Documentation: ChatConnect Application

Overview

This C source code implements a basic chat application using TCP socket programming. Users can host a connection or search for an available connection to chat with another user. The program supports real-time messaging between users over a network and it's designed to run on Unix-like systems with support for socket programming and `pthreads`.

Usage

1. Open your terminal.
2. Go to the file directory (it should be located in your desktop):

```
cd "/mnt/c/Users/moham/Desktop/CHAT CONNECT"
```

3. Compile the source code:

```
gcc -o ChatConnect -pthread ChatConnect.c
```

4. Run the compiled executable:

```
./ChatConnect
```

5. Choose to host a connection or search for an available connection.
6. Enter a username.
7. Start chatting with the other user.

Structure

1. **Header Files Inclusion:** Includes necessary header files for socket programming and other functionalities.
2. **Global Variables Declaration:** Declaration of global variables used throughout the program.
3. **Thread Functions:** Defines thread functions for sending and receiving messages for each user.
4. **Utility Functions:** Defines utility functions such as `clear_line()` and `UserPrint()`.
5. **Main Function:** Implements the main logic of the program including the menu, socket creation, connection establishment, user interaction, and thread handling.

Functions

1. `clear_line()`: Clears the current user input lines on the terminal before receiving a new message.
2. `sendThread()`, `ReceiveThread()`: Thread functions for sending and receiving messages for the first user.
3. `sendThread1()`, `ReceiveThread1()`: Thread functions for sending and receiving messages for the second user.
4. `UserPrint()`: Prints user information including username, connection details, and current time.
5. In the `main()` function, implementation of the program's menu, socket creation, connection establishment, user interaction, and thread handling.

Usage of Threads

- The program utilizes `pthreads` to handle concurrent sending and receiving of messages between users.
- Separate threads are created for sending and receiving messages for each user.
- Mutex and semaphores are used for synchronization and coordination between threads and users.

Error Handling

The program includes error handling mechanisms using `perror()` to handle various error conditions such as socket creation failure, connection establishment failure, and message sending/receiving errors.

User Interface

- The program provides a simple command-line interface for user interaction.
- Users are prompted to choose options from the menu and enter their usernames.
- Messages are displayed in real-time with proper formatting to distinguish between senders and receivers.

Conclusion

The provided C source code demonstrates the implementation of a basic chat application using TCP socket programming in C. The application showcases fundamental concepts such as socket communication, multithreading, and error handling. While the current implementation is simple, it can be extended and customized to suit more complex requirements and functionalities.

Chat flowchart

Hers is a flowchart explaining the work of the chat application: