Computer Networks Lab (CS 356)

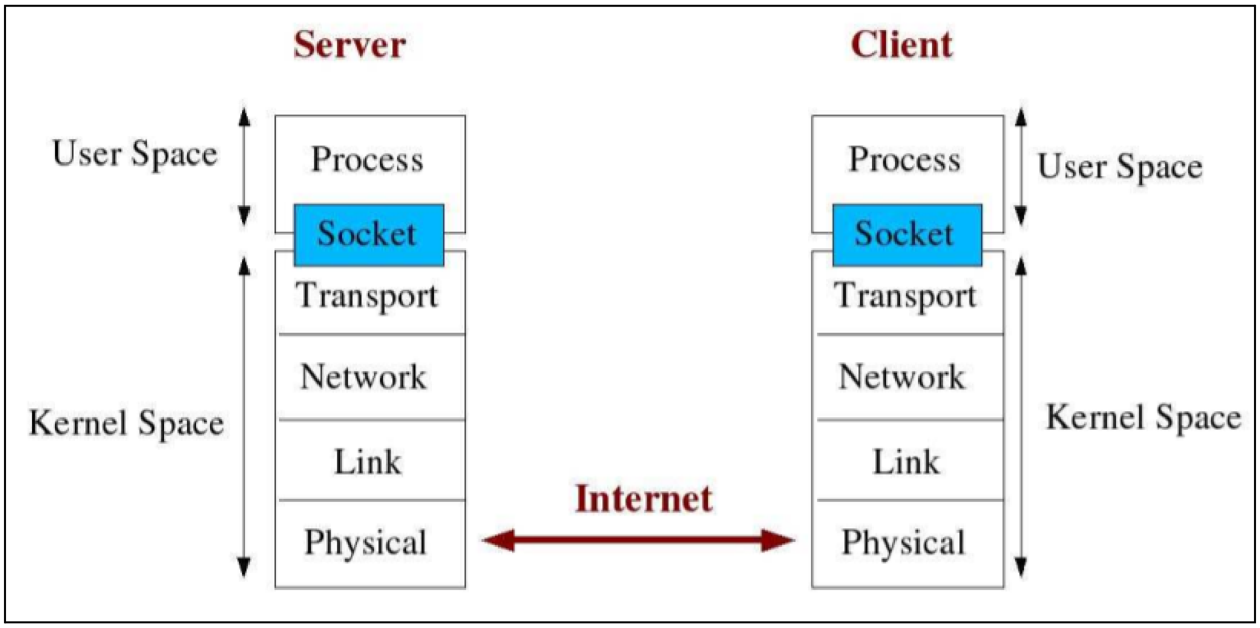
Name: Krishanu Saini
Roll No: 190001029

<u>Assignment 3</u>

Multithreaded Chatroom using socket programming

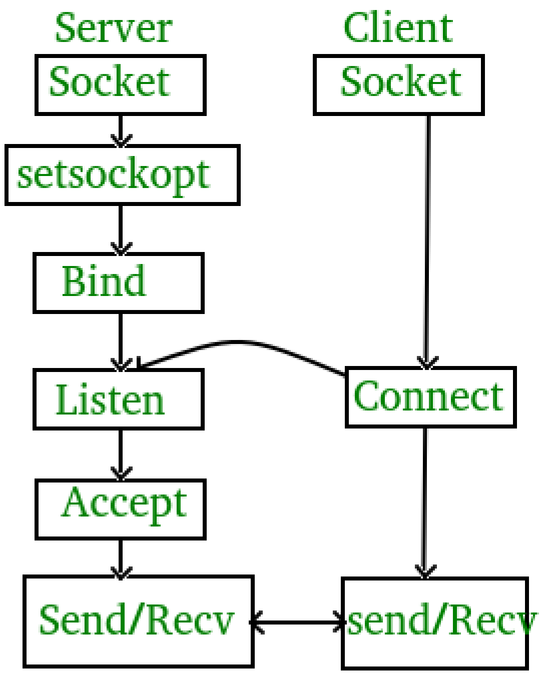
Sockets

What is a socket?
It is an interface between an application process and a transport layer.
The application process can send/receive messages to/from another application process (local or remote)via a socket.
In Unix language - a socket is a file descriptor – an integer associated with an open file.



