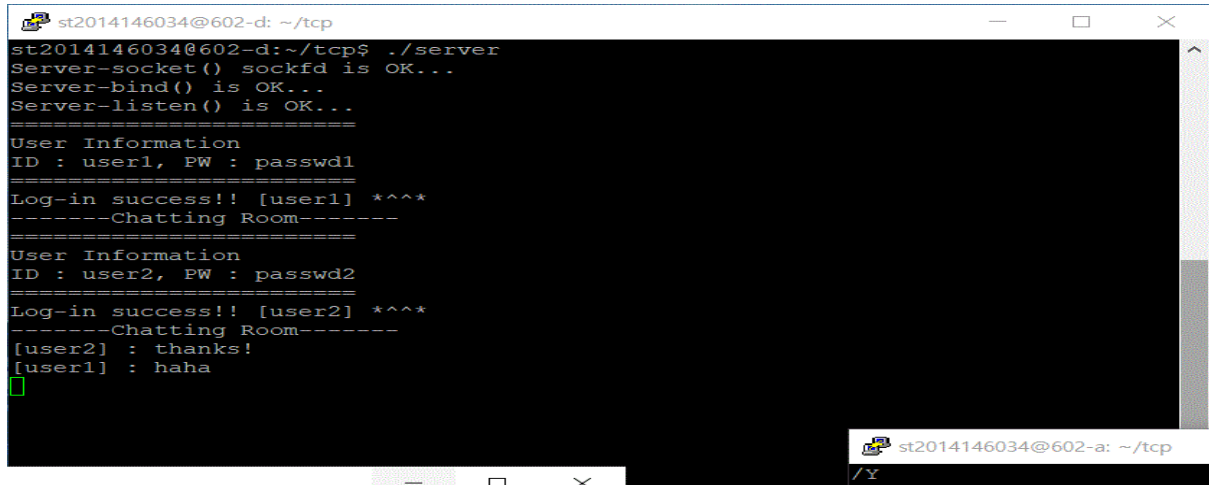
-----<P2P CLIENT>-----
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
2

P2P Success received file!
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
█
```

1. 동작 설명

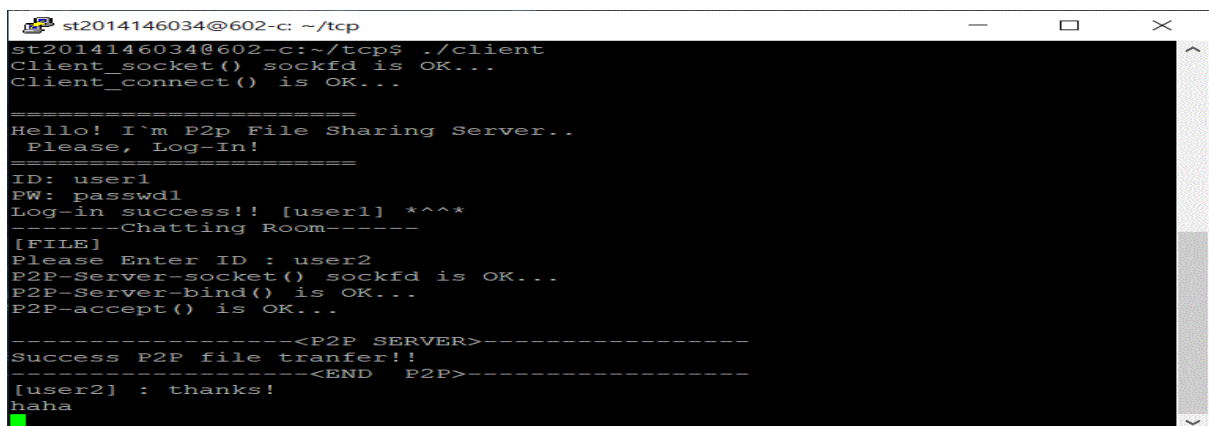
5) user2가 /q를 입력했을 때

(1) SERVER 화면 : 이제 user1과 user2의 메시지를 수신한다.



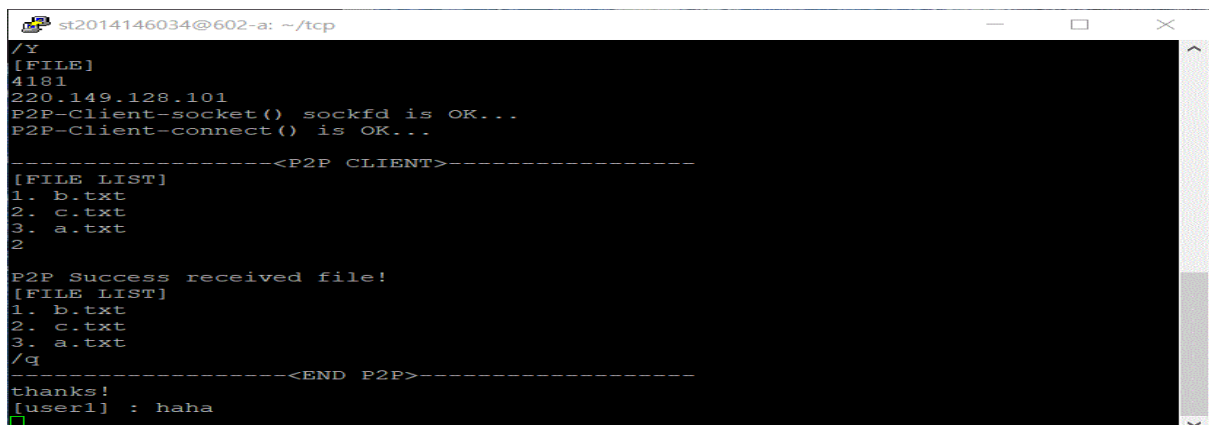
```
st2014146034@602-d: ~/tcp
st2014146034@602-d:~/tcp$ ./server
Server-socket() sockfd is OK...
Server-bind() is OK...
Server-listen() is OK...
=====
User Information
ID : user1, PW : passwd1
=====
Log-in success!! [user1] ***
-----Chatting Room-----
=====
User Information
ID : user2, PW : passwd2
=====
Log-in success!! [user2] ***
-----Chatting Room-----
[user2] : thanks!
[user1] : haha
█
```

(2) user1 화면 : User2로부터 /q를 입력받으면 P2P서버는 종료되고 다시 채팅방으로 입장된다.



