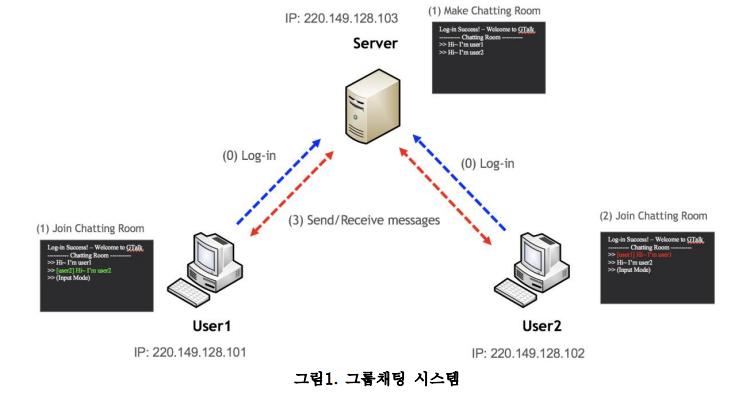
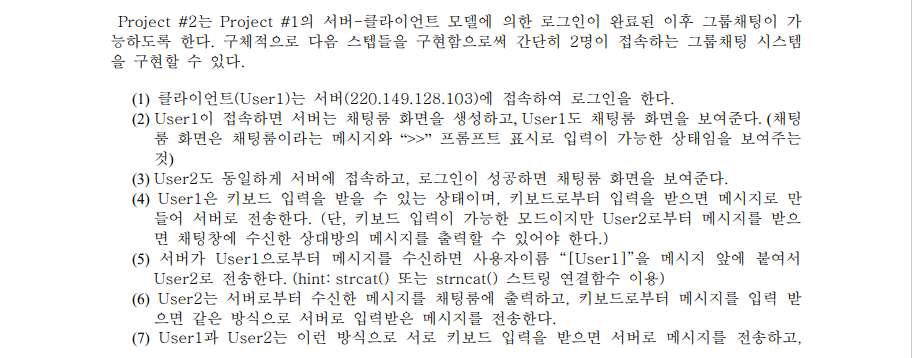
임베디드운영체제 기말 팀 프로젝트

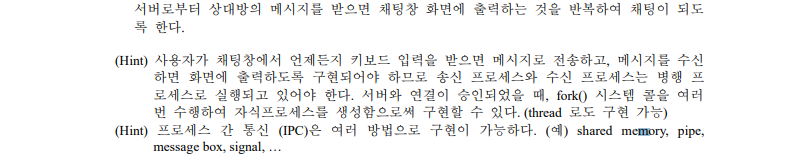
Team Project #2

2016146026 심재빈

2017146056 황경주





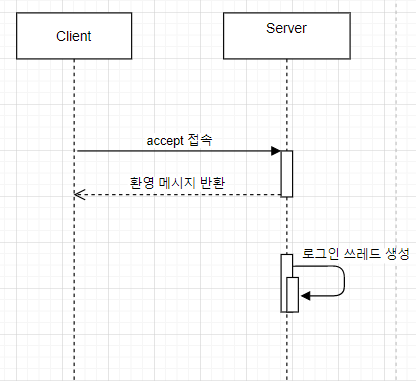


1. 설계

IPC : socket

병렬 프로세스 방식: thread(pthread)

우선 server를 동작시키고, client를 동작시키면, 가장 먼저 메인 프로세스는 다음과 같이 동작한다.



<1> 메인 프로세스 시퀀스 다이어그램

다음과 같이 accept()가 완료되면, 환영하는 메시지를 send()를 이용하여 반환해 client에서 출력하고, server는 로그인 쓰레드를 생성한다.

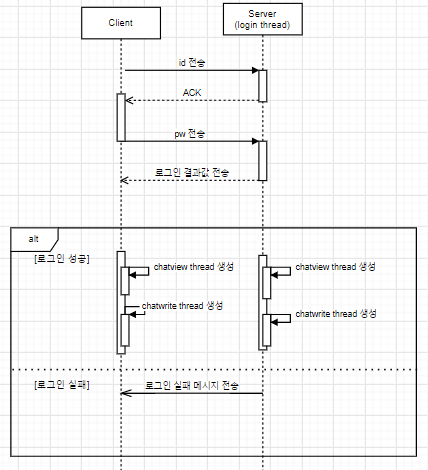
* 1. 로그인 설계
     1. 로그인을 수행하는 과정 중에서 다음과 같은 기능은 반드시 구현해야 한다.
* ID와 PW를 판별하여 어떤 유저가 로그인하였는지 찾는 기능
* 유저를 찾으면, 채팅 모드를 실행하는 것
* 한 클라이언트가 로그인을 실행 중일 때, 다른 클라이언트는 채팅 모드나 로그인 모드를 사용할 때 방해받지 않아야 한다.

3번째 기능이 수행되려면, 로그인 기능은 병렬로 실행되어야 하고, 이를 쓰레드로 구현하였다.

또한 accept()를 수행해 새로운 클라이언트가 들어올 때마다 쓰레드를 생성하여 메인 프로세스 밖에서 병렬적으로 동작하여야 하므로 로그인 동작 쓰레드의 개수는 클라이언트 개수와 같다.

또한 로그인 과정을 마치고나면, 후술할 채팅 모드에서의 쓰레드를 종료하기 위해 대기하고 있다가, 클라이언트가 종료되면 서버에서 해당 클라이언트의 소켓을 닫고, 클라이언트의 채팅 쓰레드들을 종료한다.

* + 1. Client와 login thread의 시퀀스 다이어그램은 다음과 같다.



<2> client와 login\_thread의 로그인까지의 시퀀스 다이어그램

* 1. 채팅 모드 설계

1. 채팅 모드를 구현하기 위해서는 서버는 다음과 같은 기능을 반드시 추가하여야 한다.

* 채팅에 대한 메시지를 받을 때 유저의 이름을 [ ] 안에 추가하여 메시지를 다른 클라이언트에게 전달하는 것

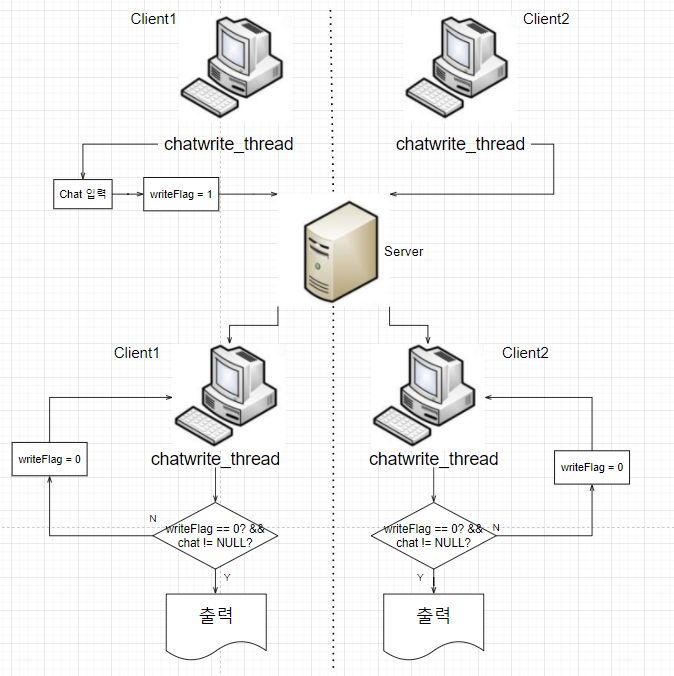
1. 채팅 모드를 구현하기 위해서는 클라이언트는 다음과 같은 기능을 반드시 추가하여야 한다.

