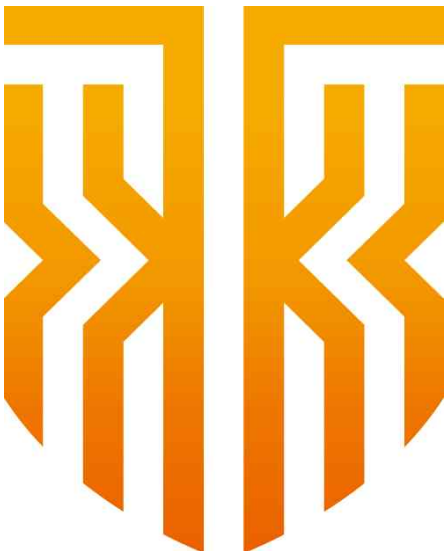
과제 5

Chatting program(Chapter 18)



목차

1. 과제 설명

가. 과제 설명

과제

다중 접속 Chatting program(Chapter 18) 을 수정하여 다음의 조건을 만족하는 프로그램을 작성하시오.
조건 : 서버는 메시지를 보낸 클라이언트에게 해당 메시지를 보내지 않는다

보고서 포함 내용 : 프로그램의 설명(의도, 해결 방법 등), 프로그램 소스 코드, 중요 코드 설명, 실행 결과 캡쳐

나. 명령 설명

1) 클라이언트 : ./cclnt 127.0.0.1 9190 name
   가) argv[0] = cclnt // 파일명
   나) argv[1] = 127.0.0.1 // 로컬 주소
   다) argv[2] = 9190 // 포트 번호
   라) argv[3] = name // 클라이언트명

2) 서버 : ./cserv 9190
   가) avgv[0] = file_server // 파일 이름
   나) argv[1] = 9190 // 로컬 주소

3) 프로그램 설명
   가) client 부분에 argv[3] 저장시키는 **문자 배열**을 sprintf로 만들고(name 처럼)
   나) recv부분에 if(!strstr(name_msg, **문자배열**)) 처리
   다) 이유
      (1) argv[3] : 클라이언트명을 받는다.
      (2) 메시지를 보낸 클라이언트를 파악하기 위해서는 클라이언트의 이름을 확인해야 한다.

4) 해결방법
   가) 클라이언트에서 받은 파일명을 받고 각각의 클라이언트를 구분하는 일은 어렵고 효율적이지 않다. 따라서 자신이 서버로 보낸 argv[3]를 이용해 recv_msg로부터 자신의 이름과 같은 경우에는 name을 받지 않는 방법을 통해 확인해보려고 한다.

2. 코드 설명

가. 전체 코드

1) chat_server.c

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

#define BUF_SIZE 100
#define MAX_CLNT 256

void * handle_clnt(void * arg);
void send_msg(char * msg, int len);
void error_handling(char * msg);

int clnt_cnt=0;
int clnt_socks[MAX_CLNT];
pthread_mutex_t mutx;

int main(int argc, char *argv[])
{
	int serv_sock, clnt_sock;
	struct sockaddr_in serv_adr, clnt_adr;
	int clnt_adr_sz;
	pthread_t t_id;
	if(argc!=2) {
		printf("Usage : %s <port>₩n", argv[0]);
		exit(1);
	}

	pthread_mutex_init(&mutx, NULL);
	serv_sock=socket(PF_INET, SOCK_STREAM, 0);

