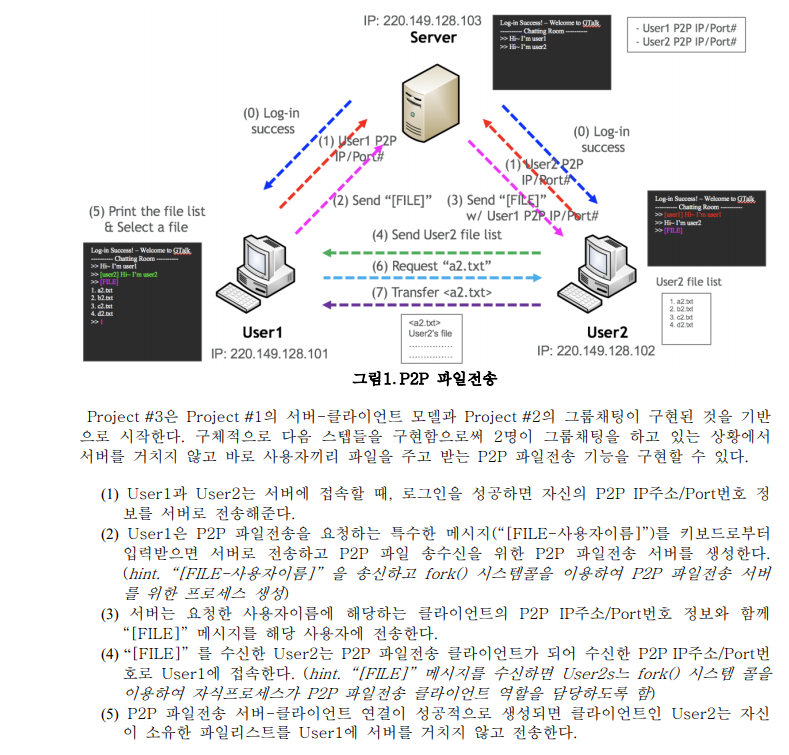
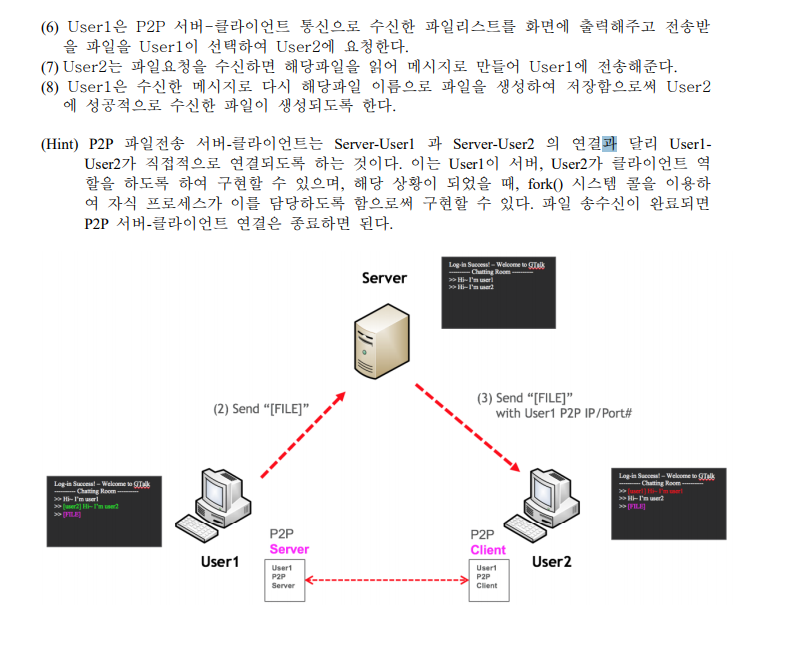
임베디드운영체제 기말 팀 프로젝트

Team Project #3

2016146026 심재빈

2017146056 황경주





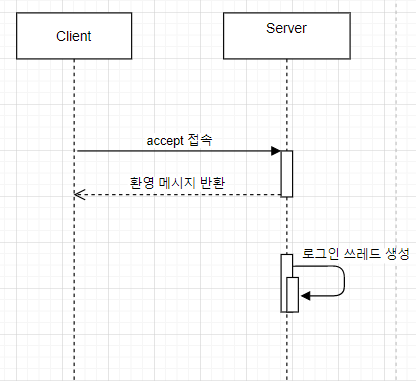
다음과 같은 글씨 색상은 project #2와 변한 부분이 없는 부분이다..

1. 설계

IPC : socket

병렬 프로세스 방식: thread(pthread)

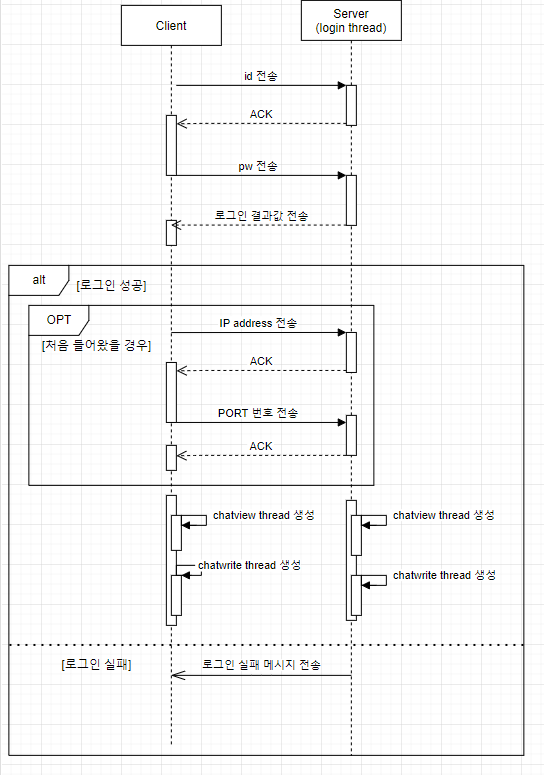
우선 server를 동작시키고, client를 동작시키면, 가장 먼저 메인 프로세스는 다음과 같이 동작한다.



<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.
* 로그인을 성공하면 클라이언트는 서버에게 자신의 포트번호와, 주소 번호를 전송하는 것
  + 1. Client와 login thread의 시퀀스 다이어그램은 다음과 같다.



<2> client와 login\_thread의 로그인까지의 시퀀스 다이어그램

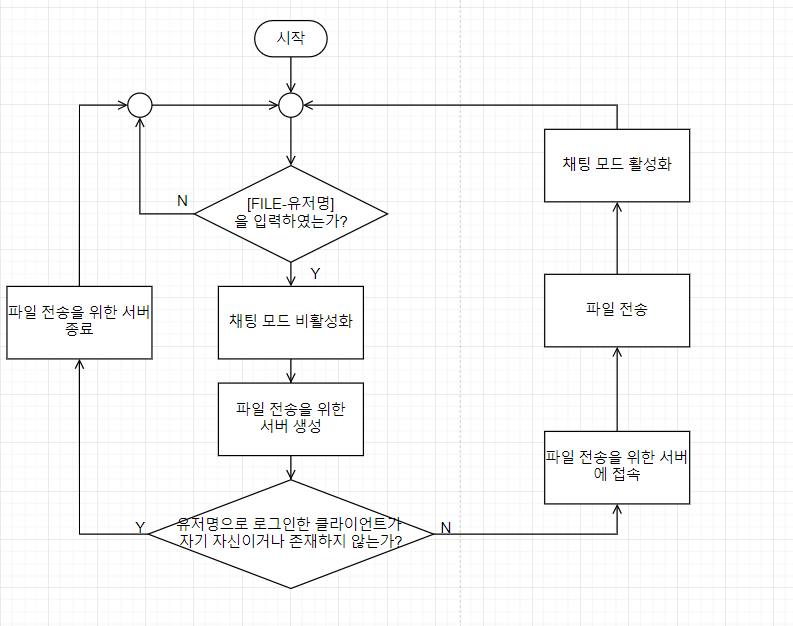
Project 2와는 달리 처음 들어왔을 경우에는 자신의 IP 주소와 포트 번호를 서버에 전송하는 부분이 추가 되었다.

