팀 프로젝트 2

교과목명	시스템프로그래밍		분반	001
제출일자	2021-12-13		교수자명	장희숙
팀명	이름	학번	이름	학번
초코파이썬칩	김기태	20173152		
	안대현	20173217		

▣ 프로젝트 주제

팀 프로젝트 2. 채팅과 파일전송이 가능한 GUI 메신저 프로그램을 구현하시오.

▣ 프로젝트 결과

1. 채팅과 파일전송이 가능한 GUI 메신저 프로그램을 구현하시오.

A) 채팅방 구성과 사용자 등록 기능을 가진 소켓 서버를 구현한다.

server.c

```
//////////////////////생략//////////////////////////
int main(int argc, char *argv[]) {
        struct sockaddr_in client_addr;
        char buf[MAX_LINE +1]; //Receive Message From client
        int i, j, nbyte, accp_sock, addrlen =sizeof(struct
                sockaddr_in);
        fd_set read_fds; //FD constructor
        pthread_t a_thread;
        if (argc !=2) {
                printf("사용법 :%s port₩n", argv[0]);
                exit(0);
        }
        // listen_tcp(host, port, backlog) Func call
        listen_sock = listen_tcp(INADDR_ANY, atoi(argv[1]), 5);
        //generate thread
        pthread_create(&a_thread, NULL, thread_function, (void *)NULL);
        while (1) {
                FD_ZERO(&read_fds);
                FD_SET(listen_sock, &read_fds);
                for (i =0; i < active_user; i++)
                        FD_SET(client_sock_list[i], &read_fds);
                max_fdp1 = get_socket_max() +1;         // max_fdp1 Re calculate
                if (select(max_fdp1, &read_fds, NULL, NULL, NULL) <0)
                        err_handle("select fail");
```

```
                if (FD_ISSET(listen_sock, &read_fds)) {
                        accp_sock = accept(listen_sock,
                                (struct sockaddr*)&client_addr, &addrlen);
                        if (accp_sock ==-1) err_handle("accept fail");
                        client_add(accp_sock, &client_addr);
                        send(accp_sock, START_STRING, strlen(START_STRING), 0);
                        ct = time(NULL);                        //Receive Current time
                        tm =*localtime(&ct);
                        write(1, "₩033[0G", 4);          //Move cursor X to 0
                        printf("[%02d:%02d:%02d]", tm.tm_hour, tm.tm_min, tm.tm_sec);
                        fprintf(stderr, "₩033[33m");//Change Text color Yellow
                        printf("사용자 1명 추가. 현재 참가자 수 = %d₩n", active_user);
                        fprintf(stderr, "₩033[32m");//Change Text color green
                        fprintf(stderr, "server>"); //print cursor
                }
                // BroadCast Meesage to Clients
                for (i =0; i < active_user; i++) {
                        if (FD_ISSET(client_sock_list[i], &read_fds)) {
                                active_chat++;                          //++ active_chat
                                nbyte = recv(client_sock_list[i], buf, MAX_LINE, 0);
                                if (nbyte <=0) {
                                        cleant_remove(i);       // exit client
                                        continue;
                                }
                                buf[nbyte] =0;

                                if (strstr(buf, EXIT_STRING) !=NULL) {
                                        cleant_remove(i);       // exit client
                                        continue;
                                }
                                // Broadcast message all clients
                                for (j =0; j < active_user; j++)
                                        send(client_sock_list[j], buf, nbyte, 0);
                                printf("₩033[0G");                      //move cursor X to 0
                                fprintf(stderr, "₩033[97m");//change text color white
                                printf("%s", buf);                      //print message