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

	if(bind(serv_sock, (struct sockaddr*) &serv_adr, sizeof(serv_adr))==-1)
		error_handling("bind() error");
	if(listen(serv_sock, 5)==-1)
		error_handling("listen() error");

	while(1)
	{
```

```
		clnt_adr_sz=sizeof(clnt_adr);
		clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_adr,&clnt_adr_sz);

		pthread_mutex_lock(&mutx);
		clnt_socks[clnt_cnt++]=clnt_sock;
		pthread_mutex_unlock(&mutx);

		pthread_create(&t_id, NULL, handle_clnt, (void*)&clnt_sock);
		pthread_detach(t_id);
		printf("Connected client IP: %s ₩n", inet_ntoa(clnt_adr.sin_addr));
	}
	close(serv_sock);
	return 0;
}

void * handle_clnt(void * arg)
{
	int clnt_sock=*((int*)arg);
	int str_len=0, i;
	char msg[BUF_SIZE];

	while((str_len=read(clnt_sock, msg, sizeof(msg)))!=0)
		send_msg(msg, str_len);

	pthread_mutex_lock(&mutx);
	for(i=0; i<clnt_cnt; i++)   // remove disconnected client
	{
		if(clnt_sock==clnt_socks[i])
		{
			while(i++<clnt_cnt-1)
				clnt_socks[i]=clnt_socks[i+1];
			break;
		}
	}
	clnt_cnt--;
	pthread_mutex_unlock(&mutx);
	close(clnt_sock);
	return NULL;
}
void send_msg(char * msg, int len)   // send to all
{
	int i;
	pthread_mutex_lock(&mutx);
	for(i=0; i<clnt_cnt; i++)
		write(clnt_socks[i], msg, len);
	pthread_mutex_unlock(&mutx);
}
void error_handling(char * msg)
{
```

```
        fputs(msg, stderr);
        fputc('₩n', stderr);
        exit(1);
}
```

2) chat_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <pthread.h>

#define BUF_SIZE 100
#define NAME_SIZE 20

void * send_msg(void * arg);
void * recv_msg(void * arg);
void error_handling(char * msg);

char name[NAME_SIZE]="[DEFAULT]";
```

1) char pass_me[NAME_SIZE]="[ ]";

```
char msg[BUF_SIZE];

int main(int argc, char *argv[])
{
        int sock;
        struct sockaddr_in serv_addr;
        pthread_t snd_thread, rcv_thread;
        void * thread_return;
        if(argc!=4) {
                printf("Usage : %s <IP> <port> <name>₩n", argv[0]);
                exit(1);
         }
```

//2) pass_me = argv[3];

sprintf(pass_me, "[%s]", argv[3]);

```
        sprintf(name, "[%s]", argv[3]);
        sock=socket(PF_INET, SOCK_STREAM, 0);

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

        if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error");

        pthread_create(&snd_thread, NULL, send_msg, (void*)&sock);
        pthread_create(&rcv_thread, NULL, recv_msg, (void*)&sock);
        pthread_join(snd_thread, &thread_return);
        pthread_join(rcv_thread, &thread_return);
```