```
st2014146034@602-c: ~/tcp
st2014146034@602-c:~/tcp$ ./client
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] ***
-----Chatting Room-----
[FILE]
Please Enter ID : user2
P2P-Server-socket() sockfd is OK...
P2P-Server-bind() is OK...
P2P-accept() is OK...
-----<P2P SERVER>-----
Success P2P file tranfer!!
-----<END P2P>-----
[user2] : thanks!
haha
█
```

(3) user2 화면 : /q를 입력하면 P2P 클라이언트는 종료되고 다시 채팅방으로 입장된다.



```
st2014146034@602-a: ~/tcp
/Y
[FILE]
4181
220.149.128.101
P2P-Client-socket() sockfd is OK...
P2P-Client-connect() is OK...
-----<P2P CLIENT>-----
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
2
P2P Success received file!
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
/q
-----<END P2P>-----
thanks!
[user1] : haha
█
```


1. 동작 설명

6) user1과 user2 정상종료

(1) SERVER 화면 : user1과 user2가 정상 종료했다.

```
st2014146034@602-d: ~/tcp
st2014146034@602-d:~/tcp$ ./server
Server-socket() sockfd is OK...
Server-bind() is OK...
Server-listen() is OK...
=====
User Information
ID : user1, PW : passwd1
=====
Log-in success!! [user1] *^^*
-----Chatting Room-----
=====
User Information
ID : user2, PW : passwd2
=====
Log-in success!! [user2] *^^*
-----Chatting Room-----
[user2] : thanks!
[user1] : haha
[user1] is left the chat...
[user2] is left the chat...
█
```

(2) user1 화면 : /exit를 입력하면 정상 종료된다..

```
st2014146034@602-c: ~/tcp
Client_socket() sockfd is OK...
Client_connect() is OK...
=====
Hello! I'm P2p File Sharing Server..
Please, Log-In!
=====
ID: user1
PW: passwd1
Log-in success!! [user1] *^^*
-----Chatting Room-----
[FILE]
Please Enter ID : user2
P2P-Server-socket() sockfd is OK...
P2P-Server-bind() is OK...
P2P-accept() is OK...
-----<P2P SERVER>-----
Success P2P file tranfer!!
-----<END P2P>-----
[user2] : thanks!
haha
/exit
st2014146034@602-c:~/tcp$ █
```

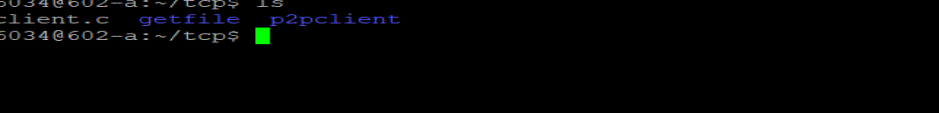
(3) user2 화면 : /exit를 입력하면 정상 종료된다..

```
st2014146034@602-a: ~/tcp
[FILE]
4181
220.149.128.101
P2P-Client-socket() sockfd is OK...
P2P-Client-connect() is OK...
-----<P2P CLIENT>-----
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
2
P2P Success received file!
[FILE LIST]
1. b.txt
2. c.txt
3. a.txt
/q
-----<END P2P>-----
thanks!
[user1] : haha
[user1] is left the chat...
/exit
st2014146034@602-a:~/tcp$ █
```

1. 동작 설명

7) 전송확인!

(1) 각각 클라이언트에는 getflile과 p2pclient의 디렉토리가 있다.



```
st2014146034@602-a: ~/tcp
st2014146034@602-a:~/tcp$ ls
client  client.c  getfile  p2pclient
st2014146034@602-a:~/tcp$
```

(2) user1 화면 : user1의 p2pclient에 있는 FILE LIST

```
st2014146034@602-c: ~/tcp/p2pclient
st2014146034@602-c:~/tcp/p2pclient$ ls
a.txt b.txt c.txt
st2014146034@602-c:~/tcp/p2pclient$ ls
a.txt b.txt c.txt
st2014146034@602-c:~/tcp/p2pclient$ vi c.txt
st2014146034@602-c:~/tcp/p2pclient$ clear
st2014146034@602-c:~/tcp/p2pclient$ vi c.txt
1 wassup?
```

(3) user2 화면 : 2번 파일인 c.txt파일이 getfile에 정상적으로 수신이 되었다.

```
st2014146034@602-a: ~/tcp/getfile
st2014146034@602-a:~/tcp/getfile$ ls
c.txt
st2014146034@602-a:~/tcp/getfile$ ls 0a
ls: '0a'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다
st2014146034@602-a:~/tcp/getfile$ ls -a
. . . c.txt
st2014146034@602-a:~/tcp/getfile$ ls
c.txt
st2014146034@602-a:~/tcp/getfile$ clear
st2014146034@602-a:~/tcp/getfile$ ls
c.txt
st2014146034@602-a:~/tcp/getfile$ vi c.txt
1 ~assup?
~
~
~
~
~
~
~
~
```

2. 코드 분석

1) server.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <errno.h>
6 #include <arpa/inet.h>
7 #include <sys/types.h>
8 #include <sys/socket.h>
9 #include <signal.h>
10 #include <sys/mman.h>
11 #include <sys/select.h>
12 #define BUFF_SIZE 1024 //buff size
13 #define SEV_IP "220.149.128.100" //
14 #define SERV_PORT 4180
15 #define INIT_MSG "=====\nHello! I'm P2p File Sharing Server..\n Please, Log-In!\n=====\n" // hello msg
16 #define User1_ID "user1"
17 #define User1_PW "passwd1"
18 #define User2_ID "user2"
19 #define User2_PW "passwd2"
20 #define User3_ID "user3"
21 #define User3_PW "passwd3"
22 #define Wrong_PW "Log-in fail: Incorrect password\n\n"
23 #define FALSE 0
24 #define TRUE 1
25 int* sockets=NULL;
26 int* cnt_socket = NULL;
27 int* clients_port = NULL;
28 char** clients_ID = NULL;
29 char** clients_IP = NULL;
30 char** clients_PORT = NULL;
31 void mystop(int signo)
32 {
33     munmap(cnt_socket,sizeof *cnt_socket);
34     munmap(sockets,sizeof (*sockets)*5);
35     for(int i=0; i<5;i++)
36         munmap(*(clients_ID+i),sizeof (*(clients_ID+i)) * 20);
37     for(int i=0; i<5;i++)
38         munmap(*(clients_IP+i),sizeof (*(clients_IP+i)) * 20);
39     for(int i=0; i<5;i++)
40         munmap(*(clients_PORT+i),sizeof (*(clients_PORT+i)) *5);
41     exit(1);
42 }
```