Using socket programming we can
1. Bind the server to a port on the host machine.
2. The server listens for any connection to this port.
3. It will accept a connection from the queue.
4. It will send and receive the data messages from the client.

The client, after connecting, can send and receive data messages from the server.



Chatroom

A chatroom involves communication between multiple clients. One user sends message and all others receive it.
We create 2 files

1. server.c
2. client.c

We use pthread library to generate threads per task in the server and client. They are synchronized using mutex to prevent concurrency issues.
We run them using
Terminal-1

```
$ gcc server.c -o server -lpthread
$ ./server 8888
```

Terminal-2

```
$ gcc client.c -o client -lpthread
$ ./client localhost 8888
```

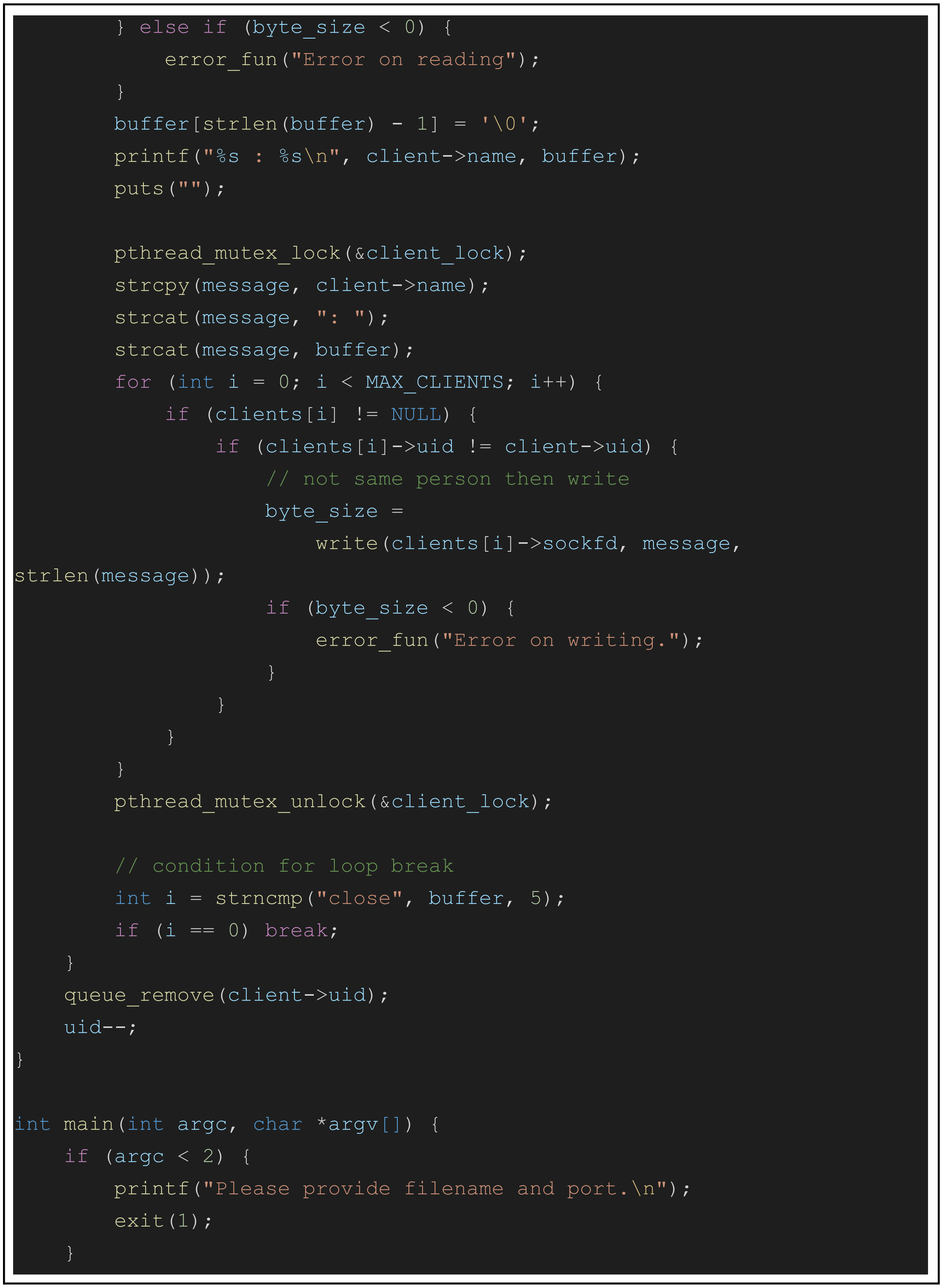
Code is given below.