* 1. 채팅 모드 설계

채팅 모드는 [FILE 유저명]을 입력하면 파일 전송 모드 변환하는 기능을 추가하였다.

또한 파일 전송 모드로 변환되면, 채팅 모드는 파일 전송 모드가 끝날 때 까지 비활성화된다.

이에 대해 파일 전송 모드와 채팅 모드에 대한 알고리즘은 다음과 같다.



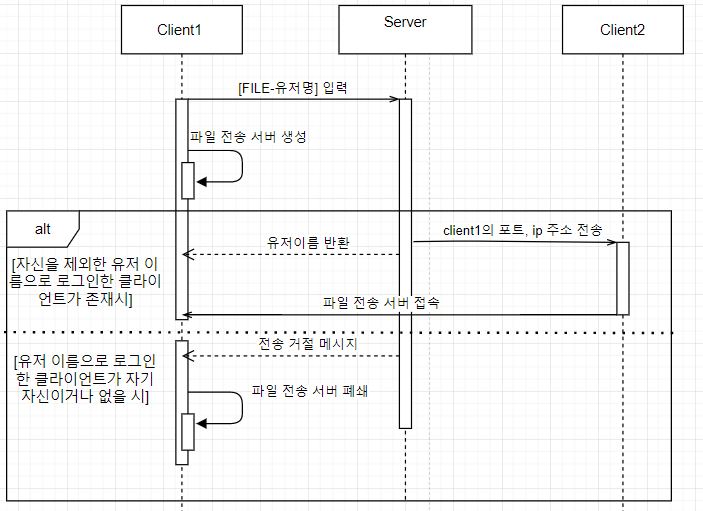
<3>파일 전송 알고리즘(채팅 모드에서 :q로 나가는 부분 제외)

* 1. 파일 전송모드

파일 전송모드는 다음과 같은 기능을 한다.

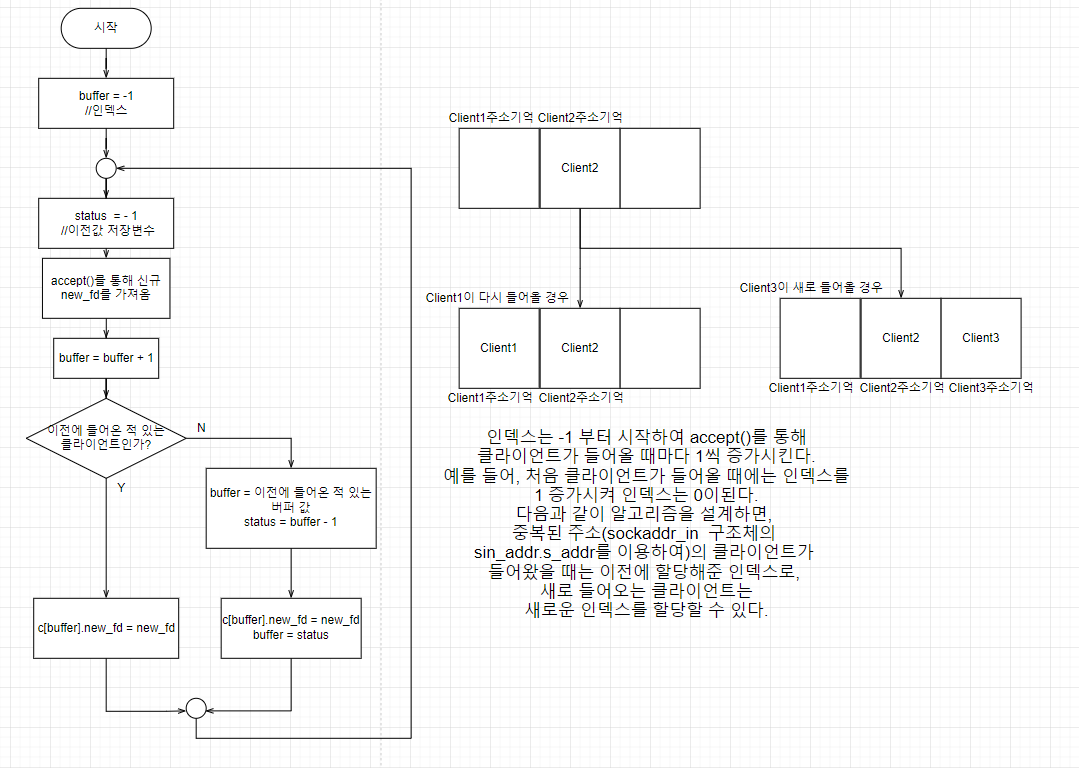
* 채팅 모드에서 [FILE 유저명]을 입력하면 입력한 클라이언트는 파일 전송을 위한 server를 생성하기 위한 쓰레드를 생성한다.
* 다음과 같은 경우에는 생성한 쓰레드에 취소 요청을 보내며, 파일 전송 쓰레드는 즉시 제거된다.
  + 현재 로그인한 클라이언트 중에 유저명이 없을 경우
  + 유저명이 자기 자신일 경우
* 유저명이 자기 자신과 다르고, 로그인한 유저명이 존재한다면, 서버는 해당 클라이언트에게 server를 새로 생성한 클라이언트의 포트 번호와 ip주소를 전송한다.
* Server 역할을 하는 클라이언트의 정보를 받은 클라이언트는 서버 역할을 하는 클라이언트에게 socket 연결을 시도한다.
* 연결이 완료되면 리스트를 보내고 서버 역할을 하는 클라이언트가 리스트의 번호를 고르면 해당 번호의 txt파일을 서버 역할을 하는 클라이언트에게 복사시켜준다.
* 복사가 완료되면 전송모드가 끝나고 채팅모드로 돌아간다.

밑의 그림은 전송모드가 수행될 때 서버가 어떻게 구현되는지에 대한 시퀀스 다이어그램이다.



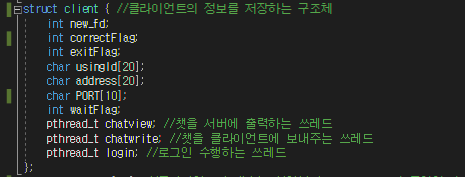
* 1. 클라이언트

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.

