Project Report

Chatting client and server

Network Programming

화요일반
학 과 : 컴퓨터과학과
학 번 : 201411706
이 름 : 김혜지
제 출 일 : 2016. 06. 11
담당교수 : 김 남기 교수님

1. usage guide

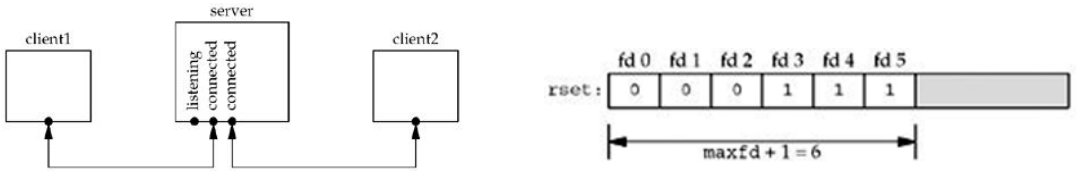
실행 터미널을 4개 준비한다. 실행 터미널 1에서 채팅 서버를 실행하고 나머지 3개의 터미널에서 채팅 클라이언트를 실행한다.

실행 터미널-1/채팅 서버측 실행	실행터미널-2,3,4/ 채팅 클라이언트측 실행
$sudo ./cser 6553	$./ccli 127.0.0.1 6553 kim
	$./ccli 127.0.0.2 6553 lee
	$./ccli 127.0.0.3 6553 han

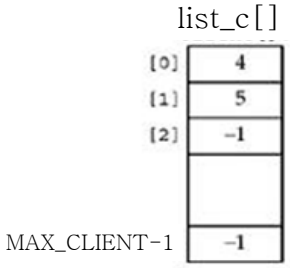
- 채팅 서버를 포트 6553으로 실행하고, 3개의 실행 터미널을 준비해서 채팅 클라이언트를 실행한다. 각각의 채팅 클라이언트틀 대화명 kim, lee, han으로 채팅 서버에 연결한다.
- 연결 후, "Welcome to chatting room" 메세지가 뜨면, 다른 클라이언트에게 보낼 메세지를 입력한다.
- 채팅에 참여하고 있는 클라이언트 목록을 보고 싶으면 /list 를 입력한다.
- 1:1 비밀 메세지를 보내고 싶으면 /smsg [대화명] [메세지] 형식으로 입력한다.
- 채팅을 종료하고 싶으면 /quit 를 입력한다.

2. design architecture

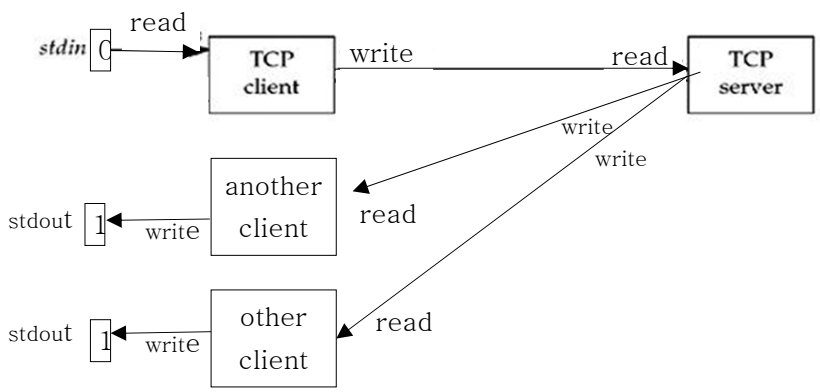
- 서버와 클라이언트 연결: 서버가 listening소켓, connected소켓 감시. FD_SET을 이용해서 connected 소켓 추가.



- list_c[] 구조체: 소켓번호, 닉네임, ip주소, 포트번호 저장.

