CSDS 325/425: Computer Networks
Project #2
Due: October 8, 11:59 PM

The second project of the semester involves writing a simple command line-based web client. The aim of this project is to write a program that exchanges information with another computer over a network and to start concretely thinking about and implementing protocols.

Overview

The usage of your program—which will be called *proj2*—is as follows:

```
./proj2 [-i] [-q] [-a] -u URL -w filename
```

Specifically:

- The "-u" option specifies the URL your web client will access. The "-u" option *must* be present on the command line.
- The "-i", "-q" and "-a" arguments are all optional. However, only *one* of these may be given. These options all trigger output that is described below. If more than one of these options is given the program must give an error and exit.
- The "-w" option specifies a filename where the downloaded contents of the supplied URL will be written. The "-w" option must be present on the command line.
- The command line arguments may appear in any order.
- Unknown command line arguments must trigger meaningful error messages and termination of the program.

Basic Operation

The main job of your project is to access the web object pointed to by the URL given on the command line (via "-u") and save that object to the local file given with the "-w" command line argument. Additionally, the "-i", "-q" and "-a" options will cause your project to write certain information about the transaction to standard output (i.e., the screen).

-u option

The "-u" option is used to supply the web server and page the client will access. The URL format your program will be expected to deal with is:

```
http://hostname[/path/to/file]
```

- Every URL must begin with "http://". While in the general case alternate protocols can be encoded in URLs—e.g., "https://" or "ftp://"—your program will only support HTTP. Further, the "http" is not case sensitive. I.e., "http", "Http", "HTTP", etc. must all be accepted. URLs that do not begin this way must trigger an error message and your project must exit.
- Following the "http://" will be a hostname. The hostname portion of the URL ends with the first "/". If a "/" does not appear after the hostname begins, the hostname continues to the end of the URL.
- The path in brackets is optional and may or may not be in the URLs your program must accept. Note: There will be no brackets in the actual options given to your program.
- Anything after the hostname is the path and filename to be sent verbatim to the web server (including the leading "/" character). If the URL does not contain a filename you must use the default filename of "/".

Hint: Strings in C and C++ are tedious. A set of standard string processing routines helps (a little!). Use the manual pages to look for routines such as *strncasecmp()*, *strstr()*, *strncmp()*, *strtok()*, etc.

-i option

The "-i" option will be used to print debugging information about the given command line parameters to standard output (i.e., the screen). When "-i" is given on the command line your program will output the following lines:

```
INFO: host: [hostname]
INFO: web_file: [url_filename]
INFO: output_file: [local_filename]
```

The format for these lines must follow these requirements:

- The "INFO:" must be at the beginning of the line.
- A single space follows "INFO:".
- The labels that appear after "INFO:<space>" must be exactly as they appear above (e.g., using all lower case letters).
- After the label, add a colon (":") and then a single space before printing the value.
- The "[hostname]" value comes from the URL given on the command line, as described in the "-u" discussion above.
- The "[url_filename]" value is the filename portion of the URL given on the command line, as described in the "-u" discussion above. If no filename is given on the command line, the default filename of "/" must be printed.
- The "[local_filename]" value is the name of the file on the local system where the web page at the given URL will be stored. This is the filename given with the "-w" option on the command line.
- The three lines must appear in the order given above.
- Do not print extra lines—including blank lines.
- Do not print any extra whitespace before, after or within the output lines.
- The "-i" output is to be printed regardless of whether there are errors fetching the web page. The "-i" output will not be printed if there are errors in the command line options given by the user.

The following are several illustrative examples with the "-i" option:

```
%%% ./proj2 -i -u http://www.icir.org -w testing.html
INFO: host: www.icir.org
INFO: web_file: /
INFO: output_file: testing.html

%%% ./proj2 -w mallman.html -i -u http://www.icir.org/mallman/
INFO: host: www.icir.org
INFO: web_file: /mallman/
INFO: output_file: mallman.html

%%% ./proj2 -u http://www.icir.org/mallman/index.html -w /tmp/mallman.html -i
INFO: host: www.icir.org
INFO: web_file: /mallman/index.html
INFO: output_file: /tmp/mallman.html
```

-q option

When the "-q" option is present on the command line, your program must print the HTTP request sent by your web client to the web server to standard output (the screen). The HTTP request you will transmit to the web server will look like this:

```
GET [url_filename] HTTP/1.0\r\n
Host: [hostname]\r\n
User-Agent: Case CSDS 325/425 WebClient 0.1\r\n
\r\n
```

Notes:

- The HTTP "GET" method must be used and expressed in all capital letters.
- The "[url_filename]" and "[hostname]" values are taken verbatim from the URL furnished on the command line and explained in the "-u" discussion above.
- "HTTP/1.0" must be the specified version of the HTTP protocol used (using all capitals, as shown).
- A single space separates "GET" and the [url_filename].
- A single space separates the [url_filename] and "HTTP/1.0".
- A single space follows "Host:".
- The "User-Agent" line above must be used verbatim. A single space appears between each word.
- All lines must end with a carriage return (\r) and a newline (\n).
- A line with only a carriage return (\r) and newline (\n) ends the HTTP request (per the HTTP specification).

