# GEBZE TECHNICAL UNIVERSITY

**SPRING 2025**

**CSE344**

**Systems Programming**

**Final Project**

**REPORT**

**Feyza Nur Küçük**

**1801042618**

## Introduction

This project implements a multithreaded TCP server and client communication system in C, designed to handle concurrent client connections. The server enables real-time messaging between users and provides a suite of commands for interactions in a chat-like environment. The main objective is to demonstrate socket programming, thread management, synchronization, and command parsing in a client-server architecture.

## Code Explanation

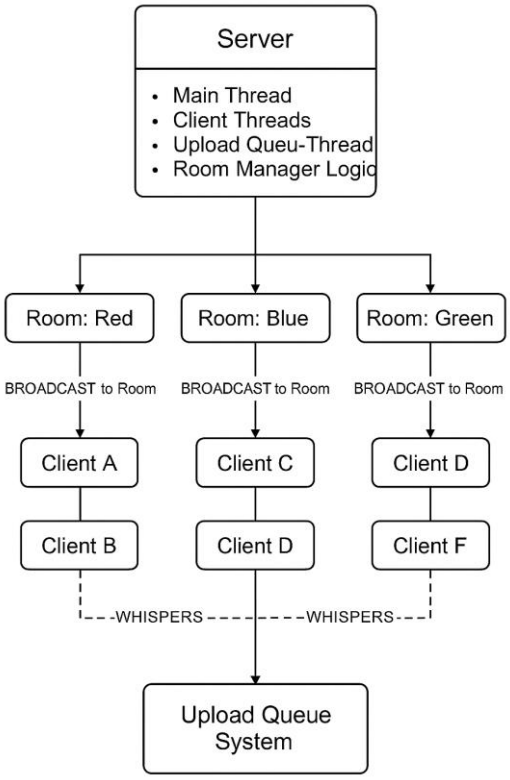
### 1. Features

- Multithreaded Server: Supports at least 15 concurrent clients using POSIX threads.
- Username Validation: Clients must provide a unique username (alphanumeric, max 16 characters).
- Rooms: Clients can join or leave chat rooms with `/join` and `/leave` commands.
- Broadcast Messaging: Use `/broadcast` to send a message to all clients in the same room.
- Private Messaging: Use `/whisper <user>` to send a private message to a specific user.
- File Transfer: The `/sendfile` command allows file uploads using a queued mechanism.
- Upload Queue: Thread-safe file upload queue using mutexes and semaphores.
- Logging: Logs important events with timestamps such as connections, disconnections, and message exchanges.
- Graceful Shutdown: Handles `SIGINT` to cleanly shut down the server and close all connections.

### 2. System Design

The system follows a client-server architecture using TCP sockets. The server listens for incoming connections and spawns a new thread for each client. The server maintains global data structures to track connected clients, their usernames, and room associations. Thread synchronization is enforced using mutexes to prevent race conditions.
Key components include:
- Main Server Thread: Accepts incoming connections and creates a thread per client.
- Client Handler Thread: Processes client input, handles commands, and communicates with other clients.
- File Upload Queue: Manages file transfer requests using a producer-consumer approach.
- Logger: Captures and writes logs to a file with timestamps for each event.

### 2.1 Archtitecture Overview




- **Main Thread**: Initializes server, listens for connections, creates per-client threads.
- **Client Threads**: Handle individual client commands (join, whisper, file send, etc.)
- **Upload Handler Thread**: Dequeues and processes file uploads.
- **Signal Handler**: Handles SIGINT for graceful shutdown.

### 2.2 Inter-Process Communication (IPC)

This project does not use traditional IPC mechanisms like pipes or shared memory across processes. Instead, it relies on:

- **TCP Sockets**: Communication between clients and server
- **Shared Memory (within process)**:
    - Global clients[], roomList[], and upload_queue[] structures
    - Access is synchronized with pthread_mutex and sem_t semaphores

2.3 Thread Model

| Component | Type | Role |
|---|---|---|
| main() | Main thread | Initializes socket, listens, and spawns clients |

| Component | Type | Role |
|---|---|---|
| handle_client() | Thread-per-client | Handles messages, room join/leave, file commands |
| upload_queue_handler() | Detached thread | Continuously processes upload tasks |
| handle_upload() | Thread-per-upload | Executes file copying using low-level I/O |

Thread safety is ensured by:

- pthread_mutex_t user_mutex – synchronizes access to clients and roomList
- pthread_mutex_t log_mutex – serializes log writes
- sem_t empty_slots/full_slots/upload_slots – controls producer-consumer queue

### 2.4 Core Components Explained

- **main() – Server Initialization & Loop**
  - Sets up TCP socket (with SO_REUSEADDR)
  - Initializes semaphores and signal handling
  - Sets server_fd to **non-blocking mode**
  - Accepts clients, spawns a thread for each via pthread_create
- **handle_client() – Client Handler Thread**

Handles:

