ECE 361 Lab #4: TCP Chat

GB 243

PRA 04: Tuesday, February 27th, 2018
PRA 02: Friday, March 02nd, 2018
PRA 03: Tuesday, March 20th, 2018
PRA 01: Friday, March 23rd, 2018

Overview

In this lab you will develop a TCP-based chat application. This requires you to develop server and client C programs using stream sockets. The server hosts the chat for multiple clients. Clients have two options once connected to the server:

1. Send a broadcast message to all other connected clients
2. Send a private message to a single connected client

NOTE: This is a programming assignment. You are provided (at least) **2 weeks** to complete your client and server applications and demonstrate it during your lab practicals. Please start early and use the discussion board for any questions.

The Lab

You are responsible for developing both the client and server using stream sockets.

Client

The client should be run as follows:

```
./chatclient [server_address] [server_port] [user_name]
```

Where **server_address** is the host address of your chat server, **server_port** is the port number the chat server is listening on and **user_name** is how the client is identified in the chat. The **user_name** must be unique and the server should not accept a new client with an already existing username. For simplicity you can assume the **user_name** will not contain any spaces.

Server

The server should be run as follows:

```
./chatserver [server_port]
```

You must handle the case where multiple clients are concurrently connected to your chat server. It is suggested that you use the **`select()`** system call. However, you may choose to use threads or processes.

When a new client connects to the server, the server must broadcast the **user_name** to the existing users.

Client Operations

After connecting to the server, the client application should accept the following commands from the command line.

- ➔ **`broadcast [message]`**
- ➔ **`[user_name] [message]`**
- ➔ **`list`**
- ➔ **`exit`**

The desired actions should be relayed to the server. You may design the exchange of messages (protocol) between the client and server in any way you choose. The server facilitates the communication between the chat clients.

Broadcast Message

The following operations should be implemented for the broadcast request:

- The message should be sent to all connected clients
- The sending client should be identified
- Each client should print the message to its terminal

Private Message

The following operations should be implemented for the private request:

- The message should only be displayed to the intended user.
- You may choose to implement this by passing the message through the server or by creating a new socket between the two clients with the help of the server. Each method has advantages and disadvantages. Regardless of which you choose be prepared to discuss the trade-offs.

List command

When a client uses the list command, the server should be queried for a list of available, connected clients and the information should be displayed on the terminal.

Exit command
When the client decides to exit, the connection should be torn down gracefully. The server should remove the client from its list of connected clients.

NOTE: Please close your programs using **Ctl + c** instead of **Ctl + z** as the programs may not exit properly and the ports will not be released for future uses. If you want to kill suspended processes, refer to the following link: https://www.linux.com/learn/intro-to-linux/2017/5/how-kill-process-command-line
You can also use the following command to check which port the server is listening on: `netstat -tulpn`

Hints: Beej's Guide

http://beej.us/guide/bgnet/html/multi/index.html
Beej's guide is an extremely useful tutorial for network programming in C. It provides a walkthrough from network basics to building applications using complex socket programming. It is highly recommended to read the guide and use it's C source code examples as starter code for your applications. For this assignment, we suggest you to go over sections **2 - 7**.

Hints: Use of Debuggers

It is highly recommended that you use either **NetBeans Debugger** or **GDB** to debug your program (especially segmentation faults).

Demonstration

Once you completed the client and server applications, **signal to the Lab TA that you are ready to perform the demo**. You must demo the functionality of your chat application and describe your implementation. The Lab TA will ask you about your approach and specifics about your code.

Submission Instructions

You must submit the following files:

- Your client application (must be named **client.c**)
- Your server application (must be named **server.c**)
- A Makefile to compile your programs. Refer to the link for creating Makefiles:

https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html

- A text file to explain your how you implemented the concurrency for your applications. Furthermore, explain the difference between these mechanisms. It must be named **explanations_lab4.txt** and should be at most 100 words. This file must also contain the names and student numbers of the group members (at the beginning of the file) prefixed by **#**. Please do not prefix other lines by **#** as this would confuse the automated scripts.

```
#first1 last1, studentnum1
#first2 last2, studentnum2
```

You must combine all these files into a tarball (tar.gz format) using the following command:

```
tar -czvf lab4.tar.gz <project directory>
```

Note: Only one student in the group needs to submit that tarball.

You can submit the file using the following command:

```
submitece361s <lab_number> <filename>
```

Example: **`submitece361s 4 lab4.tar.gz`**

You can verify if you have submitted successfully by using the following command: **`submitece361s -l 4`**

For more information regarding the command, please refer to it's man page: **`submitece361s`**

The submitted files will be used to verify your findings and check for plagiarism.

Marking

This lab is worth a total of **4** marks with the following breakdown:

- Completed Client/ Server Chat Application: **3** marks (marked as group)
- Demo : **1** mark (marked individually)

All marks will be assigned by the end of the lab session.