CSCE 5580: Computer Networks

Programming Assignment 3

Due: 11:59 PM on Wednesday, April 22, 2020

PROGRAM DESCRIPTION:

In this project, you will write a complete C program to support the server in a client/server model using Linux stream sockets. The program will consist of a server that will implement a service (like a chat room) using a simple text protocol that clients can use (through `telnet`) to message other clients on the service.

Specifically, you will *only* implement the server as follows:

- The server will set up as a server and establish a socket connection using a port number passed in as an argument to the program. For example, the server and client code may be executed as follows: **`./server <svr_port>`**

  The server will print out a usage statement and terminate the program if the user does not enter the command with the port as the command-line argument.

- The server will support up to and including 10 “registered” client connections (registration is done with the `JOIN` command). Although more than 10 clients can connect to the server, no more than 10 clients may be registered in the database at the same time. In the case of more than 10 clients attempting to register, the server will print out an error message and simply close the connection. Note that clients may come and go as they please, so if, for example, the 10th client “leaves” the service, space is now made available for another client to register.
- The server will set up a network socket (i.e., AF_INET) over TCP (i.e., SOCK_STREAM) using the port number passed in as an argument to the program.
  - o The server will bind and listen to the port specified as a command-line argument. When a client connects, a new thread will be spawned to handle the socket using POSIX threads (i.e., pthreads) in Linux.
  - o The server will support the following case-sensitive commands which are all sent as plain ASCII text:
    - ▪ `JOIN username`
      - When a client wants to join (i.e., register for) the service, it first connects to the server using the telnet command with the hostname of the server CSE machine and the port number, and then sends a `JOIN` request with the `username`. Usernames will only consist of alphanumeric characters and will not contain spaces or other “special” or control characters. You may assume that the user follows this requirement for alphanumeric characters, so no validation is needed. The server will store this username as well as the client’s socket file descriptor in the “database”. Note that

although a client may "connect" to the server, it MUST register for the service with the `JOIN` command before the client may receive any "services" provided by the server. If the database is "full" (i.e., 10 clients have registered for the service), then the server will print out a status message and send a "Too Many Users" message to the client. Once a client has already registered with a JOIN request, any subsequent JOIN requests from the same registered client will be discarded with a status message sent back to the client.

- `LIST`
  - If a registered client wants to know who is currently subscribed to the service, the client will issue a `LIST` request. Upon receipt of the `LIST` request, the server will send a list of all registered clients on individual lines and return this list (newlines and all) to the client. Note that the client must be registered for the service to receive any "services", such as this one, provided by the server. If the client who is not registered for this service sends a `LIST` request, the server will print out a status message and send an "Unregistered User" message to the client with the `JOIN` request instructions.
- `MESG username some_message_text`
  - If a registered client wants to send an individual message to another registered client, the client will issue the `MESG` request with the username of a registered client followed by whatever message he/she wants to send to the other registered client. The server will then act as a relay and forward this message to the registered user. Note that the client must be registered for the service to receive any "services", such as this one, provided by the server. If the client who is not registered for this service sends a `MESG` request, the server will print out a status message and send an "Unregistered User" message to the client with the `JOIN` request instructions. If a registered client sends a `MESG` request to an unregistered client, the server will print out a status message and send an "Unknown Recipient" message to the client.
- `BCST some_message_text`
  - If a registered client wants to broadcast a message to all other registered clients, the client will issue the `BCST` request followed by whatever message he/she wants to send to the other registered clients. The server will then act as a relay and forward this message to the registered users (but not the sender). Note that the client must be registered for the service to receive any "services", such as this one, provided by the server. If the client who is not registered for this service sends a `BCST` request, the server will print out a status message and send an "Unregistered User" message to the client with the `JOIN` request instructions.
- `QUIT`

        - When a connected client wants to leave the service, the client will issue a `QUIT` request, at which time the server will disconnect the client from the service. The database entry for registered clients should be removed after the client has been disconnected. Note that an unregistered client will still be disconnected from the service (i.e., their connection closed) with a status message at the server, though no data needs to be removed form the database since there is none for that client.
    - Unrecognizable Messages
        - If a registered client sends an unrecognizable request (i.e., one not supported by this protocol), the server will print out a status message and send an “Unknown Message” message to the client.
        - Note that the client must be registered for the service to receive any “services” provided by the server. If the client who is not registered for this service sends an unrecognizable message, the server will print out a status message and send an “Unregistered User” message to the client with the `JOIN` request instructions.
- The server will provide important status updates of messages sent and received, such as connection events and received requests, identifying the client using their socket file descriptor.
- The server will support error checking across all relevant system calls and other potential issues.

You will simply use a “telnet” client to connect to the server, specifying the hostname and port number of the server to connect to.

- When the telnet client starts up, the client can issue any request (even unrecognizable ones) to the server, but is expected to issue the `JOIN` request immediately with the client’s username using alphanumeric characters so as to be able to use the services provided by the server.
- When desired, the client will send a `LIST` request to the server to see who is currently subscribed to the service, but only the registered client will receive the listing (i.e., others will receive an error message with `JOIN` request instructions).
- When desired, the client will send a `MESG` or `BCST` request to the server who will then relay that message to an individual registered client or all registered clients, respectively.
- The client may voluntarily leave the service at any time by issuing the `QUIT` request.