- CONNECT command with username uniqueness check
- /join, /leave, /broadcast, /whisper, /sendfile, /exit commands
- Calls room and upload-related helpers
- Gracefully removes user on disconnection

- **join_room_command() / leave_room_command() / remove_from_room()**
  - Assign users to virtual chat rooms
  - Dynamically allocate usernames
  - Synchronize with user_mutex
  - Logs all changes to rooms
- **broadcast_command() / whisper_command()**
  - Broadcast: sends messages to all users in the same room

- Whisper: direct message to a specified user
- All messages are logged with user and room details

🔹 **File Upload Queue System**

**enqueue_upload()**

- Adds file upload task to the queue
- Uses semaphores for producer control

**dequeue_upload()**

- Consumer-side, blocks if queue is empty
- Returns the task to upload_queue_handler()

**handle_upload()**

- Performs file copy with read() and write()
- Handles duplicate filenames with file_exists_for_user()
- Creates uploads/<user> directory structure

🔹 **broadcast_shutdown_to_clients() and signal_handler()**

- On SIGINT, all clients are notified with "SERVER_SHUTDOWN"
- Connections and resources are cleaned
- stop = 1 flag causes loops to exit cleanly

**2.5 Synchronization Summary**

| Resource | Sync Mechanism | Purpose |
|---|---|---|
| clients[] / roomList[] | user_mutex | Prevent concurrent modifications |
| Upload Queue | queue_mutex, sem_t | Bounded queue coordination |
| Logging | log_mutex | Avoid interleaved file writes |

**2.6 Error Handling & Robustness**

- Checks for file existence and size limit before upload
- Handles username conflicts during CONNECT
- Detects and manages disconnections cleanly
- All I/O errors are logged with appropriate messages

### 3. Code Structure

- `server.c`: Implements the main server logic including connection handling, thread creation, command parsing, and file transfers.
- `client.c`: Provides a command-line client interface that connects to the server, processes user input, and handles server responses.
- `makefile`: Compilation instructions for building the server and client programs.

Each client thread operates independently and accesses shared resources with proper locking mechanisms. Commands are parsed using string comparison and are routed to appropriate handlers.

### Screenshots

**Case0:**

Username is not found in whisper command.

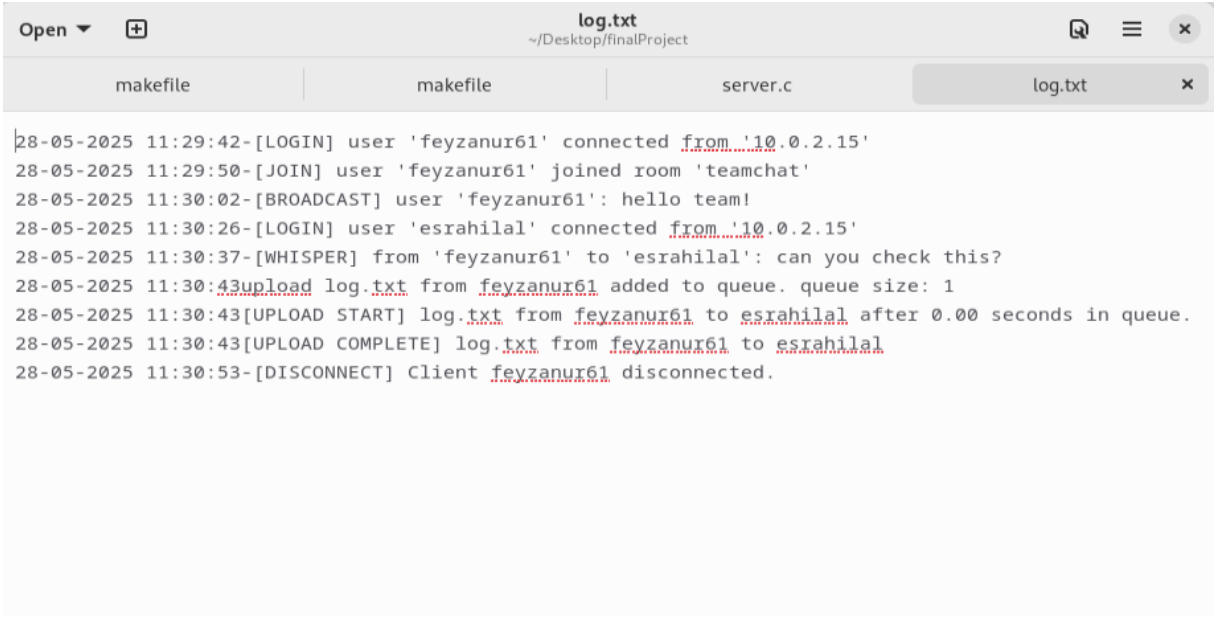
**Client Terminal:**

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: feyzanur61
> /join teamchat
> [Server]: You joined the room 'teamchat'

> /broadcast hello team!
> [SERVER]: Message sent to room 'teamchat'
> /whisper esra can you check this?
> [ERROR]: Username is not found
> /whisper esrahilal can you check this?
> [SERVER]: Whisper sent to esrahilal

> /sendfile log.txt esrahilal
> [SERVER]: File upload added to queue.