* 채팅을 입력하면 서버로 전송하는 것
* 채팅을 입력할 수 있는 상황에서 언제든지 다른 클라이언트의 메시지를 송신받아 화면에 출력하는 것

로그인과정을 마친 클라이언트가 채팅 송신 쓰레드, 채팅 수신 쓰레드 총 2개를 생성하여 채팅 과정 동작 쓰레드의 개수는 로그인을 완료한 클라이언트 \* 2개와 같다.

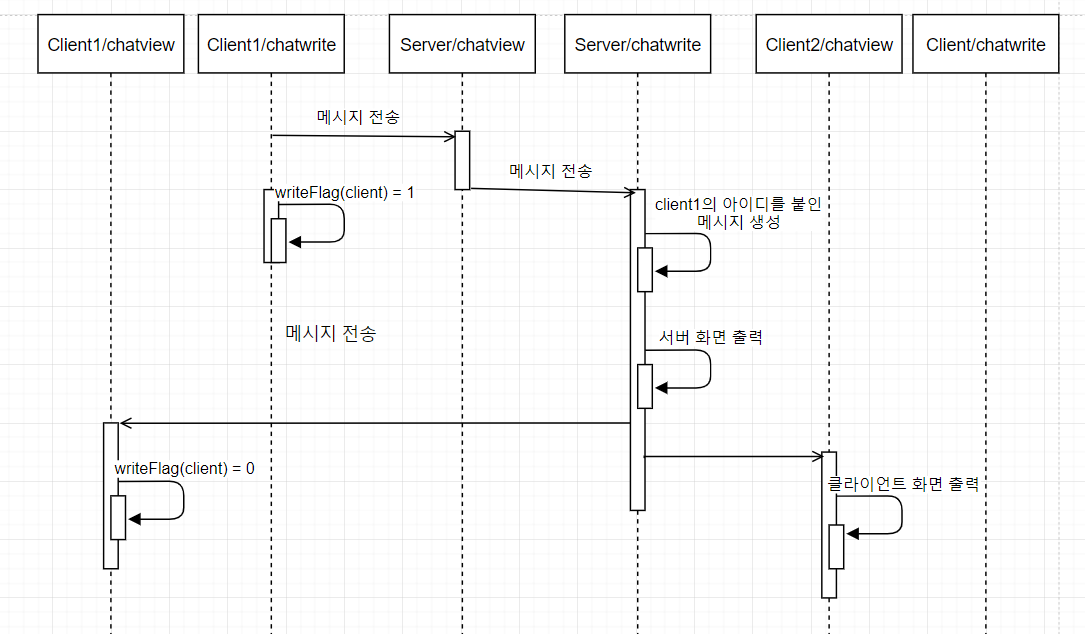
또한 2명 이상의 클라이언트가 접속하였을 때를 가정하여, 단순히 user1의 채팅을 user2뿐만 아니 라 user1을 제외한 모든 클라이언트가 확인할 수 있도록 다음과 같이 설계하였다.

채팅 모드에 대한 설계는 다음과 같다.



우선 0으로 초기화된 writeFlag라는 변수를 각 클라이언트마다 설정한다. 어떤 클라이언트가 채팅을 치면, 각 클라이언트 내에 존재하는 writeFlag라는 변수는 1로 변하고 채팅은 서버로 전송한다. 서버로 전송된 메시지는 Client1이 로그인한 계정의 아이디가 앞에 붙여져 현재 로그인된 모든 클라이언트에게 전송한다. 그렇게 전송받은 각 클라이언트들은 writeFlag가 0이고, 전송받은 메시지가 NULL이 아닌가에 대해 검사하고, 이를 만족한다면 화면에 표시하고, 그렇지 않으면 채팅을 무시하며 writeFlag를 0으로 설정한다.

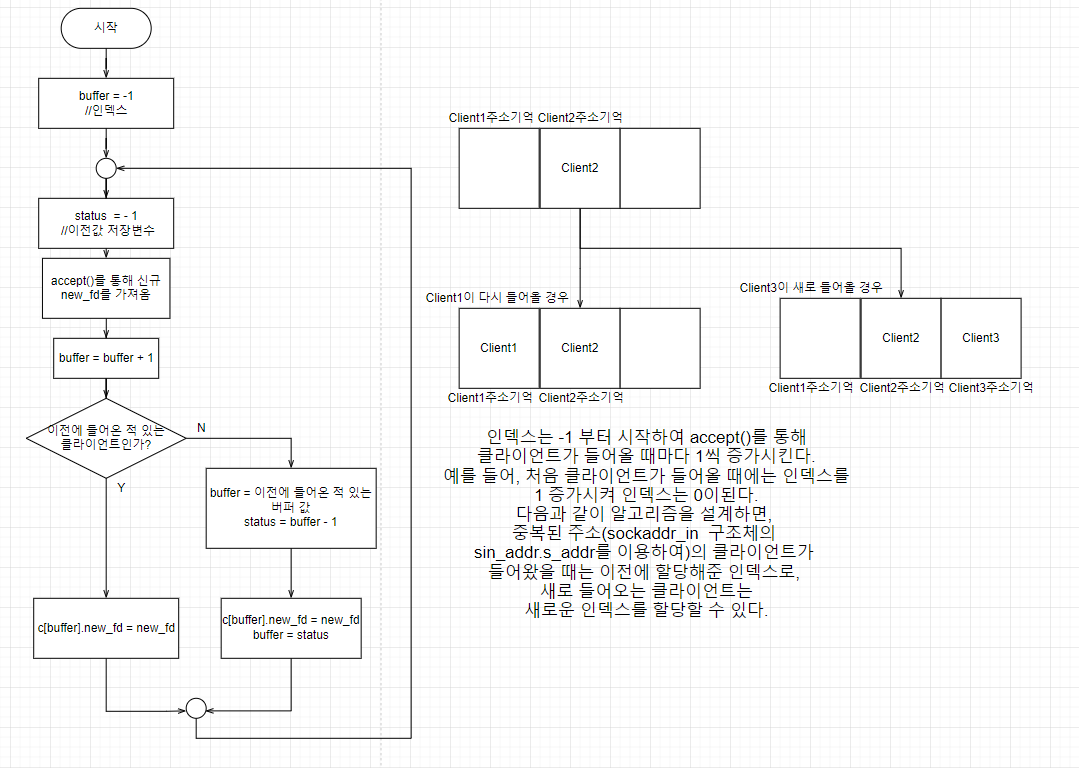
예를 들어 여러 개의 클라이언트들이 로그인 완료 상태이고, user1로 로그인된 Client1에서 채팅을 입력했다고 가정해보자. 이 때 Client1에서의 변수 writeFlag는 1로 설정되며, Client1에서 만든 채팅은 서버로 전송된다. 서버는 앞에 [user1]을 붙여 현재 로그인된 모든 클라이언트에 전송한다. 이 때 각 클라이언트들은 writeFlag가 0이고, 전송받은 메시지가 NULL이 아닌가에 대해 검사하고, writeFlag가 1로 설정된 Client1을 제외한 모든 클라이언트는 [user1]이 붙여진 메시지를 화면에 출력하고, Client1은 화면에 출력하는 대신, writeFlag을 0으로 설정한다. 이를 시퀀스 다이어그램을 표현하면 다음과 같다.



<3> client1이 메시지를 입력했을 때의 시퀀스 다이어그램

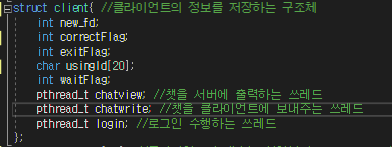
* 1. 클라이언트

Accept()를 통해 새로 들어오는 클라이언트는 아래의 플로우차트와 같이 클라이언트가 저장된다.



서버에서의 클라이언트는 구조체로 정의하였다.

