[redacted]

Due Thursday, November 9, 2023

1. **The Assignment 2**

For this assignment, the server must allow multiple clients connect the server at the same time. You are required to implement a multithreaded server (using Pthread or Java thread) and a client that can monitor the server's message and the user's input at the same time (using select() or thread).

Besides the original six commands (see the assignment 1), your new yamotd server performs two additional commands:

• SEND, forwards a private message to an active user

• WHO, returns a list of active users, including UserIDs and IP addresses.

You will need to implement MSGGET, MSGSTORE, **WHO, SEND**, SHUTDOWN, LOGIN, LOGOUT, QUIT commands on the client side and the corresponding function on the server side.

The details of the yamotd protocol depend on the command the client sends to the server.

MSGGET, **MSGSTORE**, **LOGIN, LOGOUT, and QUIT**

Same as the assignment 1.

WHO

List all active users, including the UserID and the users IP addresses.

A client-server interaction with the WHO command thus looks like:

c: WHO
s: 200 OK
The list of the active users:

john	141.215.10.30
root	127.0.0.1

SEND

Send a private message to an active user. If the UserID is invalid or the receiver is not active, the server replies the client with an error message "420 either the user does not exist or is not logged in"; otherwise, the server forwards the message to the designated user. In addition, the receiving client should process the message immediately.

A client-server interaction with the WHO command thus looks like:

At David's window
c: SEND john
s: 200 OK
c: Hello John

s: 200 OK

at John's window:
s: 200 OK you have a new message from david
david: Hello John

SHUTDOWN

The basic requirement is the same as the assignment 1. In this assignment, the SHUTDOWN **will make all clients and the server terminate**.

A client-server interaction with the SHUTDOWN command thus looks like:

c: SHUTDOWN
s: 200 OK

At the windows of all clients

s: 210 the server is about to shutdown ……

2. Programming Environment

You can use either C/C++ or Java to implement the assignments. The assignments will be tested on the UMD Login servers (login.umd.umich.edu). For easy grading, please don't use any GUI interface.

3. Requirements

The following items are required for full-credit:

- implement all eight commands: MSGGET, MSGSTORE, SEND, WHO, SHUTDOWN, LOGIN, LOGOUT, QUIT
- the users information should be maintained by the server. You must have the following users (lower case) in your system:

UserID	Password
root	root01
john	john01
david	david01
mary	mary01

- the server IP address should be a command line parameter for the client program.
- the server should output all client activities on the screen.
- make sure that you do sufficient error handling such that a user can't crash your server. For instance, what will you do if a user provides invalid input?
- your source codes must be commented
- include a **README** file in your submission.
- include a **Makefile** in your submission.

Note 1, in your README file, the following information should be included: the

commands that have been implemented, the instructions about how to run your program, known bugs, and sample outputs.

Note 2, your Makefile might be the exact same as the sample Makefile if your file names are the same as those of the sample codes.

4. Grading (100 points)

- Correctness and Robustness (90 points)
  - You will lose 10 points for any bugs that cause the system crash.
  - You will lose 5 points for any other bugs.
- Comments and style (5 points)
- README and Makefile (5 points)

5. Submission Instruction

(1) Copy all your files (only source codes, data file, README, and Makefile, no object file and executable file) in a directory. The directory should be named like lastname_firstnameinitial_p2. For example, if you name is John Smith, your directory name should **smith_j_p2**.

(2) Generate a tar file of the directory using the following command. Enter the parent directory of your current directory and type

tar cvf lastname_firstnameinitial_p2.tar lastname_firstnameinitial_p2

For example

tar cvf smith_j_p2.tar smith_j_p2

Note, only the tar file will be accepted. You are fully responsible for generating the right tar file.

(3) Submit the tar file to the canvas "P2" assignment folder.

6. Hints

- I would recommend that you use the sample programs as the base codes and add the commands implemented in the assignment 1, and then implement the new commands WHO and SEND.
- Start early. These programs will not be very long, but they may be difficult to write, and they will certainly be difficult to debug.

commands that have been implemented, the instructions about how to run your program, known bugs, and sample outputs.

Note 2, your Makefile might be the exact same as the sample Makefile if your file names are the same as those of the sample codes.

4. Grading (100 points)

- Correctness and Robustness (90 points)
  - You will lose 10 points for any bugs that cause the system crash.
  - You will lose 5 points for any other bugs.
- Comments and style (5 points)
- README and Makefile (5 points)

5. Submission Instruction

(1) Copy all your files (only source codes, data file, README, and Makefile, no object file and executable file) in a directory. The directory should be named like lastname_firstnameinitial_p2. For example, if you name is John Smith, your directory name should **smith_j_p2**.

(2) Generate a tar file of the directory using the following command. Enter the parent directory of your current directory and type

tar cvf lastname_firstnameinitial_p2.tar lastname_firstnameinitial_p2

For example

tar cvf smith_j_p2.tar smith_j_p2

Note, only the tar file will be accepted. You are fully responsible for generating the right tar file.

(3) Submit the tar file to the canvas "P2" assignment folder.

6. Hints

- I would recommend that you use the sample programs as the base codes and add the commands implemented in the assignment 1, and then implement the new commands WHO and SEND.
- Start early. These programs will not be very long, but they may be difficult to write, and they will certainly be difficult to debug.

commands that have been implemented, the instructions about how to run your program, known bugs, and sample outputs.

Note 2, your Makefile might be the exact same as the sample Makefile if your file names are the same as those of the sample codes.

4. Grading (100 points)

- Correctness and Robustness (90 points)
  - You will lose 10 points for any bugs that cause the system crash.
  - You will lose 5 points for any other bugs.
- Comments and style (5 points)
- README and Makefile (5 points)

5. Submission Instruction

(1) Copy all your files (only source codes, data file, README, and Makefile, no object file and executable file) in a directory. The directory should be named like lastname_firstnameinitial_p2. For example, if you name is John Smith, your directory name should **smith_j_p2**.

(2) Generate a tar file of the directory using the following command. Enter the parent directory of your current directory and type

tar cvf lastname_firstnameinitial_p2.tar lastname_firstnameinitial_p2

For example

tar cvf smith_j_p2.tar smith_j_p2

Note, only the tar file will be accepted. You are fully responsible for generating the right tar file.

(3) Submit the tar file to the canvas "P2" assignment folder.

6. Hints

- I would recommend that you use the sample programs as the base codes and add the commands implemented in the assignment 1, and then implement the new commands WHO and SEND.
- Start early. These programs will not be very long, but they may be difficult to write, and they will certainly be difficult to debug.