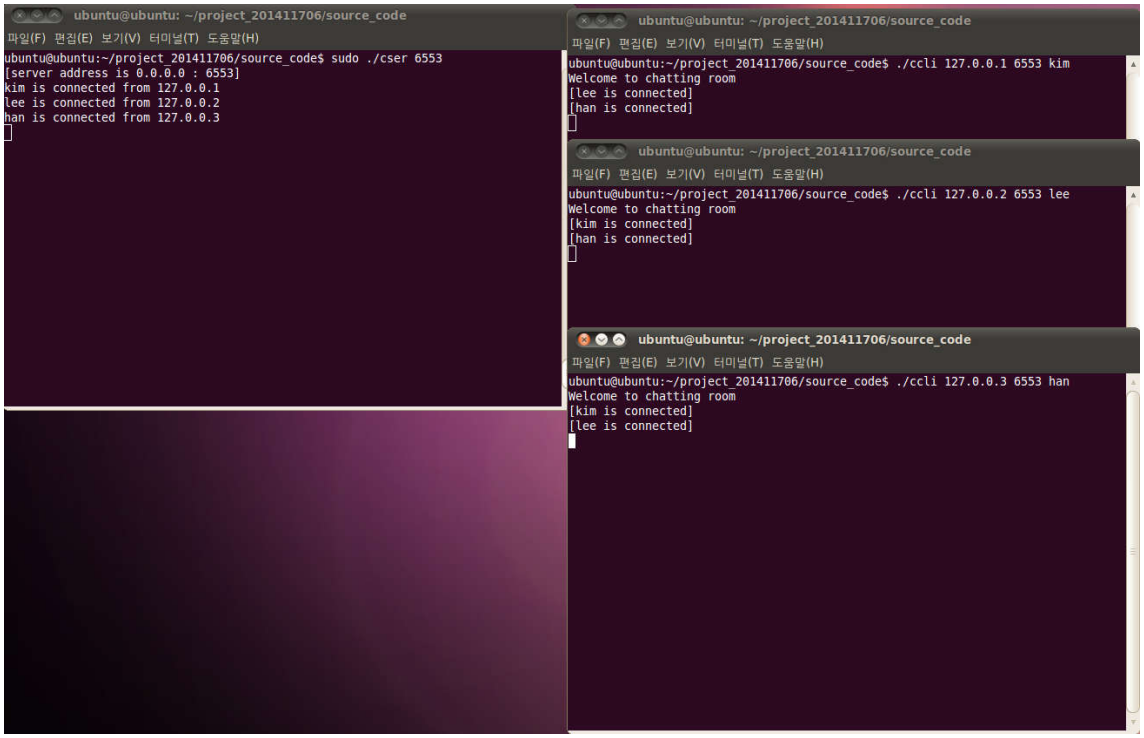


- 메세지 주고 받는 원리.