클라이언트 구조체의 요소는 다음과 같다.



new\_fd : 해당 클라이언트의 accept() 수행 후 반환 값

correctFlag: 로그인 단계에서 ID, PW와 서버의 아이디/비밀번호를 비교한 후의 반환값

exitFlag: 후술할 나가는 기능이 수행될 때에 1, 그렇지 않으면 0으로 설정되는 플래그

usingId : correctFlag를 이용하여 얻어진 계정 아이디, 이 값으로 어떤 아이디로 로그인하였는지 판별한다.

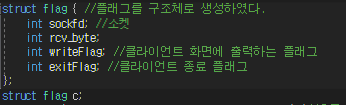
waitFlag: 1이면 채팅모드가 비활성화, 0이면 채팅모드가 활성화

chatview: 챗을 서버에 출력하는 쓰레드

charwrite : 챗을 클라이언트에 보내는 쓰레드

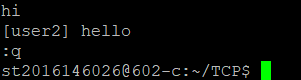
login : 로그인을 수행하는 쓰레드

또한 클라이언트에서의 구조체도 존재하는데 B에서의 writeFlag, :q가 눌릴 때 즉시 클라이언트를 종료하기위한 exitFlag가 존재한다.

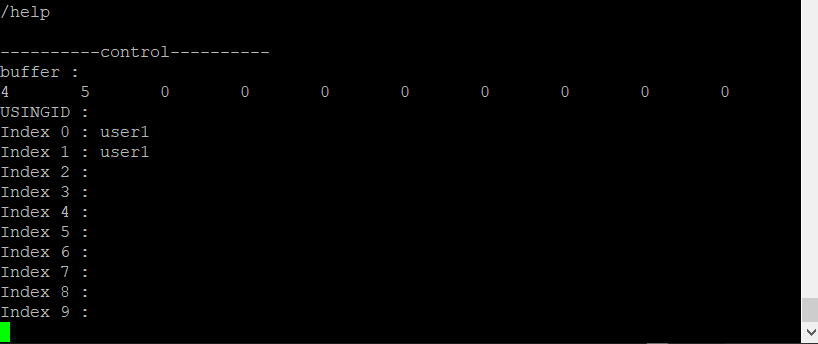


* 1. 부가기능

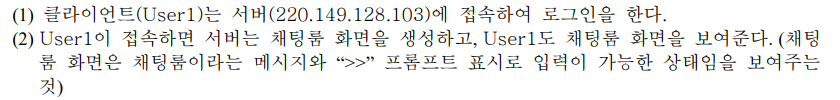
1. 클라이언트에서 채팅을 하는 도중 :q를 입력하면 클라이언트는 즉시 종료된다.

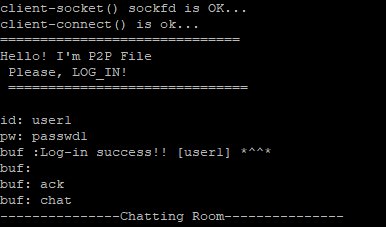


1. 서버에서 /help를 입력하면 현재 들어온 클라이언트의 accept()를 수행한 int형 자료값, 해당 클라이언트가 로그인한 아이디가 나타난다.

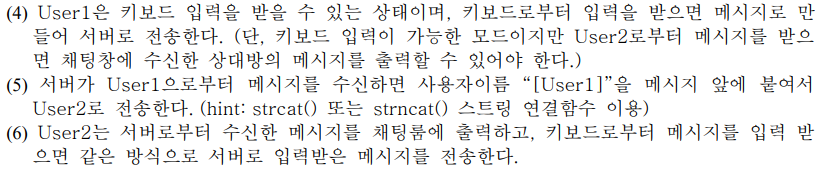


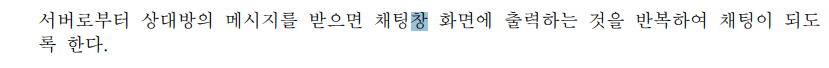
1. 구현
   * 1. 결과 화면

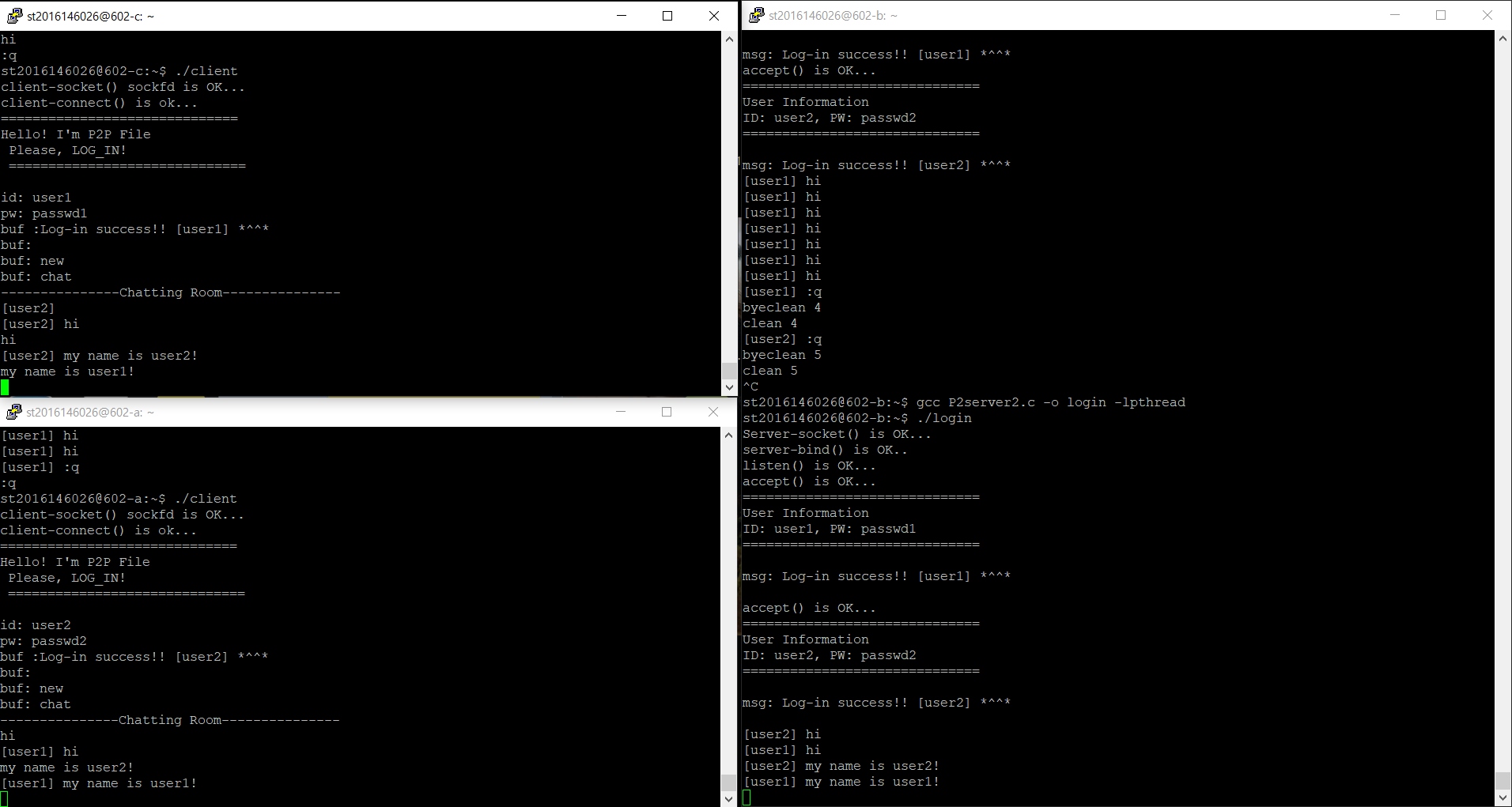


]

다음과 같이 user1으로 로그인을 성공하면, Chatting Room이 출력된다.

