INP111 Lab05 Week #6 (2022-10-20)

Date: 2022-10-20

- INP111 Lab05 Week #6 (2022-10-20)
- #1 Say Hey! to Your Friend
  - Description
  - Specification
    - Commands (received from a Client)
    - Messages (sent from the Server)
    - Server Behavior
  - Demonstration
  - Grading Policy

#1 Say *Hey!* to Your Friend

This lab aims to practice I/O multiplexing by implementing a single process chat server.

Description

1. Please implement a single process chat server by following the requirements in the spec section. You may check how to use the relevant functions select(2) (https://man7.org/linux/man-pages/man2/select.2.html), poll(2) (https://man7.org/linux/man-pages/man2/poll.2.html), or epoll(7) (https://man7.org/linux/man-pages/man7/epoll.7.html) functions on Linux.
2. Once finished your server implementation, you can test your server with the `nc` program and ensure your implementation is correct.

> ***Hint***: You may modify the sample codes (https://github.com/unpbook/unpv13e) from the textbook if you think it is easier to complete this lab.

> You must implemenet everything in C or C++.

Specification

Commands (received from a Client)

All the commands are plain-text messages encoded in UTF-8 encoding.

- A text message that starts from a leading slash `/` is considered a command. Currently we only have two commands:
    - `/name <nickname>` Set the nickname of the current user (without chevrons).
    - `/who` List all online users.
- All other text messages sent from a client are considered as text messages and must be delivered to all online users.

Messages (sent from the Server)

All the messages are plain-text messages encoded in UTF-8 encoding.

- `YYYY-MM-DD HH:MM:SS *** {message}` A system message sent from the server to a client. When sending a message, the current date and time must be prepended before the message.
- `YYYY-MM-DD HH:MM:SS <nickname> {message}` A chat message forwarded by the server. The server should deliver a chat message from a user to all online users. When delivering a message, the current date, time, and message sender's nickname must be prepended before the message.
- ***User list message*** The special message for responding `/who` command. The format of the message is shown as follows:

```
-----------------------------------
{user nickname}     {user's IP address and port number}
(... one line for each user, multiple lines ...)
-----------------------------------
```

Server Behavior

✓ 1. The server must send a two-line welcome message to a new user. The first line simply shows a banner message. The second line shows the number of the online users and a ***randomly generated*** user nickname. A sample message is given below.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

```
2022-10-05 15:45:00 *** Welcome to the simple CHAT server
2022-10-05 15:45:00 *** Total 2 users online now. Your name is <becoming babirusa
```

> We do not have a rule on generating the user nickname. It can be a simple random string. However, if you are interested in how we generate random names, you can download our dictionary from here (https://inp111.zoolab.org/lab05.1/aa.h.txt).

✓ 2. When a user has successfully connected to the server, a system notification message must be sent to all online users to let others know we have a new user online.

```
2022-10-05 21:50:08 *** User <fabulous caudata> has just landed on the server
```

✓ 3. On receipt of the `/name` (0123456) command, the server replaces the nickname of the user with the given one (one space character after the command). After updating the user's nickname, a system notification message is sent to the user and all online users. The message sent to the user and all online users is shown below.

```
(to the user)
2022-10-05 16:05:34 *** Nickname changed to <hello, world!>
(to the rest of the online users)
2022-10-05 16:06:03 *** User <jubilant cob> renamed to <hello, world!>
```

✓ 4. On receipt of the `/who` command, the server sends a list of all online users to the client using the ***user list message*** format. A sample response is shown below. Note that an asterisk `*` indicates the current user.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
--------------------------------------------------
  uncivil mayfly       140.113.1.2:55360
  browny bison         140.113.1.2:59476
* hello, world!        140.113.1.2:59478
--------------------------------------------------
```

✓ 5. On receipt of an unknown or incomplete command, the server must respond to an error message. Note That the received command should be included in the message, as shown below.

```
2022-10-05 16:13:40 *** Unknown or incomplete command </name>
```

6. On receipt of a regular text message, the server must deliver it to all online users using the ***chat message*** format.

```
2022-10-05 16:20:12 <flayed jabiru> hey guys, how's everything?
```

7. When a client is disconnected from the server, the server must notify all online users that we have one user left the server.

```
2022-10-05 16:23:05 *** User <flayed jabiru> has left the server
```

8. The server should log arrival and departure users. The sample messages are shown below.

```
* client connected from 140.113.1.1:38864
* client 140.113.1.1:38864 disconnected
```

You may want to play with our sample implementation. You can connect to our sample implementation using the command `nc inp111.zoolab.org 10005` .

Demonstration

1. You should use (at least) three terminal windows to demonstrate your implementation. The minimal setup is having one for running the server, another for client A, and the other for client B.
2. Run the following commands in these three windows, respectively, to run the server and have two clients connect to the server:

```
./server 10005        # for the server
nc localhost 10005    # for the first client
nc localhost 10005    # for the second client
```

3. Evaluate your program with all items listed in the **Grading Policy** section.

Grading Policy

- [15%] Run the server and have clients connect to the server. The clients should receive correct welcome messages.
- [15%] The `/name` command works as expected.
- [10%] The `/who` command works as expected.

- [10%] When a new client connects to the server, the server delivers a new user online notification message to all connected clients.
- [10%] Messages from each client can be delivered to all online users.
- [10%] Correct messages are delivered to all online users on user arrival and departure.
- [10%] A correct error message is displayed for unknown or incomplete commands.
- [10%] Your server can handle 1000 concurrent users online. You can use the command to test this case:
  `bash -c 'MAX=1000; I=0; while [ "$I" -lt "$MAX" ]; do I=$((I+1)); (timeout 30 nc localhost 10005 >/dev/null 2>&1 &) done'`
  The command emulates 1000 concurrent clients connecting to your server, and each client leaves the server after 30 seconds. In the meantime, you can have two additional clients connect to the server to ensure that your server works properly.
- [10%] After all the 1000 users are disconnected from your server, repeat the last experiments again to see if your server can handle another 1000 concurrent users online. ***You have to use the same server instance in the previous test case. You cannot restart your server for this test case.***