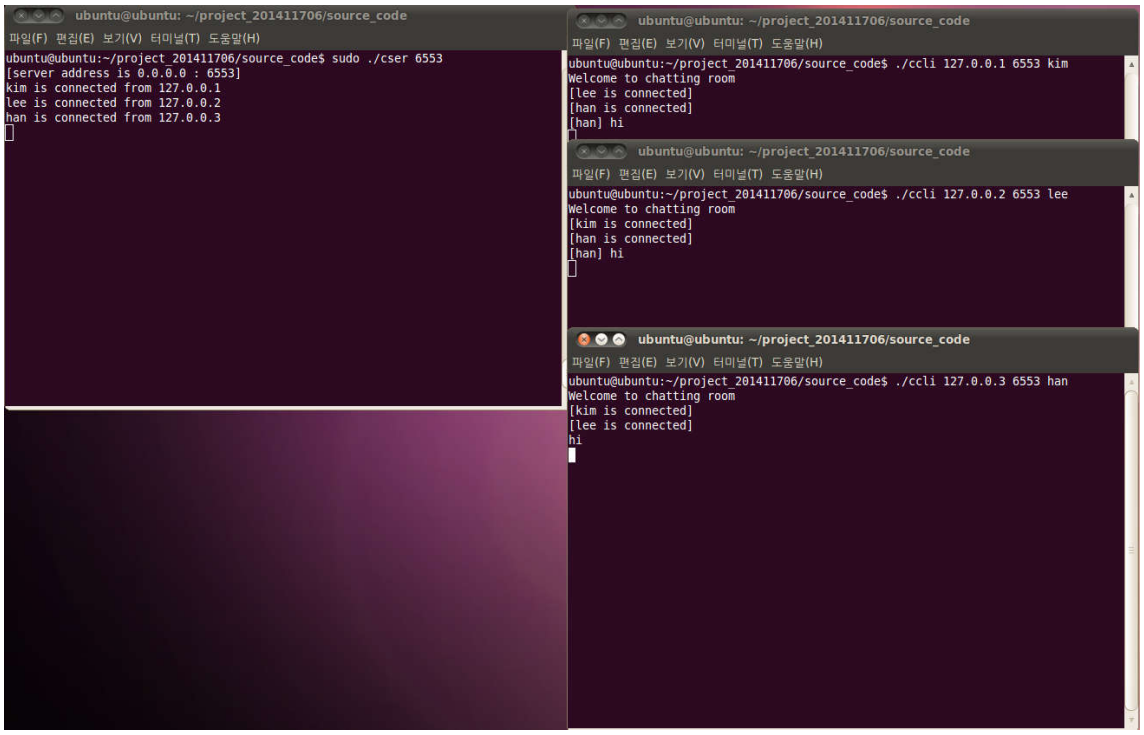


3. snapshots of executions

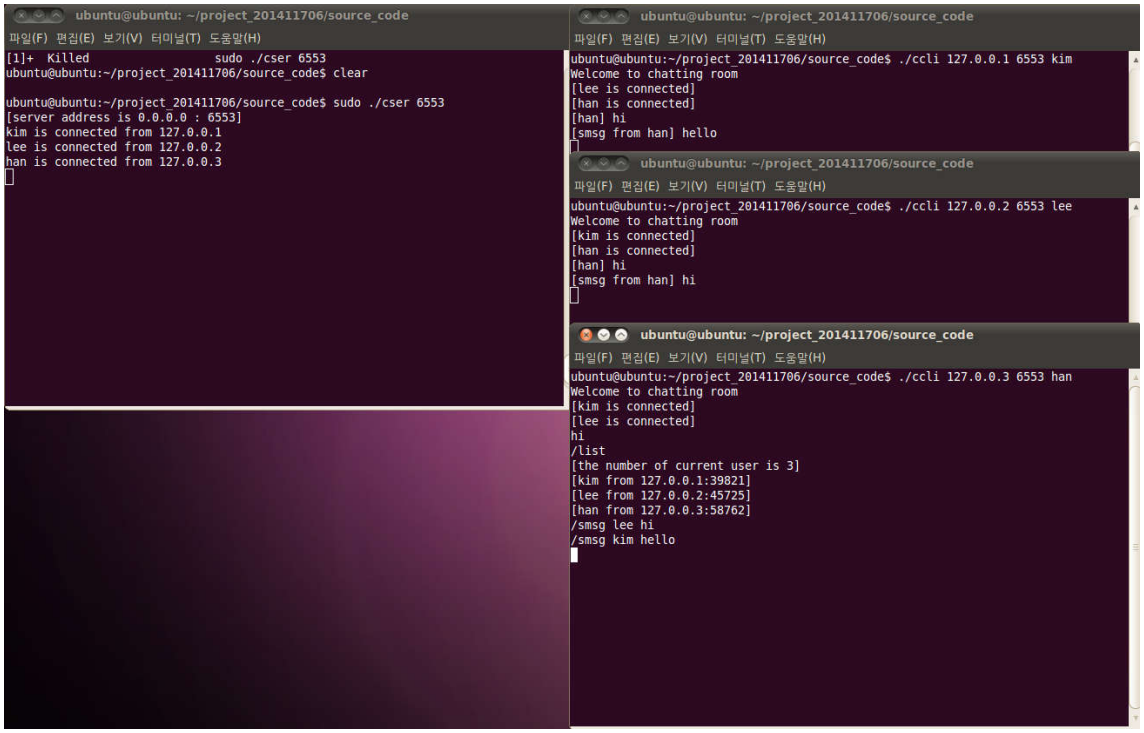
1) 연결



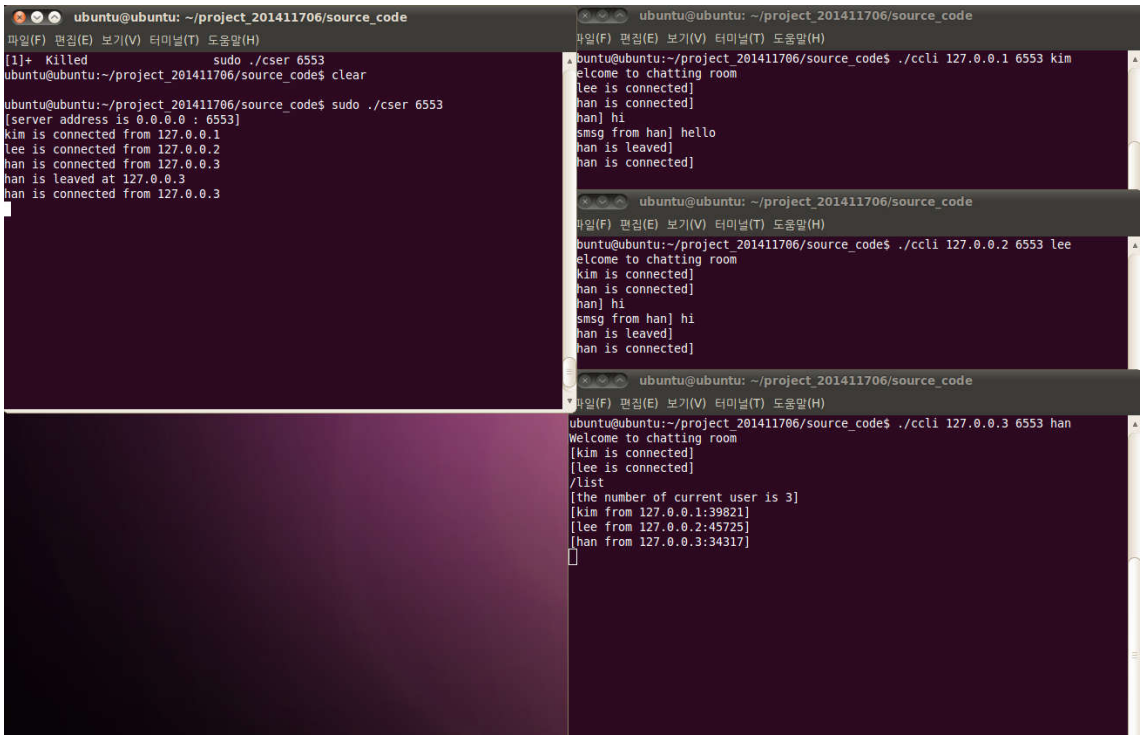
2) 메세지 보내기



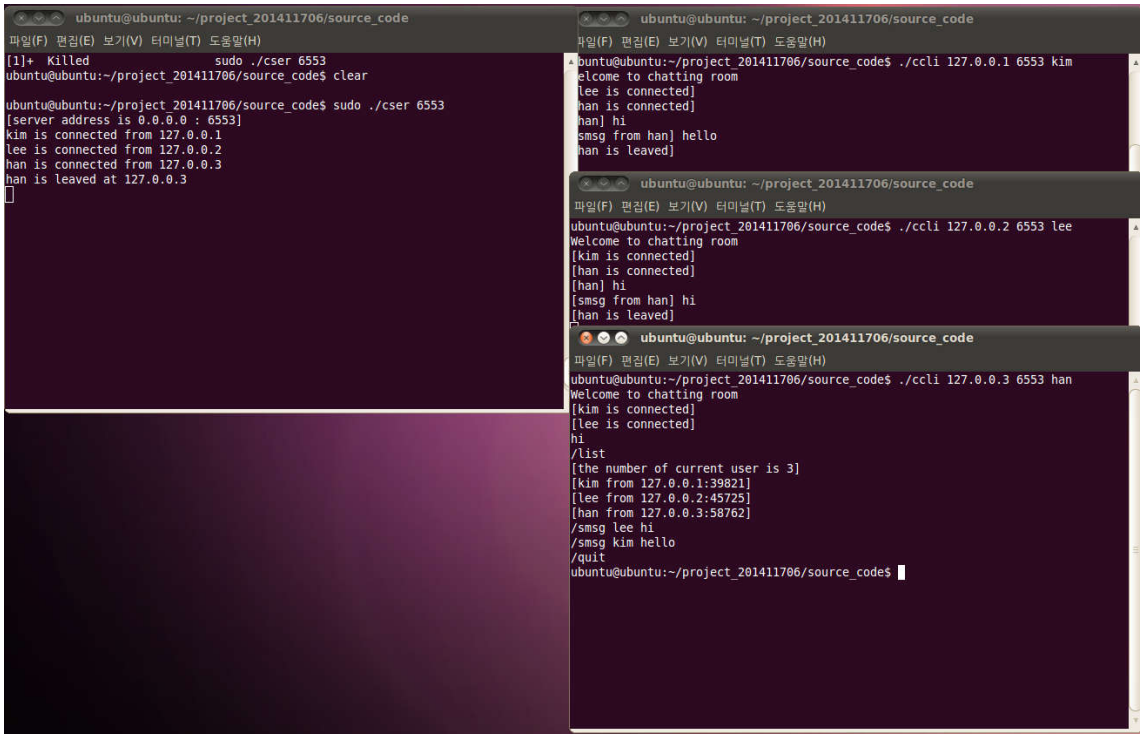
3) 1:1 메세지



4) 클라이언트 목록 띄우기



5)채팅 종료



4. source directories and compile method

project_201411706.tar.gz₩project_201411607₩source_code₩ccli.c
project_201411706.tar.gz₩project_201411607₩source_code₩cser.c

gcc -o cser cser.c
gcc -o ccli ccli.c

5. source code explanation

1) 서버 코드 cser.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<sys/select.h>

#define MAXLINE 100
#define MAX_CLIENT 1000
#define CHATDATA 1024
#define INVALID_SOCK -1

char greeting[]="Welcome to chatting room₩n";
char ERROR[]= "Sorry No More Connection₩n";
char quit[]="/quit₩n";
```

```
char list[]="/list₩n";
char smsg[]="/smsg";
char smsg_quit[]="/smsg_quit₩n";

struct List_c{          //List_c구조체 정의
        int socket_num;
        char nick[CHATDATA];
        char ip[40];
        int port;
}list_c[MAX_CLIENT];

int
pushClient(int connfd,char* c_nick,char* c_ip,int c_port){          //클라이언트 배열에 push
        int i;