LINE 1~11 : select함수와 공유메모리,signal함수를 사용하기위해 추가적으로 헤더파일 선언

LINE 12~ 24 : SERVER PORT,SERVER IP 정의와 USER들의 ID PASSWORD 정의,

LINE 25~ 30 : 공유메모리를 사용하기 위해 변수 선언

Sockets : user들의 FD를 저장하는 메모리

cnt_socket : FD의 총 개수를 저장하는 메모리

clients_port (CHAR): user의 port번호를 저장하는 메모리

clients_ID : user들의 ID를 저장하는 메모리

clients_IP : user들의 IP를 저장하는 메모리

clients_PORT (INT): user들의 ID를 저장하는 메모리

LINE 31~ 42 : 인터럽트 신호처리를 하기위한 함수(ctrl+C 종료시 함수가 실행되고 공유 메모리를 해제한다.

2. 코드 분석

1) server.c

```
43 int main(void)
44 {
45     int p2p_flag = FALSE;
46     int flag = FALSE;
47     int server_socket; //server socket
48     int client_socket; //client socket
49     int client_addr_size; // addr size
50     struct sockaddr_in server_addr;
51     struct sockaddr_in client_addr;
52     int rcv_byte;
53     char buff[BUFF_SIZE];
54     char socket_pid[BUFF_SIZE];
55     char p2p_ip_buff[BUFF_SIZE];
56     char p2p_port_buff[BUFF_SIZE];
57     char msg_file[BUFF_SIZE];
58     char id[20];
59     char pw[20];
60     char target[20];
61     char yorn[BUFF_SIZE];
62     char msg[BUFF_SIZE];
63     int option;
64
65     fd_set reads, cpy_reads;
66     struct timeval timeout;
67     int fd_max, fd_num;
68     int i;
69     /*-----Shared Memory Mapping Part-----*/
70     sockets = mmap(NULL, sizeof (*sockets)*5, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
71     cnt_socket = mmap(NULL, sizeof *cnt_socket, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
72     clients_ID = mmap(NULL, sizeof (*clients_ID)*5, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
73     for(int i=0; i<5; i++)
74         *(clients_ID+i) = mmap(NULL, sizeof (*clients_ID)*20, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
75     clients_IP = mmap(NULL, sizeof (*clients_IP)*5, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
76     for(int i=0; i<5; i++)
77         *(clients_IP+i) = mmap(NULL, sizeof (*clients_IP)*20, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
78     clients_PORT = mmap(NULL, sizeof (*clients_PORT)*5, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
79     for(int i=0; i<5; i++)
80         *(clients_PORT+i) = mmap(NULL, sizeof (*clients_PORT)*20, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
81
82     signal(SIGINT, (void*)mystop);
83
84     server_socket = socket(AF_INET, SOCK_STREAM, 0);
85     option = 1;
86     setsockopt(server_socket, SOL_SOCKET, SO_REUSEADDR, (char*)&option, sizeof(option));
87     if(-1 == server_socket) // cTeate a seRver socket
88     {
89         printf("server socket fail\n");
90         exit(1);
91     }
```

LINE 45 ~ 63 : 변수 선언

55 : p2p_ip_buff[BUFF_SIZE] : p2p를 하려는 유저의 IP번호를 저장

56 : p2p_port_buff[BUFF_SIZE] : p2p를 하려는 유저의 PORT번호를 저장

61 : yorn[BUFF_SIZE]: /Y와 /N의 수신버퍼

63 : option : BACKLOG

65~68 : SELECT(다중 입출력 함수)를 사용하기 위한 변수 선언

LINE 70 ~ 80 : 공유메모리 변수에 메모리 할당

LINE 82 : 인터럽트 처리 함수 SET

2. 코드 분석

1) server.c

```
93     server_addr.sin_family    = AF_INET;
94     server_addr.sin_port      = htons(SERV_PORT);
95     server_addr.sin_addr.s_addr= htonl(INADDR_ANY); //32bit IPv4 address
96
97     memset(&(server_addr.sin_zero), 0, 8);
98     if(setsockopt(server_socket, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option)) < 0 ) // set a socket option
99     {
100         perror("setsockopt");
101         close(server_socket);
102         return -1;
103     }
104     if(-1 == bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr))) //Associate a socket with a port and address
105     {
106         perror("bind() fail!!");
107         exit(1);
108     }
109     else printf("Server-bind() is OK...\n");
110     if(-1 == listen(server_socket, 10)){ // listen for connection
111         perror("listen fail \n");
112         exit(1);
113     }
114     else printf("Server-listen() is OK...\n");
```

LINE 84~ 116 : 클라이언트와 연결 하기 위한 소켓을 만들고 binding과 listen

```
119     FD_ZERO(&reads);
120     FD_SET(server_socket, &reads);
121     fd_max = server_socket;
```

LINE 119 ~ 121 :

119 : reads 라는 fd_set 구조체를 초기화

120: reads에 서버FD를 SET

121: fd_max에 서버FD를 넣음

```
122     while(1)
123     {
124         cpy_reads = reads;
125         timeout.tv_sec = 5;
126         timeout.tv_usec = 50000;
127         if((fd_num = select(fd_max+1, &cpy_reads, 0, 0, &timeout)) == -1)
128         {
129             perror("select() error");
130             break;
131         }
132         if(fd_num == 0) continue;
```