                                fprintf(stderr, "₩033[35m");//change text color green
                                fprintf(stderr, "server>"); //print cursor
                        }
                }
        }
        return 0;
}
// new chat vulenteer
void client_add(int s, struct sockaddr_in *new_client_addr) {
        char buf[20];
        inet_ntop(AF_INET, &new_client_addr->sin_addr, buf, sizeof(buf));
        write(1, "₩033[0G", 4);          //move cursor x pos to 0
        fprintf(stderr, "₩033[33m");      //change text color yellow
        printf("new client: %s₩n", buf); // print ip
        // add chat list
        client_sock_list[active_user] = s;
        strcpy(ip_list[active_user], buf);
        active_user++; //++ active_user
}
// generate socket and listen
int  listen_tcp(int host, int port, int backlog) {
        int sd;
        struct sockaddr_in server_addr;
```

```
        sd = socket(AF_INET, SOCK_STREAM, 0);
        if (sd ==-1) {
                perror("socket fail");
                exit(1);
        }
        // server_addr constructor setting
        bzero((char *)&server_addr, sizeof(server_addr));
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(host);
        server_addr.sin_port = htons(port);
        if (bind(sd, (struct sockaddr *)&server_addr, sizeof(server_addr)) <0) {
                perror("bind fail");  exit(1);
        }
        // waiting for connection from client
        listen(sd, backlog);
        return sd;
}
//thread by process
void *thread_function(void *arg) {
        int i;
        printf("===========Availble Commands========== ₩n");
        printf("help, active_user, active_chat, ip_list₩n");
        printf("======================================= ₩n");
        while (1) {
                char buf_msg[MAX_LINE +1];
                fprintf(stderr, "₩033[1;32m");
                printf("Socket Server > ");
                fgets(buf_msg, MAX_LINE, stdin); //input cmd
                if (!strcmp(buf_msg, "₩n")) continue;    // ignore enter key
                else if (!strcmp(buf_msg, "help₩n"))
                        printf("help, active_user, active_chat, ip_list₩n");
                else if (!strcmp(buf_msg, "active_user₩n"))
                        printf("현재 참가자 수 = %d₩n", active_user);
                else if (!strcmp(buf_msg, "active_chat₩n"))
                        printf("지금까지의 대화의 수 = %d₩n", active_chat);
                else if (!strcmp(buf_msg, "ip_list₩n"))
                        for (i =0; i < active_user; i++)
                                printf("%s₩n", ip_list[i]);
                else //handle exception
                        printf("해당 명령어가 없습니다. help 를 참조하세요.₩n");
        }
}
// find max socket num
int get_socket_max() {

        int max = listen_sock;
        int i;
        for (i =0; i < active_user; i++)
                if (client_sock_list[i] > max)
                        max = client_sock_list[i];
        return max;
}
// set exit chat
void cleant_remove(int s) {
        close(client_sock_list[s]);
        if (s != active_user -1) { //Re-sort list
                client_sock_list[s] = client_sock_list[active_user -1];
                strcpy(ip_list[s], ip_list[active_user -1]);
        }
```

```
        active_user--; //minus user numbers
        ct = time(NULL);                        // CURRENT_TIMESTAMP
        tm =*localtime(&ct);
        write(1, "₩033[0G", 4);
        fprintf(stderr, "₩033[33m");
        printf("[%02d:%02d:%02d]", tm.tm_hour, tm.tm_min, tm.tm_sec);
        printf("채팅 참가자 1명 탈퇴. 현재 참가자 수 = %d₩n", active_user);
        fprintf(stderr, "₩033[32m");
        fprintf(stderr, "server>");
}
```