address : 해당 클라이언트의 ip주소

PORT : 해당 클라이언트의 포트 주소

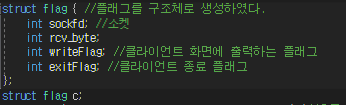
waitFlag: 1이면 채팅모드가 비활성화, 0이면 채팅모드가 활성화

chatview: 챗을 서버에 출력하는 쓰레드

charwrite : 챗을 클라이언트에 보내는 쓰레드

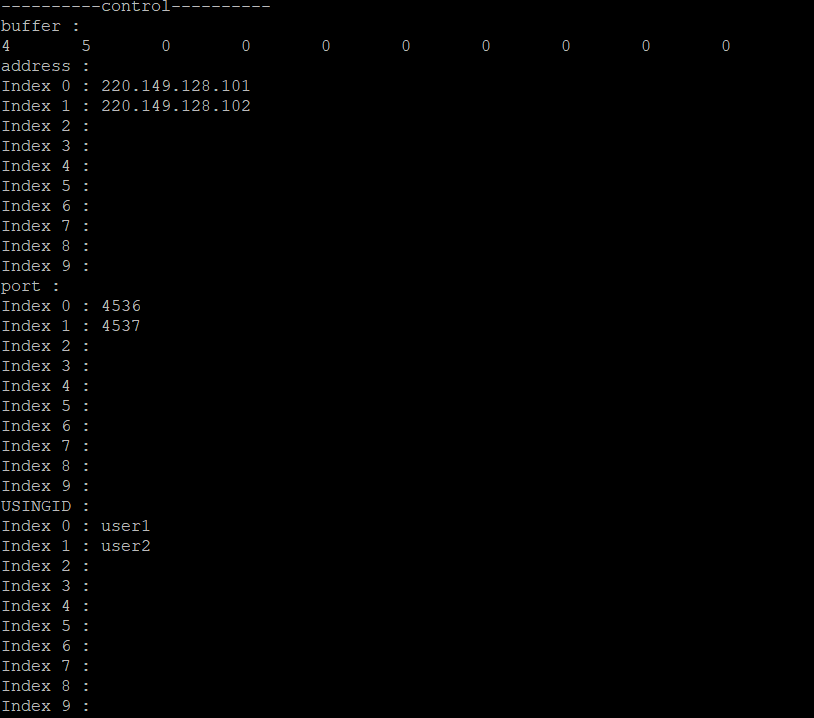
login : 로그인을 수행하는 쓰레드

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



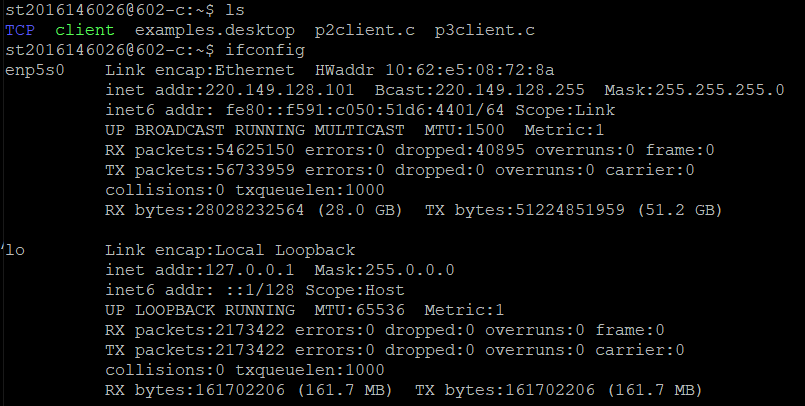
* 1. 부가기능

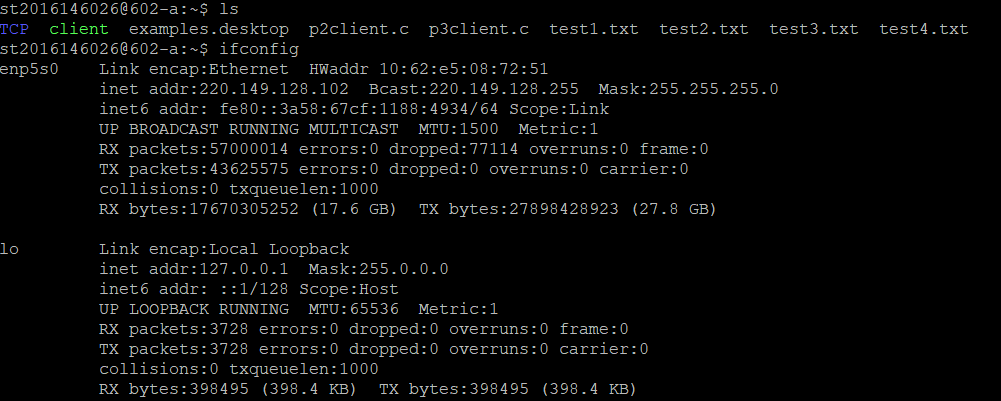
1. 서버에서 /help를 입력하면 현재 들어온 클라이언트의 accept()를 수행한 int형 자료값, ip Address, PORT 번호, 해당 클라이언트가 로그인한 아이디가 나타난다.

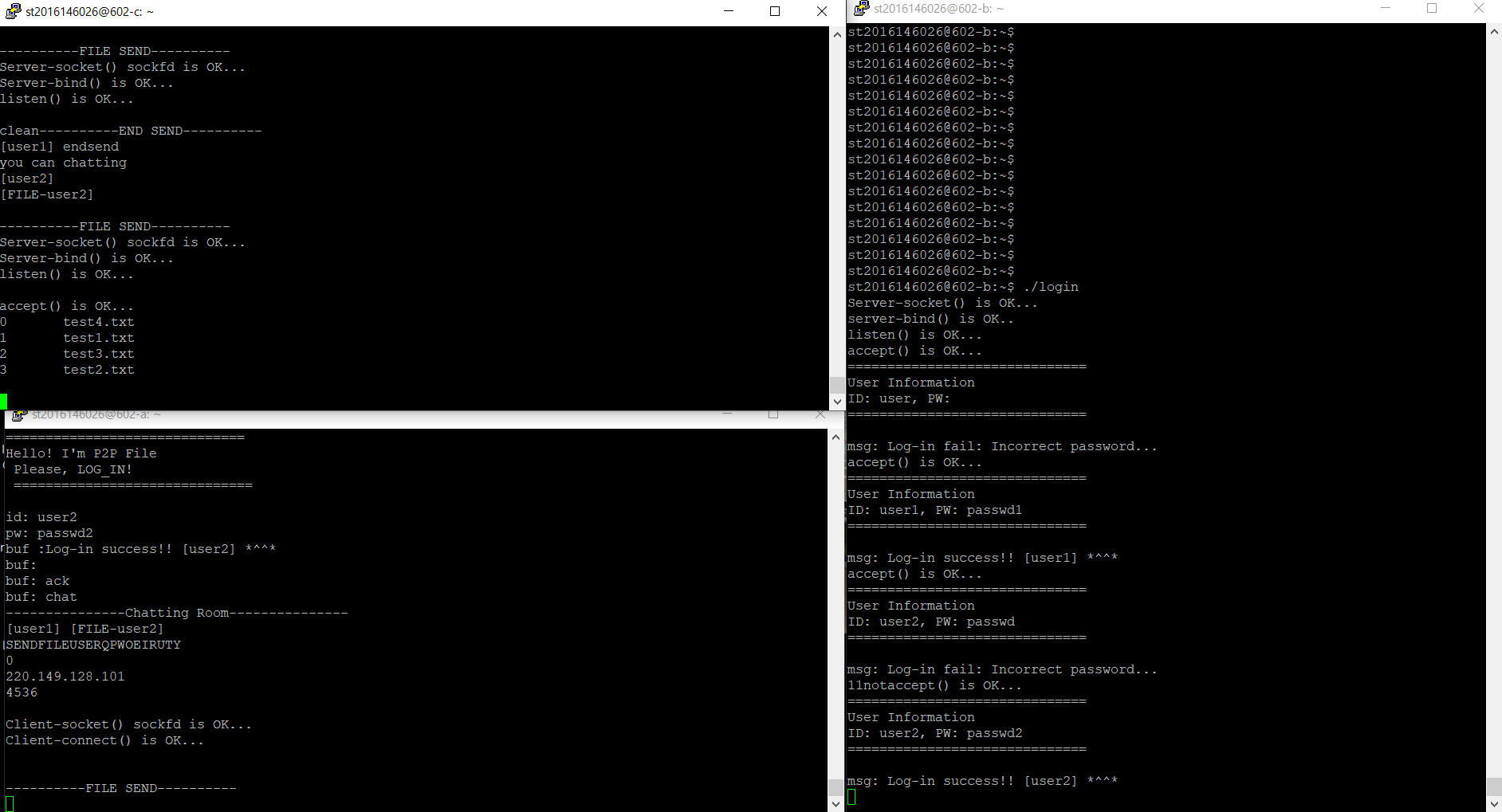


1. 구현
   * 1. 결과 화면

현재 101과 102의 txt파일은 다음과 같다.