LINE 124 : reads를 cpy_reads에 복사

LINE 127 : select는 system block함수이므로 cpy_reads중에 입출력 하고자 하는 FD가 SET될때 까지 BLOCK

LINE 132 : block대기시간이 지나면 fd_num에 0이 들어간후 다시 대기

2. 코드 분석

1) server.c

```
134     for(i=0; i < fd_max+1; i++)
135     {
136         if(FD_ISSET(i, &cpy_reads))
137         {
138             if(server_socket == i)
139             {
140                 client_socket = accept( server_socket, (struct sockaddr*)&client_addr, &client_addr_size);
141
142                 if(client_socket == -1) perror("accept() error!");
143                 FD_SET(client_socket, &reads);
144                 if(fd_max < client_socket)
145                     fd_max = client_socket;
146                 send(client_socket, INIT_MSG, strlen(INIT_MSG)+1, 0); // send hello msg
147                 rcv_byte = recv(client_socket, id, sizeof(id), 0); // recv_id
148                 rcv_byte = recv(client_socket, pw, sizeof(pw), 0); //recv_pw
149                 printf("=====\n"); //print information
150                 printf("User Information\n");
151                 printf("ID : %s, PW : %s\n", id, pw);
152                 printf("=====\n");
153
154                 /*-----Compare ID, Compare PW-----*/
155                 if((strcmp(id, User1_ID) == 0) || (strcmp(id, User2_ID) == 0) || (strcmp(id, User3_ID) == 0))
156                 {
157                     if((strcmp(pw, User1_PW) == 0) && (strcmp(id, User1_ID) == 0))
158                     {
159                         printf("Log-in success!! [%s] ***\n", id);
160                         printf("-----Chatting Room-----\n");
161                         send(client_socket, "o", strlen("o")+1, 0);
162                         flag = TRUE;
163                     }
164                     else if ((strcmp(pw, User2_PW) == 0) && (strcmp(id, User2_ID) == 0))
165                     {
166                         printf("Log-in success!! [%s] ***\n", id);
167                         printf("-----Chatting Room-----\n");
168                         send(client_socket, "o", strlen("o") + 1, 0);
169                         flag = TRUE;
170                     }
171                     else if ((strcmp(pw, User3_PW) == 0) && (strcmp(id, User3_ID) == 0))
172                     {
173                         printf("Log-in success!! [%s] ***\n", id);
174                         printf("-----Chatting Room-----\n");
175                         send(client_socket, "o", strlen("o") + 1, 0);
176                         flag = TRUE;
177                     }
178                     else
179                     {
180                         printf("%s\n", Wrong_PW);
181                         send(client_socket, "p", strlen("p")+1, 0);
182                         FD_CLR(client_socket, &reads);
```

LINE 136 : 특정 FD bit가 set되어있는 FD번호를 찾음

LINE 138 : set되어있는 FD가 서버FD이면 실행

LINE 140 : client를 accept

LINE 143 : fd_set 구조체에 accept한 client FD를 SET

LINE 144~145 : fd_max에 accept한 FD번호를 최신화

LINE 149~ : 클라이언트로부터 ID와 PW를 입력 받고 그에 해당하는 메시지를 보냄

LINE 178~182 : password 입력이 잘못된 경우 다시 fd_set 구조체의 클라이언트 FD변수를 clear

2. 코드 분석

1) server.c

```
187         else
188         {
189             printf("Log-in fail: ID Does not exist..\n\n");
190             send(client_socket,"i",strlen("i")+1,0);
191             FD_CLR(client_socket,&reads);
192             fd_max--;
193             close(client_socket);
194         }
```

LINE 187 ~ 194 : ID가 없으면 다시 fd_set 구조체의 클라이언트 FD변수를 clear

```
195         /*-----GET <ID>,<IP>,<PORT>-----*/
196         if(flag == TRUE)
197         {
198             *(sockets + *cnt_socket) = client_socket;
199             recv(client_socket,buff,sizeof(buff),0);
200             strcpy(*(clients_IP+*cnt_socket),buff);
201             recv(client_socket,buff,sizeof(buff),0);
202             strcpy(*(clients_PORT+*cnt_socket),buff);
203             strcpy(*(clients_ID+*cnt_socket),id);
204             // printf("[%s] : <%s> , <%s> \n",*(clients_ID+*cnt_socket),*(clients_PORT+*cnt_socket),*(clients_IP+*cnt_socket));
205             flag = FALSE;
206             *cnt_socket+=1;
207         }
208     }
```

LINE 195 ~ 208 : ID와 PW가 정상 입력되면 해당 클라이언트의 ID,PORT 번호,IP 주소를 공유 메모리에 저장(p2p에 사용)

LINE 206 : socket개수를 저장하는 변수 증가

flag변수 : 클라이언트가 ID와 PW가 정상 입력했을때 TRUE로 SET 아니면 FALSE로 SET

2. 코드 분석

1) server.c

```
209         else
210         {
211             rcv_byte = recv(i,msg,sizeof(msg),0);
212             strcpy(id,*(clients_ID+(i-4)));
213             if(strcmp(msg,"[FILE]") == 0 )
214             {
215                 rcv_byte = recv(i,target,sizeof(target),0);
216                 int myself;
217                 for(int j=0; j<*cnt_socket;j++)
218                 {
219                     if(strcmp(id,*(clients_ID+j))== 0)
220                         myself = j;
221                 }
222                 for(int j=0; j<*cnt_socket; j++)
223                 {
224                     if(strcmp(target,*(clients_ID+j)) == 0)
225                     {
226                         strcpy(msg_file,"[FILE]");
227                         send(*(sockets+j),"Do you accept P2P MODE?[/Y or /N]",strlen("Do you accept P2P MODE?[/Y or /N]")+1,0);
228                         rcv_byte = recv(*(sockets+j),yorn,sizeof(yorn),0);
229                         if((strcmp(yorn,"/N")==0) || (strcmp(yorn,"/n")==0))
230                         {
231                             send(i,yorn,strlen(yorn)+1,0);
232                         }
233                         else if((strcmp(yorn,"/Y")==0) || strcmp(yorn,"/y")==0)
234                         {
235                             send(i,yorn,strlen(yorn)+1,0);
236                             send(*(sockets+j),msg_file,strlen(msg_file)+1,0);
237                             sleep(1);
238                             send(*(sockets+j),*(clients_PORT+myself),strlen(*(clients_PORT+myself))+1,0);
239                             send(*(sockets+j),*(clients_IP+myself),strlen(*(clients_IP+myself))+1,0);
240                         }
241                         else
242                         {
243                             send(i,yorn,strlen(yorn)+1,0);
244                         }
245                     }
246                 }
247                 p2p_flag = TRUE;
248             }
249         }
```

LINE 209 : 클라이언트FD가 SET됐을때 진입

LINE 211 메시지 수신

LINE 213 : msg가 [FILE]일 때 진입!

LINE 215 : 클라이언트로부터 p2p하고자 하는 ID를 수신

LINE 217 ~ 221 : myself에 클라이언트의 번호 저장

LINE 222 ~ 247 : 상대 유저에게 p2p할건지의 여부를 묻는 메시지를 보내고 상대 유저로부터 /Y or /N 메시지를 수신 그 수신한 메시지를 다시 클라이언트에게 메시지를 보내준다.