        for(i=0;i<MAX_CLIENT;i++){
                if(list_c[i].socket_num==INVALID_SOCK){
                        list_c[i].socket_num=connfd;
                        strcpy(list_c[i].nick,c_nick);
                        strcpy(list_c[i].ip,c_ip);
                        list_c[i].port=c_port;
                        return i;
                }
        }

        if(i==MAX_CLIENT)
                return -1;
}

int popClient(int s)   //클라이언트 배열에 pop
{
        int i;


        for(i =0; i<MAX_CLIENT;i++){
                if(s==list_c[i].socket_num){
                        list_c[i].socket_num=INVALID_SOCK;
                        memset(list_c[i].nick,0,sizeof(list_c[i].nick));
                        memset(list_c[i].ip,0,sizeof(list_c[i].ip));
                        break;
                }
        }
        close(s);
        return 0;
}
void
constr_func(int i,int index){       //(connect string)연결 상태 메세지 띄우기
        char buf1[MAXLINE];

        memset(buf1,0,sizeof(buf1));
        sprintf(buf1,"[%s is connected]₩r₩n",list_c[index].nick);
```

```
		write(list_c[i].socket_num,buf1,strlen(buf1));

		sprintf(buf1,"[%s is connected]₩r₩n",list_c[i].nick);
		write(list_c[index].socket_num,buf1,strlen(buf1));
}
void
quit_func(int i){			//채팅 종료 메세지 띄우기
		int j;
		char* token=NULL;
		char buf1[MAXLINE];

		memset(buf1,0,sizeof(buf1));
		printf("%s is leaved at %s₩r₩n",list_c[i].nick, list_c[i].ip);
		for(j=0; j<MAX_CLIENT;j++)
			if(j!=i && list_c[j].socket_num!=INVALID_SOCK){
				sprintf(buf1,"[%s is leaved]₩r₩n",list_c[i].nick);
				write(list_c[j].socket_num,buf1,strlen(buf1));
			}
}
void
list_func(int i){			//클라이언트 목록 띄우기
		int j,cnt=0;
		char buf1[MAXLINE];

		memset(buf1,0,sizeof(buf1));
		for(j=0; j<MAX_CLIENT;j++)
			if(list_c[j].socket_num!=INVALID_SOCK)cnt++;
		sprintf(buf1,"[the number of current user is %d]₩r₩n",cnt);
		write(list_c[i].socket_num,buf1,strlen(buf1));
		for(j=0; j<MAX_CLIENT;j++)
			if(list_c[j].socket_num!=INVALID_SOCK){
				sprintf(buf1,"[%s from %s:%d]₩r₩n",list_c[j].nick,list_c[j].ip,list_c[j].port);
				write(list_c[i].socket_num,buf1,strlen(buf1));
			}
}
int
smsg_func(char* chatData,int i){		//1:1 메세지 보내기
		int j,smsg_sock;
		char* token=NULL;
		char buf1[MAXLINE];