The output corresponding to the above HTTP request when "-q" is specified will look like this:

```
REQ: GET [url_filename] HTTP/1.0
REQ: Host: [hostname]
REQ: User-Agent: Case CSDS 325/425 WebClient 0.1
```

In other words, the output should exactly mirror what was sent to the web server, with two exceptions:

- When printing to the screen, each HTTP request line must begin with "REQ:" followed by a single space.
- The blank line that terminates the HTTP request must be excluded from the "-q" output.

Examples:

```
%%% ./proj2 -q -u http://sigcomm.org -w sigcomm.html
REQ: GET / HTTP/1.0
REQ: Host: sigcomm.org
REQ: User-Agent: Case CSDS 325/425 WebClient 0.1

%%% ./proj2 -u http://www.icir.org/mallman/ -q -w mallman.html
REQ: GET /mallman/ HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: Case CSDS 325/425 WebClient 0.1
```

-a option

When the "-a" option is present on the command line, your program must print the HTTP response header received from the web server to standard output (the screen). Each line must be printed exactly as the line was received with the exception that "RSP:" followed by a single space must begin each line. Examples:

```
%%% ./proj2 -u http://www.icir.org/mallman/ -w mallman.html -a
RSP: HTTP/1.1 200 OK
RSP: Date: Mon, 16 Sep 2024 08:47:05 GMT
RSP: Server: Apache
RSP: Accept-Ranges: bytes
RSP: Content-Length: 6569
RSP: Connection: close
RSP: Content-Type: text/html; charset=UTF-8

%%% ./proj2 -a -u http://case.edu/ -w case.html
RSP: HTTP/1.1 301 Moved Permanently
RSP: Connection: close
RSP: Content-Length: 0
RSP: Server: Varnish
RSP: Retry-After: 0
RSP: Location: https://case.edu/
RSP: Accept-Ranges: bytes
RSP: Via: 1.1 varnish, 1.1 varnish
RSP: Strict-Transport-Security: max-age=300
RSP: Date: Mon, 16 Sep 2024 18:32:28 GMT
RSP: X-Served-By: cache-chi-kigq8000169-CHI, cache-chi-kigq8000169-CHI
RSP: X-Cache: HIT, MISS
RSP: X-Cache-Hits: 0, 0
RSP: X-Timer: S1726511548.446457,VS0,VE4
```

The output must not include the blank line that terminates the HTTP response header (and, hence, separates the header from the content).

-w option

The "-w" command line argument is used to tell the client where to save the contents of the downloaded URL. The web page content is everything received from the web server that follows the HTTP response header and the blank line that appears after the header. The file must contain exactly the data received from the web server. The client should only create the target file when the server returns an "OK" code of 200 in the HTTP header. When the client receives a non-200 response, it must print a meaningful error message. There are test URLs on the class web page, but you can test with arbitrary web servers, as well. The *wget* tool is available on the class servers and can help in your testing, as follows.

```
%%% wget -O wget_mallman.html http://www.icir.org/mallman/
[...]
%%% ./proj2 -w proj2_mallman.html -u http://www.icir.org/mallman/
%%% ls -l *.html
-rw-------  1 mallman  staff  6569 Sep 11 16:49 proj2_mallman.html
-rw-------  1 mallman  staff  6569 Sep 11 16:49 wget_mallman.html
%%% diff proj2_mallman.html wget_mallman.html
%%% sha1sum proj2_mallman.html wget_mallman.html
56ddd089280c8491d832ddd40de5bfd69ade5b85  proj2_mallman.html
56ddd089280c8491d832ddd40de5bfd69ade5b85  wget_mallman.html

%%% ./proj2 -a -w testing.html -u http://www.icir.org/mallman/doesnt-exist
RSP: HTTP/1.1 404 Not Found
RSP: Date: Sun, 11 Sep 2022 20:50:26 GMT
RSP: Server: Apache
RSP: Content-Length: 258
RSP: Content-Type: text/html; charset=iso-8859-1
ERROR: non-200 response code
%%% ls -l testing.html
ls: testing.html: No such file or directory
```