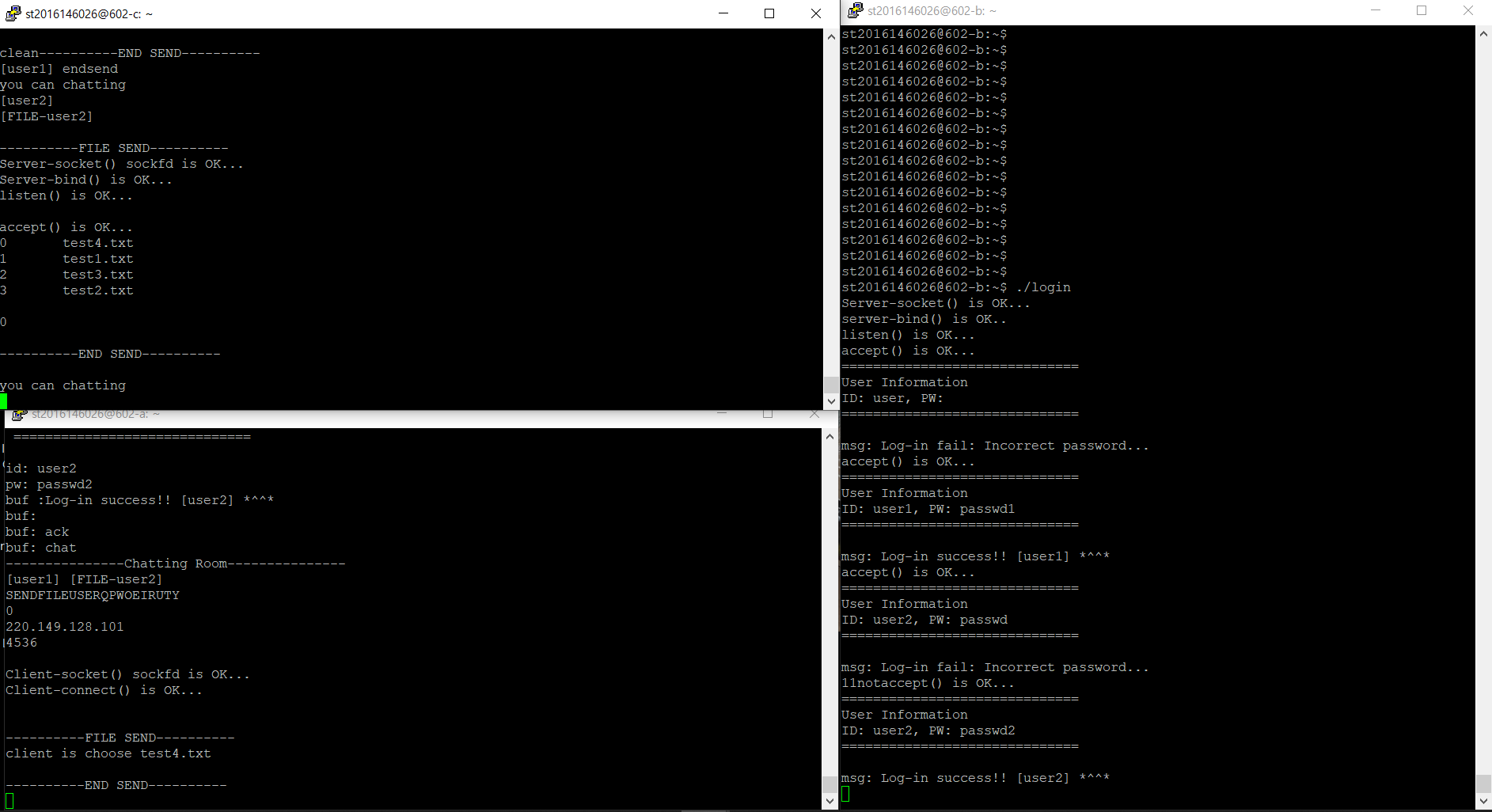
101

102

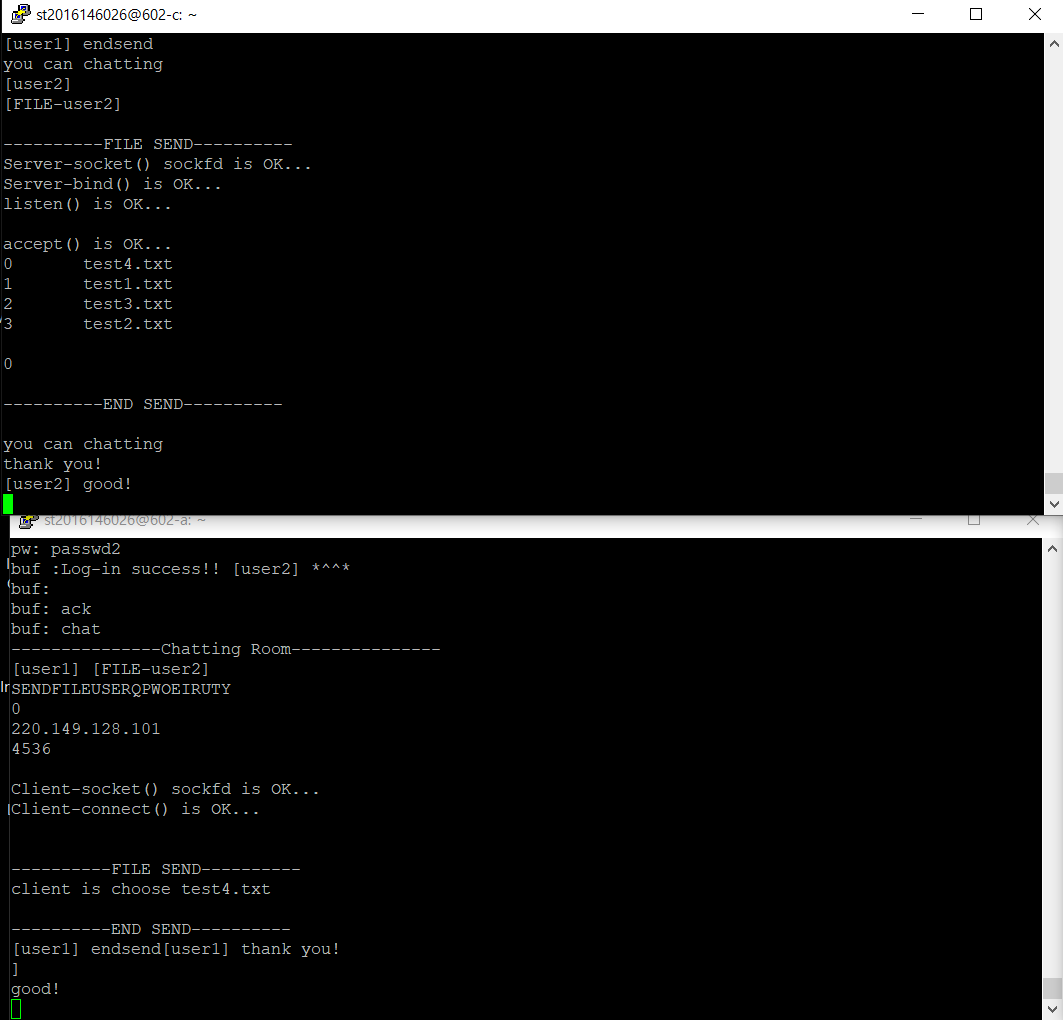


다음과 같이 로그인을 수행하고 user1이 [FILE-user2]을 입력하면 user2로 로그인된 클라이언트의디렉토리 중 txt파일로 이루어진 리스트를 받을 수 있다

왼쪽 위는 101, user1, 밑은 102,user2이다.



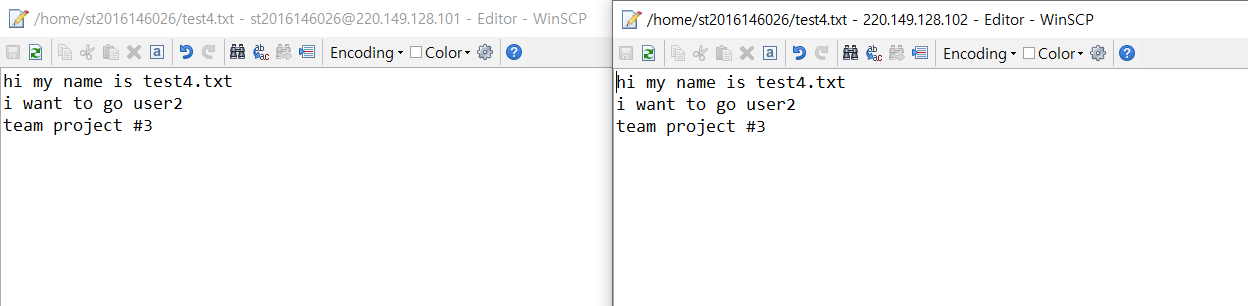
0을 누르면 test4.txt파일이 선택되었다는 문구가 뜨고, 파일 전송 모드가 종료된다



