Computer Architecture and Operating Systems

Assignment 2

Implementation

```c
int main() {
    int sockfd, newsockfd, len;
    struct sockaddr_un local, remote;
    pthread_t thread;

    // Create a socket file descriptor

    sockfd = socket(AF_UNIX, SOCK_STREAM, 0);

    if (sockfd == -1) {
        perror("socket");
    }

    // Add information about the Unix domain address

    local.sun_family = AF_UNIX;
    strcpy(local.sun_path, SOCK_PATH);
    unlink(local.sun_path);
    len = strlen(local.sun_path) + sizeof(local.sun_family);

    // Bind the socket to an address

    if (bind(sockfd, (struct sockaddr *)&local, len) == -1) {
        perror("bind");
        exit(1);
    }

    // Listen to incoming connections from client programs

    if (listen(sockfd, 5) == -1) {
        perror("listen");
        exit(1);
    }

    printf("\n\n___WELCOME TO THE CHAT___\n\n");

    while (1) {
        socklen_t remotelen = sizeof(remote);

        // We create a new socked fd which is connected to the client

        newsockfd = accept(sockfd, (struct sockaddr *)&remote, &remotelen);
```

```
    while (1) {
        socklen_t remotelen = sizeof(remote);

        // We create a new socked fd which is connected to the client

        newsockfd = accept(sockfd, (struct sockaddr *)&remote, &remotelen);

        if(newsockfd == -1) {
            perror("accept");
            exit(1);
        }

        // Check for maximum users

        if (user_count >= MAX_USERS) {
            printf("FAILED: Maximum users connected\n");
            close(newsockfd);
            continue;

        }

        // Set user details

        user_t *new_user = (user_t *) malloc(sizeof(user_t));
        new_user->userid = ++user_count;
        new_user->sockfd = newsockfd;
        new_user->addr = remote;

        // Add user to list of users

        for (int i = 0; i < MAX_USERS; i++) {
            if (list_of_users[i] == NULL) {
                list_of_users[i] = new_user;
                break;
            }
        }

        // Create a new thread for the user

        pthread_create(&thread, NULL, &welcome_user, (void *) new_user);

    }
}
```

Errors

Error checking added for:

1. send()
2. recv()
3. socket()
4. bind()
5. connect()
6. Invalid usernames
7. Maximum users

How to compile and test the program

1. make compile to compile the files
2. ./server to run the server
3. ./client to run the client
4. Open new terminal windows to have multiple clients
5. Start chatting
6. Remember to follow the instructions that are displayed when connected

Inputs the user should give

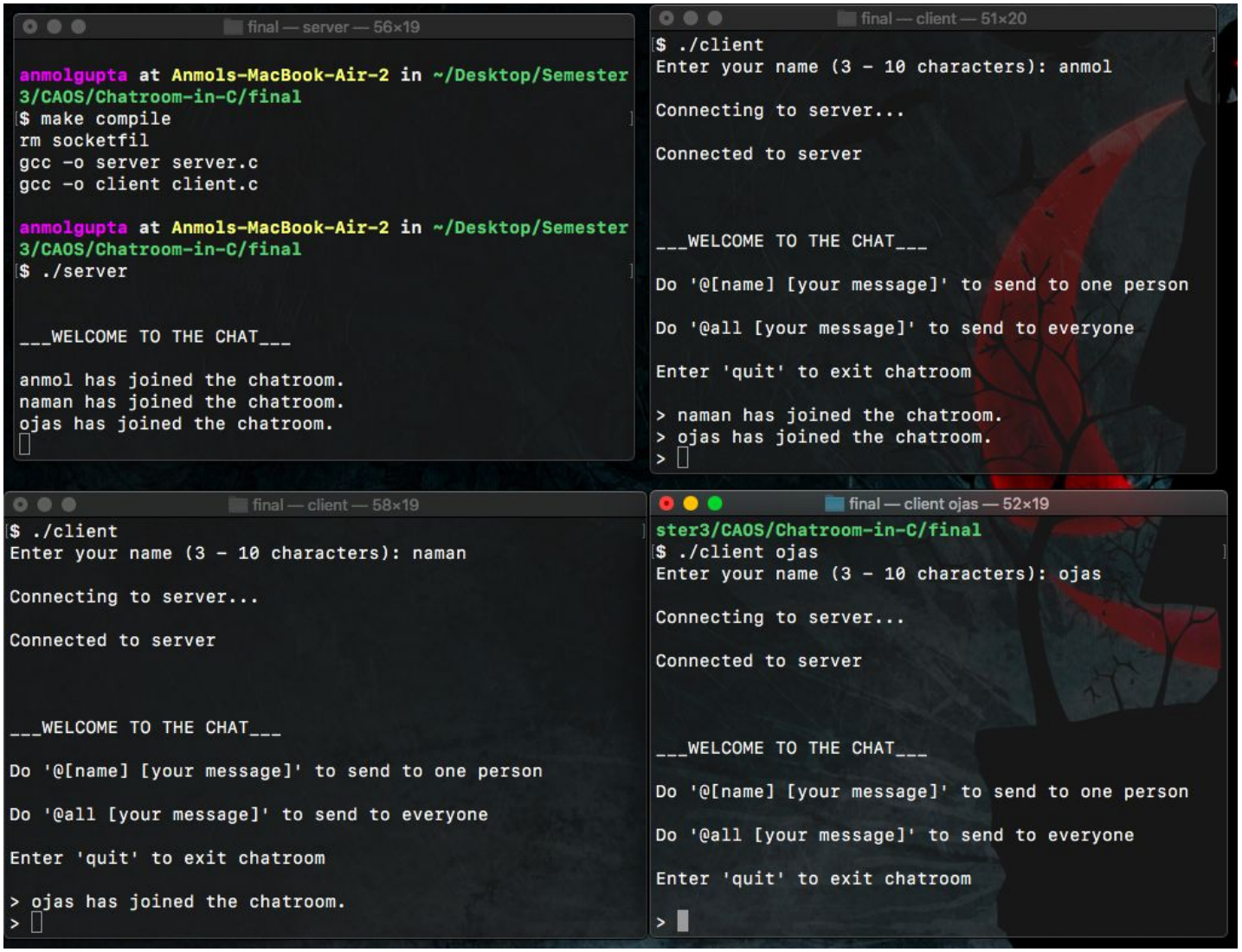
```
anmolgupta at Anmols-MacBook-Air-2 in ~/Desktop/Semester3
/CAOS/Chatroom-in-C/final
$ ./server


___WELCOME TO THE CHAT___
```

```
anmolgupta at Anmols-MacBook-Air-2 in ~/Desktop/Semester3/C
OS/Chatroom-in-C/final
$ ./client
Enter your name (3 - 10 characters):
```

No input is required to get the server running. The program asks for username after running the client.

The chat then begins normally.

Expected output



Expected output