Program 3: Text-Based HTTP Client

Assigned: Friday, October 13, 2006
Due: Friday, October 27, 2006 *before class*

Write a text-based HTTP/1.0 client that can request a single file from a web server. Your client must take a URL from the command line, connect to the server, request the file, display the server's HTTP response header to the user, and save the response data (excluding the HTTP response header) as a file. In general, the behavior of your client should be much like `wget -d`, printing both the HTTP request and the HTTP response. ***Do not wait until the last minute to start this assignment. No extensions will be given.***

Requirements

- Your program must take a URL as its only parameter. If given no parameters, your program must print a "Usage" statement and exit.
  - You may assume that all URLs will start with `http://`, contain at least one `/` after the hostname, and will not contain an alternate port (*i.e.*, no `:port` after the hostname).
- Print the HTTP request that the client sends to the server.
  - Your client *must* send the HTTP request fields `Host:` (with the proper value) and `User-Agent:` with the value `ODU-CS455` or `ODU-CS555`, as appropriate.
  - Do *not* form the HTTP request message and then issue a series of print statements with string constants. Print the contents of the buffer that is sent to the server.
- Print the HTTP response header (*and only the header!)* that the client receives from the server.
- Your client should have different actions depending upon the response code:
  - **200-level:** save the HTTP response data as a file (use filename found in the URL, or `index.html` if no filename is given in the URL).
  - **300-level:** handle the redirection by sending a new request for the URL given in the `Location:` field of the response. Note that since we are using HTTP/1.0, you must close the current socket and open a new socket before sending the new request. Your client must follow all HTTP redirections. In addition, your client must print out *all* HTTP requests that it sends and *all* HTTP response headers that it receives.
  - **400-level or 500-level:** display the HTTP header and exit - no file should be saved.
- ***CS 455 only*** – Your client needs only to be responsible for downloading and saving text-based files (*e.g.*, TXT, HTML).
- ***CS 555 only*** – Your client must allow both text and binary files (*e.g.,* PDF, JPG) to be downloaded in the proper format.
- ***CS 555 only*** – Your client must record and print the total amount of time elapsed for the object download, including redirects. (Start the timer before opening the first connection to the server and stop the timer after the object has been saved to disk.)

Note: It is in your best interest to split the program up into smaller pieces. Some example functions that you might want to write could include `getIP()`, `setupSocket()`, `parseURL()`.

Rules

- You may complete this assignment in C or C++.
- As with all projects, you are not permitted to work with anyone else – all of the coding and documentation must be your own.

Testing

Your program will be graded on how well it satisfies the requirements in handling a set of test URLs. You should test your program rigorously with various URLs before submitting.

Submission

You must name your source file `HTTPClient.c`, or `HTTPClient.cpp`, as appropriate. Make sure that you submit *all* files necessary to compile your program. *Make sure that your program compiles and runs correctly on a CS Unix machine.*

Submit a `PROG3.txt` file (*plain-text, not a Word document*) that lists your source files, describes how to compile and run your program, describes the operation of your HTTP client, and includes the output of a sample run using the following URLs:
`http://www.cs.odu.edu/~mweigle/research`
`http://www.cs.odu.edu/~clark/`
`http://www.cs.odu.edu/~mweigle/forbid/`
`http://www.cs.odu.edu/`
`http://www.cs.unc.edu/~mcweigle` (**Note:** This URL is not a typo.)

A hard-copy of PROG3.txt must be submitted at the start of class on the due date. This file is 15% of your grade on the project.

Example 1

```
% ./HTTPClient http://www.cs.odu.edu/~mweigle/foo.txt

URL: http://www.cs.odu.edu/~mweigle/foo.txt
Host: www.cs.odu.edu
Path: /~mweigle/foo.txt

> Connected to server www.cs.odu.edu on port 80
> Sending HTTP request:
GET /~mweigle/foo.txt HTTP/1.0
Host: www.cs.odu.edu
User-Agent: ODU-CS555

> Received HTTP response:
HTTP/1.1 200 OK
Date: Sat, 23 Sep 2006 15:45:14 GMT
Server: Apache/2.2.0
Last-Modified: Sat, 16 Sep 2006 17:38:59 GMT
ETag: "26ecb1e-5f-9da4d2c0"
Accept-Ranges: bytes
Content-Length: 95
Connection: close
Content-Type: text/plain

> Saved file as foo.txt
> Closing socket
```

Example 2

```
% ./HTTPClient http://www.cs.odu.edu/~mweigle

URL: http://www.cs.odu.edu/~mweigle
Host: www.cs.odu.edu
Path: /~mweigle

> Connected to server www.cs.odu.edu on port 80
> Sending HTTP request:
GET /~mweigle HTTP/1.0
User-Agent: ODU-CS555
Host: www.cs.odu.edu

> Received HTTP response:
HTTP/1.1 301 Moved Permanently
Date: Sat, 23 Sep 2006 15:48:36 GMT
Server: Apache/2.2.0
Location: http://www.cs.odu.edu/~mweigle/
Content-Length: 309
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

> Redirection to http://www.cs.odu.edu/~mweigle/
> Closing socket
-----------------------
URL: http://www.cs.odu.edu/~mweigle/
Host: www.cs.odu.edu
Path: /~mweigle/

> Connected to server www.cs.odu.edu on port 80
> Sending HTTP request:
GET /~mweigle/ HTTP/1.0
User-Agent: ODU-CS555
Host: www.cs.odu.edu

> Received HTTP response:
HTTP/1.1 200 OK
Date: Sat, 23 Sep 2006 15:45:55 GMT
Server: Apache/2.2.0
Last-Modified: Mon, 18 Sep 2006 18:56:53 GMT
ETag: "26ecb23-1490-efeb2740"
Accept-Ranges: bytes
Content-Length: 5264
Connection: close
Content-Type: text/html

> Saved file as index.html
> Closing socket
```