Code

Server.c

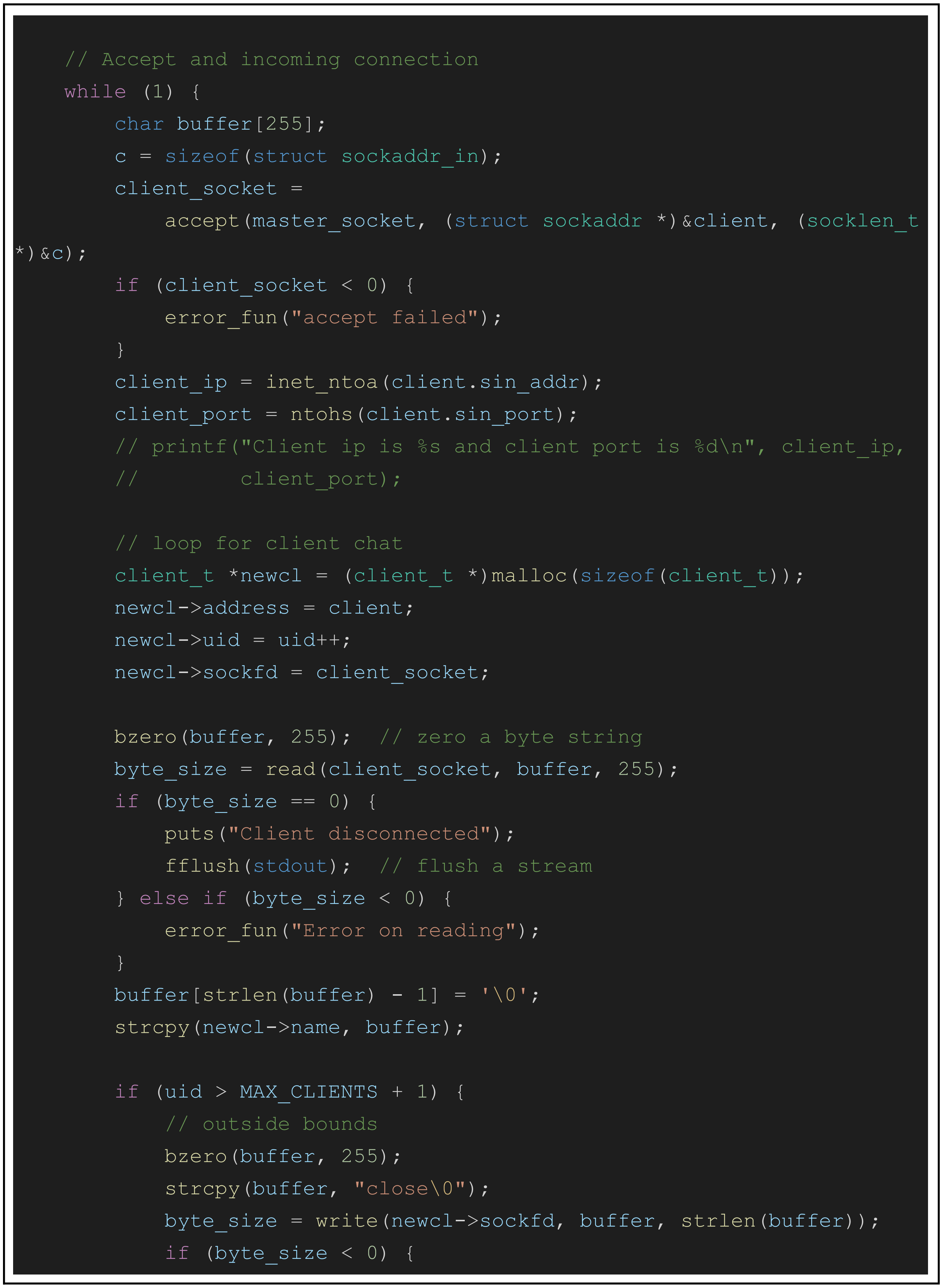
```c
#include <arpa/inet.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Assn: (A3) multithread - server
 */

const int MAX_CLIENTS = 5;

int client_count;

typedef struct {
    struct sockaddr_in address;
    int sockfd;
    int uid;
    char name[32];
} client_t;

client_t *clients[5];
pthread_mutex_t client_lock = PTHREAD_MUTEX_INITIALIZER;
char message[255];
int uid;

void error_fun(const char *msg) {
    perror(msg);
    exit(1);
```

```c
}

void queue_add(client_t *cl) {
    // add to client list
    int i = 0;
    while (i < MAX_CLIENTS) {
        if (clients[i] == NULL) {
            clients[i] = cl;
            break;
        }
        i++;
    }
}

void queue_remove(int uid) {
    int i = 0;
    while (i < MAX_CLIENTS) {
        if (clients[i] != NULL) {
            if (clients[i]->uid == uid) {
                clients[i] = NULL;
                break;
            }
        }
        i++;
    }
}

void *handle_client(void *cl) {
    client_t *client = (client_t *)cl;
    // read user input in loop
    int byte_size;
    char buffer[255];
    printf("user here: %s\n", client->name);
    while (1) {
        bzero(buffer, 255);  // zero a byte string
        byte_size = read(client->sockfd, buffer, 255);
        if (byte_size == 0) {
            puts("Client disconnected");
            fflush(stdout);  // flush a stream
```



```
        } else if (byte_size < 0) {
            error_fun("Error on reading");
        }
        buffer[strlen(buffer) - 1] = '\0';
        printf("%s : %s\n", client->name, buffer);
        puts("");

        pthread_mutex_lock(&client_lock);
        strcpy(message, client->name);
        strcat(message, ": ");