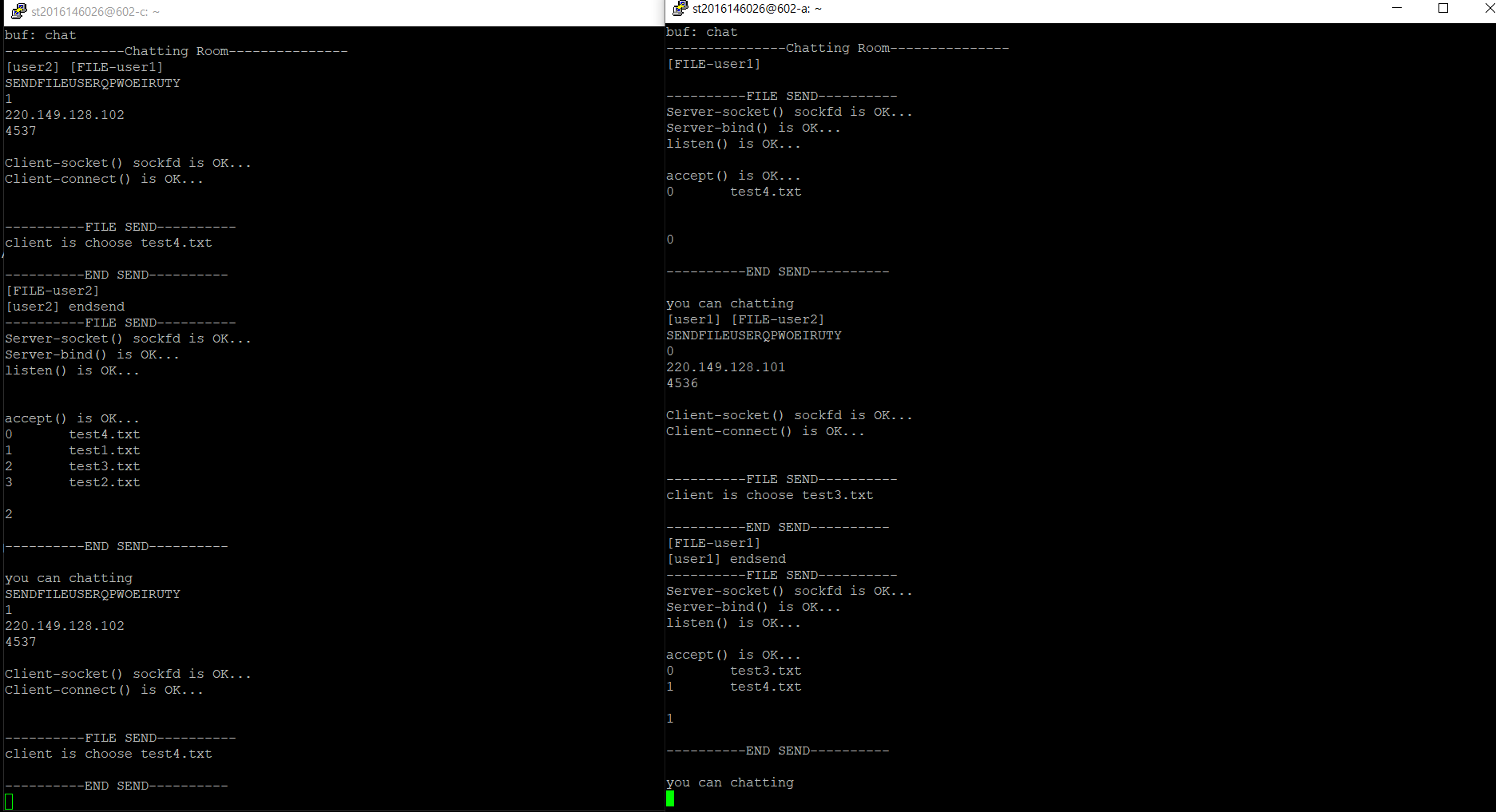
파일전송 모드가 끝나면 채팅 모드로 돌아간다

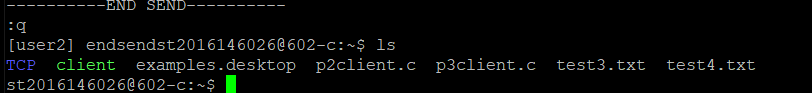


클라이언트를 종료하고, ls로 확인하면 220.149.128.101에 test4.txt가 보내진 것을 확인할 수 있다.



또한 동일한 내용이 복사되어있는 것이 확인되었다.





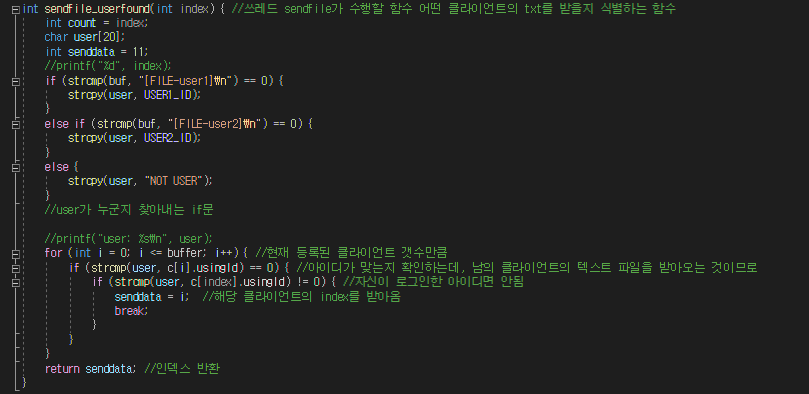
또한 파일 전송이 종료되면 바로 채팅 모드로 돌아가기 때문에 여러 번 파일 전송 모드를 사용할 수 있다.

* + 1. 모듈

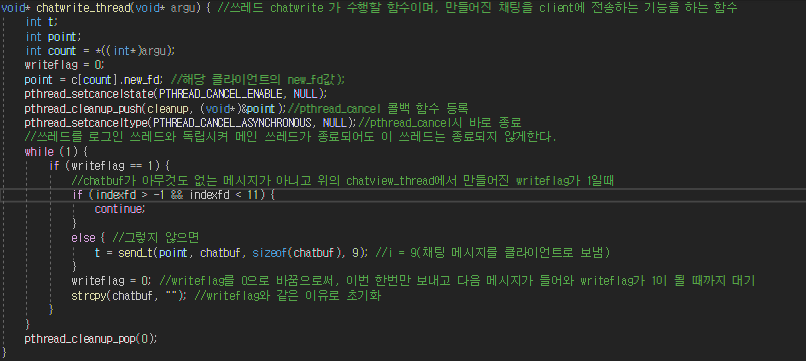
Server.c



앞에 [FILE 5글자가 포함된 메시지를 전송받으면 누구의 txt파일을 받을지 판별하는 알고리즘을 수행한 뒤 인덱스 값을 반환받아, 이 값에 속해있는 클라이언트에게 [FILE-유저명]을 입력한 클라이언트의 포트값과 iP주소값을 전송한다. 하지만 해당 유저명으로 로그인한 클라이언트가 없으면, [FILE-유저명]을 입력한 클라이언트에게 파일을 보내지 않겠다는 메시지를 전송한다…..(1)

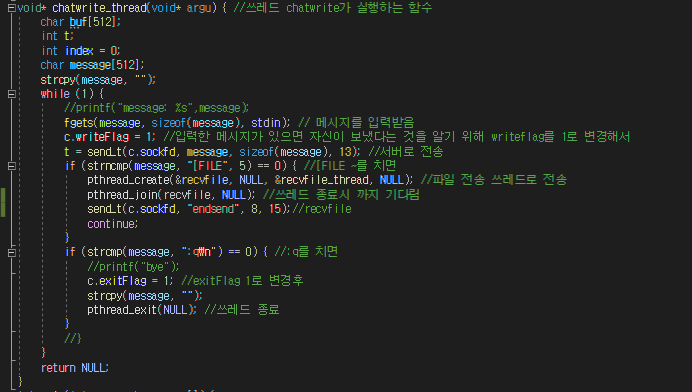


클라이언트를 찾는 알고리즘은 다음과 같다.

