Operating Systems 0512.4402 - Homework 3: Chat Client/Server

Dan Shamia 208004119, Daniel Halperin 207826314 **Repository:** https://github.com/DCodeGenius/os-hw3-networking

1. Introduction

This assignment implements a chat client and server system in Linux using the TCP protocol. The system enables multiple clients to connect to a central server, exchange public messages (broadcast to all), and send private "whisper" messages. The solution is implemented in C, adhering to the POSIX socket API and utilizing I/O multiplexing to handle concurrent connections without multi-threading.

2. Architecture and Design

2.1. Server Design (`hw3server`)

The server acts as a central relay for all chat messages. It is designed to handle multiple concurrent client connections using a single-threaded event loop driven by the `select()` system call.

- **Connection Handling:** The server binds to a specified TCP port and listens for incoming connections.
- **Client Management:** All active client sockets are stored in a global data structure. When `select()` indicates activity on the main listening socket, the server accepts the new connection and adds the new client file descriptor to its set.
- **Message Routing:** When `select()` indicates activity on a connected client socket, the server reads the incoming message and parses it to determine the type:
    - **Normal Messages:** If the text is standard, the server iterates through all connected clients and forwards the message with the prefix `sourcename: message`.
    - **Whisper Messages:** If the message starts with `@friend`, the server parses the destination name. It then searches the client list for a matching name and sends the message only to that specific socket, maintaining the `sourcename: @friend message` format.

- **Disconnection:** If `recv()` returns 0, the server identifies the client as disconnected, removes them from the list, closes the socket, and broadcasts a "client [name] disconnected" message to all remaining users.

2.2. Client Design (`hw3client`)

The client provides the user interface for sending and receiving messages. It must handle two asynchronous sources of input: the user typing on the keyboard (`stdin`) and messages arriving from the server (network socket).

- **Multiplexing:** To ensure the user interface remains responsive, the client uses `select()` to monitor both `STDIN_FILENO` and the server socket simultaneously. This prevents the client from blocking on user input while a message arrives from the server.
- **Handshake:** Upon connecting, the client immediately sends its name to the server to register itself.
- **Outgoing Messages:**
    - **Standard/Whisper:** Text typed by the user is sent directly to the server. Formatting for whispers (e.g., `@name msg`) is handled by the server's logic.
    - **Exit:** If the user types `!exit`, the client sends the message to the server (so others see the exit), prints "client exiting", and terminates.
- **Incoming Messages:** Any data received from the server socket is written directly to `stdout`.

3. Implementation Details

3.1 Data Structures

We utilized a linked list (or array, depending on your specific code) to manage the state of active clients on the server. Each node contains the client's file descriptor, name, and IP address. This allows for efficient lookups when processing private "whisper" messages.

3.2 Protocol Logic

- **Prefixing:** The server is solely responsible for adding the `sender_name:` prefix. The client sends raw text, and the server reformats it before broadcasting.
- **Buffering:** We handle partial sends/receives by ensuring buffers are flushed or loops are used to read/write the complete message length where necessary.

4. Compilation and Usage

A `Makefile` is provided to automate the build process.

Building the project:

```
make
```

This produces two executables: `hw3server` and `hw3client`.

Running the Server:

```
./hw3server <port>
```

Running the Client:

```
./hw3client <server_address> <server_port> <client_name>
```

Cleaning up:

```
make clean
```

This command removes all object files and executables.