        strcat(message, buffer);
        for (int i = 0; i < MAX_CLIENTS; i++) {
            if (clients[i] != NULL) {
                if (clients[i]->uid != client->uid) {
                    // not same person then write
                    byte_size =
                        write(clients[i]->sockfd, message,
strlen(message));
                    if (byte_size < 0) {
                        error_fun("Error on writing.");
                    }
                }
            }
        }
        pthread_mutex_unlock(&client_lock);

        // condition for loop break
        int i = strncmp("close", buffer, 5);
        if (i == 0) break;
    }
    queue_remove(client->uid);
    uid--;
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Please provide filename and port.\n");
        exit(1);
    }
```

```c
    int master_socket, client_socket, c, client_port, portno, byte_size;
    char *client_ip;
    struct sockaddr_in server, client;
    pthread_t tid[MAX_CLIENTS + 1];
    uid = 1;

    // Create socket
    master_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (master_socket == -1) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    bzero((char *)&server, sizeof(server));
    portno = atoi(argv[1]);

    // Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(portno);

    // Bind
    if (bind(master_socket, (struct sockaddr *)&server, sizeof(server))
< 0) {
        error_fun("bind failed");
    }
    printf("bind done\n");

    // Listen
    // try to specify maximum of 3 pending connections for the master
socket
    if (listen(master_socket, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    client_count = 0;
    int server_len = sizeof(server);
    puts("waiting for connections\n");
```