main 함수에서 클라이언트가 접속함에따라 소켓을 읽게 되면 사용자 1명 추가라는 메세지를 출력하며 클라이언트와 통신을 하게된다. 이후 브로드캐스트 방식으로 접속한 클라이언트에게 메시지를 전송하는 역할을 구현한다.
client_add 함수는 소켓을 읽게되면 서버에서 클라이언트가 들어왔음을 읽게되며 스레드를 통해 매개변수를 입력하여 서버 소켓을 읽거나 참가자 수를 출력하는 역할을 담당한다. get_socket_max를 통해 총 들어온 클라이언트의 소켓 개수를 파악하며 client_remove를 통해 채팅 참가자가 나감을 구현한다.

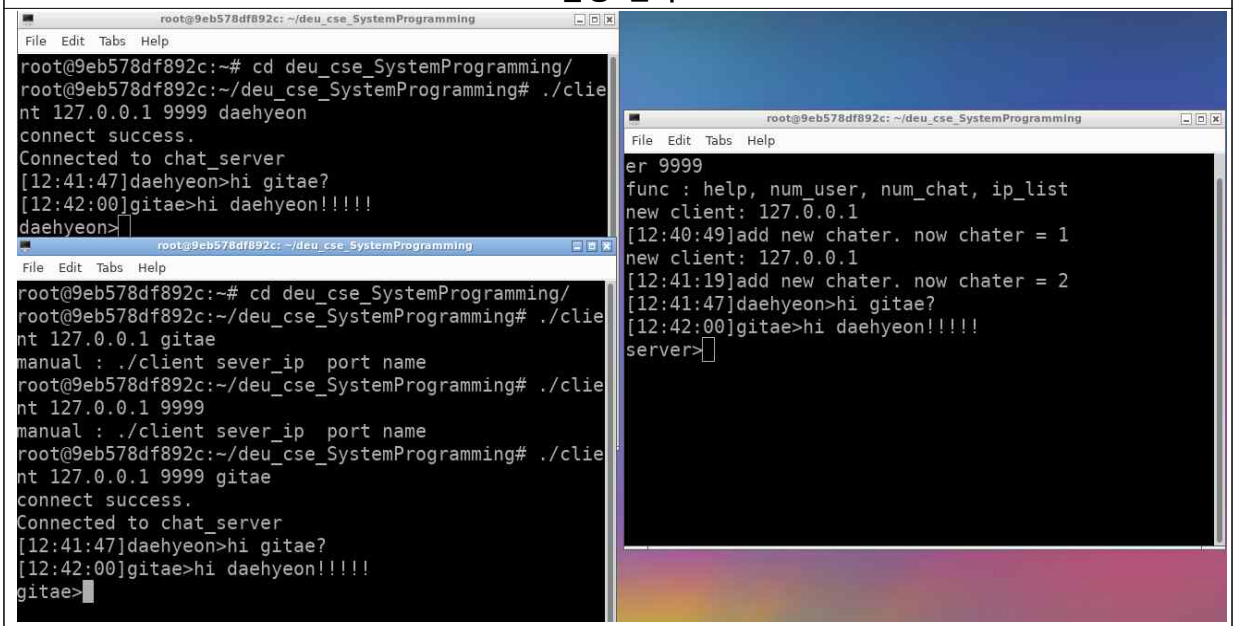
B) 위 5번에서 구현한 소켓 채팅 프로그램을 활용하여 채팅 기능을 구현한다.

client_tcp.c

```
int main(int argc, char *argv[]) {
        char bufname[NAME_LEN];       //name
        char bufmsg[MAXLINE]; //message
        char bufall[MAXLINE + NAME_LEN];
        int maxfdp1;     //max socket descriper
        int s;           //socket
        int namelen;
        fd_set read_fds;
        time_t ct;
        struct tm tm;
        if (argc !=4) {
                printf("manual : %s sever_ip  port name ₩n", argv[0]);
                exit(0);
        }
        s = tcp_connect(AF_INET, argv[1], atoi(argv[2]));
        if (s ==-1)
                errquit("tcp_connect fail");
        puts("connect success.");
        maxfdp1 = s +1;
        FD_ZERO(&read_fds);
        while (1) {
                FD_SET(0, &read_fds);
                FD_SET(s, &read_fds);
                if (select(maxfdp1, &read_fds, NULL, NULL, NULL) <0)
                        errquit("select fail");
                if (FD_ISSET(s, &read_fds)) {
                        int nbyte;
                        if ((nbyte = recv(s, bufmsg, MAXLINE, 0)) >0) {
                                bufmsg[nbyte] =0;
                                write(1, "₩033[0G", 4);
                                printf("%s", bufmsg);
                                fprintf(stderr, "%s>", argv[3]);
                        }
                }
                if (FD_ISSET(0, &read_fds)) {
                        if (fgets(bufmsg, MAXLINE, stdin)) {
```

```
                                fprintf(stderr, "₩033[1A");
                                ct = time(NULL);
                                tm =*localtime(&ct);
                                sprintf(bufall,    "[%02d:%02d:%02d]%s>%s",    tm.tm_hour,
tm.tm_min, tm.tm_sec, argv[3], bufmsg);
                                if (send(s, bufall, strlen(bufall), 0) <0)
                                        puts("Error : Write error on socket.");
                                if (strstr(bufmsg, EXIT_STRING) !=NULL) {
                                        puts("Good bye.");
                                        close(s);
                                        exit(0);
                                }
                        }
                }
        } // end of while
}
int tcp_connect(int af, char *servip, unsigned short port) {
        struct sockaddr_in servaddr;
        int  s;
        // create socket
        if ((s = socket(af, SOCK_STREAM, 0)) <0)
                return -1;
        bzero((char *)&servaddr, sizeof(servaddr));
        servaddr.sin_family = af;
        inet_pton(AF_INET, servip, &servaddr.sin_addr);
        servaddr.sin_port = htons(port);
        // connect response
        if (connect(s, (struct sockaddr *)&servaddr, sizeof(servaddr))
                <0)
                return -1;
        return s;
}
```

실행 결과



tcp_connect 함수를 통해 port 번호를 시스템 인자로 받아 소켓을 생성하며 같은 포트의 서버에게 메시지를 보내고, 서버에서 메시지를 받는 역할을 구현한다.

C) GTK+ 또는 Qt를 이용하여 간단한 채팅 프로그램을 위한 GUI 클라이언트 프로그램을 작성하고

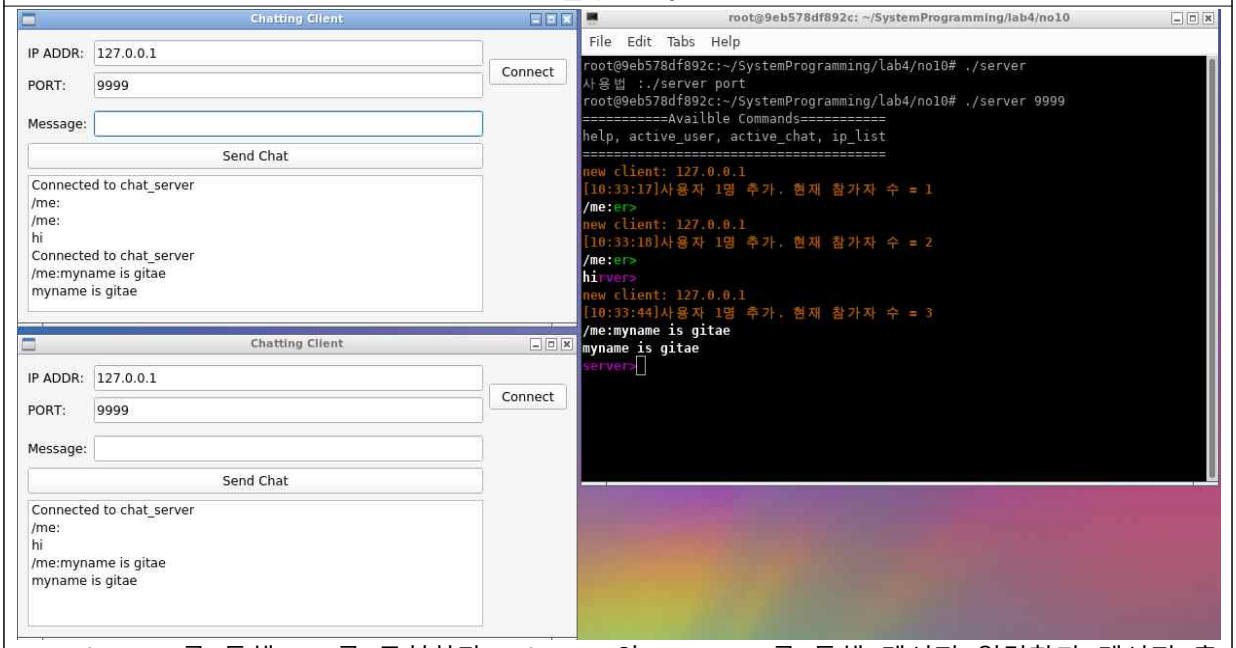
구현한 채팅 프로그램에 적용하여 본다.

chat_001.pro && main.cpp && widget.cpp && widget.ui

rlarlxo2323 add no10 ... 2 hours ago History

..		
Makefile	add no10	2 hours ago
chat_001	add no10	2 hours ago
chat_001.pro	add no10	2 hours ago
main.cpp	add no10	2 hours ago
main.o	add no10	2 hours ago
moc_widget.cpp	add no10	2 hours ago
moc_widget.o	add no10	2 hours ago
ui_widget.h	add no10	2 hours ago
widget.cpp	add no10	2 hours ago
widget.h	add no10	2 hours ago
widget.o	add no10	2 hours ago
widget.ui	add no10	2 hours ago

실행 결과



Qt5 designer를 통해 GUI를 구현한다. Edit Text와 Line Text를 통해 메시지 입력창과 메시지 출력창을 구현하며 이를 서버로 보내고 받는 역할을 담당한다. 서버와 포트 번호를 통해 같은 서버와 클라이언트를 연결하는 역할을 담당한다.

D) 파일 전송 기능을 구현한다. FTP 프로토콜 등을 참고하는 것도 좋다. - 구현 실패