IRC Chat

Project implementing some IRC features in C++.

Description of the IRC Chat Application

This IRC (Internet Relay Chat) application is designed to enable communication between multiple users through text channels. The application includes functionalities for channel creation, message sending, access control, and user management, with special privileges for administrators.

Application Functionality

The application consists of a server and multiple clients. The server manages client connections, chat channels, and user permissions. Each client connects to the server, chooses a nickname, and can join channels to send and receive messages.

Code Structure

- **Server (`server.cpp`):** Responsible for accepting client connections, managing channels, handling user commands, and transmitting messages between clients.
- **Client (`client.cpp`):** Allows the user to connect to the server, send commands and messages, and receive messages from the server.

Available Commands

The following are the commands that users can use in the application:

General Commands

1. **/nick `<nickname>`**
   - **Description:** Sets or changes the user's nickname.
   - **Example:** `/nick JohnDoe`
2. **/join `<channel>`**
   - **Description:** Joins an existing channel. A user can only be in one channel at a time. If the user is already in a channel, they leave the current channel before joining the new one.
   - **Example:** `/join general`
3. **/list**
   - **Description:** Lists all available channels and the users present in each channel.
   - **Example:** `/list`

Administrator Commands

The following commands can only be used by administrators. To become an administrator, a user must log in with the administrator password.

1. **/login `<password>`**
   - **Description:** Authenticates the user as an administrator using the provided password.
   - **Example:** `/login adminpass`
2. **/create `<channel>`**
   - **Description:** Creates a new channel. Only administrators can create channels.
   - **Example:** `/create new_channel`
3. **/mute `<nickname>`**
   - **Description:** Mutes a user in the current channel, preventing them from sending messages.
   - **Example:** `/mute JohnDoe`
4. **/unmute `<nickname>`**
   - **Description:** Unmutes a user in the current channel, allowing them to send messages again.
   - **Example:** `/unmute JohnDoe`
5. **/kick `<nickname>`**
   - **Description:** Removes a user from the current channel.
   - **Example:** `/kick JohnDoe`

Rules and Restrictions

- **Nickname:** A user must set a nickname before they can join a channel or send messages.
- **Channel:** A user must join a channel before sending messages. If the user tries to send messages without being in a channel, they will receive an error message.
- **Muting:** Muted users cannot send messages in the channel they were muted in.
- **Administration:** Only administrators can create channels, mute, unmute, and remove users.

How to Use

1. **Run the Server:** Compile and run the `server.cpp` file on the server.
2. **Connect Clients:** Compile and run the `client.cpp` file on each client. Each client should connect to the server's IP address.
3. **Set Nickname:** Use the `/nick` command to set a nickname.
4. **Join a Channel:** Use the `/join` command to join an existing channel.
5. **Interact:** Send messages, list channels and users, and use the administration commands as needed.

Example Usage

```
# On the server
make server
make run-server

# On the client
make client
make run-client

# Commands on the client
/nick Alice
/join general
Hello everyone!
/list
```

Makefile

```
all: server client

server: server.cpp
    g++ src/server.cpp -o exe/server -pthread -Wall -Werror

client: client.cpp
    g++ src/client.cpp -o exe/client -pthread -Wall -Werror

run-server:
    ./exe/server

run-client:
    ./exe/client

clean:
    rm exe/server exe/client
```

To compile the project and create server and client executables in `/exe` folder, run:

```
make
```