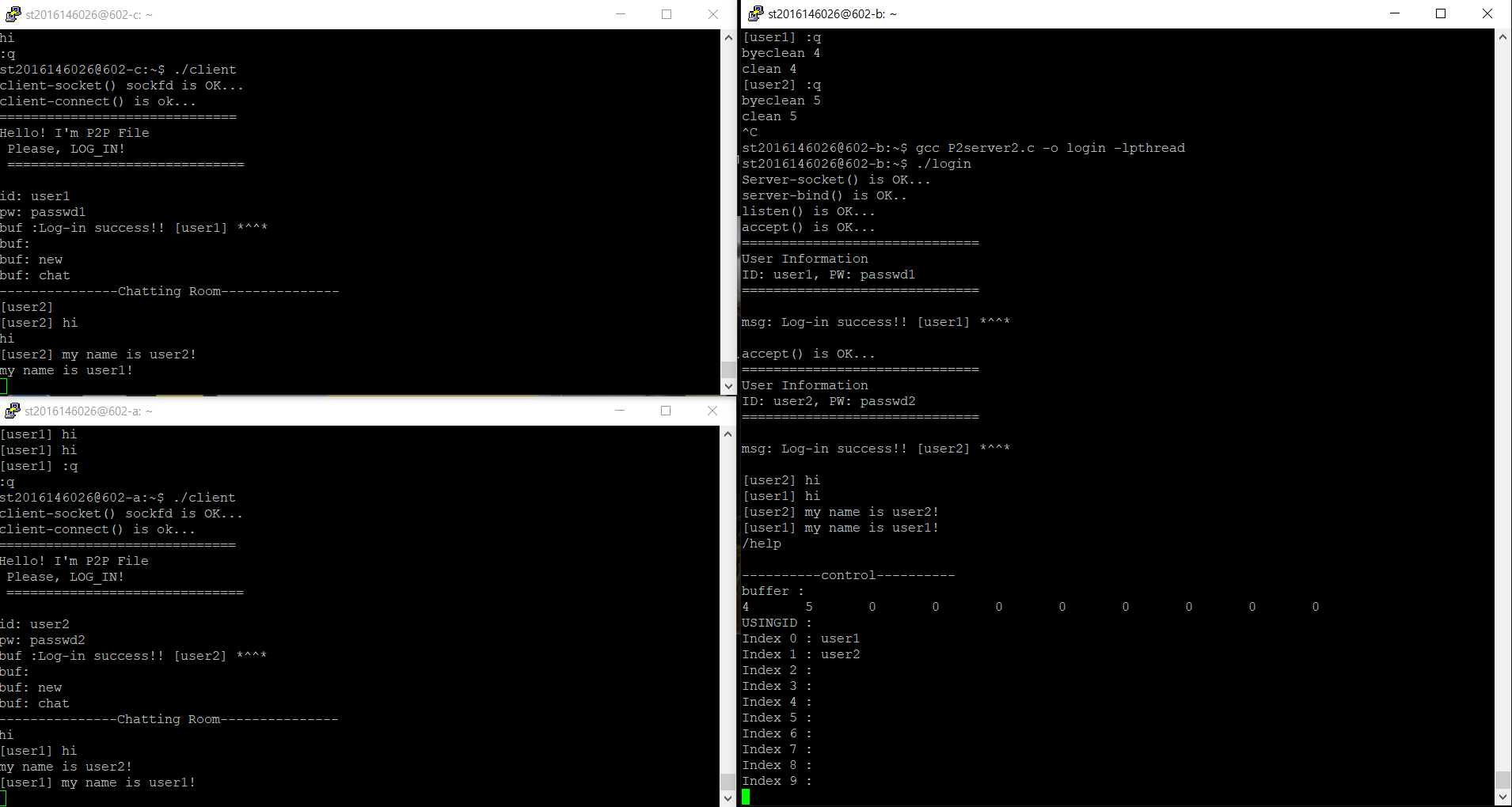




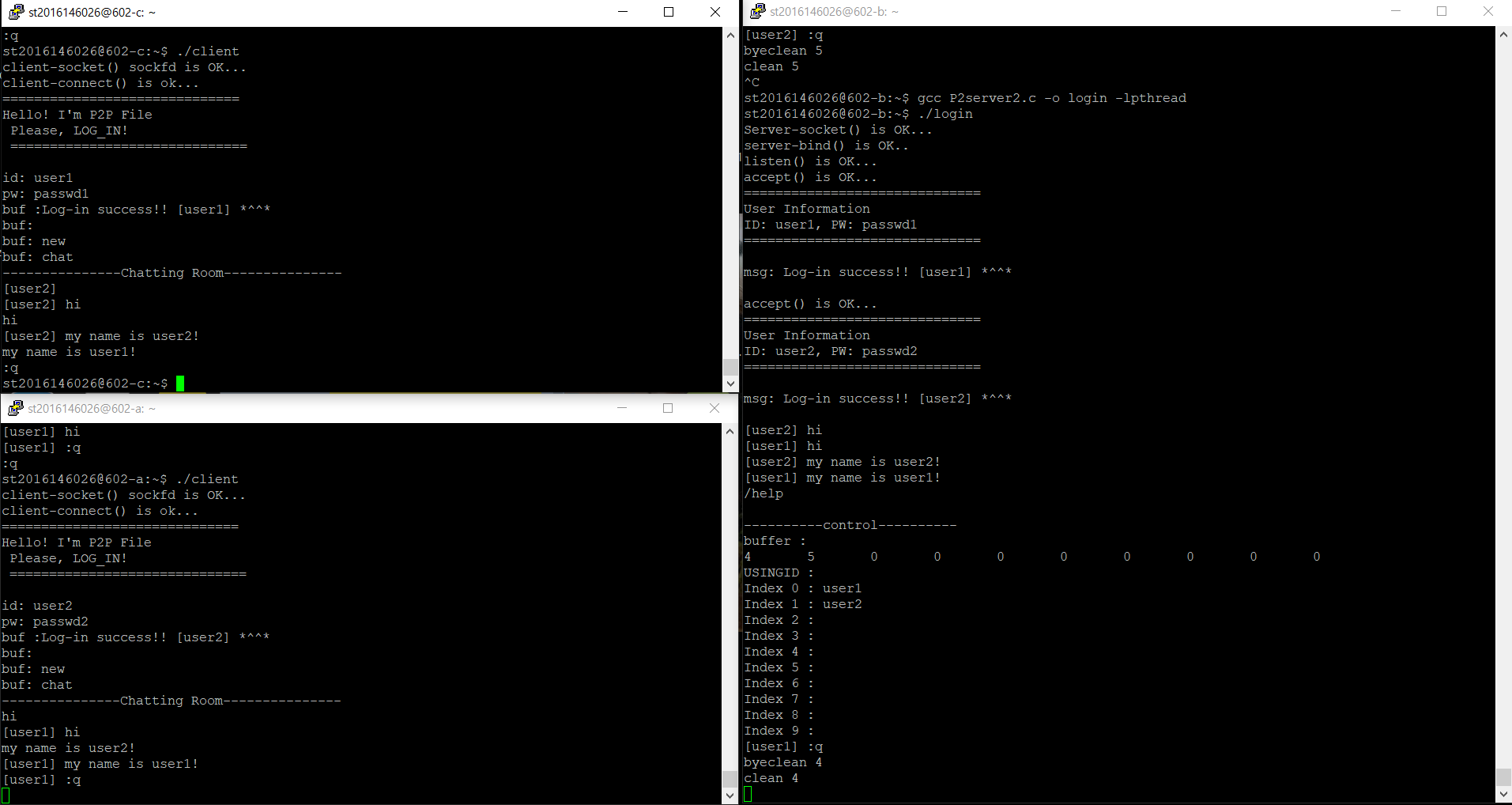


다음과 같이 user1이 채팅을 치면, user2가 [user1]이 앞에 붙은 채팅을 받는 것을 확인할 수 있으며, user2가 채팅을 하면 user1이 [user2]가 앞에 붙은 채팅을 받는 것을 확인할 수 있다. 또한 옆을 보면, 서버는 user1의 채팅과 user2의 채팅을 모두 받고 있는 것을 확인할 수 있다.

