**Introduction to Computer Networks**

**Lab 05: 채팅 서버 구현**

## 1. 실습의 목표

본 실습에서는 채팅 서버를 직접 구현해본다. 채팅 서버의 가장 기본적인 기능은, 어떤 클라이언트로부터 메시지를 받으면 **그 메시지를 보낸 클라이언트를 제외한 나머지 클라이언트들에게 보내주는 것이다.** 본 실습을 마치고 나면, 소켓을 이용한 채팅 프로그램 구현의 기본 단계를 마치게 되고, 이후 실습에서는 기능을 추가하는 형태로 진행된다. 따라서, 반드시 코드를 이해할 수 있도록 한다.

## 2. 작성해 볼 프로그램

**chatserver**: socket을 이용한 채팅 서버 프로그램.

## 3. 프로그램 예제 설명

아래의 예제를 따라 chatserver.cc 프로그램을 작성한다. 음영처리 된 부분이 기존의 서버 코드에서 달라진 부분이다.

```
// chatserver.cc -- a simple chat server.

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>

#define MAXDATASIZE 1000
#define BACKLOG 10

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

    int sockfd, newfd;
    struct addrinfo hints, *servinfo;
    struct sockaddr_storage their_addr;
    socklen_t sin_size;
    char s[INET_ADDRSTRLEN];
    int rv;

    // for select()
    fd_set master;
    fd_set read_fds;
```

```
    int fdmax;

    char buf[MAXDATASIZE];
    int numbytes;

    if(argc != 2) {
        printf("usage: server portnum\n");
        exit(1);
    }

    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;    // use my IP

    if((rv = getaddrinfo(NULL, argv[1], &hints, &servinfo)) != 0) {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
        exit(1);
    }

    if((sockfd = socket(servinfo->ai_family, servinfo->ai_socktype, servinfo-
>ai_protocol)) == -1) {
        perror("server: socket");
        exit(1);
    }

    if(bind(sockfd, servinfo->ai_addr, servinfo->ai_addrlen) == -1) {
        close(sockfd);
        perror("server: bind");
        exit(1);
    }

    freeaddrinfo(servinfo);

    if(listen(sockfd, BACKLOG) == -1) {
        perror("listen");
        exit(1);
    }


    FD_SET(sockfd, &master);
    fdmax = sockfd;

    while(1) {
        read_fds = master;
        if(select(fdmax+1, &read_fds, NULL, NULL, NULL) == -1) {
            perror("select");
            exit(1);
        }

        for(int i=0; i<=fdmax; i++) {
            if(FD_ISSET(i, &read_fds)) {
                if(i == sockfd) {
                    sin_size = sizeof their_addr;
                    newfd = accept(sockfd, (struct sockaddr*)&their_addr,
&sin_size);

                    if(newfd == -1) {
```

```
                    perror("accept");
                } else {
                    FD_SET(newfd, &master);
                    if(newfd > fdmax) {
                        fdmax = newfd;
                    }
                    printf("selectserver: new connection from %s on socket %d\n",
                        inet_ntop(their_addr.ss_family, &((struct
sockaddr_in*)&their_addr)->sin_addr, s, sizeof s), newfd);
                }
            }
            else {
                if((numbytes = recv(i, buf, sizeof buf, 0)) <= 0) {
                    if(numbytes == 0) {
                        printf("selectserver: socket %d hung up\n", i);
                    } else {
                        perror("recv");
                    }
                    close(i);
                    FD_CLR(i, &master);
                }
                else {
                    buf[numbytes] = '\0';
                    printf("server received \"%s\" from socket %d.\n", buf, i);

                    for(int j=0; j<=fdmax; j++) {
                        if(j==sockfd) continue;
                        if(j==i) continue;
                        if(!FD_ISSET(j, &master)) continue;
                        //printf("sending to socket %d\n", j);
                        if(send(j, buf, strlen(buf), 0) == -1) {
                            perror("send");
                        }
                    }
                }
            }
        }
    }
}
```

#### 4. 프로그램 예제 설명

아래에는 chatserver.cc의 내용 중 전과 달라진 부분을 라인별로 설명하였다.

```
                    for(int j=0; j<=fdmax; j++) {
                        if(j==sockfd) continue;
                        if(j==i) continue;
                        if(!FD_ISSET(j, &master)) continue;
                        //printf("sending to socket %d\n", j);
                        if(send(j, buf, strlen(buf), 0) == -1) {
                            perror("send");
                        }
                    }
```

클라이언트로부터 메시지를 받았을 때, 이 메시지를 다른 소켓에 보내주는 것이 서버의 임무지만, 다음의 소켓들은 제외시켜야 한다.