		memset(buf1,0,sizeof(buf1));
		token=strtok(chatData," ");
		char * end;

		if(strcmp(token,smsg)){
			if((end=strtok(NULL,"₩n"))==NULL)
				sprintf(buf1,"%s",token);
			else sprintf(buf1,"%s %s",token,end);
```

```
                sprintf(chatData,"[%s] %s₩n",list_c[i].nick,buf1);
                return 1;
        }
        else{
                token=strtok(NULL," ");
                for(j=0;j<MAX_CLIENT;j++)
                        if(!strcmp(list_c[j].nick,token))
                                smsg_sock=list_c[j].socket_num;
                token=strtok(NULL,"₩n");
                memset(buf1,0,sizeof(buf1));
                sprintf(buf1, "[smsg from %s] %s₩r₩n", list_c[i].nick, token);
                write(smsg_sock, buf1, strlen(buf1));
                return 0;
        }
}

main(int argc,char *argv[])
{
        int connfd,listenfd;
        struct sockaddr_in servaddr,cliaddr;
        int clilen;
        int maxfd=0;
        int i,j,n;
        fd_set rset;

        int index;

        char* token=NULL;
        char buf1[MAXLINE];
        char buf2[MAXLINE];
        char chatData[CHATDATA];



        if(argc<2){
                printf("usage: %s port_number₩n",argv[0]);
                exit(-1);
        }

        listenfd=socket(AF_INET,SOCK_STREAM,0);
        memset(&servaddr,0,sizeof(servaddr));
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(atoi(argv[1]));

        if(bind(listenfd,(struct sockaddr *) &servaddr,sizeof(servaddr))==-1){
                printf("Can not Bind₩n");
                return -1;
        }
        if(listen(listenfd,MAX_CLIENT)==-1){
```

```
                printf("listen Fail₩n");
                return -1;
        }

        for(i=0;i<MAX_CLIENT;i++)
                list_c[i].socket_num=INVALID_SOCK;

        memset(buf1,0,sizeof(buf1));
        inet_ntop(AF_INET,&servaddr.sin_addr,buf1,sizeof(buf1));
        printf("[server address is %s : %d]₩r₩n", buf1, ntohs(servaddr.sin_port));

        for ( ; ; )
        {
                maxfd=listenfd;

                FD_ZERO(&rset);
                FD_SET(listenfd,&rset);
                for(i=0;i<MAX_CLIENT;i++){
                        if(list_c[i].socket_num!=INVALID_SOCK){
                                FD_SET(list_c[i].socket_num,&rset);
                                if(list_c[i].socket_num > maxfd) maxfd=list_c[i].socket_num;
                        }
                }
                maxfd++;

                if(select(maxfd,&rset,(fd_set *)0, (fd_set *)0, (struct timeval *)0) < 0){
                        printf("Select error₩n");
                        exit(1);
                }

                if(FD_ISSET(listenfd,&rset)){//클라이언트 push하고 연결메세지 띄우기
                        clilen=sizeof(cliaddr);
                        if((connfd=accept(listenfd,(struct sockaddr *)&cliaddr, &clilen)) > 0) {
                                memset(buf1,0,sizeof(buf1));
                                memset(buf2,0,sizeof(buf2));
                                read(connfd,buf1,sizeof(buf1));//read client's nickname
                                inet_ntop(AF_INET,&cliaddr.sin_addr,buf2,sizeof(buf2));
                                index=pushClient(connfd,buf1,buf2,ntohs(cliaddr.sin_port));//push
socknum,nick,ip,port_num of client
                                printf("%s is connected from %s₩r₩n",list_c[index].nick,list_c[index].ip );

                                if(index<0){
                                        write(connfd,ERROR,strlen(ERROR));
                                        close(connfd);
                                }else{
                                        write(connfd,greeting, strlen(greeting));
                                        for(i=0; i<MAX_CLIENT;i++)
                                                if(i!=index                                     &&
list_c[i].socket_num!=INVALID_SOCK){
                                                        constr_func(i,index);
                                                }
```