/Y를 받았을 경우에만 상대유저에게 클라이언트의 IP,PORT번호를 보내줌

2. 코드 분석

1) server.c

```
249         else if(strcmp(msg, "/exit") == 0)
250         {
251             for(int j=0; j<5; j++)
252             {
253                 if(*(sockets+j) == i)
254                 {
255                     send(*(sockets+j), msg, strlen(msg)+1, 0);
256                     *(sockets+j) = 0;
257                     *cnt_socket--;
258                 }
259             }
260             FD_CLR(i, &reads);
261             close(i);
262             sprintf(buff, "[%s] is left the chat...", id);
263         }
264         else
265         {
266             sprintf(buff, "[%s] : %s", id, msg);
267         }
268         if(p2p_flag == FALSE)
269         {
270             printf("%s\n", buff);
271             for(int j=0; j<5; j++)
272             {
273                 if(i != *(sockets+j))
274                 {
275                     send(*(sockets+j), buff, strlen(buff)+1, 0);
276                 }
277             }
278         }
279         p2p_flag = FALSE;
280     }
281 }
282 }
283 }
```

LINE 249 : 클라이언트로부터 /exit(채팅 방 나가는 메시지)를 수신하면 진입

LINE 251 ~ 263 : 공유 메모리에 있는 해당 클라이언트의 정보를 없애고 소켓을 닫아준다.

p2p_flag : [FILE] 메시지를 수신할 때에만 SET시켜서 필요없는 메시지를 보내지 못하도록 사용

2. 코드 분석

2) client.c

```
1 #include <stdio.h>
2 #include <sys/select.h>
3 #include <sys/socket.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <arpa/inet.h>
7 #include <string.h>
8 #include <unistd.h>
9 #include <errno.h>
10 #include <signal.h>
11 #include <stdlib.h>
12 #include <netinet/in.h>
13 #include <sys/mman.h>
14 #include <dirent.h>
15 #define MAXLINE 1024 //def bufsize
16 #define SERV_IP "220.149.128.100"
17 #define SERV_PORT 4180
18 #define MY_IP "220.149.128.101"
19 #define MY_PORT "4181"
20 #define OK "o" // def connect success
21 #define WP "p" // def passwd error
22 #define WI "i" // def id error
23 int *mutex = NULL;
24 char* p2p_ip = NULL;
25 char* p2p_port = NULL;
26 void p2p_server();
27 void p2p_client();
28 void mystop(int signo)
29 {
30     munmap(p2p_ip, sizeof (*p2p_ip)*20);
31     munmap(p2p_port, sizeof (*p2p_port)*20);
32     munmap(mutex, sizeof *mutex);
33     exit(1);
34 }
```

LINE 1~14 : select함수와 공유메모리,signal함수,dirent.h를 사용하기위해 추가적으로 헤더파일 선언

LINE 15~ 22 : 버퍼 사이드와 서버의 IP PORT정보 클라이언트의 IP PORT정보 저장,id PW 에대한 서버의 메시지 정의

LINE 23~ 25 : 공유메모리를 사용하기 위해 변수 선언

Sockets : user들의 FD를 저장하는 메모리

p2p_ip: p2p요청하는 클라이언트의 IP를 저장하는 메모리

p2p_port: p2p요청하는 클라이언트의 PORT를 저장하는 메모리

LINE 26~27 : P2P 서버와 클라이언트 함수 정의

LINE 28 ~ 34 : 인터럽트 신호처리를 하기위한 함수(ctrl+C 종료시 함수가 실행되고 공유 메모리를 해제한다.

2. 코드 분석

2) client.c

```
35 int main(int argc, char **argv)
36 {
37     struct sockaddr in clientaddr;
38     int client_sockfd;
39     int client_len;
40     int rcv_byte;
41     int pipe_[2];
42     char pipe_buff[MAXLINE];
43     char id[20]; // id recieve buff
44     char pw[20]; // passwd recieve bff
45     char buf[MAXLINE]; //buff, buffer size
46     char last[MAXLINE]; // error or success buff
47     char msg[MAXLINE];
48     char p2p_id[20];
49     char p2p_ip[10];
50     char p2p_ip_buff[MAXLINE];
51     char p2p_port_buff[MAXLINE];
52     pid_t pid;
53
54     mutex = mmap(NULL, sizeof *mutex, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
55
56     p2p_ip = mmap(NULL, sizeof (*p2p_ip)*20, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
57     p2p_port = mmap(NULL, sizeof (*p2p_port)*20, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
58     *mutex = 1;
59     signal(SIGINT, (void*)mystop);
60     if(-1 == pipe(pipe_))
61     {
62         perror("create PIPE() failed..");
63         exit(1);
64     }
```

LINE 37~ 51 : 변수 선언

LINE 54 ~ 57 : 공유메모리 변수에 메모리 할당

mutex = 임계영역을 만들기 위한 공유 메모리!

LINE 59 : 인터럽트 처리 함수 SET

LINE 60 ~ 64 : 부모 자식간 통신을 하기위한 PIPE

```
65     if((client_sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) { // create server socket
66         perror("error : ");
67         return 1;
68     }
69     else printf("Client_socket() sockfd is OK...\n");
70
71     clientaddr.sin_family = AF_INET;
72     clientaddr.sin_addr.s_addr = inet_addr(SERV_IP); //init server ip
73     clientaddr.sin_port = htons(SERV_PORT); // init server port
74     memset(&(clientaddr.sin_zero),0,8);
75
76     client_len = sizeof(clientaddr);
77
78     if(connect(client_sockfd, (struct sockaddr*)&clientaddr, client_len) == -1) { //connect part
79         perror("connect error : ");
80         exit(1);
81     }
82     else printf("Client_connect() is OK...\n\n");
83 }
```

