CS260 Assignment #3:
HTTP Client

Task

This assignment builds off assignment 2 to create a simple HTTP client.

As with assignment 2, you will connect over TCP, send data, listen for the response, then close the socket and exit. In addition to the core set of tasks you had to perform in assignment 2, you will also need to do the following:

- Get the destination URL from the first command-line argument (argv[1]). You can *not* simply hard-code the destination IP address as in the first two assignments.
- Parse the destination URL to extract the hostname. The hostname will be everything in between the second and third slashes (so, for example, if the URL was "http://echo.cs260.net/test" the hostname would be "echo.cs260.net".
- Use DNS to fetch the IP address associated with the hostname.
- Compose a valid HTTP request.
- Parse the server's HTTP response.

For a good presentation of the minimal set of headers you need to send and receive to fetch a web page, read https://www.jmarshall.com/easy/http/#http1.1clients. Just as in assignment 2, you will receive multiple TCP packets. Accumulate the data into a buffer, watching for the “Content-Length” header so that you know how many bytes to expect from the body. You should *not* assume that you will see recv() return 0 when the body is complete—the server may choose to keep the connection alive. Once you have received Content-Length, seen the start of the body and received the specified number of bytes, dump that body to stdout and exit.

Linux Only

You do not need to support Windows with this assignment, nor do you need to support cross-platform compilation (though you may if you wish). Submit only a Linux project.

Testing

Technically, since you will be implementing the real HTTP standard, you can test your program by connecting to any web server. If you connect to http://echo.cs260.net/raw, the response will be the contents of your request. Note that the packet-logging page at http://echo.cs260.net/packets is not set up to capture traffic from port 80 and will not be used in this assignment. Also note that a valid URL is not guaranteed to have anything after the trailing slash.

Submission

Your Linux project must have a makefile and compile with nothing installed except build-essential. Once finished, upload your project folder and upload it to Moodle as a zip or tar. You should clear out any build detritus (executable files, object files, symbol tables, etc.).

Grading

From a base score of 100:

- If your program can't be compiled with make on Linux Mint, with nothing but build-essential installed, you lose 50 points.
- If your program doesn't connect to the server, you lose 70 points.

- If your program does not correctly compose the HTTP request, you lose up to 40 points, depending on how close to correct formatting you came.
- If your program does not correctly parse the HTTP response and then print out exactly the body of the document, you lose 30 points, depending on how close to correct formatting you came. No debug text or anything from previous assignments should be printed out.
- If your program has memory leaks, you lose 10 points.
- If your submission has build detritus, you lose 10 points.

If you score 89 or fewer points, you may correct errors and resubmit. Each resubmission carries a 10 point penalty.