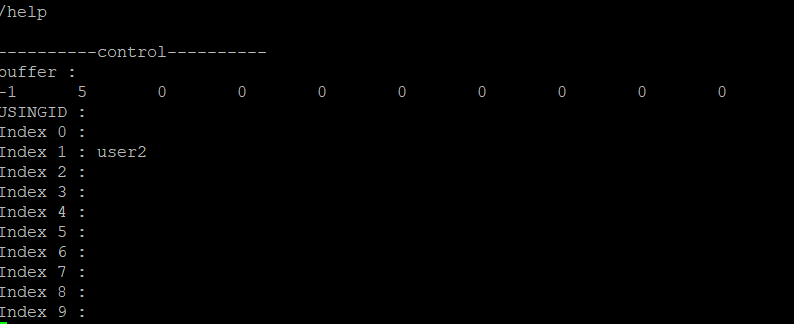
* 부가기능



또한 서버에서 /help를 누르면 들어온 클라이언트들의 new\_fd 값, 로그인한 아이디가 나타난다

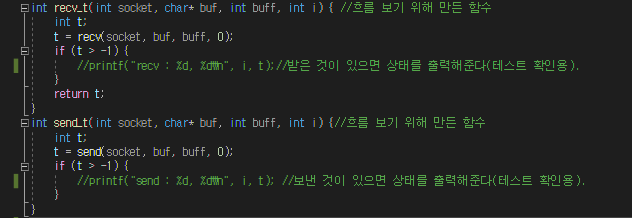


:q를 누르면 클라이언트가 즉시 종료됨을 확인할 수 있다.



다음과 같이 user1이 사라지고 buffer에는 -1이 들어가있는 것을 확인할 수 있다.

* + 1. 모듈
* Recv\_t, send\_t



Int i에 대한 부분은 send()와 recv()를 채팅 모드가 되기 전까지 동기식으로 한번씩 주고 받기 위해 구현한 것을 확인하기 위한 것으로 사실상 recv\_t 함수는 recv()와 같고, send\_t 함수는 send()와 같다.

또한 주석에 i=(숫자)의 부분은 send와 recv의 순서를 뜻하며, server프로그램에서 send가 i값이 홀수고, recv에서의 i값이 짝수일 경우 client 프로그램에서는 recv에서의 i값이 홀수이고, send에서의 i값이 짝수이다.

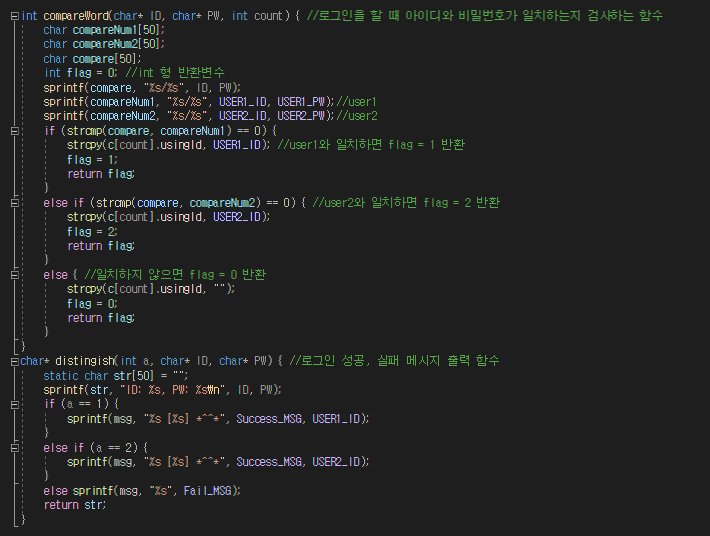
* Login\_thread



왼쪽은 server, 오른쪽은 client 에서의 login 구현부이다.

서버에서는 로그인 성공시 chat을 전송하고, 실패시 not을 전송한다.

클라이언트에서는 chat을 받을시 채팅모드 쓰레드가 생성된다.

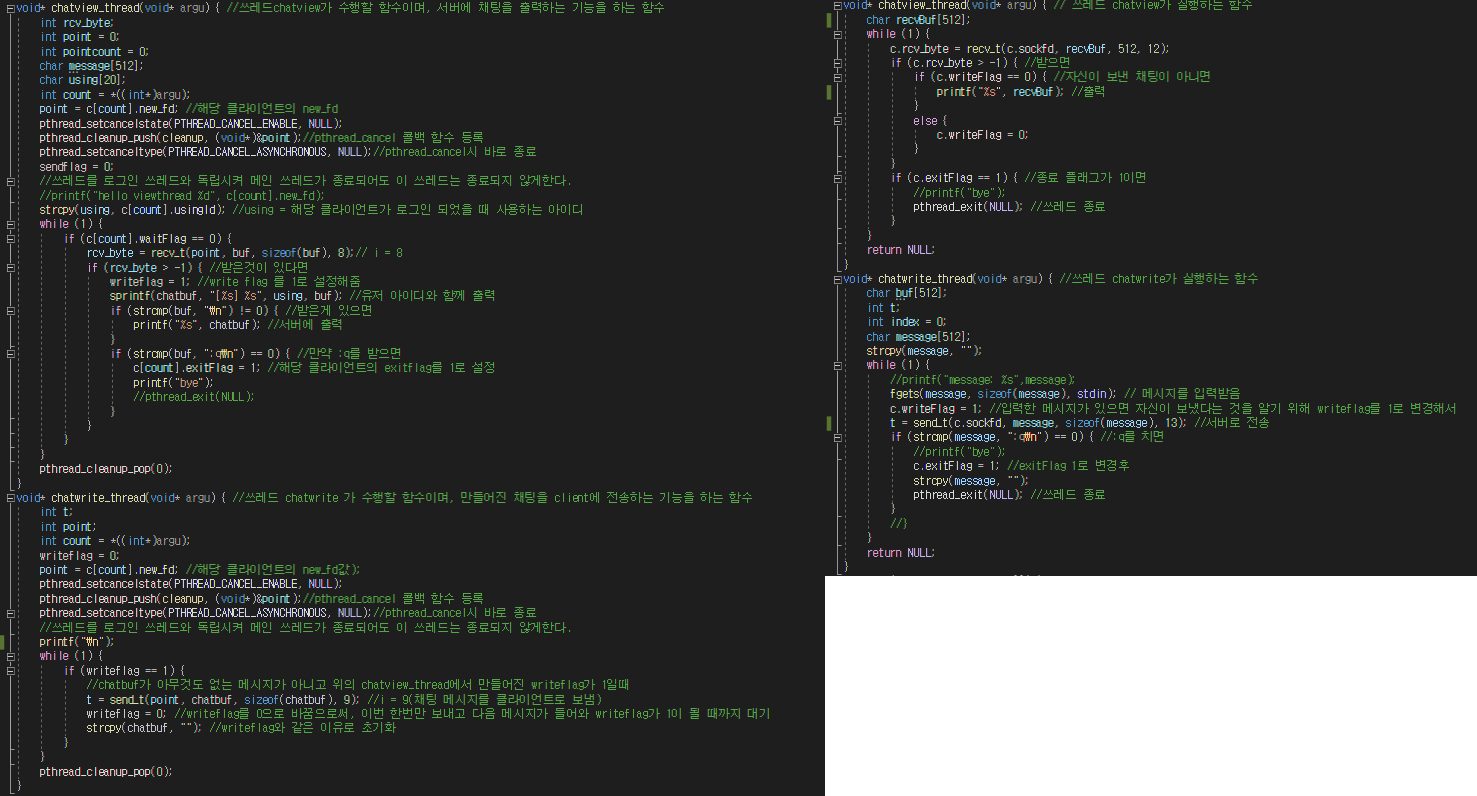


서버에서의 compareWord, distinguish함수는 다음과 같은 기능을 한다

compareWord : id와 pw값이 맞는지 판별하는 함수로 correctFlag를 결정짓는다.

distinguish: compareWord에 따라 로그인 축하 메시지를 생성하는 함수

* Chatview, chatwrite



왼쪽은 server, 오른쪽은 client 에서의 채팅모드 구현부이다.