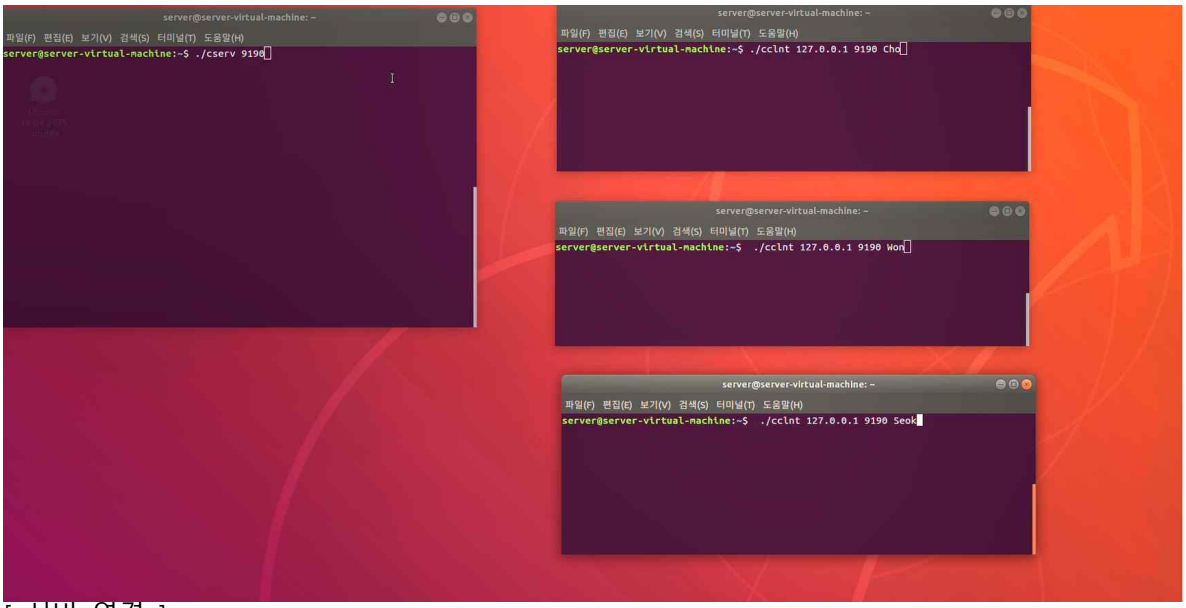
```
	close(sock);
	return 0;
}

void * send_msg(void * arg)   // send thread main
{
	int sock=*((int*)arg);
	char name_msg[NAME_SIZE+BUF_SIZE];
	while(1)
	{
		fgets(msg, BUF_SIZE, stdin);
		if(!strcmp(msg,"q₩n")||!strcmp(msg,"Q₩n"))
		{
			close(sock);
			exit(0);
		}
		sprintf(name_msg,"%s %s", name, msg);
		write(sock, name_msg, strlen(name_msg));
	}
	return NULL;
}

void * recv_msg(void * arg)   // read thread main
{
	int sock=*((int*)arg);
	char name_msg[NAME_SIZE+BUF_SIZE];
	int str_len;
	while(1)
	{
		str_len=read(sock, name_msg, NAME_SIZE+BUF_SIZE-1);
		if(str_len==-1)
			return (void*)-1;
		name_msg[str_len]=0;
		//char* e = strtok(name_msg ,"]");
		//char* s = strtok(e, "[");
		3) if(!strstr(name_msg, pass_me))
			fputs(name_msg, stdout);
	}
	return NULL;
}

void error_handling(char *msg)
{
	fputs(msg, stderr);
	fputc('₩n', stderr);
	exit(1);
}
```

나. 중요코드 설명

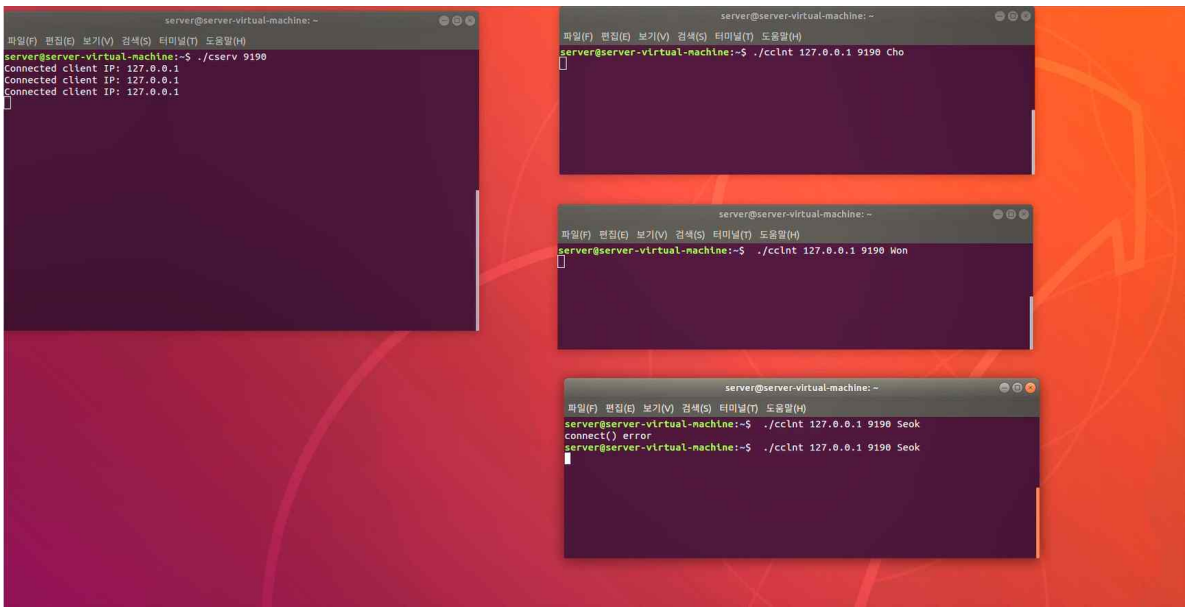
1) client에서 이름을 받기 위해 선언했던 name 배열과 같이 pass_me라는 이름의 배열을 선언한다.
2) pass_me에 argv[3]를 입력하기 위해 sprintf을 입력받는다. "[%s]"의 형태로 입력받은 이유는 name과 pass_me를 비교하여 추후에 recv에서 출력할 때 서로 비교하여 제외하기 위해 선택하였다.
3) if(!strstr(name_msg, pass_me)) fputs(name_msg, stdout);에서 strstr이라는 메서드를 통해 2)에서 name과 pass_me를 비교하는 과정을 통해 name_msg에 pass_me가 포함되지 않을 경우에만 메시지와 이름을 출력시키도록 구현하였다.