> /exit
> [SERVER]: Disconnected. Goodbye!
```

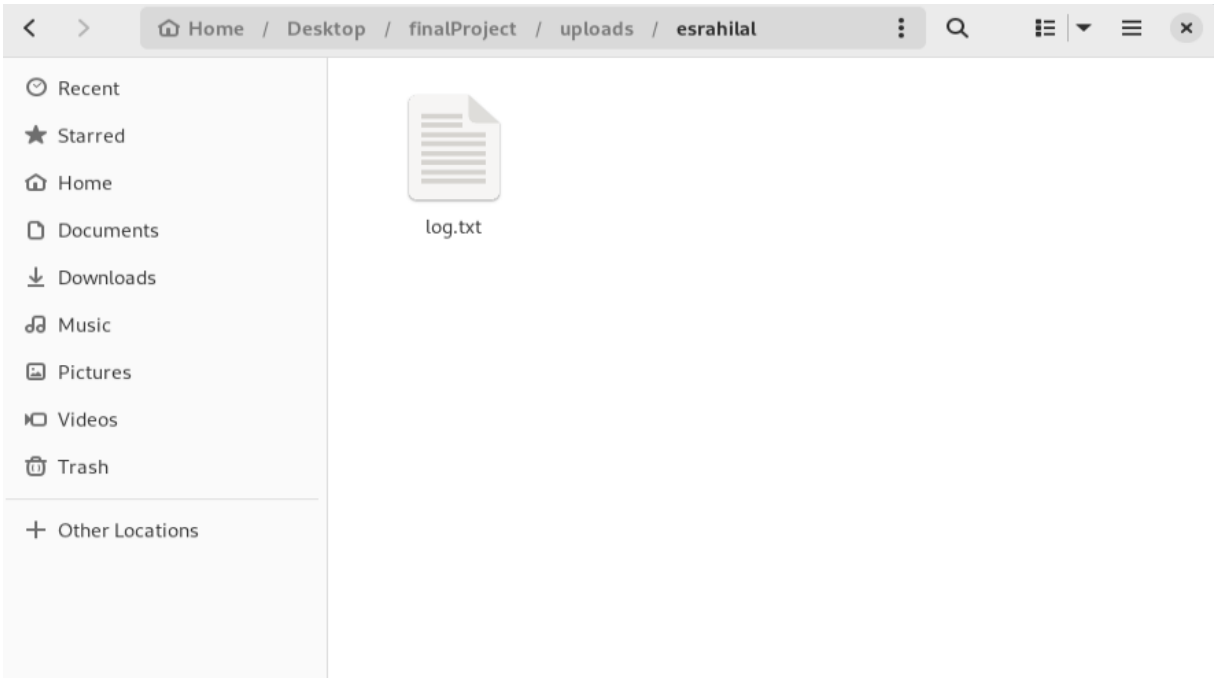
**Server Console:**

```
feyza@debian:~/Desktop/finalProject$ ./chatserver 8080
[INFO] Server listening on port: 8080
[CONNECT] New client connected: feyzanur61 from 10.0.2.15
[COMMAND] feyzanur61 joined room 'teamchat'
[COMMAND] feyzanur61 broadcast to 'teamchat'
[CONNECT] New client connected: esrahilal from 10.0.2.15
[COMMAND] feyzanur61 sent whisper to esrahilal
[DISCONNECT] Client feyzanur61 disconnected.
```

**Log.txt:**




**The sent file log.txt copied under uploads/targetUser directory.**




**Case1:**

The broadcasted message is not delivered to user that left the room.

## Client1 Terminal:

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: elifnisa17
> /join teamA
> [Server]: You joined the room 'teamA'

> /broadcast Hey team!! We have a new project.
> [SERVER]: Message sent to room 'teamA'
> [teamA] esrahilal: Well, you guys do it. I'm done.

> /broadcast ok:)
> [SERVER]: Message sent to room 'teamA'
>
```

## Client2 Terminal:

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: esrahilal
> [feyzanur61]: can you check this?
> /join teamA
> [Server]: You joined the room 'teamA'

> [teamA] elifnisa17: Hey team!! We have a new project.