- 새로운 클라이언트를 기다리는 소켓 (sockfd)
- 메시지를 보낸 클라이언트 (i)
- master에 등록되어있지 않은 소켓들 ( if(!FD_ISSET(j, &master) )

따라서 0부터 fdmax까지의 모든 소켓 중 이들 소켓을 제외하고는 메시지를 보내주면 된다.

**5. 컴파일 및 실행**

채팅 서버를 컴파일하여 실행시키고, 지난 랩에서 작성한 client를 여러 개 띄워 접속한 다음 채팅이 잘 되는지 해보도록 한다.

- 프로그램을 컴파일 하여 실행파일을 얻는다.

```
$ g++ -o chatserver chatserver.cc
```

- **세 개**의 ssh 창을 띄운다. 먼저 **chatserver**를 실행시킨다.

```
$ ./chatserver 55555
```

- **lab5_client 두 개를 실행시켜서 서버에 접속한다**. 클라이언트를 실행시킨다.

```
$ ./lab4_client 127.0.0.1 55555 user1
```

- 또 하나의 클라이언트를 실행시킨다.

```
$ ./lab4_client 127.0.0.1 55555 user2
```

각각의 클라이언트에서 메시지를 입력해보고 어떤 식으로 동작이 되는지 확인한다.
(만약 lab4_client를 구현하지 못했다면, selectclient를 사용해도 무방하다.)

**6. 과제**

chatserver를 다음과 같이 수정하여 lab5_server로 만든다.

- Lab3의 과제는 서버에 접속된 사용자의 수를 보여주는 것이었다. 여기서는 서버에 접속한 사용자의 리스트를 보여주는 기능을 추가할 것이다.

- 가정은, 클라이언트가 Lab4의 과제로 구현한 형태로 메시지를 보낸다는 것이다. 즉 클라이언트의 아이디가 alice일 경우 사용자가 "hello"라는 메시지를 입력하면

**alice: hello**

와 같은 형태로 메시지가 전송된다.

- 서버는 소켓 번호와 유저 아이디를 매핑한 자료구조를 관리한다. (3번은 alice, 4번은 bob, 5번은 dragon과 같은 식으로...)

- 서버가 메시지를 받으면, 다른 클라이언트에게 보내주기 전에 해당 소켓번호의 유저 아이디를 업데이트 해주도록 한다. (클라이언트가 중간에 아이디를 바꿀 수 있다고 가정하여 메시지를 받을 때마다 유저 아이디를 업데이트 해주는 것으로 한다.)

- 클라이언트가 "/user"라는 메시지를 보내는 상황을 가정하자. 그러면 해당 소켓의 유저 아이디가 alice인 경우 서버는 "alice: /user"와 같은 형태로 메시지를 받게 될 것이다. 이 경우는 특수하게 처리하여 다른 클라이언트에게 메시지를 전송하지 않고, alice에게 user의 list를 보내준다. User의 리스트는 다음과 같은 형태로 보내주면 된다.

만약 서버에 접속된 유저의 리스트가 다음과 같았다고 가정하자.

alice
bob
dragon

그러면, 스트링을 아래와 같이 만든다.

| a | l | i | c | e | \n | b | o | b | \n | d | r | a | g | o | n | \n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

\n은 줄바꿈 문자를 의미한다.

클라이언트는 이 메시지를 받으면 그대로 출력할 것이므로, 사용자의 리스트를 표시하는 효과를 가져올 것이다.

- 테스트를 위해 여러 클라이언트를 서버에 접속시킨 다음 /user 명령을 실행시켜 본다.

※ 주의 1: 이 과제를 테스트하기 위해서는 lab4_client 와 같이 유저의 아이디를 붙여서 보내는 클라이언트를 사용해야 한다. selectclient를 사용하면 테스트가 되지 않을 것이다.
※ 주의 2: **유저 아이디에는 특수문자나 공백이 들어가지 않도록 한다.**