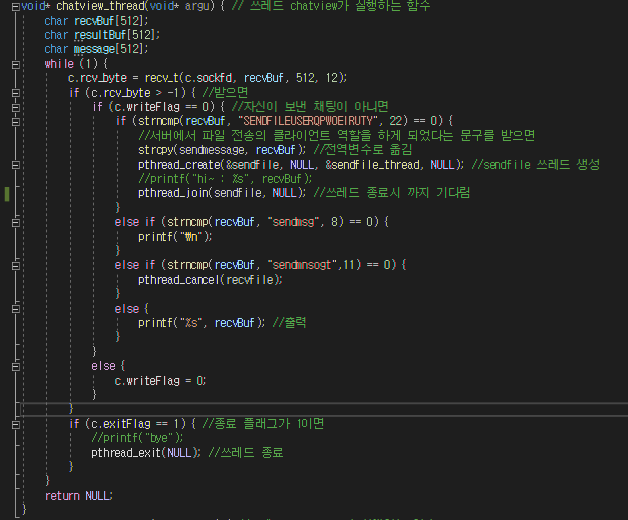


또한 파일 전송모드를 사용하려는 클라이언트가 존재하면 채팅 모드를 비활성화한다.

Client.c



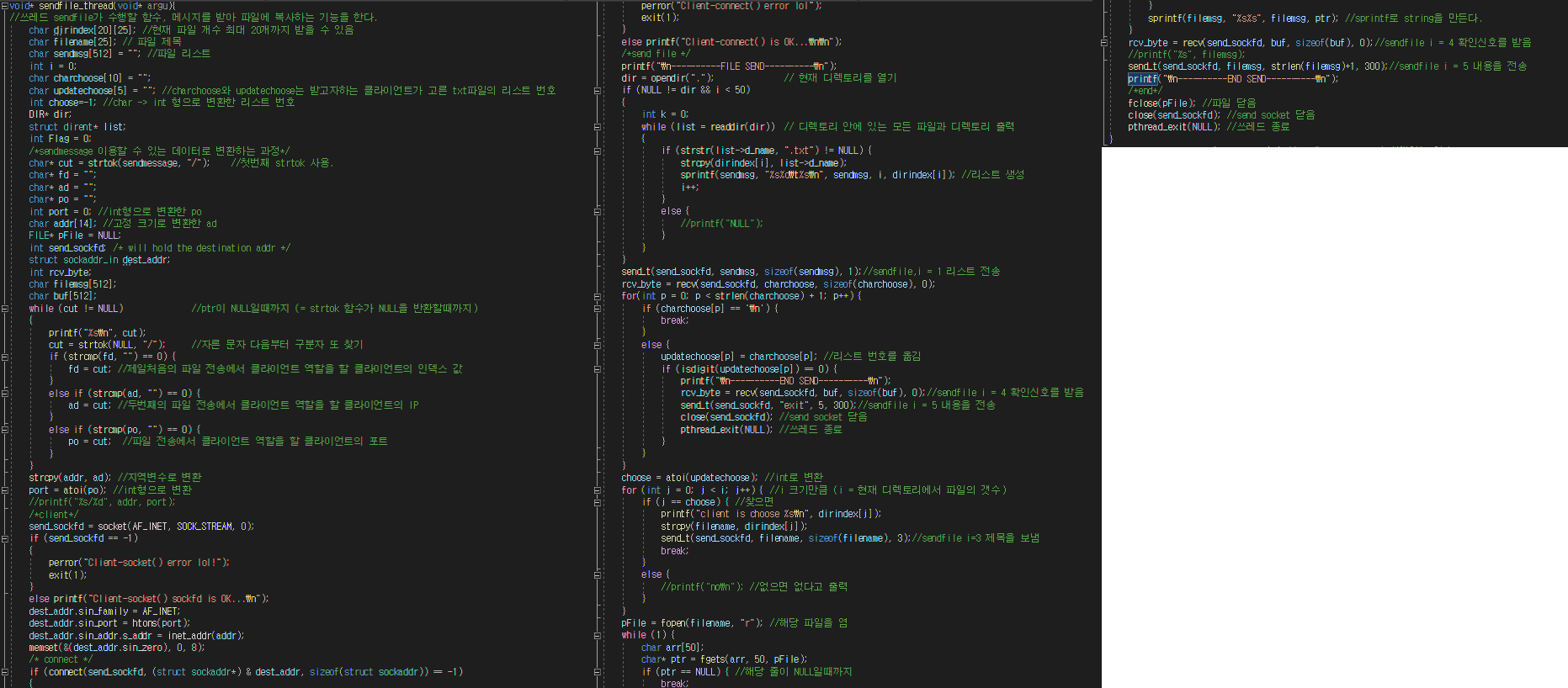
클라이언트에서 [FILE로 시작하는 단어를 입력하면 즉시 recvfile, 파일 전송을 하기위해 서버를 생성하는 쓰레드를 만든다.



아무것도 하지 않는 클라이언트가 (1)에서 서버에서 포트와 IP번호를 받으면, 이 포트와 IP번호를 이용하여 sendfile 쓰레드를 생성한다.

또한, (1)에서 파일 전송 거부 메시지를 받게 되는 클라이언트는 [FILE-유저명]을 입력해 서버로 보낸 클라이언트밖에 없으므로, 서버를 생성한 쓰레드에게 취소요청을 보내며, 후술할 recvfile 쓰레드는 즉시 소켓을 닫고 쓰레드를 종료한다.

이렇게 설계를 하면, 클라이언트에서 [FILE-유저명]을 입력하자마자 파일 전송 서버를 먼저 생성하기 때문에 메인 서버에서 유저명을 판별후 해당 파일 전송 서버에 소켓 연결을 할 것인지, 하지 않을것인지 결정하도록 할 수 있고, 서버보다 요청하는 클라이언트가 먼저 실행되지 않는다.



Sendfile\_thread

Sendfile\_thread는 크게 4가지의 단계로 나뉜다.

연결 - 리스트 전송 - 반환값을 받은 후 해당 파일선택 – 파일을 열어 메시지를 저장

1. 연결

파일 전송 서버에 소켓 연결을 시도한다.

1. 리스트 전송

해당 .c파일이 존재하는 디렉토리를 기준으로, .txt파일을 모두 리스트에 추가한다.

모든 파일을 탐색하여 .txt파일 리스트를 완성하였다면, 이 리스트를 파일 전송 서버에 전송한다.

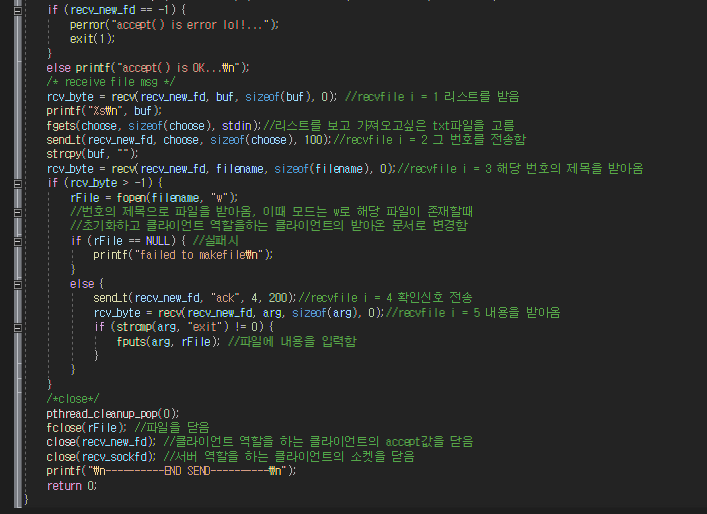
1. 반환값을 받아 해당 파일 선택

전송한 서버는 후술할 recvfile\_thread에서 해당 리스트에서 번호를 입력하고, 입력받은 번호에 해당하는 txt파일을 선택하며, 해당하는 파일의 이름을 전송한다.