> /broadcast Well, you guys do it. I'm done.
> [SERVER]: Message sent to room 'teamA'
> /leave
> [Server]: You left the room 'teamA'

> /exit
> [SERVER]: Disconnected. Goodbye!
```

## Server Console:

```
feyza@debian:~/Desktop/finalProject$ ./chatserver 8080
[INFO] Server listening on port: 8080
[CONNECT] New client connected: feyzanur61 from 10.0.2.15
[COMMAND] feyzanur61 joined room 'teamchat'
[COMMAND] feyzanur61 broadcast to 'teamchat'
[CONNECT] New client connected: esrahilal from 10.0.2.15
[COMMAND] feyzanur61 sent whisper to esrahilal
[DISCONNECT] Client feyzanur61 disconnected.
[COMMAND] esrahilal joined room 'teamA'
[CONNECT] New client connected: elifnisa17 from 10.0.2.15
[COMMAND] elifnisa17 joined room 'teamA'
[COMMAND] elifnisa17 broadcast to 'teamA'
[COMMAND] esrahilal broadcast to 'teamA'
[COMMAND]: User esrahilal left room: teamA
[COMMAND] elifnisa17 broadcast to 'teamA'
[DISCONNECT] Client esrahilal disconnected.
```

## Log.txt

```
28-05-2025 11:34:33-[JOIN] user 'esrahilal' joined room 'teamA'
28-05-2025 11:34:47-[LOGIN] user 'elifnisa17' connected from '10.0.2.15'
28-05-2025 11:36:45-[JOIN] user 'elifnisa17' joined room 'teamA'
28-05-2025 11:37:26-[BROADCAST] user 'elifnisa17': Hey team!! We have a new project.
28-05-2025 11:37:49-[BROADCAST] user 'esrahilal': Well, you guys do it. I'm done.
28-05-2025 11:37:54User esrahilal left room: teamA
28-05-2025 11:38:03-[BROADCAST] user 'elifnisa17': ok:)
28-05-2025 11:40:41-[DISCONNECT] Client esrahilal disconnected.
```

**Case2:**

Client SIGINT handling

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: elifnisa17
> /join teamA
> [Server]: You joined the room 'teamA'

> /broadcast Hey team!! We have a new project.
> [SERVER]: Message sent to room 'teamA'
> [teamA] esrahilal: Well, you guys do it. I'm done.

> /broadcast ok:)
> [SERVER]: Message sent to room 'teamA'
> ^C
[INFO] SIGINT received. Shutting down the server gracefully...
feyza@debian:~/Desktop/finalProject$
```

**Case3:**

Wrong port number

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8081
connect: Connection refused
```

**Case4:**

Missing entry

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1
Usage: ./chatclient <IP> <PORT>
```

**Case5:**

Server SIGINT

**Server Console:**

```
[CONNECT] New client connected: feyzanur from 10.0.2.15
[CONNECT] New client connected: esrahilal34 from 10.0.2.15
[COMMAND] esrahilal34 joined room 'team'
[COMMAND] esrahilal34 sent whisper to feyzanur
^C
[SHUTDOWN] SIGINT received. Shutting down the server gracefully...
[DISCONNECT] Client feyzanur disconnected.
[DISCONNECT] Client esrahilal34 disconnected.
feyza@debian:~/Desktop/finalProject$
```

**Client1 Terminal:**

```
> feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: esrahilal34
> /join team
> [Server]: You joined the room 'team'

> /whisper feyzanur hey
> [SERVER]: Whisper sent to feyzanur

>
[INFO] Server is shutting down. Disconnecting...
feyza@debian:~/Desktop/finalProject$
```

**Client2 Terminal:**

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: feyzanur
> [esrahilal34]: hey
>
[INFO] Server is shutting down. Disconnecting...
feyza@debian:~/Desktop/finalProject$
```

**Log.txt**