LINE 65~ 82 : 서버와 연결 하기 위한 소켓을 만들고 connecting

2. 코드 분석

2) client.c

```
84     rcv_byte = recv(client_sockfd,buf,sizeof(buf),0);
85     printf("%s",buf);
86
87     printf("ID: ");
88     scanf("%s", id);
89     send(client_sockfd,id,strlen(id)+1,0);    // send id
90
91     printf("PW: ");
92     scanf("%s", pw);
93     send(client_sockfd, pw,strlen(pw)+1,0);    // send pw
94
95     rcv_byte = recv(client_sockfd,last,sizeof(last),0); //recieve errors
96
97     if(strcmp(last,OK) == 0)    //connect success
98     {
99         printf("Log-in success!! [%s] *^^*\n",id);
100        printf("-----Chatting Room-----\n");
101    }
102    else if(strcmp(last,WP) == 0)    //wrong passwd
103    {
104        printf("Log-in fail: Incorrect password...\n");
105        close(client_sockfd);
106        exit(1);
107    }
108    else if(strcmp(last,WI) == 0 )    //worng id
109    {
110        printf("Log-in fail: ID does not exist...\n");
111        close(client_sockfd);
112        exit(1);
113    }
114    send(client_sockfd,MY_IP,strlen(MY_IP)+1,0);
115    send(client_sockfd,MY_PORT,strlen(MY_PORT)+1,0);
```

LINE 84 : Log-in 해달라는 서버의 메시지 수신

LINE 87 ~ 93 : ID PW를 입력후 서버에 송신

LINE 95 : ID PW 일치 불일치 메시지를 수신

LINE 97 ~ 113 : 메시지에 따른 결과 값 출력

LINE 114 ~ 115 : 접속 완료되면 자신의 IP와 PORT번호를 서버에 송신

2. 코드 분석

2) client.c

```
116 pid = fork();
117 // buf : 1024 msg : 1024
118 if(pid == 0)
119 {
120     close(pipe_[1]);
121     while(1)
122     {
123         while(*mutex<=0);
124         //scanf("%s",buf);
125         fgets(buf,MAXLINE,stdin);
126         buf[strlen(buf)-1] = '\0';
127         send(client_sockfd,buf,strlen(buf)+1,0);
128         if(strcmp(buf,"[FILE]") == 0)
129         {
130             printf("Please Enter ID : ");
131             scanf("%s",p2p);
132             send(client_sockfd,p2p,strlen(p2p)+1,0);
133             read(pipe_[0],pipe_buff,MAXLINE);
134             if((strcmp(pipe_buff,"/N")==0) || (strcmp(pipe_buff,"/n")==0))
135             {
136                 *mutex = 1;
137             }
138             else if((strcmp(pipe_buff,"/Y")==0) || (strcmp(pipe_buff,"/y")==0))
139             {
140                 p2p_server();
141                 *mutex = 1;
142             }
143         }
144         else if(strcmp(buf,"/exit")==0)
145         {
146             close(client_sockfd);
147             exit(1);
148         }
149         else if((strcmp(buf,"/Y")==0) || (strcmp(buf,"/y")==0))
150         {
151             *mutex = 0;
152         }
153     }
154 }
```

<자식프로세스 부분>

LINE 120 : 부모로부터 메시지를 읽기만 하기 때문에 쓰는 부분은 닫아줌

LINE 128 : 해당 유저가 P2P를 하고싶을 때 진입

LINE 131 : P2P하고싶은 유저 아이디를 입력하고 송신

LINE 132 : 부모프로세스로부터 상대방의 수락 거절 여부 메시지를 받는다.

LINE 134 ~ 137 : 거절시 다시 채팅방으로 진입

LINE 138 ~ 142 : 수락시 클라이언트가 P2P의 서버를 만듦

LINE 144 : 채팅방 종료시 소켓을닫고 자식 프로세스 종료

LINE 149 ~ 152 : P2P 파일을 받는 유저의 코드쪽에서 승락을 한다면 LINE123 번줄에 올라가 임계영역에 진입을 못하여 채팅방 사용을 금지시킴!

2. 코드 분석

2) client.c

```
155     else if(pid > 0)
156     {
157         close(pipe_[0]);
158         while(1)
159         {
160             while(*mutex<=0);
161             rcv_byte = recv(client_sockfd,msg,sizeof(msg),0);
162             if(strcmp(msg,"[FILE]") == 0)
163             {
164                 printf("%s\n",msg);
165                 rcv_byte = recv(client_sockfd,p2p_port_buff,sizeof(p2p_port_buff),0);
166                 rcv_byte = recv(client_sockfd,p2p_ip_buff,sizeof(p2p_ip_buff),0);
167                 printf("%s\n%s\n",p2p_port_buff,p2p_ip_buff);
168                 strcpy(p2p_port,p2p_port_buff);
169                 strcpy(p2p_ip,p2p_ip_buff);
170                 p2p_client();
171                 *mutex = 1;
172             }
173             else if(strcmp(msg,"/exit")==0)
174             {
175                 close(client_sockfd);
176                 exit(1);
177             }
178             else if((strcmp(msg,"/Y")==0) || (strcmp(msg,"/y")==0))
179             {
180                 strcpy(pipe_buff,msg);
181                 write(pipe_[1],pipe_buff,strlen(pipe_buff));
182             }
183             else if((strcmp(msg,"/N")==0) || (strcmp(msg,"/n")==0))
184             {
185                 strcpy(pipe_buff,msg);
186                 write(pipe_[1],pipe_buff,strlen(pipe_buff));
187             }
188             else
189                 printf("%s\n",msg);
190         }
191     }
192     else;
193     munmap(p2p_ip,sizeof (*p2p_ip)*20);
194     munmap(p2p_port,sizeof (*p2p_port)*20);
195     munmap(mutex,sizeof *mutex);
196     return 0;
197 }
```

<부모 프로세스>

LINE 157 : PIPE를 통해 쓰기만 하기때문에 읽는 부분은 다음

LINE 162 : 어떤 유저와의 P2P 통신을 하겠다고 /Y 메시지를 보내면 [FILE]메시지를 서버측에서 받으면서 진입

LINE 165~ 169 : 상대방 IP와 PORT번호를 받음 (추후 P2P 클라이언트에서 P2P 서버와 접속하기 위함)

LINE 170 : P2P 클라이언트 함수로 진입

LINE 173 ~ 177 : 채팅방 종료를 하기위해 소켓을 닫고 프로세스 종료

LINE 178 ~ 187 : 서버측에서 받은 P2P 수락 거절 메시지를 수신, 파이프를 통해 자식프로세스에 메시지를 송신해서 P2P서버 함수로 진입할 수 있도록 해줌