```
                                        }
                                }
                        }
                        for(i=0; i<MAX_CLIENT;i++){//메세지 읽고 메세지 비교 후 동작하기
                                if((list_c[i].socket_num != INVALID_SOCK) && FD_ISSET(list_c[i].socket_num,&rset)){
                                        memset(chatData,0,sizeof(chatData));
                                        if((n=read(list_c[i].socket_num,chatData, sizeof (chatData)))>0){//클라이
언트한테 메세지 읽기
                                                if(!strcmp(chatData,quit)){//클라이언트i 채팅 종료
                                                        quit_func(i);
                                                        popClient(list_c[i].socket_num);
                                                        continue;
                                                }
                                                if(!strcmp(chatData,list)){//클라이언트목록 띄우기
                                                        list_func(i);
                                                        continue;
                                                }
                                                if(strstr(chatData,smsg)!=NULL){//1:1 메세지
                                                        if(smsg_func(chatData,i)==0) continue;
                                                        else;
                                                }
                                                for(j=0;j<MAX_CLIENT;j++){//메세지 보내기
                                                        if(list_c[i].socket_num !=INVALID_SOCK)
                                                                if(j!=i)

write(list_c[j].socket_num,chatData,sizeof(chatData));
                                                }


                                        }
                                }
                        }

                }
}
```

2) 클라이언트 코드 ccli.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/select.h>

#define CHATDATA 1024

char quit[]="/quit₩n";
char list[]="/list₩n";
char smsg[]="/smsg";
char smsg_quit[]="/smsg_quit₩n";
```

```
void
chatting(int sockfd, int maxfdp1, fd_set rset, char *argv[])
{
	char chatData[CHATDATA];
	char buf[CHATDATA];
	int n;

	while(1){
		FD_ZERO(&rset);
		FD_SET(0,&rset);
		FD_SET(sockfd, &rset);

		if(select(maxfdp1, &rset, (fd_set *)0, (fd_set *)0, (struct timeval *)0) <0) {
			printf("select error₩n");
			exit(1);
		}

		if(FD_ISSET(sockfd,&rset)){		//서버에게 메세지 받으면 모니터에 출력
			memset(chatData, 0, sizeof(chatData));
			if((n=read(sockfd, chatData, sizeof(chatData))) >0){
				write(1, chatData, n);
			}
		}

		if(FD_ISSET(0, &rset)) {
			memset(buf, 0, sizeof(buf));
			if((n=read(0,buf,sizeof(buf)))>0){		//키보드로 메세지 입력하기
				if(!strcmp(buf, quit)){ //입력한 메세지가 "/quit"이면 "quit"만 보내기
					write(sockfd, buf, strlen(buf));
					break;
				}
				if(!strcmp(buf,list)){	//입력한 메세지가 "/list"이면 "list"만 보내기
					write(sockfd, buf, strlen(buf));
					continue;
				}
				if(strstr(buf,smsg)!=NULL){ //입력한 메세지에 "/smsg가 들어가면 그대
로 보내기
					write(sockfd, buf, strlen(buf));
					continue;
				}
				sprintf(chatData, "[%s] %s", argv[3], buf);	//나머지는 대화명 붙여서
보내기
				write(sockfd, chatData, strlen(chatData));



			}
		}
	}
}
```

```
main(int argc,char *argv[])
{
	int sockfd;
	struct sockaddr_in servaddr;
	int maxfdp1;
	fd_set rset;

	int len;
	char chatData[CHATDATA];
	char buf[CHATDATA];
	int n;
	char* token=NULL;

	if(argc<4){
		printf("usage:%s ip_address port_number nickname₩n",argv[0]);
		exit(-1);
	}

	sockfd=socket(AF_INET, SOCK_STREAM,0);

	memset(&servaddr, 0, sizeof(servaddr));
	servaddr.sin_addr.s_addr=inet_addr(argv[1]);
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(atoi(argv[2]));


	if(connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr))==-1){
		printf("Can not connect₩n");
		return -1;
	}

	write(sockfd, argv[3], strlen(argv[3])); //send client's nickname
	maxfdp1=sockfd +1;

	chatting(sockfd, maxfdp1, rset, argv);

	close(sockfd);
	//exit(0);
}
```