```
28-05-2025 13:43:56-[LOGIN] user 'feyzanur' connected from '10.0.2.15'
28-05-2025 13:44:01-[LOGIN] user 'esrahilal34' connected from '10.0.2.15'
28-05-2025 13:44:05-[JOIN] user 'esrahilal34' joined room 'team'
28-05-2025 13:44:12-[WHISPER] from 'esrahilal34' to 'feyzanur': hey
28-05-2025 13:44:19[SHUTDOWN] SIGINT received. Disconnecting 2 clients, saving logs.
```

**Case6 & 7:**

-File not exists.

-User not found

-File conflict

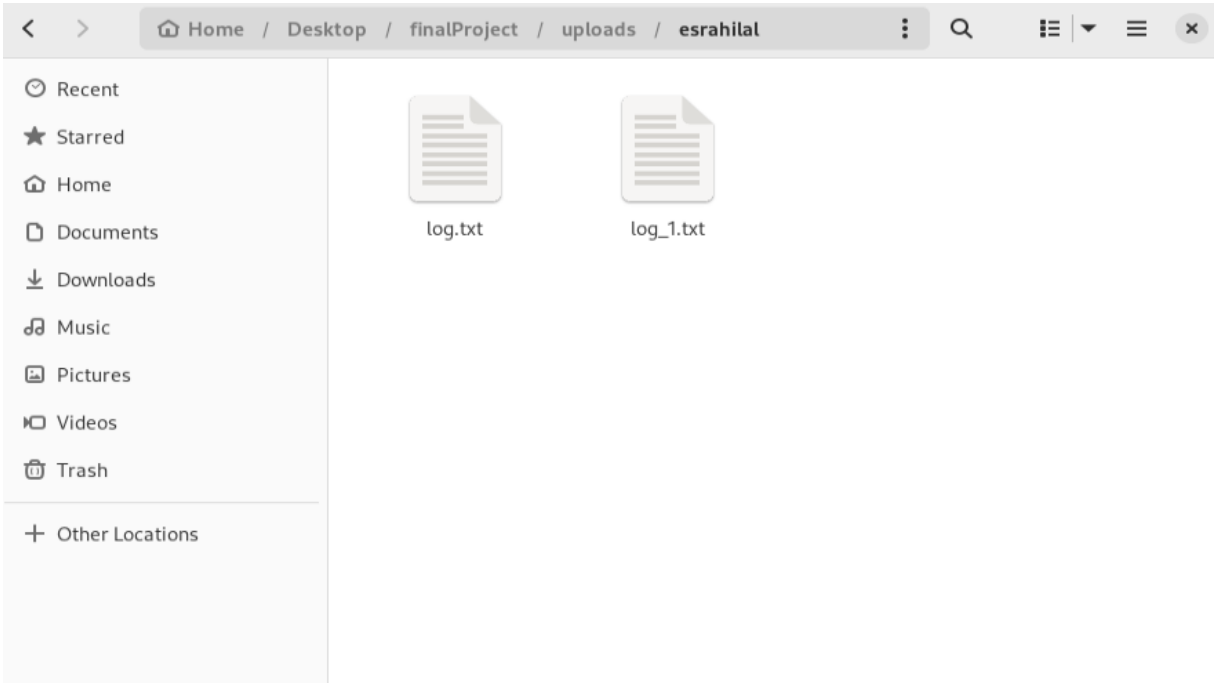
**Client Terminal:**

```
feyza@debian:~/Desktop/finalProject$ ./chatclient 127.0.0.1 8080
Enter your username: feyzanur61
> /sendfile file.txt esrahilal
> [ERROR]: File does not exist.

> /sendfile log.txt esra
> [ERROR]: Target user not found.

> /sendfile log.txt esrahilal
> [SERVER]: File upload added to queue.

