README - Multiple Connection Server with User Registration and Login (Linux)

Project Overview

This project implements a multi-connection server in C, specifically designed to run on **Linux**. The server supports user registration and login, handling multiple client connections simultaneously. After logging in, users can store and manage personalized text files.

Features:

- **Multiple connections**: The server supports multiple clients at once using threads, making it scalable for handling simultaneous users.
- **User Registration**: New users can create accounts with a unique username and password.
- **User Login**: Registered users can log in by providing valid credentials, which are checked against the `password.txt` file.
- **Personalized Text Storage**: Each user has a `.data` file where their personalized text is stored. This file is created upon successful login.
- **Logout**: Users can log out to terminate their session.

Requirements

- **Linux OS**: The server is designed and tested on Linux. It requires standard libraries such as `<pthread.h>` for multi-threading and `<sys/socket.h>` for networking.
- **Compiler**: GCC or any compatible C compiler that supports POSIX threads.

How to Run

1. **Compile the server**: Use the following command to compile the server code on Linux:

   ```
   bash
   make
   ```

2. **Run the server**: Start the server by executing:

   ```
   bash
   ./server
   ```

File Structure

- `server.c`: The core server code, which handles networking, user registration, login, and multi-threaded client support.
- `password.txt`: A text file storing usernames and passwords in a simple `username:password` format.
- `<username>.data`: A text file for each user, created upon login, where their personalized content is saved.

Example Workflow

1. Start the server: `./server`
2. **`Run:`** `index.hteml`
3. Register: `register user1 password1`
4. Login: `login user1 password1`
5. Write and save personal text.
6. Logout: `logout`

Notes

- Ensure the `password.txt` file exists in the same directory as the server, and has appropriate write permissions for the server to update.
- Each user's `.data` file is created upon their first login and will persist for future logins.