ASSESSED INDIVIDUAL WORK

COMP1007
LAB EXERCISE #4

Steven R. Bagley

Submission Deadline:
26/11/2021 15:00

Version: 1.00

A change from the usual HDL this week (although we will return to it later in the course). For this coursework, you are to implement a simple network client that can connect to a server and fetch a number of 'Quotes of the Day' from the server, and then print them out to the client. There is actually a standard Internet protocol for this which is defined in RFC[1] 865, but we are going to use a slightly modified version that enables more than one quote to be fetched. Details of the protocol are given below.

A skeleton C file for the client is available via `git` in the repository below. Please ensure you use this for your implementation. It is expected that you will be developing and testing this coursework on the school's Linux systems. In particularly, you will be unable to access the supplied testing server from outside of the University network. The skeleton file can be found in the repository at:

`https://projects.cs.nott.ac.uk/2021-COMP1007/2021-COMP1007-LabExercise04`

Good luck…

Implementation Detail

Inside the git repository, you forked and cloned via `git` in the usual fashion, you will find the skeleton `.c` file you need to implement the client. It is **strongly recommended** that you use this skeleton otherwise your program may not work in the pipeline. The pipeline will not be opened till the 21st November, this is so that you think about how you can test your program yourself…

The pipeline for this exercise will be opened on Monday 21st November.

Assessment Notes

The pipeline will mark this coursework automatically by running your client against a test server, and then assess the quality of your client. A mark will then be awarded for your client out of 10, which will be your mark for this exercise. A mark of 40% (4/10) or above is a pass mark for the exercise.

Combined, the lab exercises for COMP1007 form part of your programming portfolio will account for 10% of the mark for the module.

[1] Most Internet protocols are defined in documents called Requests for Comments (or RFC, for short)

Multi-quote of the Day Protocol

This lab exercise revolves around the following network protocol that allows a client to retrieve multiple quotes from a server. This protocol runs on port 1818.

Upon connection, the server will immediately respond with a quote. The server will then wait for a command from the client to be sent over the network connection. All communication between the client and server is sent as complete lines of ASCII text that are terminated by a carriage return (`CR`, ASCII code 13) and a line feed (`LF`, ASCII code 10). All quotes are sent as a single line of text.

The client command can either be a line containing the word '`CLOSE`' (in capitals), in which case the server will send the response '`BYE`' and then close the connection to the client. Alternatively, the client can respond with the command '`ANOTHER`' (in capitals) in which case the server should send another quote and await further commands from the client. All commands and server responses (except the quotes) are in capitals.

If the server receives any command from the client other than '`ANOTHER`' or '`CLOSE`', it will send the response '`ERROR`' and wait for a proper response from the client.

If you were to peek at the data travelling over the network, an example exchange of data between the server (**`S:`**) and client (**`C:`**) might look like this… (the **`S:`** and **`C:`** are for illustrative purposes only — the communication would not normally be seen by the user since it is only sent via the TCP/IP connection)

```
[C connects]
S: He's dead, Jim...
C: ANOTHER
S: One day I shall come back, yes, I shall come back
C: FRED
S: ERROR
C: CLOSE
S: BYE
[S closes connection]
[C closes connection]
```

Multi-quote client

For this exercise, you should implement a multi-quote client, `MultiQuoteClient.c` as per the following description that can use the multi-quote protocol defined above to fetch a number of quotes from a server. The multi-quote client should take two parameters on the command line. The first parameter is the address of the server to connect too (this can either be an IP address or a domain name). The second parameter is the number of quotes to fetch from the specified server, e.g. like this:

```
$ ./mq-client some.server.name 5
```

The program will then output the quotes fetched from the server to the standard output (e.g. by using `printf`) separated by a blank lines, the number of quotes being defined by the second

command line parameter (which must be greater than or equal to zero. Your program should take no input from the user other than via the command line.

The quotes that are printed should be fetched from the specified server. Your client should connect to the server on port 1818, and read the first quote from the server, before printing it to the standard output. It should then automatically send the necessary commands (as described in the protocol section above) to the server, to cause the server to send the required number of quotes, before printing each quote to the standard output, separated by a single blank line.

Hint: Again, remember that you might have to read from the network connection multiple times to obtain a complete quote.

Compiling

Compile your program using the standard `gcc` compiler as follows (you **should not** use any flags that enable ANSI compatibility or similar, or your program may not compile correctly):

```
gcc -o mq-client MultiQuoteClient.c
```

Testing

To enable you to test your client independently of the pipeline, there is a test multi-quote server running on the machine `arrow.cs.nott.ac.uk` on port 1818. You can point your client at this machine to test your implementation.

Please note that you will only be able to access the test server from a machine on the University network. If you are off campus, trying to connect to the test server will fail.

If you want to experiment with the protocol manually, then you can use the `nc` program to do so, by executing the following command:

```
nc -C arrow.cs.nott.ac.uk 1818
```

You will then be able to type the commands for the protocol manually into the keyboard. Please note that depending on which version of nc you have installed you may need to use a lower-case -c flag (the school Linux's machines require an upper-case `-C`, macOS's version requires a lower-case `-c`).