>
```

**Log.txt**

```
28-05-2025 13:47:27-[LOGIN] user 'feyzanur61' connected from '10.0.2.15'
28-05-2025 13:47:32-[LOGIN] user 'esrahilal' connected from '10.0.2.15'
28-05-2025 13:47:56[ERROR]: Target user not found.
28-05-2025 13:48:04upload log.txt from feyzanur61 added to queue. queue size: 1
28-05-2025 13:48:04[FILE] Conflict : 'log.txt' received twice -> renamed 'log_1.txt'
28-05-2025 13:48:04[UPLOAD START] log_1.txt from feyzanur61 to esrahilal after 0.00 seconds in
queue.
28-05-2025 13:48:04[UPLOAD COMPLETE] log.txt from feyzanur61 to esrahilal
```

**Uploads/username**




**Case8:**

Run Client using valgrind

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==4919== Memcheck, a memory error detector
==4919== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==4919== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==4919== Command: ./chatclient 127.0.0.1 8080
==4919==
Enter your username: tuba1808
> /join teamB
> [Server]: You joined the room 'teamB'

> /broadcast Are there any homework to do
> [SERVER]: Message sent to room 'teamB'
> [teamB] emre0234: no the semester ended!

> /whisper emre0234 thanks
> [SERVER]: Whisper sent to emre0234

> /exit
> [SERVER]: Disconnected. Goodbye!

> ==4919==
==4919== HEAP SUMMARY:
==4919==     in use at exit: 272 bytes in 1 blocks
==4919==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==4919==
==4919== LEAK SUMMARY:
==4919==    definitely lost: 0 bytes in 0 blocks
==4919==    indirectly lost: 0 bytes in 0 blocks
==4919==      possibly lost: 272 bytes in 1 blocks
==4919==    still reachable: 0 bytes in 0 blocks
==4919==         suppressed: 0 bytes in 0 blocks
==4919== Rerun with --leak-check=full to see details of leaked memory
==4919==
==4919== For lists of detected and suppressed errors, rerun with: -s
==4919== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

**Case 9:** Server Console:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatserver 8080
==5315== Memcheck, a memory error detector
==5315== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5315== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5315== Command: ./chatserver 8080
==5315==
[INFO] Server listening on port: 8080
[CONNECT] New client connected: emre0234 from 10.0.2.15
[COMMAND] emre0234 joined room 'teamB'
[COMMAND] emre0234 broadcast to 'teamB'
[DISCONNECT] Client emre0234 disconnected.
^C
[SHUTDOWN] SIGINT received. Shutting down the server gracefully...
==5315==
==5315== HEAP SUMMARY:
==5315==     in use at exit: 272 bytes in 1 blocks
==5315==   total heap usage: 27 allocs, 26 frees, 12,692 bytes allocated
==5315==
==5315== LEAK SUMMARY:
==5315==    definitely lost: 0 bytes in 0 blocks
==5315==    indirectly lost: 0 bytes in 0 blocks
==5315==      possibly lost: 272 bytes in 1 blocks
==5315==    still reachable: 0 bytes in 0 blocks
==5315==         suppressed: 0 bytes in 0 blocks
==5315== Rerun with --leak-check=full to see details of leaked memory
==5315==
==5315== For lists of detected and suppressed errors, rerun with: -s
==5315== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

Client Terminal:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==5317== Memcheck, a memory error detector
==5317== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5317== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5317== Command: ./chatclient 127.0.0.1 8080
==5317==
Enter your username: emre0234
> /join teamB
> [Server]: You joined the room 'teamB'

> /broadcast no work to do
> [SERVER]: Message sent to room 'teamB'
> /whisper tuba1808 bye
> [ERROR]: Username is not found
> /exit
> [SERVER]: Disconnected. Goodbye!

> ==5317==
==5317== HEAP SUMMARY:
==5317==     in use at exit: 272 bytes in 1 blocks
==5317==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==5317==
==5317== LEAK SUMMARY:
==5317==    definitely lost: 0 bytes in 0 blocks
==5317==    indirectly lost: 0 bytes in 0 blocks
==5317==      possibly lost: 272 bytes in 1 blocks
==5317==    still reachable: 0 bytes in 0 blocks
==5317==         suppressed: 0 bytes in 0 blocks
==5317== Rerun with --leak-check=full to see details of leaked memory
==5317==
==5317== For lists of detected and suppressed errors, rerun with: -s
==5317== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

**Case10:**

Valgrind SIGINT Server sends termination info to all clients

Server Console:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatserver 8080
==5418== Memcheck, a memory error detector
==5418== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5418== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5418== Command: ./chatserver 8080
==5418==
[INFO] Server listening on port: 8080
[CONNECT] New client connected: feyzanur61 from 10.0.2.15
[CONNECT] New client connected: esra from 10.0.2.15
[CONNECT] New client connected: tuba1808 from 10.0.2.15
^C
[SHUTDOWN] SIGINT received. Shutting down the server gracefully...
==5418==
==5418== HEAP SUMMARY:
==5418==     in use at exit: 1,088 bytes in 4 blocks
==5418==   total heap usage: 35 allocs, 31 frees, 13,275 bytes allocated
==5418==
==5418== LEAK SUMMARY:
==5418==    definitely lost: 0 bytes in 0 blocks
==5418==    indirectly lost: 0 bytes in 0 blocks
==5418==      possibly lost: 1,088 bytes in 4 blocks
==5418==    still reachable: 0 bytes in 0 blocks
==5418==         suppressed: 0 bytes in 0 blocks
==5418== Rerun with --leak-check=full to see details of leaked memory
==5418==
==5418== For lists of detected and suppressed errors, rerun with: -s
==5418== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

Client1 terminal:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==5420== Memcheck, a memory error detector
==5420== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5420== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5420== Command: ./chatclient 127.0.0.1 8080
==5420==
Enter your username: feyzanur61
>
[INFO] Server is shutting down. Disconnecting...
==5420==
==5420== HEAP SUMMARY:
==5420==     in use at exit: 272 bytes in 1 blocks
==5420==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==5420==
==5420== LEAK SUMMARY:
==5420==    definitely lost: 0 bytes in 0 blocks
==5420==    indirectly lost: 0 bytes in 0 blocks
==5420==      possibly lost: 272 bytes in 1 blocks
==5420==    still reachable: 0 bytes in 0 blocks
==5420==         suppressed: 0 bytes in 0 blocks
==5420== Rerun with --leak-check=full to see details of leaked memory
==5420==
==5420== For lists of detected and suppressed errors, rerun with: -s
==5420== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

Client2 Terminal:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==5435== Memcheck, a memory error detector
==5435== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5435== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5435== Command: ./chatclient 127.0.0.1 8080
==5435==
Enter your username: esra
>
[INFO] Server is shutting down. Disconnecting...
==5435==
==5435== HEAP SUMMARY:
==5435==     in use at exit: 272 bytes in 1 blocks
==5435==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==5435==
==5435== LEAK SUMMARY:
==5435==    definitely lost: 0 bytes in 0 blocks
==5435==    indirectly lost: 0 bytes in 0 blocks
==5435==      possibly lost: 272 bytes in 1 blocks
==5435==    still reachable: 0 bytes in 0 blocks
==5435==         suppressed: 0 bytes in 0 blocks
==5435== Rerun with --leak-check=full to see details of leaked memory
==5435==
==5435== For lists of detected and suppressed errors, rerun with: -s
==5435== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

Client3 terminal:

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==5446== Memcheck, a memory error detector
==5446== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5446== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5446== Command: ./chatclient 127.0.0.1 8080
==5446==
Enter your username: tuba1808
>
[INFO] Server is shutting down. Disconnecting...
==5446==
==5446== HEAP SUMMARY:
==5446==     in use at exit: 272 bytes in 1 blocks
==5446==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==5446==
==5446== LEAK SUMMARY:
==5446==    definitely lost: 0 bytes in 0 blocks
==5446==    indirectly lost: 0 bytes in 0 blocks
==5446==      possibly lost: 272 bytes in 1 blocks
==5446==    still reachable: 0 bytes in 0 blocks
==5446==         suppressed: 0 bytes in 0 blocks
==5446== Rerun with --leak-check=full to see details of leaked memory
==5446==
==5446== For lists of detected and suppressed errors, rerun with: -s
==5446== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$
```