다. 실행한 화면 및 전송된 파일 확인 화면 캡쳐

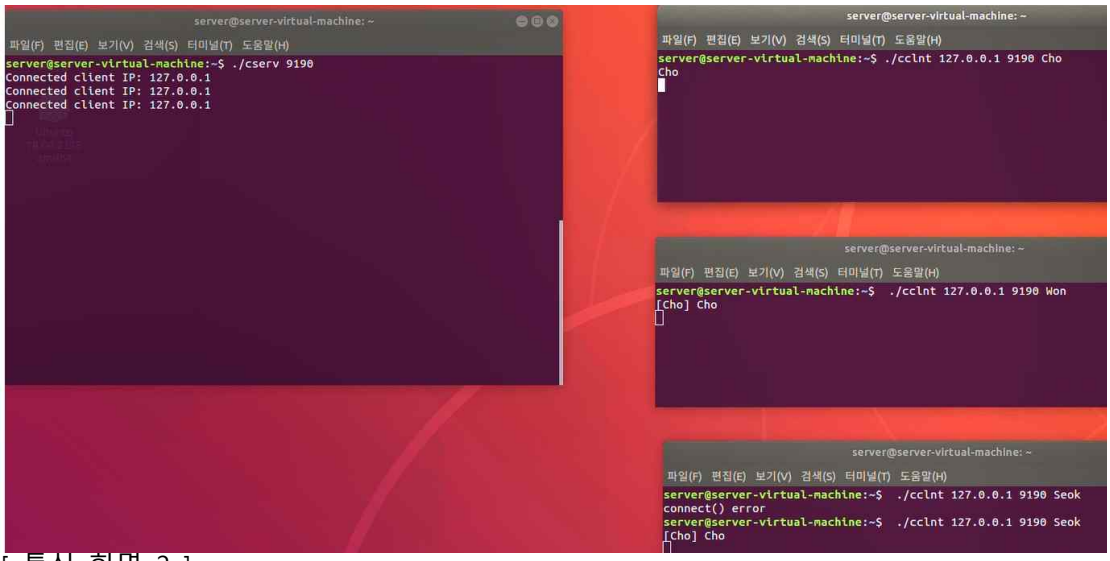
[초기 상태]