1. 파일을 열어 메시지를 저장

해당하는 파일을 열어 파일의 내용을 sprintf를 이용하여 모두 스트링 형태로 저장하여, 이를 파일전송서버에 전송한다.

Recvfile\_thread(서버 생성 구현부 제외)



리스트를 받으면 번호를 입력하면, 해당 파일의 제목을 전송받아, 그 제목의 이름으로 파일을 생성한다. 그 후 내용을 받으면, 그 내용을 fputs를 이용하여 입력하고, 파일과 소켓을 닫는다.

* + 1. 개선점
* 리스트에서 값을 고르고 반환시 recv받은 char \* 형태의 자료형을 int형으로 변환하는 과정에서 atoi()를 사용하는데, 문자를 입력할 경우 0으로 인식하여서 0번째에 해당하는 텍스트 파일을 복사해온다.

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

gcc [파일 명] -o [실행 파일] -lpthread

1. 소스코드

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 4538 //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

//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.

}

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.

}

}

struct client { //클라이언트의 정보를 저장하는 구조체

int new\_fd;

int correctFlag;

int exitFlag;

char usingId[20];

char address[20];

char PORT[10];

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("address :\n"); //주소(new\_fd와 다르게 한번 들어오면 영구적으로 주소가 저장됨)

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

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

}

printf("port :\n");//PORT(new\_fd와 다르게 한번 들어오면 영구적으로 주소가 저장됨)

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

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

}

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;

}

int sendfile\_userfound(int index) { //쓰레드 sendfile가 수행할 함수 어떤 클라이언트의 txt를 받을지 식별하는 함수

int count = index;

char user[20];

int senddata = 11;

//printf("%d", index);

if (strcmp(buf, "[FILE-user1]\n") == 0) {

strcpy(user, USER1\_ID);

}

else if (strcmp(buf, "[FILE-user2]\n") == 0) {

strcpy(user, USER2\_ID);

}

else {

strcpy(user, "NOT USER");

}

//user가 누군지 찾아내는 if문

//printf("user: %s\n", user);

for (int i = 0; i <= buffer; i++) { //현재 등록된 클라이언트 갯수만큼

if (strcmp(user, c[i].usingId) == 0) { //아이디가 맞는지 확인하는데, 남의 클라이언트의 텍스트 파일을 받아오는 것이므로

if (strcmp(user, c[index].usingId) != 0) { //자신이 로그인한 아이디면 안됨

senddata = i; //해당 클라이언트의 index를 받아옴

break;

}

}

}

return senddata; //인덱스 반환

}

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);

}

else if (strncmp(buf, "[FILE", 5) == 0) { //만약 [FILE이 앞의 5글자에 포함되어있다면

indexfd = sendfile\_userfound(count);

if (indexfd > -1 && indexfd < 10) {

send\_t(point, "sendmsg", 8, 131);

sprintf(sendmessage, "SENDFILEUSERQPWOEIRUTY/%d/%s/%s\n", count, c[count].address, c[count].PORT);

send\_t(c[indexfd].new\_fd, sendmessage, sizeof(sendmessage), 14);

}

else {

printf("%dnot", indexfd);

send\_t(point, "sendmnsogt", 11, 132);

}

}

else if (strcmp(buf, "endsend") == 0) { //만약 :q를 받으면

indexfd = -1;

send\_t(point,"\nyou can chatting\n", 50, 9); //(채팅 메시지)

}

}

}

}

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.

while (1) {

if (writeflag == 1) {

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

if (indexfd > -1 && indexfd < 11) {

continue;

}

else { //그렇지 않으면

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.

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, "");

if (c[val].waitFlag == 0) { //새로 들어온 클라이언트일 경우에 (3번 문제 해당)

//c[val].waitFlag = 0;

send\_t(c[val].new\_fd, "ack", 4, 7);//분기 1, i = 7 확인신호를 보냄

rcv\_byte = recv\_t(c[val].new\_fd, c[val].address, sizeof(c[val].address), 8);//분기 1,i = 8 클라이언트의 IP주소를 받음

send\_t(c[val].new\_fd, "ack", 4, 9);//분기 1, i = 9 확인 신호를 보냄

rcv\_byte = recv\_t(c[val].new\_fd, c[val].PORT, sizeof(c[val].PORT), 10);//분기 1, i =10 클라이언트의 포트 번호를 받음

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

send\_t(c[val].new\_fd, "chat", 5, 11); //분기 1-1 , i = 11채팅모드를 시작하겠다는 메시지를 출력한다.

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);

}

else {

send\_t(c[val].new\_fd, "not", 4, 11); //분기 1-2 , i = 11, 로그인 실패했다는 신호를 보낸다

}

}

else { //한번 exitflag가 1이어서 나간 적이 있던 클라이언트가 다시 들어올 때

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

send\_t(c[val].new\_fd, "new", 4, 70); //분기 2, i = 7 IP주소와 포트번호가 저장되어있기 때문에 굳이 보낼 필요없다는 신호를 보냄

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

send\_t(c[val].new\_fd, "chat", 4, 90);//분기 2, 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); //분기 2, i = 7 IP주소와 포트번호가 저장되어있기 때문에 굳이 보낼 필요없다는 신호를 보냄

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

send\_t(c[val].new\_fd, "not", 4, 190); //분기 2-2, i = 7 IP주소와 포트번호가 저장되어있기 때문에 굳이 보낼 필요없다는 신호를 보냄

}

}

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.

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;

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

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

exit(1);

}

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

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 <ctype.h>

#include <pthread.h> //pthread 헤더

#define SERV\_IP "220.149.128.103"

#define SERV\_PORT 4538

#define CLIT\_IP "220.149.128.102" //클라이언트 IP

#define CLIT\_PORT "4537" //클라이언트 포트 번호

#define CLIT\_PORT\_INT 4537

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

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

pthread\_t sendfile;//파일을 보내기 위해 만든 쓰레드

pthread\_t recvfile;//파일을 받기 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 cleanup(void\* argu) {

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

printf("clean");

close(socket); //서버 역할을 하는 클라이언트의 소켓을 닫음

printf("----------END SEND----------\n");

}

void\* recvfile\_thread(void\* argu) {

//쓰레드 recvfile가 수행할 함수, 메시지를 받아 파일에 복사하는 기능을 한다.

int recv\_sockfd, recv\_new\_fd;

struct sockaddr\_in my\_addr;

struct sockaddr\_in their\_addr;

unsigned int sin\_size;

FILE\* pFile = NULL;

int rcv\_byte;

char buf[512];

char arg[512];

char filename[25]; //파일 제목

char choose[10] = "";

int val = 1;

printf("\n----------FILE SEND----------\n");

FILE\* rFile = NULL;

/\*server\*/

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

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

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

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

if (recv\_sockfd == -1) {

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

exit(1);

}

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

/\* host byte order \*/

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

/\* short, network byte order \*/

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

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

/\* zero the rest of the struct \*/

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

/\* to prevent ‘Address already in use…’ \*/

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

{

perror("setsockopt");

close(recv\_sockfd);

exit(1);

//return -1;

}

/\* bind \*/

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

{

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

exit(1);

}

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

/\* listen \*/

if (listen(recv\_sockfd, 10) == -1)

{

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

exit(1);

}

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

/\* ...other codes to read the received data... \*/

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

recv\_new\_fd = accept(recv\_sockfd, (struct sockaddr\*) & their\_addr, &sin\_size);

if (recv\_new\_fd == -1) {

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

exit(1);

}

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

/\* receive file msg \*/

rcv\_byte = recv(recv\_new\_fd, buf, sizeof(buf), 0); //recvfile i = 1 리스트를 받음

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

fgets(choose, sizeof(choose), stdin);//리스트를 보고 가져오고싶은 txt파일을 고름

send\_t(recv\_new\_fd, choose, sizeof(choose), 100);//recvfile i = 2 그 번호를 전송함

strcpy(buf, "");

rcv\_byte = recv(recv\_new\_fd, filename, sizeof(filename), 0);//recvfile i = 3 해당 번호의 제목을 받아옴

if (rcv\_byte > -1) {

rFile = fopen(filename, "w");