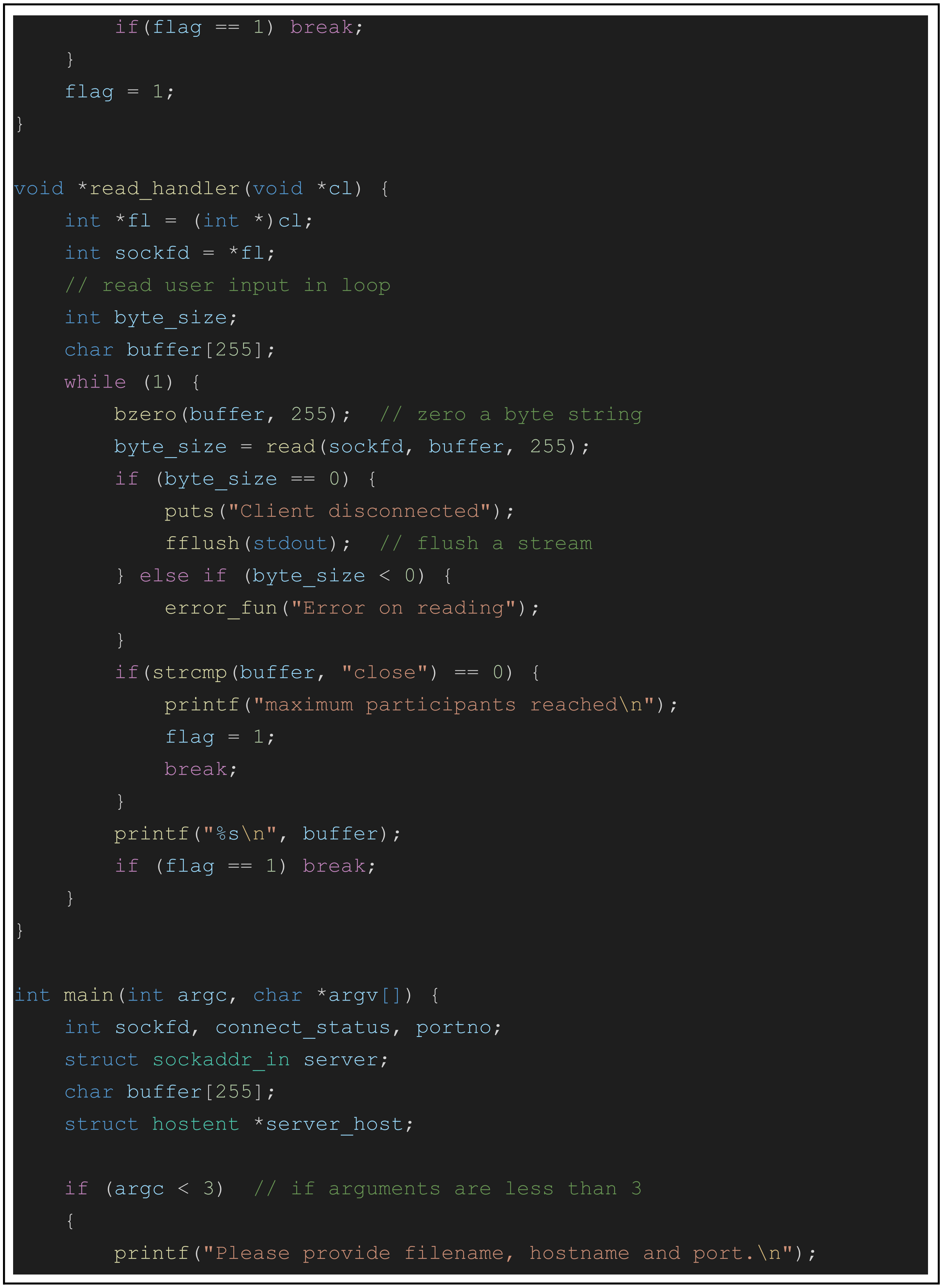


```
    // Accept and incoming connection
    while (1) {
        char buffer[255];
        c = sizeof(struct sockaddr_in);
        client_socket =
            accept(master_socket, (struct sockaddr *)&client, (socklen_t
*)&c);
        if (client_socket < 0) {
            error_fun("accept failed");
        }
        client_ip = inet_ntoa(client.sin_addr);
        client_port = ntohs(client.sin_port);
        // printf("Client ip is %s and client port is %d\n", client_ip,
        //        client_port);

        // loop for client chat
        client_t *newcl = (client_t *)malloc(sizeof(client_t));
        newcl->address = client;
        newcl->uid = uid++;
        newcl->sockfd = client_socket;

        bzero(buffer, 255);  // zero a byte string
        byte_size = read(client_socket, buffer, 255);
        if (byte_size == 0) {
            puts("Client disconnected");
            fflush(stdout);  // flush a stream
        } else if (byte_size < 0) {
            error_fun("Error on reading");
        }
        buffer[strlen(buffer) - 1] = '\0';
        strcpy(newcl->name, buffer);

        if (uid > MAX_CLIENTS + 1) {
            // outside bounds
            bzero(buffer, 255);
            strcpy(buffer, "close\0");
            byte_size = write(newcl->sockfd, buffer, strlen(buffer));
            if (byte_size < 0) {
```

```c
                error_fun("Error on closing.");
            }
            continue;
        }

        queue_add(newcl);
        pthread_create(&tid[uid], NULL, &handle_client, (void *)newcl);
    }

    for (int i = 1; i < uid; i++) {
        pthread_join(tid[i], NULL);
    }

    pthread_mutex_destroy(&client_lock);
    close(client_socket);
    close(master_socket);
    return 0;
}
```

