CIS527 Fall 2023 Programming Assignment 1

Professor Jinhua Guo

Due Wednesday, September 27, 2023

1. Introduction

This programming assignment is designed to let you familiar with the socket interface and client-server application. This assignment weights 10 % of your final grade.

In this programming assignment, you will create a server process that implements yet another "message of the day" (yamotd) protocol with a client process that you will also create. The client and server processes will communicate using TCP sockets and will implement the yamotd protocol discussed below.

The yamotd server performs four functions:

- returns a message of the day to any user that sends the server a MSGGET message
- allows a user, whose identity has been verified by the server, to send the server a MSGSTORE message to upload one or more messages of the day to the server. These messages will be stored in a file on the server and can be returned to other clients that later send MSGGET messages to the server.
- allows the “root” user to send the server a SHUTDOWN message, which will cause the server to close any open sockets, and then terminate.
- verifies the identity of a user through LOGIN

2. The Assignment 1

You will write two programs, a server and a client. The server creates a socket in the Internet domain bound to port SERVER_PORT (a constant you should define in both programs, you may use last 4-5 digits of your UM-ID or your SSN). The server receives requests through this socket, acts on those requests, and returns the results to the requester. The client will also create a socket in the Internet domain, send requests to the SERVER_PORT of a computer specified on the **command-line**, and receive responses through this socket from a server.

For this assignment, you just need to allow one active client connect to the server. In the next assignment, you should allow multiple clients connect the server at the same time.

Your client and server should operate as follows. Your server begins execution by opening a file that you have created and that initially contains five messages of the day. (A "message of the day" is a short phrase that may or may not be meaningful, such as "One can never be too rich, too thin, or have too much bandwidth.", or "Anyone who has never made a mistake has never tried anything new."). Your server should read these messages of the day into a server-internal data structure and keep track of the number of messages of the day. You may assume that you will never have to store more than some fixed number, say 20, messages of the day. Once the

server has initialized its data structures, it should wait for the connection requests from the clients.

Your client operates by sending a MSGGET, MSGSTORE, SHUDOWN, LOGIN, LOGOUT, QUIT commands to the server. You should create a client that is able to send any of the six commands above, and allows a user to specify which of the commands the client should send to the server.

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

MSGGET

The MSGGET command, which is sent from the client to the server, consists solely of the ASCII string "MSGGET" followed by the newline character (i.e., '\n'). After sending the MSGGET command, the client will wait to receive a message of the day message back from the server via the socket. After displaying the message of the day, the client should loop back and allow the user to indicate the next command to be sent.

When your server receives a MSGGET command from a client, it should return the string "200 OK" (terminated with a newline), followed by one of the messages of the day (also terminated with a newline). The server should just return one message. That message should be chosen sequentially by cycling through its messages.

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

c: MSGGET
s: 200 OK
Anyone who has never made a mistake has never tried anything new.

Note, all these messages should be displayed at the client side.

MSGSTORE

The MSGSTORE command allows a logged in user to upload one message to the server. A client that wants to store one message should begin by sending the ASCII string "MSGSTORE", followed by the newline character (i.e., '\n'). The client should then wait for the server to return a "200 OK" message (indicating that the client is authorized to upload messages), or a "401 You are not currently logged in, login first." If the client has been authorized to upload a message, the client responds to the "200 OK message" by sending one message to be added to the server's message store. The message should be terminated with a newline. After sending the MSGSTORE command, and the messages of the day, the client should read and display the return code sent by the server.

When your server receives a MSGSTORE command from a client, it should check the login status of the client. If the client has not logged in yet, the server should return the string "401

You are not currently logged in, login first," otherwise it should return the "200 OK message," as discussed above.

If the user is allowed to execute the MSGSTORE command, the server should then read in and store (in its internal data structures and in the file), the new additional message of the day. If the message of the day is received correctly, your server should return the string "200 OK" (terminated with a newline).

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

c: MSGSTORE
s: 200 OK
c: Imagination is more important than knowledge.
s: 200 OK

SHUTDOWN

The SHUTDOWN command, which is sent from the client to the server, is a single line message that allows the root user to shutdown the server. The root user that wants to shutdown the server should send the ASCII string "SHUTDOWN" followed by the newline character (i.e., '\n'). The client should then read and display the return code sent by the server, and then close the connection.

When your server receives a SHUTDOWN command from a client, it should check if the current user is the root. If it is not the root user, the server should return the string "402 User not allowed to execute this command."

If the user is allowed to execute the SHUTDOWN command, the server should return the string "200 OK" (terminated with a newline), close all open sockets and files, and then terminate. In case of an error, the string "300 message format error" should be returned.

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

c: SHUTDOWN
s: 200 OK

LOGIN

Identify the user to the remote server. A client that wants to login should begin by sending the ASCII string “LOGIN" followed by a space, followed by a UserID, followed by a space, followed by a Password, and followed by the newline character (i.e., '\n').

Your server should be initialized with the UserIDs and Passwords of at least four users who will be allowed to execute the MSGSTORE and SHUTDOWN (the root user only) commands at the server.

When the server receives a LOGIN command from a client, it should check if the UserID and Password are correct and match each other. If they are not correct or don't match each other, the server should return the string "410 Wrong UserID or Password," otherwise it should return the "200 OK" message.

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

c: LOGIN john john01
s: 200 OK

LOGOUT

Logout from the server. A user is not allowed to send MSGSTORE and SHUTDOWN commands after logout, but it can still send MSGGET command.

A client-server interaction with the LOGOUT command looks like:

c: LOGOUT
s: 200 OK

QUIT
Terminate the client with the remote server. The client exits when it receives the confirmation message from the server.

A client-server interaction with the LOGOUT command looks like:

c: QUIT
s: 200 OK

3. Format

You should form a group of two students (or work individually) and then jointly design your project. While each project should be an integrated effort, you should identify in your README file what part of the project each member is responsible for.

Note, please form your group as soon as possible. Every student needs to sign up for one of the programming groups on canvas no later than 9/15.

4. 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.

Requirements

The following items are required for full-credit:

- implement all six commands: MSGGET, MSGSTORE, 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

- both the server and client should be able to run on the UMD Login servers
- the server IP address should be a command line parameter for the client program.
- the server should print out all messages received from clients on the screen.
- when the previous client exits, the server should allow the next client connect.
- 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 functions 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.

5. Grading (100 points)

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

6. 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_p1. For example, if you name is John Smith, your directory name should **smith_j_p1**.

(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_p1.tar lastname_firstnameinitial_p1

For example

tar cvf smith_j_p1.tar smith_j_p1

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 website "P1" assignment folder.

7. Hints

- I would strongly suggest that everyone begin by writing a client and server that execute the simple MSGGET and QUIT commands first. Then program the LOGIN and LOGOUT commands. Finally, program the MSGSTORE and SHUTDOWN commands.

- Start early. These programs will not be very long, but they may be difficult to write, and they will certainly be difficult to debug.
- The sample programs (both C/C++ and java) are available at the canvas website.
- **SSH**

  Mac OS and Linux come with a ssh client you can run from a Terminal. You do not need to install anything. Microsoft Windows users may download the SSH software, from the following site: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

- **Accessing the UM-Dearborn Home Drive**

  https://umdearborn.teamdynamix.com/TDClient/2019/Portal/KB/ArticleDet?ID=42879