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

/***********/
typedef struct s_client {
	int	fd;
	int	id;
	char *buf;
} t_client;

t_client	clients[1024];
int max_fd = 0, next_id = 0;
fd_set read_set, write_set, active_set;

void	err(char *msg, int exit_code) {
	if (!msg)
		msg = "Fatal error\n";
	write(2, msg, strlen(msg));
	exit(exit_code);
}

void	send_all(int except, char *msg) {
	for (int i = 0; i <= max_fd; i++)
		if (clients[i].fd > 0 && i !
=except)
			send(clients[i].fd,
msg, strlen(msg), 0);
}
/**************************/


int extract_message(char **buf, char **msg)
{
	char	*newbuf;
	int	i;

	*msg = 0;
	if (*buf == 0)
		return (0);
	i = 0;
	while ((*buf)[i])
	{
		if ((*buf)[i] == '\n')
		{
			newbuf = calloc(1, s
izeof(*newbuf) * (strlen(*buf + i + 1) + 1))
;
			if (newbuf == 0)
				return (-1);
			strcpy(newbuf, *buf
+ i + 1);
			*msg = *buf;
			(*msg)[i + 1] = 0;
			*buf = newbuf;
			return (1);
		}
		i++;
	}
	return (0);
}

char *str_join(char *buf, char *add)
{
	char	*newbuf;
	int		len;

	if (buf == 0)
		len = 0;
	else
		len = strlen(buf);
	newbuf = malloc(sizeof(*newbuf) * (l
en + strlen(add) + 1));
	if (newbuf == 0)
		return (0);
	newbuf[0] = 0;
	if (buf != 0)
		strcat(newbuf, buf);
	free(buf);
	strcat(newbuf, add);
	return (newbuf);
}


int main(int ac, char **av) {
	int sockfd, connfd; //
	struct sockaddr_in servaddr; //
	/**********/
	if (ac != 2) {
		err("Wrong number of argumen
ts\n", 1);
	}
	/***************/

	// socket create and verification
	sockfd = socket(AF_INET, SOCK_STREAM
, 0);
	if (sockfd == -1) {
		//printf("socket creation fa
iled...\n");
		//exit(0);
		err(NULL, 1);
	}
	//else
	//	printf("Socket successfully
created..\n");
	max_fd = sockfd; //
	bzero(&servaddr, sizeof(servaddr));

	// assign IP, PORT
	servaddr.sin_family = AF_INET;
	servaddr.sin_addr.s_addr = htonl(213
0706433); //127.0.0.1
	servaddr.sin_port = htons(atoi(av[1]
)); //

	// Binding newly created socket to g
iven IP and verification
	if ((bind(sockfd, (const struct sock
addr *)&servaddr, sizeof(servaddr))) != 0) {

		//printf("socket bind failed
...\n");
		//exit(0);
		err(NULL,1); //
	}
```

```
        //else
        //      printf("Socket successfully
binded..\n");
        if (listen(sockfd, 10) != 0) {
                //printf("cannot listen\n");

                //exit(0);
                err(NULL,1); //
        }
        /*****************/
        FD_ZERO(&active_set);
        FD_SET(sockfd, &active_set);
        bzero(clients, sizeof(clients));
        while (1) {
                read_set = write_set = activ
e_set;
                if(select(max_fd +1, &read_s
et, &write_set, NULL, NULL) < 0 )
                        continue;
                for (int fd = 0; fd <= max_f
d; fd++) {
                        if(!FD_ISSET(fd, &re
ad_set)) {
                                continue;
                        }
                        if (fd == sockfd) {
                                connfd= acce
pt(sockfd, NULL, NULL);
                                if (connfd <
 0) continue;
                                if (connfd >
max_fd) max_fd = connfd;
                                clients[conn
fd].fd = connfd;
                                clients[conn
fd].id = next_id++;
                                clients[conn
fd].buf = NULL;
                                FD_SET(connf
d,&active_set);
                                char msg [10
0];
                                sprintf(msg,
 "server: client %d just arrived\n", clients
[connfd].id);
                                send_all(con
nfd,msg);
                        }
                        else {
                                char buf[102
4];
                                int r = recv
(fd, buf, sizeof(buf) - 1, 0);
                                if (r <= 0)
{
                                        char
 msg [100];
                                        spri
ntf(msg, "server: client %d just left\n", cl
ients[fd].id);
                                        send
_all(fd,msg);
                                        FD_C
LR(fd,&active_set);
                                        clos
e(fd);
```

```
                                        free
(clients[fd].buf);
                                        clie
nts[fd].fd = 0;
                                }
                                else {
                                        buf[
r] = 0;
                                        clie
nts[fd].buf = str_join(clients[fd].buf, buf)
;
                                        if(!
clients[fd].buf)

err(NULL,1);
                                        char
 *msg;
                                        whil
e(extract_message(&clients[fd].buf, &msg) ==
1) {

char prefix[50];

sprintf(prefix, "client %d: ", clients[fd].i
d);

char *full = malloc(strlen(prefix)+strlen(ms
g) +1);

if (!full)

        err(NULL,1);

full[0] = 0;

strcat(full,prefix);

strcat(full, msg);

send_all(fd,full);

free(full);

free(msg);
                                        }
                                }
                        }
                }
        }
}
/***************/
        //len = sizeof(cli);
        //connfd = accept(sockfd, (struct so
ckaddr *)&cli, &len);
        //if (connfd < 0) {
    //    printf("server acccept failed...\n
");
    //    exit(0);
   // }
    //else
     //   printf("server acccept the client.
..\n");
//}
```