//번호의 제목으로 파일을 받아옴, 이때 모드는 w로 해당 파일이 존재할때

//초기화하고 클라이언트 역할을하는 클라이언트의 받아온 문서로 변경함

if (rFile == NULL) { //실패시

printf("failed to makefile\n");

}

else {

send\_t(recv\_new\_fd, "ack", 4, 200);//recvfile i = 4 확인신호 전송

rcv\_byte = recv(recv\_new\_fd, arg, sizeof(arg), 0);//recvfile i = 5 내용을 받아옴

if (strcmp(arg, "exit") != 0) {

fputs(arg, rFile); //파일에 내용을 입력함

}

}

}

/\*close\*/

pthread\_cleanup\_pop(0);

fclose(rFile); //파일을 닫음

close(recv\_new\_fd); //클라이언트 역할을 하는 클라이언트의 accept값을 닫음

close(recv\_sockfd); //서버 역할을 하는 클라이언트의 소켓을 닫음

printf("\n----------END SEND----------\n");

return 0;

}

void\* sendfile\_thread(void\* argu){

//쓰레드 sendfile가 수행할 함수, 메시지를 받아 파일에 복사하는 기능을 한다.

char dirindex[20][25]; //현재 파일 개수 최대 20개까지 받을 수 있음

char filename[25]; // 파일 제목

char sendmsg[512] = ""; //파일 리스트

int i = 0;

char charchoose[10] = "";

char updatechoose[5] = ""; //charchoose와 updatechoose는 받고자하는 클라이언트가 고른 txt파일의 리스트 번호

int choose=-1; //char -> int 형으로 변환한 리스트 번호

DIR\* dir;

struct dirent\* list;

int Flag = 0;

/\*sendmessage 이용할 수 있는 데이터로 변환하는 과정\*/

char\* cut = strtok(sendmessage, "/"); //첫번째 strtok 사용.

char\* fd = "";

char\* ad = "";

char\* po = "";

int port = 0; //int형으로 변환한 po

char addr[14]; //고정 크기로 변환한 ad

FILE\* pFile = NULL;

int send\_sockfd; /\* will hold the destination addr \*/

struct sockaddr\_in dest\_addr;

int rcv\_byte;

char filemsg[512];

char buf[512];

while (cut != NULL) //ptr이 NULL일때까지 (= strtok 함수가 NULL을 반환할때까지)

{

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

cut = strtok(NULL, "/"); //자른 문자 다음부터 구분자 또 찾기

if (strcmp(fd, "") == 0) {

fd = cut; //제일처음의 파일 전송에서 클라이언트 역할을 할 클라이언트의 인덱스 값

}

else if (strcmp(ad, "") == 0) {

ad = cut; //두번째의 파일 전송에서 클라이언트 역할을 할 클라이언트의 IP

}

else if (strcmp(po, "") == 0) {

po = cut; //파일 전송에서 클라이언트 역할을 할 클라이언트의 포트

}

}

strcpy(addr, ad); //지역변수로 변환

port = atoi(po); //int형으로 변환

//printf("%s/%d", addr, port);

/\*client\*/

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

if (send\_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(port);

dest\_addr.sin\_addr.s\_addr = inet\_addr(addr);

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

/\* connect \*/

if (connect(send\_sockfd, (struct sockaddr\*) & dest\_addr, sizeof(struct sockaddr)) == -1)

{

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

exit(1);

}

else printf("Client-connect() is OK...\n\n");

/\*send file \*/

printf("\n----------FILE SEND----------\n");

dir = opendir("."); // 현재 디렉토리를 열기

if (NULL != dir && i < 50)

{

int k = 0;

while (list = readdir(dir)) // 디렉토리 안에 있는 모든 파일과 디렉토리 출력

{

if (strstr(list->d\_name, ".txt") != NULL) {

strcpy(dirindex[i], list->d\_name);

sprintf(sendmsg, "%s%d\t%s\n", sendmsg, i, dirindex[i]); //리스트 생성

i++;

}

else {

//printf("NULL");

}

}

}

send\_t(send\_sockfd, sendmsg, sizeof(sendmsg), 1);//sendfile,i = 1 리스트 전송

rcv\_byte = recv(send\_sockfd, charchoose, sizeof(charchoose), 0);

for(int p = 0; p < strlen(charchoose) + 1; p++) {

if (charchoose[p] == '\n') {

break;

}

else {

updatechoose[p] = charchoose[p]; //리스트 번호를 옮김

if (isdigit(updatechoose[p]) == 0) {

printf("\n----------END SEND----------\n");

rcv\_byte = recv(send\_sockfd, buf, sizeof(buf), 0);//sendfile i = 4 확인신호를 받음

send\_t(send\_sockfd, "exit", 5, 300);//sendfile i = 5 내용을 전송

close(send\_sockfd); //send socket 닫음

pthread\_exit(NULL); //쓰레드 종료

}

}

}

choose = atoi(updatechoose); //int로 변환

for (int j = 0; j < i; j++) { //i 크기만큼 (i = 현재 디렉토리에서 파일의 갯수)

if (j == choose) { //찾으면

printf("client is choose %s\n", dirindex[j]);

strcpy(filename, dirindex[j]);

send\_t(send\_sockfd, filename, sizeof(filename), 3);//sendfile i=3 제목을 보냄

break;

}

else {

//printf("no\n"); //없으면 없다고 출력

}

}

pFile = fopen(filename, "r"); //해당 파일을 염

while (1) {

char arr[50];

char\* ptr = fgets(arr, 50, pFile);

if (ptr == NULL) { //해당 줄이 NULL일때까지

break;

}

sprintf(filemsg, "%s%s", filemsg, ptr); //sprintf로 string을 만든다.

}

rcv\_byte = recv(send\_sockfd, buf, sizeof(buf), 0);//sendfile i = 4 확인신호를 받음

//printf("%s", filemsg);

send\_t(send\_sockfd, filemsg, strlen(filemsg)+1, 300);//sendfile i = 5 내용을 전송

printf("\n----------END SEND----------\n");

/\*end\*/

fclose(pFile); //파일 닫음

close(send\_sockfd); //send socket 닫음

pthread\_exit(NULL); //쓰레드 종료

}

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

char recvBuf[512];

char resultBuf[512];

char message[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

if (strncmp(recvBuf, "SENDFILEUSERQPWOEIRUTY", 22) == 0) {

//서버에서 파일 전송의 클라이언트 역할을 하게 되었다는 문구를 받으면

strcpy(sendmessage, recvBuf); //전역변수로 옮김

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

//printf("hi~ : %s", recvBuf);

pthread\_join(sendfile, NULL); //쓰레드 종료시 까지 기다림

}

else if (strncmp(recvBuf, "sendmsg", 8) == 0) {

printf("\n");

}

else if (strncmp(recvBuf, "sendmnsogt",11) == 0) {

pthread\_cancel(recvfile);

}

else {

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 (strncmp(message, "[FILE", 5) == 0) { //[FILE ~를 치면

pthread\_create(&recvfile, NULL, &recvfile\_thread, NULL); //파일 전송 쓰레드로 전송

pthread\_join(recvfile, NULL); //쓰레드 종료시 까지 기다림

send\_t(c.sockfd, "endsend", 8, 15);//recvfile

continue;

}

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);

if (strcmp(buf, "new") != 0) { //만약 재접속이 아니라면

send\_t(c.sockfd, CLIT\_IP, sizeof(CLIT\_IP), 8);//분기 1, 8 이 클라이언트의 IP를 보냄

c.rcv\_byte = recv\_t(c.sockfd, buf, sizeof(buf), 9);//분기 1, 9확인 신호를 받음

send\_t(c.sockfd, CLIT\_PORT, sizeof(CLIT\_PORT), 10);//분기 1, 10 이 클라이언트의 PORt를 받음

}

else {

send\_t(c.sockfd, "ack", 4, 8);//분기 2, i = 8

}

//chat

c.rcv\_byte = recv\_t(c.sockfd, buf2, sizeof(buf2), 11);//분기 1일땐 i = 9, 분기 2일땐 i = 9

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

if (strcmp(buf2, "chat") == 0 || strcmp(buf2, "new") == 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

}