**Case11:**

Client Terminal:

Tries to login with a username with already exists username

```
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
==5441== Memcheck, a memory error detector
==5441== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5441== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5441== Command: ./chatclient 127.0.0.1 8080
==5441==
Enter your username: esra
> [ERROR]-Username already taken.

> ==5441==
==5441== HEAP SUMMARY:
==5441==     in use at exit: 272 bytes in 1 blocks
==5441==   total heap usage: 8 allocs, 7 frees, 5,948 bytes allocated
==5441==
==5441== LEAK SUMMARY:
==5441==    definitely lost: 0 bytes in 0 blocks
==5441==    indirectly lost: 0 bytes in 0 blocks
==5441==      possibly lost: 272 bytes in 1 blocks
==5441==    still reachable: 0 bytes in 0 blocks
==5441==         suppressed: 0 bytes in 0 blocks
==5441== Rerun with --leak-check=full to see details of leaked memory
==5441==
==5441== For lists of detected and suppressed errors, rerun with: -s
==5441== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
feyza@debian:~/Desktop/finalProject$ valgrind ./chatclient 127.0.0.1 8080
```

Log.txt

```
28-05-2025 14:06:44-[LOGIN] user 'feyzanur61' connected from '10.0.2.15'
28-05-2025 14:06:48-[LOGIN] user 'esra' connected from '10.0.2.15'
28-05-2025 14:06:53[REJECTED]-Duplicate username attempted: esra
28-05-2025 14:06:58-[LOGIN] user 'tuba1808' connected from '10.0.2.15'
28-05-2025 14:07:03[SHUTDOWN] SIGINT received. Disconnecting 3 clients, saving logs.
```

**Case12**: Rejoin example

```
feyza@debian:~/Downloads/1801042618_FeyzaNur_Kucuk$ valgrind ./chatclient 127.0.
0.1 8080
==8184== Memcheck, a memory error detector
==8184== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==8184== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==8184== Command: ./chatclient 127.0.0.1 8080
==8184==
Enter your username: feyzanur
> /join team
> [Server]: You joined the room 'team'

> [team] esra: hey team!!

> /leave
> [Server]: You left the room 'team'

> /join team
> [Server]: Welcome back! You rejoined the room 'team'

> /exit
> [SERVER]: Disconnected. Goodbye!
```

Log.txt

```
29-05-2025 16:01:45-[LOGIN] user 'feyzanur' connected from '10.0.2.15'
29-05-2025 16:01:48-[JOIN] user 'feyzanur' joined room 'team'
29-05-2025 16:01:58-[LOGIN] user 'esra' connected from '10.0.2.15'
29-05-2025 16:02:01-[JOIN] user 'esra' joined room 'team'
29-05-2025 16:02:06-[BROADCAST] user 'esra': hey team!!
29-05-2025 16:02:11-[ROOM] User feyzanur left room: team
29-05-2025 16:02:15-[JOIN] user 'feyzanur'  rejoined room 'team'
29-05-2025 16:02:18-[DISCONNECT] Client feyzanur disconnected.
```

**Issues Faced and Solutions**

- **Problem:** When the server was stopped using Ctrl+C (SIGINT), the clients remained running and did not close properly.
  **Solution:** Implemented a signal handler for SIGINT on the server side that sends a shutdown message to all connected clients before closing. Clients listen for this message and exit cleanly, ensuring both server and clients close together.
- **Problem:** After stopping the server and trying to restart it immediately, the server could not bind to the same port again due to the port being in use.
  **Solution:** Enabled the socket option SO_REUSEADDR on the server socket. This allows the server to reuse the port immediately after shutdown without waiting for the OS to release it.

- **Problem:** Multiple threads accessing shared data structures like the list of connected clients and upload queue caused race conditions and inconsistent data.
  **Solution:** Used pthread mutex locks to protect these shared resources so that only one thread can modify them at a time, preventing conflicts and data corruption.
- **Problem:** File upload queue access by multiple threads sometimes caused deadlocks or missed signals.
  **Solution:** Introduced semaphores combined with mutexes to manage the upload queue properly, ensuring threads wait when the queue is empty or full and proceed safely when data is available.

**Conclusion**

The project successfully demonstrates multithreaded socket programming in C. Key achievements include:

- Stable handling of 15+ concurrent clients.
- Real-time messaging with support for rooms and private messages.
- Secure and synchronized file uploads via a queue mechanism.
- Robust logging and graceful shutdown via signal handling.