LINE 193 ~ 197 : 공유메모리 해제

2. 코드 분석

2) client.c

```
198 void p2p_server()
199 {
200     DIR *dir_ptr = NULL;
201     struct dirent *file = NULL;
202     char my_dirname[MAXLINE];
203     int sockfd, new_fd;
204     struct stat buf;
205     struct sockaddr_in my_addr;
206     struct sockaddr_in their_addr;
207     unsigned int sin_size;
208     char p2p_buff[MAXLINE];
209     char p2p_msg[MAXLINE];
210     char txt_lists[10][MAXLINE];
211     char ch_num[5];
212     char filename[MAXLINE];
213     char p2p_path[256];
214     int get_list_num;
215     int p2p_rcv_byte;
216     int val = 1;
217     int err;
218     pid_t pid2;
```

<P2P server 함수>

LINE 200 : 디렉토리를 가리키는 dir pointer 선언

LINE 201 : 디렉토리의 정보를 담는 구조체 선언

202 ~ 217: 변수 선언

2. 코드 분석

2) client.c

```
219     sockfd = socket(AF_INET, SOCK_STREAM, 0);
220     if(sockfd == -1)
221     {
222         perror("Server-socket() error lol!");
223         return;
224     }
225     else printf("P2P-Server-socket() sockfd is OK...\n");
226
227     my_addr.sin_family = AF_INET;
228     strcpy(p2p_buff, MY_PORT);
229     int my_port = atoi(p2p_buff);
230     my_addr.sin_port = htons(my_port);
231     my_addr.sin_addr.s_addr = INADDR_ANY;
232     memset(&(my_addr.sin_zero), 0, 0);
233
234     if(setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (char*)&val, sizeof(val)) < 0)
235     {
236         perror("P2P-setsockopt");
237         close(sockfd);
238         return;
239     }
240
241     if(bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1)
242     {
243         perror("P2P-Server-bind() error lol!");
244         return;
245     }
246     else printf("P2P-Server-bind() is OK...\n");
247
248     if(listen(sockfd, 5) == -1)
249     {
250         perror("P2P-listen() error lol!");
251         return;
252     }
253
254     sin_size = sizeof(struct sockaddr_in);
255     new_fd = accept(sockfd, (struct sockaddr *)&their_addr, &sin_size);
256     if(new_fd == -1) perror("P2P-accept() error lol!");
257     else {printf("P2P-accept() is OK...\n\n");}
```

<P2P server 함수>

LINE 219 ~ 257 : p2p클라이언트와 통신하기 위한 소켓생성 ,binding,listen,accept()

new_fd에 FD저장

```
259     printf("-----<P2P SERVER>-----\n");
260     strcpy(my_dirname, "/home/st2014146034/tcp/p2pclient");
261     if((dir_ptr = opendir(my_dirname)) == NULL)
262     {
263         printf("<%s> can not read information...\n", my_dirname);
264         return;
265     }
266     int i=0;
267     while((file = readdir(dir_ptr)) != NULL)
268     {
269         printf(filename, "%s/%s", my_dirname, file->d_name);
270         err = lstat(filename, &buf);
271
272         if(err == -1) perror("error");
273         else
274         {
275             if(S_ISREG(buf.st_mode))
276             {
277                 strcpy(txt_lists[i], file->d_name);
278                 i++;
279             }
280         }
281     }
```

LINE 261 : my_dirname의 경로를 사용하여 해당 디렉토리를 open

이때 dir_ptr에는 첫 디렉토리를 가리킴

LINE 267 ~ 281 : filename에 my_dirname/파일이름 을 저장 후 lstat함수를 통해 해당 디렉토리의 정보를 buf에 저장 S_ISREG()함수를 통해 해당 디렉토리가 .txt면 txt_list에 파일이름을 저장

2. 코드 분석

2) client.c

```
282 while(1)
283 {
284     send(new_fd, "[FILE LIST]", strlen("[FILE LIST]") + 1, 0);
285     for(int j=0; j<i; j++)
286     {
287         sleep(1);
288         sprintf(p2p_msg, "%d. %s", j+1, txt_lists[j]);
289         send(new_fd, p2p_msg, strlen(p2p_msg) + 1, 0);
290     }
291     recv(new_fd, p2p_buff, sizeof(p2p_buff), 0);
292     if(strcmp(p2p_buff, "/q") == 0)
293     {
294         send(new_fd, "/q", strlen("/q") + 1, 0);
295         break;
296     }
297     get_list_num = atoi(p2p_buff);
298     send(new_fd, "file", strlen("file") + 1, 0);
299     sleep(1);
300     send(new_fd, txt_lists[get_list_num-1], strlen(txt_lists[get_list_num-1]) + 1, 0);
301     sprintf(p2p_path, "%s/%s", my_dirname, txt_lists[get_list_num-1]);
302     char sdbuf[512];
303     FILE *fs = fopen(p2p_path, "r");
304     if(fs == NULL)
305     {
306         printf("P2Pfile path error\n");
307         break;
308     }
309     bzero(sdbuf, 512);
310     int fs_block_sz;
311     sleep(1);
312     while((fs_block_sz = fread(sdbuf, sizeof(char), 512, fs)) > 0)
313     {
314         if(send(new_fd, sdbuf, fs_block_sz, 0) < 0)
315         {
316             printf("P2P can not send file\n");
317             break;
318         }
319         bzero(sdbuf, 512);
320         sleep(1);
321     }
322     printf("Success P2P file tranfer!!\n");
323 }
324 printf("-----<END P2P>-----\n");
325 close(new_fd);
326 close(sockfd);
327 closedir(dir_ptr);
328 return;
```

<P2P server 함수>

LINE 284 ~ 291 : p2p client에게 FILE LIST를 전송

LINE 292 : p2p client로부터 파일 번호를 수신 /q 일 경우에 p2p server를 종료 하기 위해 break

LINE 300 : file을 보내겠다는 신호를 p2p client에게 송신

LINE 302 : p2p client에게 원하는 파일 이름을 전송

LINE 305 : 해당 파일을 OPEN

LINE 314 ~ 324 : sdbuf만큼 파일을 읽고 저장 한후 sdbuf를 p2p client에 송신, 완료되면 다시 while() 첫줄로 올라가 FILE LIST를 보내줌

