Lab 1 Documentation

How To Use:

1. cd into the folder lab01: “cd lab0”
2. There are 4 folders and 2 files. The 2 files are the makefile and the README.txt. The folders are bin, build, doc and src. In src, there is my source code. The build folder is for .o files, the doc folder is for documentation, and the bin folder is for the executable.
3. Compile the source code using: “make”
4. cd into the folder bin: “cd bin”
5. To run the executable, there are a few command line options you can choose from. The structure is as follows:
    a. ./myweb *hostname IP-Address:(optional port)/document-path*
    b. Eg: ./myweb www.example.com 93.184.216.34/index.html
    c. The output of the program will be placed into a file called output.dat, which is also in the bin directory.
6. You can also choose to see the headers only. The structure of this command is as follows:
    a. ./myweb *hostname IP-Address:(optional port)/document-path -h*
    b. Eg: ./myweb www.example.com 93.184.216.34/index.html -h
    c. The output of this will be directly in stdout.

Internal Design:

- The internal design of this program is fairly straightforward. In my main function, I call a function called get_commandline_args() that gathers the command line arguments into an organized struct. Then I pass this information to a function called send_request() which performs all of the socket system calls. Once I am connected to the server via a socket file descriptor, I read from this file descriptor. First I gather the headers, then parse for the status code and content length. Once I have the content length, I read bytes from the file descriptor into the file output.dat until bytes_read == content length. Once bytes read = content length, I know that there's no more bytes to read and I close the files and exit.

Shortcomings:

- In my testing of the finished program, I have not come across any shortcomings. Everything I've tested so far matches curl's output exactly.

Test Cases: (along with curl commands)

1. 301 code:
    a. ./myweb canvas.ucsc.edu 34.231.92.70/
    b. curl 34.231.92.70/ -H "Host: canvas.ucsc.edu" -o output1.dat
    c. This test should yield a 301: moved permanently code.
    d. "diff output1.dat output.dat:" returns no difference
2. Chunked encoding:
    a. ./myweb www.google.com 142.250.72.196/
    b. curl 142.250.72.196/ -H "Host: www.google.com" -o output1.dat
    c. This test should detect that the request sent back chunked encoding and should exit with an error message.
3. Big file:
    a. ./myweb www.washington.edu 128.95.155.134/
    b. curl 128.95.155.134/ -H "Host: www.washington.edu" -o output1.dat
    c. This test returns quite a large webpage. This confirmed to me that my program works with big files.
    d. "diff output1.dat output.dat:" returns no difference
4. Another Big File:
    a. ./myweb go.com 23.236.60.174/
    b. curl 23.236.60.174/ -H "Host: go.com" -o output1.dat
5. Regular Given Test From Doc:
    a. ./myweb www.example.com 93.184.216.34/index.html
    b. curl 93.184.216.34:80/index.html -H "Host: www.example.com" -o output1.dat
    c. "diff output1.dat output.dat:" returns no difference
6. Testing The Header Functionality:
    a. ./myweb www.example.com 93.184.216.34/index.html
    b. curl 93.184.216.34/index.html -I -H "Host: www.example.com"