**Potential Improvements**

- Persistent Storage: Store chat logs and uploaded files on disk or database.
- Authentication System: Add login/password functionality.
- User Interface: Develop a GUI version using ncurses or a separate frontend app.
- Timeout Handling: Disconnect idle clients after a configurable period.

**Java Implementation**

**Server.java**

Core Components

- clients: Maps usernames to their handler threads.

- rooms: Maps room names to sets of usernames.

- userRooms: Tracks which room a user belongs to.

Sockets and Threads

The server listens on a user-defined port and creates a new thread for each incoming connection using a ServerSocket. Each client is managed independently through a ClientHandler thread. Example run: java Server 8080

ClientHandler Class

Handles the logic for one client:

- run(): reads commands and responds

- sendMessage(): sends a message to the client

- getIP(): retrieves the client's IP address

Command Handling

- /join <room>: joins a room, max 15 users

- /leave: exits the current room

- /broadcast <msg>: sends message to room

- /whisper <user> <msg>: private message

- /sendfile <filename> <user>: logs simulated file transfer

-/exit: disconnects user and remove them from users list.

Logging System

Each event is recorded in log.txt with a timestamp using write_to_log(). Example:

2025-05-29 18:06:30 - [BROADCAST] user feyzanur61:Hello team!

Multithreading

Each client has its own thread, enabling concurrent interactions. Threads are created with new Thread(handler).start().

Error Handling

Exceptions such as IO errors or disconnections are handled gracefully. Client sockets are closed and removed from server memory.

**Client.java**

Socket Communication

It creates a Socket with the server's IP and port. The client uses:

- BufferedReader: to receive server messages

- PrintWriter: to send messages to the server

Command-Line Arguments

The IP and port must be passed at startup:

java Client <server-ip> <port>

Example:

java Client 127.0.0.1 8080

Username Input

After connecting, the client is prompted to enter a username, which is sent to the server for registration.

Receive Thread

A new thread is created to listen for incoming messages from the server. This allows the user to send messages and receive messages concurrently.

Supported Commands

The client supports the following commands:

- /join <room>

- /leave

- /broadcast <msg>

- /whisper <user> <msg>

- /sendfile <filename> <user>

- /exit

Error Handling

If the server connection is lost, the client prints a message and exits. IOExceptions are handled to avoid crashes.

Test run:

Client1

```
[CLIENT] Server connection lost. Exitingjava Client 192.1
68.1.7 8080\feyza\Desktop\finalProject>
[CLIENT] Connected to the server.
Enter your username: feyzanur
[INFO] Commands: /join roomName, /leave, /whisper usernam
e message, /broadcast message, /sendfile filename targetU
ser, exit
[SERVER] Welcome, feyzanur!
/join team
[SERVER] Joined room 'team'.
[team]esra: Hello team!
/leave
[SERVER] Left room 'team'.
/whisper esra i left the chatroom
[Whisper from esra]: oh, i see.
/sendfile log.txt esra
[SERVER] File added to the upload queue.
/exit
[CLIENT] Server connection lost. Exiting.
PS C:\Users\feyza\Desktop\finalProject>
```

Client2

```
 max 16 chars). Connection closing.     javac Client.java
          \feyza\Desktop\finalProject>
PS C:\Users\feyza\Desktop\finalProject> java Client 192.1
68.1.7 8080
[CLIENT] Connected to the server.
Enter your username: esra
[INFO] Commands: /join roomName, /leave, /whisper usernam
e message, /broadcast message, /sendfile filename targetU
ser, exit
[SERVER] Welcome, esra!
/join team
[SERVER] Joined room 'team'.
/broadcast Hello team!
/broadcast is there any news?
[Whisper from feyzanur]: i left the chatroom
/whisper feyzanur oh, i see.
/exit
[CLIENT] Server connection lost. Exiting.
PS C:\Users\feyza\Desktop\finalProject>
```

Server

```
[DISCONNECT] Client esra disconnected.
PS C:\Users\feyza\Desktop\finalProject> javac Server.java
PS C:\Users\feyza\Desktop\finalProject> java Server 8080
[INFO] Server listening on port 8080...
[CONNECT] user 'feyzanur' connected from 192.168.1.7
[CONNECT] user 'esra' connected from 192.168.1.7
[COMMAND] user 'feyzanur' joined room 'team'
[COMMAND] user 'esra' joined room 'team'
[COMMAND] user 'esra' broadcasted to 'team'
[COMMAND] user 'esra' broadcasted to 'team'
[COMMAND] user 'feyzanur' whispered to 'esra'
[COMMAND] user 'esra' whispered to 'feyzanur'
[COMMAND] 'feyzanur' initiated file transfer to esra
[DISCONNECT] Client esra disconnected.
[DISCONNECT] Client feyzanur disconnected.
```