Client

```c
#include <arpa/inet.h>  //inet_addr
#include <netdb.h>      //gethostbyname
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>  //strlen
#include <string.h>  //strlen
#include <sys/socket.h>
#include <unistd.h>
/*
 * Name: Krishanu Saini
 * Roll: 190001029
 * Assn: (A3) multithread - client
 */
void error_fun(const char *msg) {
    perror(msg);
    exit(1);
}

int flag = 0;

void *write_handler(void *cl) {
    int *fl = (int *)cl;
    int sockfd = *fl;
    // read user input in loop
    int byte_size;
    char buffer[255];
    while (1) {
        bzero(buffer, 255);
        fgets(buffer, 255, stdin);
        // Send some data
        if (send(sockfd, buffer, strlen(buffer), 0) < 0) {
            error_fun("Send failed");
        }
        int i = strncmp("close", buffer, 5);
        if (i == 0) break;
```

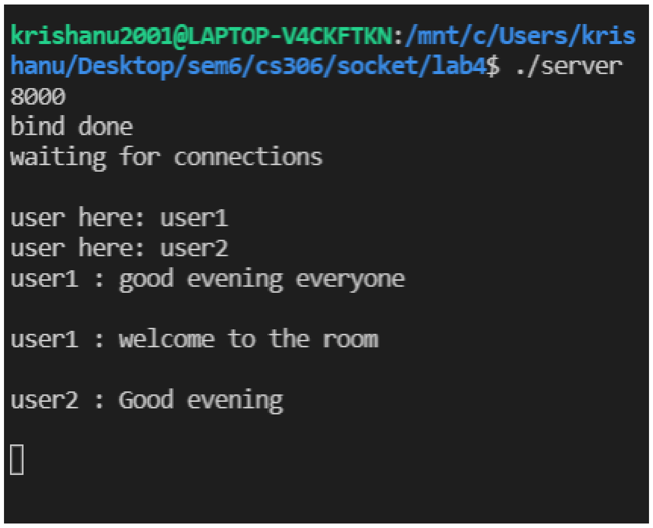


```
        if(flag == 1) break;
    }
    flag = 1;
}

void *read_handler(void *cl) {
    int *fl = (int *)cl;
    int sockfd = *fl;
    // read user input in loop
    int byte_size;
    char buffer[255];
    while (1) {
        bzero(buffer, 255);  // zero a byte string
        byte_size = read(sockfd, buffer, 255);
        if (byte_size == 0) {
            puts("Client disconnected");
            fflush(stdout);  // flush a stream
        } else if (byte_size < 0) {
            error_fun("Error on reading");
        }
        if(strcmp(buffer, "close") == 0) {
            printf("maximum participants reached\n");
            flag = 1;
            break;
        }
        printf("%s\n", buffer);
        if (flag == 1) break;
    }
}

int main(int argc, char *argv[]) {
    int sockfd, connect_status, portno;
    struct sockaddr_in server;
    char buffer[255];
    struct hostent *server_host;

    if (argc < 3)  // if arguments are less than 3
    {
        printf("Please provide filename, hostname and port.\n");
```

```c
        exit(1);
    }

    portno = atoi(argv[2]);
    // Create socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        error_fun("Socket can not be created");
    }
    server_host = gethostbyname(argv[1]);
