CSCI-4220 Network Programming Project #3

Shigeru Imai (imais@rpi.edu)

April 19, 2014

1 Compilation (20 points)

- Compiled languages (C, C++, Java, ...): Graded based on compilation-time errors. Warnings are ignored. Only apparent errors are penalized (5 points per one type of errors).
- Interpreted languages (Python, JavaScript, Ruby...): 20 points are given unconditionally; however, run-time errors are graded harshly compared to the compiled languages.

Observed programming languages/frameworks: C, C++, Java, Nodejs, Python2, Python3, Go, Ruby, The Kawa Scheme language.

2 Testing with TCP Clients (50 points)

Preparation

- The server is started with a verbose (`-v`) option and port 8888. In case of a python server, it looks like:
  `$ python char_server.py -v 8888`
- The test client is `nc`, which is also started on the same computer:
  `$ nc localhost 8888`
- Three users (`gaga, selena, paris`) are used for testing, and each user uses one `nc` client.

(a) **ME IS** (10 points)

- **OK case** (5 points)
  Test case:
  Each user logs in from an `nc` client. For example, in case of `gaga`, type the following:
  `ME IS gaga[hit enter here]`

  Expected result:
  An `OK` response from the server for each user.
- **ERROR case** (5 points)
  Test case:
  After the three users successfully logged in, again, send the following command from any of the three `nc` clients:
  `ME IS gaga[hit enter here]`

  If this does not work, it is also acceptable to send this command from another `nc` client on a new terminal.

Expected result
An `ERROR` response from the server.

(b) **SEND** (20 points)

- **Regular message** (10 points)
  Test case:
  From `gaga`'s client, type either of the following commands to send a message to `selena`:
  - `SEND gaga selena[hit enter here]`
    `5[hit enter here]`
    `Hello[hit enter here]`
  - `SEND gaga selena^J5^JHello[hit enter here]`
    Note that `^J` represents a new line character (`'\n'`). To type `^J`, hit `Ctrl-V Ctrl-J`.

  It is also OK to use 6 instead of 5 to account for a new line character at the end of the message (*i.e.*, `Hello'\n'`).

  Expected result:
  The `Hello` message is displayed on `selena`'s client.
- **Chunked message** (10 points)
  Test case:
  From `gaga`'s client, type either of the following commands to send a chunked message to `selena`:
  - `SEND gaga selena[hit enter here]`
    `C5[hit enter here]`
    `Hello[hit enter here]`
    `C0[hit enter here]`
  - `SEND gaga selena^JC5^JHello^JC0[hit enter here]`

  It is also OK to use C6 instead of C5.

  Expected result:
  The chunked `Hello` message is displayed on `selena`'s client.

(c) **BROADCAST** (10 points)
Test case:
From `gaga`'s client, type either of the following commands to broadcast a message:

- `BROADCAST gaga[hit enter here]`
  `5[hit enter here]`
  `Hello[hit enter here]`
- `BROADCAST gaga^J5^JHello[hit enter here]`

It is also OK to use 6 instead of 5.

Expected result:
The `Hello` message is displayed on all the clients.

(d) **WHO HERE** (5 points)
Test case:
From `gaga`'s client, type the following command:
`WHO HERE gaga[hit enter here]`

Expected result:
Active users (`gaga selena paris`) are shown on `gaga`'s client.

(e) **LOGOUT** (5 points)
Test case:
First, from `gaga`'s client, type the following to logout `gaga`:
`LOGOUT gaga[hit enter here]`
And then, from `selena`'s client, type the following to show the current users:
`WHO HERE selena[hit enter here]`

Expected result:
Active users (`selena paris`) are shown on `selena`'s client.

3 Testing with UDP Clients (20 points)

Preparation

- The server is started with a verbose (`-v`) option and port 8888.
- The test client is `nc`, but with a UDP (`-u`) option:
  `$ nc localhost 8888 -u`
- Two users (`gaga, selena`) are used for testing, and each user uses one `nc` client.

(a) **ME IS** (10 points)
Each user logs in from an `nc` client. For example, in case of `gaga`, type the following:
`ME IS gaga[hit enter here]`

Expected result:
An `OK` response from the server for each user.

(b) **SEND** (10 points)
Test case:
From `gaga`'s client, type one of the following commands to send a message to `selena`:

- `SEND gaga selena[hit enter here]`
  `5[hit enter here]`
  `Hello[hit enter here]`
- `SEND gaga selena^J5^JHello[hit enter here]`

Expected result:
The `Hello` message is displayed on `selena`'s client.

4 Common Server Functions (10 points)

- **Random messages** (5 points)
  After every three messages sent to a client,the server must send a random message from one of the previous three senders (selected randomly).
- **Server output** (5 points)
  The server should display all messages sent to and from it.

Extra Credits (10 points)

- **SEND request to multiple users** (+5 points)
  With the same setting used in "Testing with TCP clients", we check if `gaga` can send a message to `selena` and `paris` simultaneously by one of the following commands:

  - `SEND gaga selena paris[hit enter here]`
    `5[hit enter here]`
    `Hello[hit enter here]`
  - `SEND gaga selena paris^J5^JHello[hit enter here]`

- **Separate "Chat universe"** (+5 points)

  - The server is started with a verbose (`-v`) option and ports 8888 and 9999. In case of a python server it looks like:
    `$ python chat_server.py -v 8888 9999`
  - `gaga` and `selena` login to port 8888, whereas `paris` logs in to port 9999.
  - `gaga` broadcasts a message to the port 8888 universe. We check if only `gaga` and `selena` receive the message.

Notes

- If you are not satisfied with your grade, please run the test cases described in this document. If you think that I made a mistake in grading, come to my office hour or send me an email.