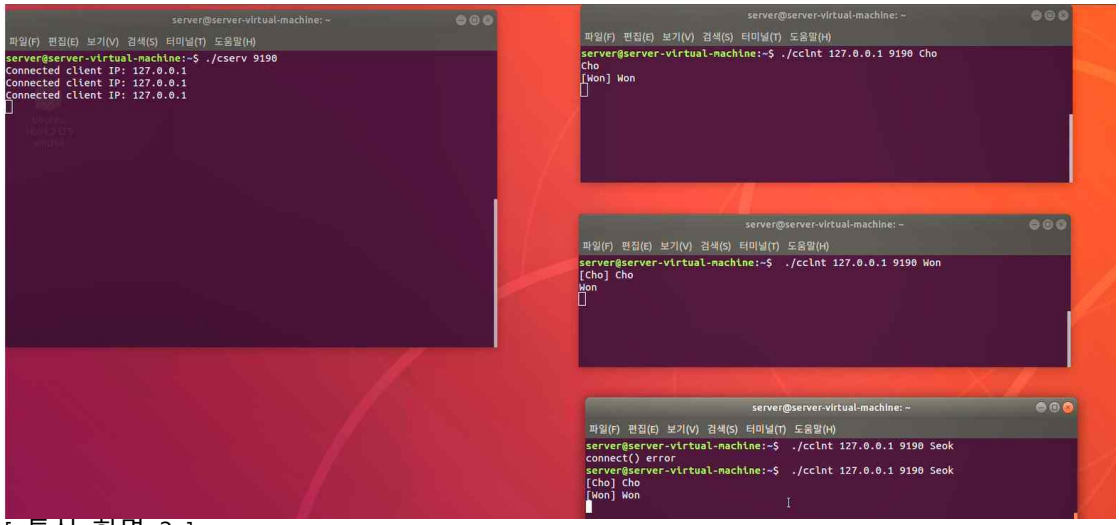
[서버 연결]



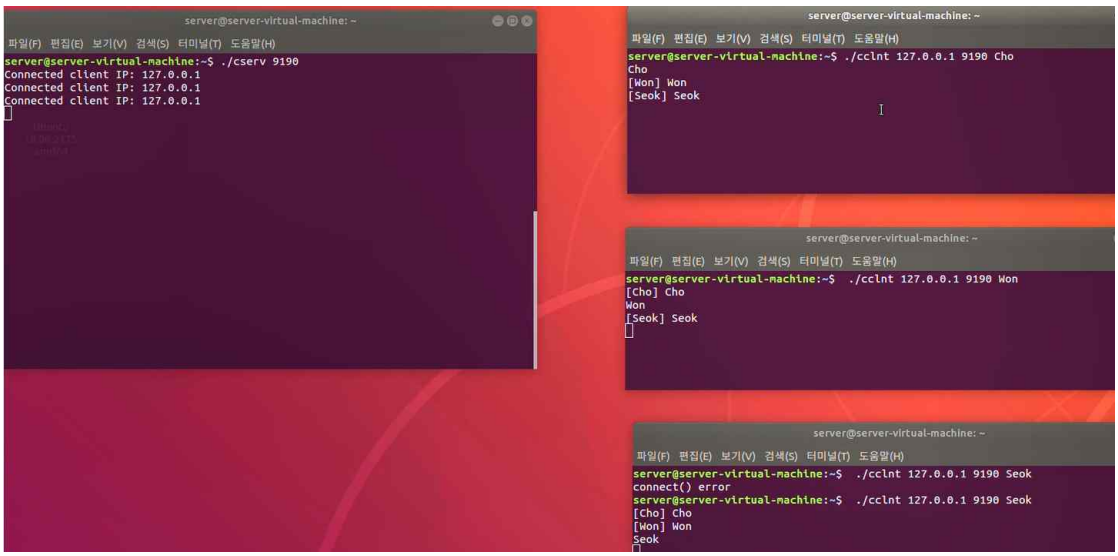
[통신 화면 1]