설계 과정에서의 writeflag는 c.writeflag이다.

client에서 chatwrite\_thread에서 먼저 입력을 받아 server로 전송함과 동시에 c.writeflag를 1로 설정하며, 서버에서는 메시지를 받으면 서버에 출력하고, 전역변수 writeflag를 1로 설정하여 chatwrite가 반응할 수 있도록 하고, 자신의 클라이언트에 보내면 writeflag를 0으로 설정하여 send를 한번만 보내도록 하였다. 서버에서 제작한 메시지를 로그인이 완료된 클라이언트들이 받게 되면, writeflag가 0이면 화면에 출력하고, 1이면 화면에 출력하는 대신 writeflag를 0으로 설정한다.

Pthread를 이용하는데, gcc 컴파일러를 이용하여 컴파일하려면 다음과 같은 명령어를 사용하여야 한다.

gcc [파일 명] -o [실행 파일] -lpthread

1. 소스 코드

Server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <pthread.h> //pthread 헤더

#define SERV\_IP "220.149.128.103" //서버 IP

#define SERV\_PORT 4534 //서버 PORT

#define BACKLOG 10

#define INIT\_MSG "==============================\nHello! I'm P2P File \n Please, LOG\_IN!\n ==============================\n"

#define Success\_MSG "Log-in success!!"

#define Fail\_MSG "Log-in fail: Incorrect password..."

#define USER1\_ID "user1"

#define USER1\_PW "passwd1"

#define USER2\_ID "user2"

#define USER2\_PW "passwd2"

#define UI1 "==============================\nUser Information\n"

#define UI2 "==============================\n"

char msg[512]; //로그인 성공 메시지

char buf[512] = ""; //채팅 내용을 char\* 형태로 저장한 변수

char chatbuf[512]; //채팅 내용에 클라이언트가 로그인한 만든 변수

int writeflag = 0; // chatbuf를 만들고 다른 클라이언트에 보내주기 위해 만든 flag(0이면 준비 안됨, 1이면 준비됨)

int sendflag = 0;

char sendmessage[512];

int new\_fd;

//char user[50];

int buffer = -1; //모든 쓰레드에서 관리되는 INDEX 값(main()문 포함)

int indexfd = -1; // 3번의 [FILE user2]등을 수행할 때 뒤의 user2가 어떤 클라이언트에서 로그인 되었는지 판별하기 위해 설정한 변수

int add[10]; //main문에서 이용할 their\_addr.sin\_addr.s\_addr을 이용하여

int retry[10] = { 0,0,0,0,0,0,0,0,0,0 };

//새 IP가 들어온다면 새 INDEX를 발급하여 로그인을 수행하고, 그렇지 않으면 IP가 등록된 INDEX로 로그인을 수행한다..

pthread\_t controlbox; //main문에서 /help를 치면 현재 서버의 정보를 출력해주는 쓰레드

int recv\_t(int socket, char\* buf, int buff, int i) { //흐름 보기 위해 만든 함수

int t;

t = recv(socket, buf, buff, 0);

if (t > -1) {

//printf("recv : %d, %d\n", i, t);//받은 것이 있으면 상태를 출력해준다(테스트 확인용).

}

return t;

}

int send\_t(int socket, char\* buf, int buff, int i) {//흐름 보기 위해 만든 함수

int t;

t = send(socket, buf, buff, 0);

if (t > -1) {

//printf("send : %d, %d\n", i, t); //보낸 것이 있으면 상태를 출력해준다(테스트 확인용).

}

}

struct client{ //클라이언트의 정보를 저장하는 구조체

int new\_fd;

int correctFlag;

int exitFlag;

char usingId[20];

int waitFlag;

pthread\_t chatview; //챗을 서버에 출력하는 쓰레드

pthread\_t chatwrite; //챗을 클라이언트에 보내주는 쓰레드

pthread\_t login; //로그인 수행하는 쓰레드

};

struct client c[10]; //클라이언트의 개수는 선언부의 Thread, add와 동일한 숫자여야한다.

//최대 10개까지 소켓 가능

void cleanup(void\* argu) {

int socket = \*((int\*)argu);

printf("clean %d\n",socket);

}

void\* control(void\* argu) { //control 쓰레드가 수행할 함수

char message[50];

while (1) {

fgets(message, sizeof(message), stdin); // /help라는 문자가 들어오면

if (strcmp(message, "/help\n") == 0) {

printf("\n----------control----------\n");

printf("buffer :\n"); //현재 서버에 들어온 클라이언트의 new\_fd

for (int i = 0; i < 10; i++) {

printf("%d\t", c[i].new\_fd);

}

printf("\n");

printf("USINGID :\n"); //클라이언트가 로그인한 아이디

for (int i = 0; i < 10; i++) {

printf("Index %d : %s\n", i, c[i].usingId);

}

//를 출력한다

}

}

pthread\_exit(NULL);

}

int compareWord(char\* ID, char\* PW, int count) { //로그인을 할 때 아이디와 비밀번호가 일치하는지 검사하는 함수

char compareNum1[50];

char compareNum2[50];

char compare[50];

int flag = 0; //int 형 반환변수

sprintf(compare, "%s/%s", ID, PW);

sprintf(compareNum1, "%s/%s", USER1\_ID, USER1\_PW);//user1

sprintf(compareNum2, "%s/%s", USER2\_ID, USER2\_PW);//user2

if (strcmp(compare, compareNum1) == 0) {

strcpy(c[count].usingId, USER1\_ID); //user1와 일치하면 flag = 1 반환

flag = 1;

return flag;

}

else if (strcmp(compare, compareNum2) == 0) { //user2와 일치하면 flag = 2 반환

strcpy(c[count].usingId, USER2\_ID);

flag = 2;

return flag;

}

else { //일치하지 않으면 flag = 0 반환

strcpy(c[count].usingId, "");

flag = 0;

return flag;

}

}

char\* distingish(int a, char\* ID, char\* PW) { //로그인 성공, 실패 메시지 출력 함수

static char str[50] = "";

sprintf(str, "ID: %s, PW: %s\n", ID, PW);

if (a == 1) {

sprintf(msg, "%s [%s] \*^^\*", Success\_MSG, USER1\_ID);

}

else if (a == 2) {

sprintf(msg, "%s [%s] \*^^\*", Success\_MSG, USER2\_ID);

}

else sprintf(msg, "%s", Fail\_MSG);

return str;

}

void\* chatview\_thread(void\* argu) { //쓰레드chatview가 수행할 함수이며, 서버에 채팅을 출력하는 기능을 하는 함수

int rcv\_byte;

int point = 0;

int pointcount = 0;

char message[512];

char using[20];

int count = \*((int\*)argu);

point = c[count].new\_fd; //해당 클라이언트의 new\_fd

pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL);

pthread\_cleanup\_push(cleanup, (void\*)&point);//pthread\_cancel 콜백 함수 등록

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS, NULL);//pthread\_cancel시 바로 종료

sendflag = 0;

//쓰레드를 로그인 쓰레드와 독립시켜 메인 쓰레드가 종료되어도 이 쓰레드는 종료되지 않게한다.

