Quote Server and Client

Build a network server and client to implement the "quote of the day service". The protocol is based on RFC 865 "Quote of the Day protocol" with the following modifications:

- The server and client will use port 8017 *
- Only TCP service is supported, no UDP service
- The quotes may be randomly determined or sequentially served from a file
- An example "quotes.txt" file is linked on the class web page

There are no command line arguments or menu for the server (no interface - it is a daemon). Server shutdown should be handled by sending a CTRL-C (SIGINT) to the server from the console. We will learn how to improve this later.

The client has one required command line argument, the hostname or IP address of the server. On success, the client will display the received string to the console and then terminate.

The server and client should handle errors gracefully. Graceful means your programs should recover from the error and continue if possible, otherwise they should display an informative message, cleanup resources, and terminate.

Your program must conform to the **UNA CS Code Style Guide** linked on the class web page. Your program must compile without any compiler warnings.

Your programs must be developed in *C* (-std=c17) using the BSD socket API on the CS server. The programs must compile and run on the CS server with cc (clang).

Testing your server without your client:

See the note below about ports for testing your programs.

You may test your server, before you create your client, with the netcat utility nc(1). The netcat utility will send and receive messages over the network. To test your TCP server listening on port 8017, run your server in one terminal window (server's terminal). Then in a different terminal window (client's terminal) you would type the following:

```
echo "\r" |nc localhost 8017
```

If your server is working correctly you'll see the string displayed in the client's terminal. The port number must match the port your server is listening on in order to work.

Instead of logging in twice to the server, you may use the terminal multiplexer tmux(1). The following list shows the tmux commands needed to open two screens for testing:

- `tmux` - start the tmux program
- `CTRL+b "` - split the current pane into two panes, top and bottom
- `CTRL+b` ↑ - move the cursor up a pane
- `CTRL+b` ↓ - move the cursor down a pane
- `exit` - close a pane (closing the last pane exits tmux)

See the man page tmux(1) for additional information.

★ Note: To avoid conflicting with your classmate's programs when testing on the CS server, use the last three digits of your student ID L# plus 11000 for your port number.

Due dates:

Friday 16th before 11:59 p.m.
Completed server commit, tag it "server"

Wednesday 21 before 11:59 p.m.
Completed client commit, tag it "client"

Use an annotated tag *after* your code commit.

How to submit:

Create an empty git(1) repository in a folder named "program-1". Commit and tag your files as described above in "Due dates". You may, and should, commit more frequently than the required commit dates. You should build your server and client programs in stages (skeleton, startup, etc.) and commit the working version of each stage as you complete it.

Program evaluation:

Your program must conform to the **UNA C Code Style**. The "UNA C Code Style" is linked on the class web page. Code will be evaluated for correctness, efficiency, and style.

Helpful resources:

```
Command line arguments: https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/
--
man pages - err(3), printf(3)
            fopen(3), getline(3), fread(3), fseek(3), ferror(3)
            style(9), nc(1)
```

Additional resources:

```
https://beej.us/guide/bgnet/
https://csrc.nist.gov/Projects/ssdf
https://sourceware.org/gdb/current/onlinedocs/gdb.html
```