Your program (i.e., a server program) should run on the INET domain using SOCK_STREAM (i.e., TCP) sockets so that the server and client execute on different CSE machines at the same time. The server will accept the port number to communicate on as an argument to the server program. Your code will handle errors appropriately, such as printing an error message, disconnecting the client, or termination of the program.

SAMPLE OUTPUT (user input shown in **bold**):

==> SERVER on cse06

```
$ ./prog3svr
usage: ./prog3svr <svr_port>
$ ./prog3svr 8001
Waiting for Incoming Connections...
Client (5): Connection Accepted
Client (5): Connection Handler Assigned
Unable to Locate Client (5) in Database. Discarding LIST.
Unable to Locate Client (5) in Database. Discarding BCST.
Client (5): QUIT
Unable to Locate Client (5) in Database. Disconnecting User.
Client (5): Disconnecting User.
Client (6): Connection Accepted
Client (6): Connection Handler Assigned
Client (6): JOIN mat0299
Client (6): LIST
Client (6): Unrecognizable Message. Discarding UNKNOWN Message.
Client (5): Connection Accepted
Client (5): Connection Handler Assigned
Client (5): JOIN xyz0123
Client (5): LIST
Unable to Locate Recipient (brb5678) in Database. Discarding MESG.
Client (7): Connection Accepted
Client (7): Connection Handler Assigned
Client (7): JOIN brb5678
Client (7): LIST
Client (6): LIST
Client (5): QUIT
Client (5): Disconnecting User.
Client (6): QUIT
Client (6): Disconnecting User.
Client (7): QUIT
Client (7): Disconnecting User.
^C
```

==> CLIENT 1 on cse05

```
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
LIST
Unregistered User. Use "JOIN <username>" to Register.
BCST Is anyone there?
Unregistered User. Use "JOIN <username>" to Register.
QUIT
Connection closed by foreign host.
$ telnet cse06 8001
Trying 129.120.151.99...
```

```
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
JOIN mat0299
JOIN mat0299 Request Accepted
LIST
USERNAME     FD
----------------------------
mat0299       6
----------------------------
HELO
Unknown Message. Discarding UNKNOWN Message.
FROM xyz0123: Hello!
FROM brb5678: Who else is on?
LIST
USERNAME     FD
----------------------------
mat0299       6
xyz0123       5
brb5678       7
----------------------------
QUIT
Connection closed by foreign host.
```

==> CLIENT 2 on cse04

```
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
JOIN xyz0123
JOIN xyz0123 Request Accepted
LIST
USERNAME     FD
----------------------------
mat0299       6
xyz0123       5
----------------------------
MESG mat0299 Hello!
MESG brb5678 Are you there?
Unknown Recipient (brb5678). MESG Discarded.
FROM brb5678: I'm here!
FROM brb5678: Who else is on?
QUIT
Connection closed by foreign host.
```

==> CLIENT 3 on cse03

```
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
JOIN brb5678
JOIN brb5678 Request Accepted
LIST
USERNAME     FD
----------------------------
mat0299       6
xyz0123       5
brb5678       7
----------------------------
MESG xyz0123 I'm here!
BCST Who else is on?
QUIT
Connection closed by foreign host.
```

The following SAMPLE OUTPUT documents when 10 users are currently registered with the service and 1 more client attempts to register (note that "register" implies a difference to "connection"). There are two cases here: (1) more than 10 users have connected, but not yet registered when the 11th client attempts to register, and (2) 10 users have connected and registered when the 11th client attempts to connect.

==> SERVER on cse06

```
...
Client (25): Database Full. Disconnecting User.
...
Error: Too Many Clients Connected
...
```

==> CLIENT 11 on cse04

```
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
JOIN asd6789
Too Many Users. Disconnecting User.
Connection closed by foreign host.
...
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
Connection closed by foreign host.
```

The following SAMPLE OUTPUT documents when an already registered user sends another JOIN request with (1) the same username as well as (2) a different username.

==> SERVER on cse06

```
$ ./prog3svr 8001
Waiting for Incoming Connections...
Client (5): Connection Accepted
Client (5): Handler Assigned
Client (5): JOIN mat0299
Client (5): PONG
Client (5): LIST
Client (5): User Already Registered. Discarding JOIN.
Client (5): User Already Registered. Discarding JOIN.
Client (5): LEAVE
Client (5): Disconnecting User.
^C
```

==> CLIENT 1 on cse04

```
$ telnet cse06 8001
Trying 129.120.151.99...
Connected to cse06.cse.unt.edu.
Escape character is '^]'.
JOIN mat0299
PING
PONG
LIST
USERNAME     FD
----------------------------
mat0299       5
----------------------------
JOIN mat0299
User Already Registered: Username (mat0299), FD (5)
QUIT
Connection closed by foreign host.
```

REQUIREMENTS:

This assignment must be submitted via Canvas with the following elements:

- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.
- Your C program file should be named "**`prog3svr.c`**", without the quotes, for the server code.
- Your program will be graded based largely on whether it works correctly on the CSE machines (e.g., `cse01`, `cse02`, …, `cse06`), so you should make sure that your program compiles and runs on a CSE machine.
- Please pay attention to the **SAMPLE OUTPUT** for how this program is expected to work. If you have any questions about this, please contact your instructor or TA assigned to this course to ensure you understand these directions.
- This is an individual programming assignment that must be the sole work of the individual student. Any instance of academic dishonesty will result in a grade of "F" for the course, along with a report filed into the Academic Integrity Database.

SUBMISSION:

- You will electronically submit your server C program to the **Program 3** dropbox in Canvas by the due date.