Documentation for the Multi-Client Chat Server

Overview

This is a multi-client chat server implemented in C using the TCP protocol. The server supports:

1. Multiple clients connected simultaneously.
2. Broadcasting messages to all clients.
3. Sending private (whisper) messages.
4. Graceful handling of client disconnections.
5. Server shutdown triggered by a Ctrl+C interrupt.

Features

1. Server

- Listens for incoming connections on a specified TCP port.
- Handles multiple client connections concurrently using threads.
- Broadcasts messages from one client to all connected clients.
- Supports private (whisper) messages to specific clients.
- Monitors client disconnections and notifies all connected clients.
- Handles server shutdown gracefully using Ctrl+C.

2. Clients

- Connects to the server using its IP and port.
- Sends messages to the server, which are broadcast to all other clients.
- Sends private messages to specific clients using the @username message format.
- Exits the chat using the !exit command.

Code Architecture

Main Components

1. Server Code:
   - Manages incoming client connections.
   - Handles message broadcasting and whisper functionality.
   - Detects and handles client disconnections.
   - Provides a graceful shutdown mechanism.

2. Client Code:
   - Sends user input to the server.
   - Displays messages from other clients or server notifications.

Key Data Structures

- Client: Represents a connected client.
  - Fields:

- socket: Socket descriptor for the client.
- name: Name of the client.
- ip: IP address of the client.
- port: Port number of the client.
- thread: Thread handling the client.

How It Works

Server

1. Initialization:
   - The server starts by creating a socket and binding it to a specified port.
   - Listens for incoming connections.

2. Accepting Clients:
   - When a client connects, the server accepts the connection and creates a Client structure.
   - Spawns a thread to handle the client.

3. Message Handling:
   - Broadcasts messages to all clients except the sender.
   - Handles private messages (whispers) using the @username message format.
   - Detects the !exit command to disconnect clients gracefully.

4. Disconnection:
   - On disconnection, notifies all other clients and removes the client from the active client list.

5. Shutdown:
   - Handles Ctrl+C to shut down gracefully.
   - Waits for all threads to finish and cleans up resources.

Usage

Running the Server

1. Compile the server code:
   gcc -o hw3server hw3server.c -pthread
2. Run the server with a specified port:
   ./hw3server <port>

Running the Client

1. Compile the client code:
   gcc -o hw3client hw3client.c -pthread
2. Run the client with the server IP, port, and username:
   ./hw3client <server_ip> <port> <username>

Example Session

Server Terminal:
$ ./hw3server 9900
Server listening on port 9900...
john connected from 127.0.0.1 using port 12345
doe connected from 127.0.0.1 using port 12346
(Whisper from john): Hi, doe!
doe has left the chat.
Interrupt received. Shutting down server...
Server shut down successfully.

Client Terminal 1:
$ ./hw3client 127.0.0.1 9900 john
Connected to server as john.
doe has joined the chat
@doe Hi, doe!
!exit

Client Terminal 2:
$ ./hw3client 127.0.0.1 9900 doe
Connected to server as doe.
john: Hello, everyone!
(Whisper from john): Hi, doe!
!exit