    if (server_host == NULL) {
        error_fun("Error, No such host");
    }

    bzero((char *)&server, sizeof(server));
    bcopy((char *)server_host->h_addr, (char *)&server.sin_addr.s_addr,
          server_host->h_length);
    // server.sin_addr.s_addr =
inet_addr("142.250.194.196");//INADDR_ANY
    server.sin_family = AF_INET;
    server.sin_port = htons(portno);

    // Connect to remote server

    connect_status =
        connect(sockfd, (struct sockaddr *)&server, sizeof(server));
    printf("Connecting...%d", connect_status);
    if (connect_status < 0) {
        error_fun("connection error");
    }

    printf("\n=== Welcome to Chat Channel ==\n");

    printf("enter your name: ");
    bzero(buffer, 255);         // zero a byte string
    fgets(buffer, 255, stdin);  // input of characters and strings
    // Send some data
    if (send(sockfd, buffer, strlen(buffer), 0) < 0) {
        error_fun("Send failed\n");
    }

    pthread_t tid[2];

    // loop for client group chat
    pthread_create(&tid[0], NULL, &write_handler, (void *)&sockfd);
    pthread_create(&tid[1], NULL, &read_handler, (void *)&sockfd);

    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);

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

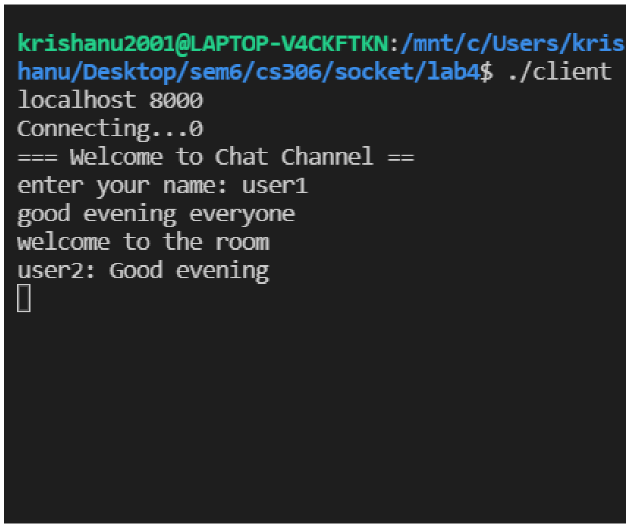
Output

Server



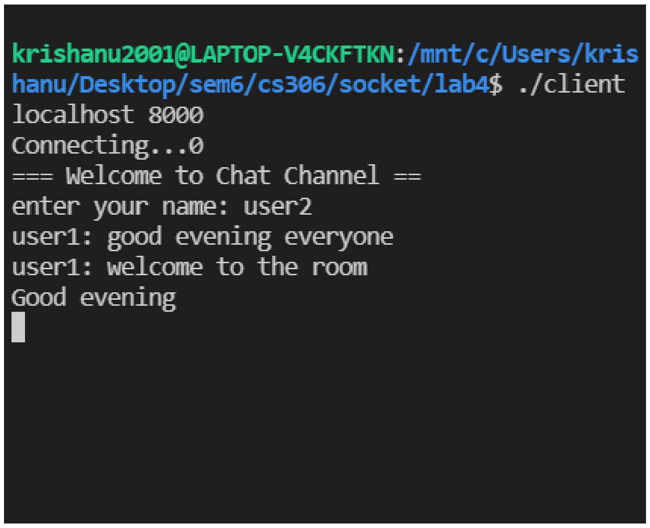
```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem6/cs306/socket/lab4$ ./server 8000
bind done
waiting for connections

user here: user1
user here: user2
user1 : good evening everyone

user1 : welcome to the room

user2 : Good evening

```

user1



```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem6/cs306/socket/lab4$ ./client localhost 8000
Connecting...0
=== Welcome to Chat Channel ==
enter your name: user1
good evening everyone
welcome to the room
user2: Good evening
```

user2



```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem6/cs306/socket/lab4$ ./client localhost 8000
Connecting...0
=== Welcome to Chat Channel ==
enter your name: user2
user1: good evening everyone
user1: welcome to the room
Good evening
```