LINE 326 ~ 330 : 공유메모리 서버소켓, 클라이언트 소켓, dir_ptr을 닫음

2. 코드 분석

2) client.c

```
332 void p2p_client()
333 {
334     int count=0;
335     int sockfd;
336     struct sockaddr_in dest_addr;
337     char p2p_buff[MAXLINE];
338     char p2p_msg[MAXLINE];
339     char p2p_file_list[10][MAXLINE];
340     char p2p_path[512];
341     char revbuf[512];
342     char filename[256];
343     int p2p_rcv byte;
344     int p2p_server_port = atoi(p2p_port);
345     char path[256] = "/home/st2014146034/tcp/getfile";
346     pid_t pid2;
347     sockfd = socket(AF_INET, SOCK_STREAM, 0);
348     if(sockfd == -1)
349     {
350         perror("P2P-Client-socket() error lol!");
351         return;
352     }
353     else printf("P2P-Client-socket() sockfd is OK...\n");
354
355     dest_addr.sin_family = AF_INET;
356
357     dest_addr.sin_port = htons(p2p_server_port);
358     dest_addr.sin_addr.s_addr = inet_addr(p2p_ip);
359
360     memset(&(dest_addr.sin_zero), 0, 0);
361
362     if(connect(sockfd, (struct sockaddr *)&dest_addr, sizeof (struct sockaddr)) == -1)
363     {
364         perror("P2P-Client-connect() error lol!");
365         return;
366     }
367     else printf("P2P-Client-connect() is OK...\n\n");
```

<P2P client 함수>

LINE 334 ~ 345 : 변수 선언

LINE 345 : 받은 파일을 저장할 위치를 절대경로로 저장

LINE 347~ 367 : 서버와 통신하기 위한 sockt생성 및 connecting

```
370     pid2 = fork();
371
372     if(pid2 == 0)
373     {
374         while(1)
375         {
376             scanf("%s", p2p_buff);
377             send(sockfd, p2p_buff, strlen(p2p_buff)+1, 0);
378             if(strcmp(p2p_buff, "/q") == 0)
379             {
380                 close(sockfd);
381                 exit(1);
382             }
383         }
384     }
```

<자식 프로세스 >

LINE 372 ~ 384 : p2p server에게 메시지를 보내고 /q를 입력할경우 소켓을 닫고 자식 프로세스 종료

2. 코드 분석

2) client.c

```
385     else if(pid2 > 0)
386     {
387         while(1)
388         {
389             int i=0;
390             p2p_rcv_byte = recv(sockfd,p2p_msg,sizeof(p2p_msg),0);
391             if(strcmp(p2p_msg,"/q") == 0)
392             {
393                 break;
394             }
395             else if(strcmp(p2p_msg,"file") == 0)
396             {
397                 count =0;
398                 p2p_rcv_byte = recv(sockfd,filename,sizeof(filename),0);
399                 sprintf(p2p_path,"%s/%s",path,filename);
400                 FILE *fr = fopen(p2p_path,"a");
401
402                 if(fr == NULL)
403                     printf("P2P cannot be opened\n");
404                 else
405                 {
406                     bzero(revbuf, 512);
407                     int fr_block_sz = 0;
408                     while((fr_block_sz = recv(sockfd,revbuf, 512,0))>0)
409                     {
410                         int write_sz = fwrite(revbuf, sizeof(char),fr_block_sz,fr);
411                         if(write_sz < fr_block_sz)
412                         {
413                             perror("P2P file write failed");
414                         }
415                         bzero(revbuf,512);
416                         if(fr_block_sz==0 || fr_block_sz != 512)
417                         {
418                             break;
419                         }
420                     }
421                     if(fr_block_sz < 0)
422                     {
423                         if(errno == EAGAIN)
424                         {
425                             printf("P2P recv() timed out\n");
426                         }
427                         else
428                         {
429                             printf("P2P recv() failed due to error");
430                         }
431                     }
432                     printf("P2P Success received file!\n");
433                     fclose(fr);

```

<P2P client 함수>

<자식 프로세스 >

LINE 391 ~ 394 : /q를 입력하면 while문 빠져나간후 p2p 통신 종료, 다시 채팅방으로 진입

LINE 395 : p2p 서버로부터 file이라는 신호를 받으면 진입

LINE 398 ~ 400 : 서버로부터 원하는 파일의 이름은 받고 절대 경로로 저장 한후 파일을 생성 후 OPEN

LINE 408 ~ 420 : revbuf만큼 파일은 내용을 수신후 내용의 크기만큼 fr에 write, bzero()로 버퍼를 비워준다.

LINE 421 ~ 431 : recv 오류 전송

LINE 432 ~ 433 : p2p 통신 완료 메시지를 출력하고 fr포인터를 닫음

2. 코드 분석

2) client.c

```
434         }  
435     }  
436     else  
437     {  
438         printf("%s\n", p2p_msg);  
439     }  
440 } //end while  
441 } //end parents_process  
442 else  
443     perror("P2P fork() error!");  
444  
445 printf("-----<END P2P>-----\n");  
446 close(sockfd);  
447 return;  
448
```

<P2P client 함수>

LINE 446 : client 소켓을 닫음

3. 성능 수준

기본적인 파일 송수신은 전체적으로 다 구현하였다. 추가적인 부분은 클라이언트 2개 이상 채팅방에 있을 경우 P2P통신을 하기 위해 자신이 원하는 유저와 P2P통신이 가능하다. 또한 P2P 통신을 원하지 않는 경우도 있기 때문에 추가적인 special msg(/N or /Y)를 이용하여 수락 거절이 가능하다. 받고 싶은 파일을 원하는 만큼 다운로드 하고 /q 메시지를 통해 파일 송수신을 멈추고 채팅방으로 진입한다.

[느낀점 및 앞으로의 방향]

부족한 부분은 아직 아이디 3개를 제외한 다른 유저의 아이디 패스워드를 새로 생성할 수 있는 부분이 없어 그러한 기능을 추가시키는 부분과 ID와 PW의 제거도 있었으면 더욱 괜찮을 것 같다는 생각입니다. C언어뿐만 아니라 다양한 언어[python or c++]를 사용해서 서버와 클라이언트 사이의 소켓 통신을 구현 해보고 싶고, 또한 fork가 아닌 쓰레드로 하지 못했다는 점이 살짝 아쉬웠습니다. 추후 방학때는 쓰레드를 사용해서 팀 프로젝트와 동일한 과제를 둘이 도전해 볼 생각입니다.

이 팀프로젝트를 통해 메모리 사용과 할당에 대해 많은 어려움을 느꼈습니다. 수많은 메모리 종류에 대해 처음 알게되었고 그것을 어떤 용도로 어떻게 사용해야 하는지 많은 구글링을 통해서 조금이나마 알게 되었습니다. 스스로 노력해서 성공했다는 데에 뿌듯함과 자부심을 느낍니다.