CSE 156/L

Programming Assignment 1
Due: Jan [ ]

The purpose of this project is to develop the client side of a simple HTTP-based download application similar to `wget`. The user of the application can send requests with GET or HEAD methods to any public HTTP server and retrieve the corresponding document or just response headers as appropriate.

Requirements

- The client must accept as argument a hostname for the server hosting the document, e.g. `www.example.com`. The second argument is the URL path of the document. The URL would have the server's IP address, an optional port number (80), and the document path itself (e.g. `index.html`). For example,

  `./myweb www.example.com 93.184.216.34:80/index.html`

  Which would be equivalent to:

  `wget 93.184.216.34:80/index.html --header="Host:  www.example.com"`

  The corresponding basic HTTP request (using C string notation for CR and LF) would be:

  `GET /index.html HTTP/1.1\r\nHost: www.example.com\r\n\r\n`

- An optional parameter -h can be given by a user after all other parameters to only get the response and headers and not the document. For example with `-h`:

  `./myweb www.example.com 93.184.216.34/index.html -h`

  Equivalent to:

  `wget 93.184.216.34/index.html --header="Host:  www.example.com" --server-response --spider`

  With corresponding HTTP request:

  `HEAD /index.html HTTP/1.1\r\nHost: www.example.com\r\n\r\n`

- The client **must** output the document to the file `output.dat` unless the option `-h` is present, in which case there is no file output.
- If the client receives the option `-h`, it must output the complete server response onto `stdout` exactly as received. If the option is not present, then the client should output **nothing** on `stdout`.
- You are free to output any needed debug messages on `stderr`.
- The client must be robust and not crash if there are network errors, or if it received invalid parameters, it should instead close itself properly.

  An explanation of the error delivered on `stderr` is optional. It can, however, generate a partially completed (or even empty) `output.dat` file if it is interrupted while downloading the contents.
- The memory use of the client should not depend on the size of the file being downloaded, furthermore, the client should be able to work with documents of any size, as long as the computer has enough free space.

What to submit?

You must submit all files in a single compressed tar file (with `tar.gz` extension). The files should include

1. A README file including your name, student ID, and a list of files in the submission with a brief description. It should be in the top directory.
2. Organize the files into sub-directories (`src`, `bin`, and `doc`)
3. The source files should be in the `src` sub-directory.
4. A `Makefile` that can be used to build the client program from the required source files. It should be in the top directory.
5. You should not include the compiled program. The `Makefile` will be used to generated the executable program. The make step should put the executable program in the `bin` sub-directory.
6. Documentation of your application in plain text or pdf. Do not include any Microsoft Word files or other formats. The documentation should describe how to use your application and the internal design, as well as any shortcomings it might have. The documentation should list 5 test cases done by you to validate the functionality. This file should be in the `doc` sub-directory.
7. Name your file myweb-CruzID.tar.gz where CruzID is your UCSC email username, and myweb is the name of this assignment. Submit this file.

Grading

Each submission will be tested to make sure it works properly and can deal with errors. Grades are allocated using the guidelines below.

Basic Functionality:	50%
Dealing with Errors:	20%
Documentation:	20%
Style/Code Structure, etc.:	10%

Note that 20% of the grade will be based on how well your code deals with errors. Good practice includes checking all system calls for errors and avoiding unsafe situations such as a buffer overflow.

The files must be submitted before he due date and time. Please make sure to submit on time. Late policy commences immediately after the due date/time. Reminder: Late policy deducts 10% per day from your grade.

Honor Code

All the code must be developed independently. All the work must be your own.

Useful references

- HTTP requests: `https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html`
- HTTP responses: `https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html`
- Some examples of plain http websites for testing
  - `http://neverssl.com/`
  - `http://www.softwareqatest.com/`
  - `http://www.testingmcafeesites.com/`
  - `http://pudim.com.br/`

FAQ

- What is expected result when the input IP address doesn't correspond to a real server? The connection could be refused or failed, the program should terminate gracefully.
- Where should the response from a server be printed when there is a 404, 403, or any other status code? Should it go into output.dat or just write it to standard out? The response code line should go into **stderr**, the same way as if the response had a 200 code.
- I am having trouble leaving the while loop after I send the message and read the response. (You need to parse the head of the HTTP response to find information describing the body, so you can save the body to a file and end the connection. You should look at the Content-Length header)
- What if the Content-Length header is missing from the response? Read until the EOF is encountered.
- You can read more about HTTP requests here (https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html), and more about the inet_pton function here (https://linux.die.net/man/3/inet_aton). The hostname will be used when composing the HTTP request, you can read more about that here (https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.23).
- What if Transfer-Encoding header with a value of "chunked" is specified in the response? The client should print an error message to stderr that chunked encoding is not supported for this lab.

  `https://en.wikipedia.org/wiki/Chunked_transfer_encoding`