//printf("hello viewthread %d", c[count].new\_fd);

strcpy(using, c[count].usingId); //using = 해당 클라이언트가 로그인 되었을 때 사용하는 아이디

while (1) {

if (c[count].waitFlag == 0) {

rcv\_byte = recv\_t(point, buf, sizeof(buf), 8);// i = 8

if (rcv\_byte > -1) { //받은것이 있다면

writeflag = 1; //write flag 를 1로 설정해줌

sprintf(chatbuf, "[%s] %s", using, buf); //유저 아이디와 함께 출력

if (strcmp(buf, "\n") != 0) { //받은게 있으면

printf("%s", chatbuf); //서버에 출력

}

if (strcmp(buf, ":q\n") == 0) { //만약 :q를 받으면

c[count].exitFlag = 1; //해당 클라이언트의 exitflag를 1로 설정

printf("bye");

//pthread\_exit(NULL);

}

}

}

}

pthread\_cleanup\_pop(0);

}

void\* chatwrite\_thread(void\* argu) { //쓰레드 chatwrite 가 수행할 함수이며, 만들어진 채팅을 client에 전송하는 기능을 하는 함수

int t;

int point;

int count = \*((int\*)argu);

writeflag = 0;

point = c[count].new\_fd; //해당 클라이언트의 new\_fd값);

pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL);

pthread\_cleanup\_push(cleanup, (void\*)&point);//pthread\_cancel 콜백 함수 등록

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS, NULL);//pthread\_cancel시 바로 종료

//쓰레드를 로그인 쓰레드와 독립시켜 메인 쓰레드가 종료되어도 이 쓰레드는 종료되지 않게한다.

printf("\n");

while (1) {

if (writeflag == 1) {

//chatbuf가 아무것도 없는 메시지가 아니고 위의 chatview\_thread에서 만들어진 writeflag가 1일때

t = send\_t(point, chatbuf, sizeof(chatbuf), 9); //i = 9(채팅 메시지를 클라이언트로 보냄)

writeflag = 0; //writeflag를 0으로 바꿈으로써, 이번 한번만 보내고 다음 메시지가 들어와 writeflag가 1이 될 때까지 대기

strcpy(chatbuf, ""); //writeflag와 같은 이유로 초기화

}

}

pthread\_cleanup\_pop(0);

}

void\* login\_thread(void\* argu) { //로그인함수

int rcv\_byte;

char idbuf[20];//id받는 변수

char pwbuf[20];//passwd받는 변수

char buf[512];

char UI[512];

char\* UIstr;

int val = \*((int\*)argu);

pthread\_cleanup\_push(cleanup, (void\*)&val);//pthread\_cancel 콜백 함수 등록

pthread\_setcancelstate(PTHREAD\_CANCEL\_ENABLE, NULL);

pthread\_setcanceltype(PTHREAD\_CANCEL\_ASYNCHRONOUS, NULL);//pthread\_cancel시 바로 종료

c[val].exitFlag = 0; //exitFlag 초기화 = :q가 눌릴때까지 exitFlag는 반드시 0이어야 한다.

//printf("%d", c[val].waitFlag);

rcv\_byte = recv\_t(c[val].new\_fd, idbuf, sizeof(idbuf), 2);//i = 2 id를 받음

send\_t(c[val].new\_fd, "idack", 6, 3);// i = 3 확인 신호

rcv\_byte = recv\_t(c[val].new\_fd, pwbuf, sizeof(pwbuf), 4);//i = 4 pw를 받음

c[val].correctFlag = compareWord(idbuf, pwbuf, val);

UIstr = distingish(c[val].correctFlag, idbuf, pwbuf);

sprintf(UI, "%s%s%s", UI1, UIstr, UI2);

printf("%s\n", UI);

printf("msg: %s\n", msg);

send\_t(c[val].new\_fd, msg, strlen(msg) + 1, 5);//i = 5 확인 신호대신 로그인 성공 메시지

rcv\_byte = recv\_t(c[val].new\_fd, buf, sizeof(buf), 6);//i = 6 client에서 로그인 완료 확인 신호를 받음

strcpy(buf, "");

//printf("%d", c[val].waitFlag);

if (c[val].correctFlag > 0) { //로그인 성공시

send\_t(c[val].new\_fd, "new", 4, 70); // i = 7 확인 신호를 보냄

rcv\_byte = recv\_t(c[val].new\_fd, buf, sizeof(buf), 80); //i = 8 확인신호를 받음

send\_t(c[val].new\_fd, "chat", 4, 90);// i = 9 채팅모드 실행

pthread\_create(&c[val].chatview, NULL, &chatview\_thread, (void\*)&val); //해당 클라이언트의 채팅 쓰레드 생성

pthread\_create(&c[val].chatwrite, NULL, &chatwrite\_thread, (void\*)&val);

c[val].waitFlag = 0;

}

else {

send\_t(c[val].new\_fd, "new", 4, 170); // i = 7 확인신호를 보냄

rcv\_byte = recv\_t(c[val].new\_fd, buf, sizeof(buf), 180); //확인신호를 받음

send\_t(c[val].new\_fd, "not", 4, 190); // i = 9 로그인 실패 신호 전송

}

while (1) {

if (c[val].exitFlag == 1) { //exitflag, 즉 종료할때

strcpy(c[val].usingId, ""); //로그인을 취소시키고

close(c[val].new\_fd); //해당 클라이언트의 new\_fd소켓을 닫고

pthread\_cancel(c[val].chatview);

pthread\_cancel(c[val].chatwrite);

c[val].new\_fd = -1; //해당하는 new\_fd값을 -1로 변환

c[val].waitFlag = 1; //또한 accept 대기 플래그를 1로 만들어줌

c[val].exitFlag = 0;

}

//wait

}

pthread\_cleanup\_pop(0);

}

int main(void) //accept문

{

int sockfd;

int flag = 0;

struct sockaddr\_in my\_addr;

struct sockaddr\_in their\_addr;

unsigned int sin\_size;

int th;

int val = 1;

int status = -1;//중복된 주소번호일때

//char buf[512];

//int rcv\_byte;

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

perror("Server-socket() error lol!");

exit(1);

}

else printf("Server-socket() is OK...\n");

my\_addr.sin\_family = AF\_INET;

my\_addr.sin\_port = htons(SERV\_PORT);

my\_addr.sin\_addr.s\_addr = INADDR\_ANY;

memset(&(my\_addr.sin\_zero), 0, 8);

if (setsockopt(sockfd, SOL\_SOCKET, SO\_REUSEADDR, (char\*)&val, sizeof(val)) < 0) {

perror("setsockopt");

close(sockfd);

return -1;

}

if (bind(sockfd, (struct sockaddr\*)&my\_addr, sizeof(struct sockaddr)) == -1)

{

perror("server-bind() error lol!");

close(sockfd);

exit(1);

}

else printf("server-bind() is OK..\n");

if (listen(sockfd, BACKLOG) == -1)

{

perror("listen() is error lol!");

exit(1);

}

else printf("listen() is OK...\n");

pthread\_create(&controlbox, NULL, &control, (void\*)&buffer); // accept전,클라이언트 상태 정보 확인하는 쓰레드 생성

while (1) {

//새로운 클라이언트가 들어왔을 때 새로운 인덱스를 할당하고

//이전 accept된 값이 한번 이상 재접속한 클라이언트일 경우

//이전에 accept되었을 때 할당받은 인덱스를 받아야하는데,

//만약 이때 buffer 할당받은 인덱스로 돌아가면, buffer + 1이 새로운 인덱스가 아니라면,

//다음 accept를 수행할 때 해당 인덱스를 가진 클라이언트와 충돌한다.