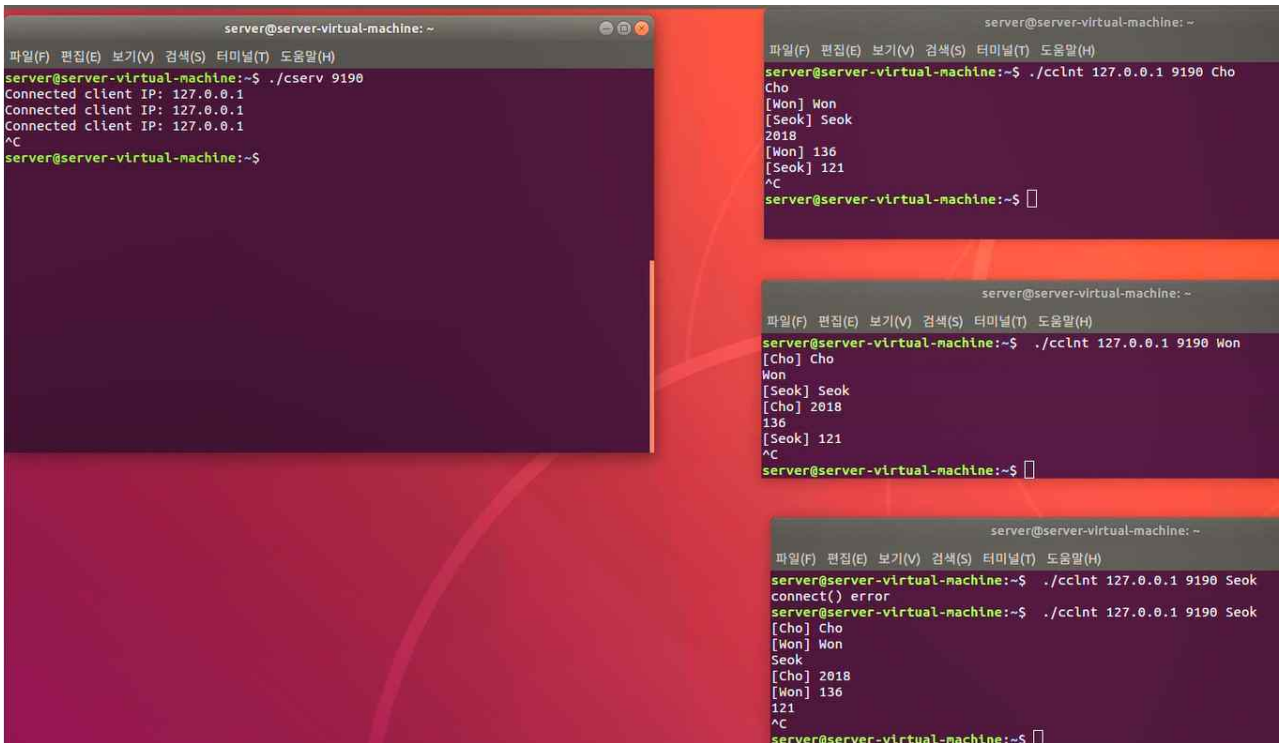
[통신 화면 2]



[통신 화면 3]



[결과 화면]


server@server-virtual-machine: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
server@server-virtual-machine:~$ ./cserv 9190
Connected client IP: 127.0.0.1
Connected client IP: 127.0.0.1
Connected client IP: 127.0.0.1
^C
server@server-virtual-machine:~$
server@server-virtual-machine: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
server@server-virtual-machine:~$ ./cclnt 127.0.0.1 9190 Cho
Cho
[Won] Won
[Seok] Seok
2018
[Won] 136
[Seok] 121
^C
server@server-virtual-machine:~$
server@server-virtual-machine: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
server@server-virtual-machine:~$ ./cclnt 127.0.0.1 9190 Won
[Cho] Cho
Won
[Seok] Seok
[Cho] 2018
136
[Seok] 121
^C
server@server-virtual-machine:~$
server@server-virtual-machine: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
server@server-virtual-machine:~$ ./cclnt 127.0.0.1 9190 Seok
connect() error
server@server-virtual-machine:~$ ./cclnt 127.0.0.1 9190 Seok
[Cho] Cho
[Won] Won
Seok
[Cho] 2018
[Won] 136
121
^C
server@server-virtual-machine:~$