Computer Networks Programming Assignment 1

Fall 2019

Due: 07 October 2019

The big picture here is that you will write two simple programs: One that acts like a basic POP3 server and one that acts like a simple POP3 client. Your programs do not have to implement the full POP3 spec, but all of the behavior is inspired by it.

1 The Setup

You will write two programs. The server should run forever, listening for **TCP** requests. When the request arrives, a conversation similar to that on slide 5 of the email lecture will occur. See section 2 for a complete list of commands the server must support. After the `QUIT` command, the TCP connection should be closed and the server returns to listening for new connections.

The client will be an interactive program, rather than the "behind the scenes" behavior of a real email client. The user should be prompted to type any of the commands listed in section 2.

For simplicity, let all of the emails exist as separate plaintext files. (This eliminates the nastiness of file pointers or figuring out some way to build a custom message store.

Posted with this assignment are three plaintext (RFC 5322) emails. Upon starting up, the server should detect that these files exist and respond to the client's queries based on this information. For example, if the client issues the `LIST` command, the server should report those three emails and their respective sizes. A `DELE` command should have the effect of truly deleting that message from the "store" (i.e., deleting the file). If I fire up the client again, deleted messages should not be available for download again.

The client is only truly responsible for getting the email and saving it locally. You do not need to display them (unless you would like some extra points — see section 5). After a session communicating with the server, the client should exit.

2 Some Details

Your server should be listening on a port number that is passed as a command-line argument (e.g., `./server 4272`). If you want, you can use a default value if nothing is passed on the command line. If you are developing on MTU machines, there is a possibility that two servers may try to use the same port. To reduce this possibility, use the last four digits of your M-number (provided they are greater than 1024). This has historically worked pretty well.

The client should be passed the IP address and port of the server as command line arguments (e.g., `./client localhost 4272`). The program should accept `localhost` or a numeric IP address.

The server should support the following POP3 commands:

- STAT
- LIST
- RETR
- DELE
- TOP
- QUIT

See the slides or a POP3 reference for descriptions of each of these commands.

3 Deliverables

You will be submitting the source code for the two programs. They may be written in C or Python 3. If you choose to use Python 3 and somehow feel the need for some fancy package, explicitly indicate in your submission how to install that. I will either run your code on my Mac or on the CS Colossus machine, so your code should run in one of those environments. No "notebooks" (e.g., Jupyter) are permitted.

If you choose to develop in C, it would be appreciated if you provide a makefile. (Appreciated, not required.)

Be sure to submit the test email files along with your source code. I will deduct 5 points from your final score if I have to copy those files to test your code.

If any special instructions are needed for compilation or running, please include those as a simple README file. If you have no special instructions, just omit that file.

It is preferred that all files be zipped up and submitted as a single package. Tarballs, Gzipped and "plain" Zip files are great. 7-zip and RAR archives are much less great.

4 Grading

Each program is worth 50 points. I will test all of the commands listed in section 2, which will be worth 8 points each. 2 points per program are just for successful compilation/execution of *something*. (In other words, you could submit two programs that just print out "Email is cool" and get 4 points.)

5 Extra Credit

For 20 extra points, modify the client so that it can display the emails retrieved from the server.