//이를 방지하기 위해 status는 새로운 인덱스를 잠시 저장한다.

if (status != -1) {

buffer = status;

status = -1; //status 초기화

}

sin\_size = sizeof(struct sockaddr\_in);

new\_fd = accept(sockfd, (struct sockaddr\*)&their\_addr, &sin\_size);

buffer++;//제일 처음에는 buffer가 -1이므로 -1 + 1 = 0부터 시작한다.

//printf("their\_addr : %d, new\_fd : %d\n", their\_addr.sin\_addr.s\_addr, their\_addr);

//printf("new\_fd %d", new\_fd);

for (int i = 0; i < 10; i++) {

if (add[i] == their\_addr.sin\_addr.s\_addr) { //클라이언트 재접속 여부 판별(재접속일 경우)

status = buffer-1; //카운트값저장(이때 buffer는 accept가 되면 필수적으로 1을 더하기 때문에 1을 빼서 저장하여야 한다)

pthread\_cancel(c[i].login);

buffer = i;

c[buffer].new\_fd = new\_fd; //new\_fd를 이전에 할당받은 인덱스에 저장한다.

break;

}

if (i == buffer) { //새로 접속일 경우

c[buffer].new\_fd = new\_fd;

break;

}

}

add[buffer] = their\_addr.sin\_addr.s\_addr;

//printf("count = %d\n", c[buffer].new\_fd);

if (c[buffer].new\_fd == -1) { //accept판별

perror("accept() is error lol!...");

exit(1);

}

else printf("accept() is OK...\n");

//printf("buffer = %d\n",buffer);

send\_t(c[buffer].new\_fd, INIT\_MSG, strlen(INIT\_MSG) + 1, 1); //i = 1 환영하는 메시지 전송

th = pthread\_create(&c[buffer].login, NULL, &login\_thread, (void\*)&buffer); //로그인 쓰레드 생성

if (th < 0) {

printf("error");

}

}

}

Client.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <dirent.h>

#include <pthread.h> //pthread 헤더

#define SERV\_IP "220.149.128.103"

#define SERV\_PORT 4534

pthread\_t chatview;//채팅을 출력 위해 만든 쓰레드

pthread\_t chatwrite;//채팅을 입력해 서버로 전송하기 위해 만든 쓰레드

char sendmessage[100]; //3번 문제에서 서버에서 받은 포트와 IP의 정보값

struct flag { //플래그를 구조체로 생성하였다.

int sockfd; //소켓

int rcv\_byte;

int writeFlag; //클라이언트 화면에 출력하는 플래그

int exitFlag; //클라이언트 종료 플래그

};

struct flag c;

int recv\_t(int socket, char\* buf, int buff, int i) {//흐름 보기 위해 만든 함수

int t;

t = recv(socket, buf, buff, 0);

return t;

}

int send\_t(int socket, char\* buf, int buff, int i) {//흐름 보기 위해 만든 함수

int t;

t = send(socket, buf, buff, 0);

if (t > -1) {

//printf("send : %d, %d\n", i,t);

}

return t;

}

void\* chatview\_thread(void\* argu) { // 쓰레드 chatview가 실행하는 함수

char recvBuf[512];

while (1) {

c.rcv\_byte = recv\_t(c.sockfd, recvBuf, 512, 12);

if (c.rcv\_byte > -1) { //받으면

if (c.writeFlag == 0) { //자신이 보낸 채팅이 아니면

printf("%s", recvBuf); //출력

}

else {

c.writeFlag = 0;

}

}

if (c.exitFlag == 1) { //종료 플래그가 1이면

//printf("bye");

pthread\_exit(NULL); //쓰레드 종료

}

}

return NULL;

}

void\* chatwrite\_thread(void\* argu) { //쓰레드 chatwrite가 실행하는 함수

char buf[512];

int t;

int index = 0;

char message[512];

strcpy(message, "");

while (1) {

//printf("message: %s",message);

fgets(message, sizeof(message), stdin); // 메시지를 입력받음

c.writeFlag = 1; //입력한 메시지가 있으면 자신이 보냈다는 것을 알기 위해 writeflag를 1로 변경해서

t = send\_t(c.sockfd, message, sizeof(message), 13); //서버로 전송

if (strcmp(message, ":q\n") == 0) { //:q를 치면

//printf("bye");

c.exitFlag = 1; //exitFlag 1로 변경후

strcpy(message, "");

pthread\_exit(NULL); //쓰레드 종료

}

//}

}

return NULL;

}

int main(int argc, char\* argv[]) {

struct sockaddr\_in dest\_addr;

char buf[512];

char buf2[512];

char id[20];

char pw[20];

char chatReceive[512];

c.writeFlag = 0;

c.exitFlag = 0;

c.sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

c.exitFlag = 0;

if (c.sockfd == -1)

{

perror("client-socket() error lol!");

exit(1);

}

else printf("client-socket() sockfd is OK...\n");

dest\_addr.sin\_family = AF\_INET;

dest\_addr.sin\_port = htons(SERV\_PORT);

dest\_addr.sin\_addr.s\_addr = inet\_addr(SERV\_IP);

memset(&(dest\_addr.sin\_zero), 0, 8);

if (connect(c.sockfd, (struct sockaddr\*) & dest\_addr, sizeof(struct sockaddr)) == -1) {

perror("client-connect() error lol");

exit(1);

}

else printf("client-connect() is ok...\n");

c.rcv\_byte = recv\_t(c.sockfd, buf, sizeof(buf), 1);// i = 1 서버에서 환영하는 메시지 받음

printf("%s\n", buf);

printf("id: ");

scanf("%s", id);

send\_t(c.sockfd, id, strlen(id) + 1, 2);//i = 2 id를 전송

c.rcv\_byte = recv\_t(c.sockfd, buf, sizeof(buf), 3);//i = 3 확인 받음

printf("pw: ");

scanf("%s", pw);

send\_t(c.sockfd, pw, strlen(pw) + 1, 4);//i = 4 pw를 전송

c.rcv\_byte = recv\_t(c.sockfd, buf, sizeof(buf), 5);//i = 5 서버에서 로그인 메시지 받음

printf("buf :%s\n", buf); //로그인 메시지 출력

send\_t(c.sockfd, "ack", 4, 6);//i = 6 로그인 완료 확인 신호를 보냄

//port

strcpy(buf, "");

printf("buf: %s\n", buf);

c.rcv\_byte = recv\_t(c.sockfd, buf, sizeof(buf), 7);//7 로그인 서버에서 확인 신호를 받음

printf("buf: %s\n", buf);

send\_t(c.sockfd, "ack", 4, 8);//i = 8

//chat

c.rcv\_byte = recv\_t(c.sockfd, buf2, sizeof(buf2), 11);//i = 9

printf("buf: %s\n", buf2);

if (strcmp(buf2, "chat") == 0) { //buf2가

printf("---------------Chatting Room---------------\n");

pthread\_create(&chatview, NULL, &chatview\_thread, NULL); //채팅 쓰레드 생성

pthread\_create(&chatwrite, NULL, &chatwrite\_thread, NULL);

pthread\_join(chatwrite, NULL);// :q 눌려서 쓰레드 끝날때까지 대기

pthread\_join(chatview, NULL);

}

//printf("exit %d", c.exitFlag);

close(c.sockfd); //:q눌리면 소켓을 닫고

exit(0); // 프로세스 종료

}