Final Bits

1. Submission specifications:
   (a) All project files must be submitted to Canvas in a gzip-ed tar file called "[CaseID]-proj2.tar.gz" (without the brackets).
   (b) Your submission must contain all code and a Makefile that by default produces an executable called "`proj2`" (i.e., when typing "make").
   (c) Do not include executables or object files in the tarball.
   (d) Do not include sample input or output files in your tarball.
   (e) Do not include multiple versions of your program in your submission.
   (f) Do not include directories in your tarball.
   (g) Do not use spaces in file names.
   (h) Every source file must contain a header comment that includes (*i*) your name, (*ii*) your Case network ID, (*iii*) the filename, (*iv*) the date created and (*v*) a brief description of the code contained in the file.
2. Your project must be written using sockets-based code and implement the required parts of HTTP. You may not leverage a third-party library for these tasks. Projects that rely on extra libraries will be returned ungraded.
3. Your submission may include a "notes.txt" file for any information you wish to convey during the grading process. We will review the contents of this file, but not of arbitrary files in your tarball (e.g., "readme.txt").
4. There will be sample reference output on the class web page by the end of the day on September 17. (Not all sample input will have (all) corresponding reference output.)
5. Print only what is described above. Extra debugging information must not be included. Adding an extra option (e.g., "-v" for verbose mode) to dump debugging information is always fine.
6. Do not make assumptions about the size of the URL contents. Your project should handle arbitrary size downloads.
7. If you encounter or envision a situation not well described in this assignment, please Do Something Reasonable in your code and include an explanation in the "notes.txt" file in your submission. If you'd like to ensure you're on the right track, please feel free to discuss these situations with me.
8. Hints / tips:
   (a) Needlessly reserving large amounts of memory in case it may be needed (e.g., for a large content download) is unreasonable.
   (b) Errors will be thrown at your project.
   (c) You may leverage the example sockets code we reviewed in class. This code is available from the project web page.
   (d) A simple C program and Makefile are available on the class web page. The program illustrates the use of *getopt()* to parse the command line arguments. The Makefile can be easily adapted for this project.
9. *WHEN STUCK, ASK QUESTIONS!*

CSDS 425: Computer Networks
Project #2 Extensions
Due: October 8, 11:59 PM

Graduate students will be responsible for writing the basic web client described above, as well as the following extension. Undergraduates can implement the extension for extra credit.

Following Redirects

When the "-r" option is given on the command line the client will follow redirections from the web server. For instance, consider this sample invocation of the web client:

```
%%% ./proj2 -a -w testing.html -u http://www.icir.org/mark/
RSP: HTTP/1.1 301 Moved Permanently
RSP: Date: Mon, 16 Sep 2024 08:51:21 GMT
RSP: Server: Apache
RSP: Location: http://www.icir.org/mallman/
RSP: Content-Length: 298
RSP: Connection: close
RSP: Content-Type: text/html; charset=iso-8859-1
ERROR: non-200 response code
```

In this case, the desired web page is not at the URL given on the command line ("http://www.icir.org/mark/"). Rather, the web server uses a response with a code of 301 to redirect the client to a different URL—which is given in the "Location:" line of the response header. When the "-r" option is given, the web client will download the URL given in the redirection message. Redirection can happen more than once. E.g., "www.foo.com" could redirect to "www.bar.com" which could in turn redirect to "bar.com". When redirects happen and the "-q" or "-a" options are given the client must print every request and/or response header encountered in the order encountered. The output file given with "-w" will contain the contents of the ultimate (last) response. An example:

```
%%% ./proj2 -a -u http://www.icir.org/mark/ -w mark.html -r
RSP: HTTP/1.1 301 Moved Permanently
RSP: Date: Mon, 16 Sep 2024 08:51:21 GMT
RSP: Server: Apache
RSP: Location: http://www.icir.org/mallman/
RSP: Content-Length: 298
RSP: Connection: close
RSP: Content-Type: text/html; charset=iso-8859-1
RSP: HTTP/1.1 200 OK
RSP: Date: Mon, 16 Sep 2024 08:52:16 GMT
RSP: Server: Apache
RSP: Accept-Ranges: bytes
RSP: Content-Length: 6569
RSP: Connection: close
RSP: Content-Type: text/html; charset=UTF-8
%%% wget -O wget_mark.html http://www.icir.org/mark/
[...]
%%% ls -l mark.html wget_mark.html
-rw-------  1 mallman  staff  6569 Sep 11 17:09 mark.html
-rw-------  1 mallman  staff  6569 Sep 11 17:09 wget_mark.html
%%% sha1sum mark.html wget_mark.html
56ddd089280c8491d832ddd40de5bfd69ade5b85  mark.html
56ddd089280c8491d832ddd40de5bfd69ade5b85  wget_mark.html
```