# EECS 325/425: Computer Networks
# Project #2
Due: October 1, 11:59 PM

The second project of the semester involves writing a simple command line-based web client. The aim of this project is (*i*) to get your feet wet with writing C/C++, (*ii*) to write a program that exchanges information with another computer over a network and (*iii*) to start concretely thinking about protocols.

## Overview

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

```
./proj2 -u URL [-d] [-r] [-R] -o filename
```

Specifically:

- The "-u" option specifies the URL your web client will access. The "-u" option must be present on the command line.
- The "-d", "-r" and "-R" arguments are all optional and any number and combination of these could be given. These options all trigger output that is described below.
- The "-o" option specifies a filename where the downloaded contents of the supplied URL will be written. The "-o" option must be present on the command line.
- The command line arguments may appear in any order.
- Unknown command line arguments must trigger errors.

## -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[:port][/path/to/file]
```

- The items in brackets are optional and may or may not be in the URLs your program must accept.
- 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. should all be accepted.
- Following the "http://" will be a hostname. The hostname portion of the URL ends with the first "/" or ":" encountered after the beginning of the hostname. If neither of these appear after the hostname begins, the hostname continues to the end of the URL.
- If a colon (":") ends the hostname, a port number follows the colon and continues until a "/" or the end of the URL string is encountered. The port is optional and if not given the default web port of "80" will be assumed.
- Anything after the hostname and optional port is the filename and path 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.

## -d option

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

```
DET: hostname = [hostname]
DET: port = [port]
DET: web_filename = [url_filename]
DET: output_filename = [local_filename]
```

The format for these lines must follow these requirements:

- The "DET:" must be at the beginning of the line.
- A single space follows "DET:".
- The labels that appear after "DET:<space>" above must be exactly as they appear above (e.g., using all lower case letters).
- After the label a single space, an equals sign and another single space separates the label from the value.
- The "[hostname]" value comes from the URL given on the command line, as described in the "-u" discussion above.
- The "[port]" value is the port specified in the URL given on the command line, as described in the "-u" discussion above. Specifying the port in the URL is optional. If no port is specified in the given URL then the value printed here should be the default port ("80").
- 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 "/" should 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 "-o" option on the command line.
- The four lines must appear in the order given above.
- Do not print extra lines—including blank lines.
- The "-d" output is to be printed regardless of whether there are errors fetching the web page. The "-d" 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 "-d" option:

```
./proj2 -d -u http://www.icir.org -o testing.html
DET: hostname = www.icir.org
DET: port = 80
DET: web_filename = /
DET: output_filename = testing.html

./proj2 -o mallman.html -d -u http://www.icir.org:8080/mallman/
DET: hostname = www.icir.org
DET: port = 8080
DET: web_filename = /mallman/
DET: output_filename = mallman.html

./proj2 -u http://www.icir.org/mallman/index.html -o /tmp/mallman.html -d
DET: hostname = www.icir.org
DET: port = 80
DET: web_filename = /mallman/index.html
DET: output_filename = /tmp/mallman.html
```

## -r option

When the "-r" option is present on the command line, your program must print the HTTP request sent 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: CWRU EECS 325 Client 1.0\r\n
\r\n
```

Notes:

- The HTTP "GET" method must be used.
- The "[url_filename]" and "[hostname]" values are taken 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.
- 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 corresponding output when "-r" is specified will look like this:

```
REQ: GET /mallman HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: CWRU EECS 325 Client 1.0
```

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 not be included in the "-r" output.

Examples:

```
./proj2 -r -u http://case.edu -o case.html
REQ: GET / HTTP/1.0
REQ: Host: case.edu
REQ: User-Agent: CWRU EECS 325 Client 1.0

./proj2 -u http://www.icir.org/mallman/ -r -o mallman.html
REQ: GET /mallman/ HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: CWRU EECS 325 Client 1.0
```

## -R option

When the "-R" 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/ -o mallman.html -R
RSP: HTTP/1.1 200 OK
RSP: Date: Thu, 30 Aug 2018 16:00:42 GMT
RSP: Server: Apache/2.4.34 (Fedora)
RSP: Accept-Ranges: bytes
RSP: Content-Length: 6470
RSP: Connection: close
RSP: Content-Type: text/html; charset=UTF-8

./proj2 -R -u http://case.edu -o case.html
RSP: HTTP/1.1 200 OK
RSP: Date: Thu, 30 Aug 2018 16:01:28 GMT
RSP: Server: Apache
RSP: Set-Cookie: www-case-edu=129.22.12.4.1535644888279283; path=/; expires=Fri, 30-Aug-19 16:01:28 G
RSP: Accept-Ranges: bytes
RSP: Vary: User-Agent
RSP: Connection: close
RSP: Content-Type: text/html
```

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

## Printing Order

Note: any number, order and combination of "-d", "-r" and "-R" is allowed. This includes no printing options. When multiple options are given, the output will be printed in a standard order. First the "-d" output will be given, if needed (the "DET:" lines). Second, the "-r" output will be given, if needed (the "REQ:" lines). Finally, the "-R" output will be given, if needed (the "RSP:" lines). This order holds regardless of the order the options are provided on the command line. Examples:

```
./proj2 -u http://www.icir.org/mallman/ -o mallman.html -r -R -d
DET: hostname = www.icir.org
DET: port = 80
DET: web_filename = /mallman/
DET: output_filename = mallman.html
REQ: GET /mallman/ HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: CWRU EECS 325 Client 1.0
RSP: HTTP/1.1 200 OK
RSP: Date: Thu, 30 Aug 2018 16:07:22 GMT
RSP: Server: Apache/2.4.34 (Fedora)
RSP: Accept-Ranges: bytes
RSP: Content-Length: 6470
RSP: Connection: close
RSP: Content-Type: text/html; charset=UTF-8
```

```
./proj2 -r -u http://www.icir.org/mallman/ -o mallman.html -d
DET: hostname = www.icir.org
DET: port = 80
DET: web_filename = /mallman/
DET: output_filename = mallman.html
REQ: GET /mallman/ HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: CWRU EECS 325 Client 1.0

./proj2 -R -r -u http://www.icir.org/mallman/ -o mallman.html
REQ: GET /mallman/ HTTP/1.0
REQ: Host: www.icir.org
REQ: User-Agent: CWRU EECS 325 Client 1.0
RSP: HTTP/1.1 200 OK
RSP: Date: Thu, 30 Aug 2018 16:09:28 GMT
RSP: Server: Apache/2.4.34 (Fedora)
RSP: Accept-Ranges: bytes
RSP: Content-Length: 6470
RSP: Connection: close
RSP: Content-Type: text/html; charset=UTF-8

./proj2 -u http://www.icir.org/mallman/ -o mallman.html
```

### -o option

The "-o" 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 an 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 -o proj2_mallman.html -u http://www.icir.org/mallman/
ls -l *.html
-rw-------  1 mallman  staff  6470 Aug 30 13:29 proj2_mallman.html
-rw-------  1 mallman  staff  6470 Aug 30 13:29 wget_mallman.html
diff proj2_mallman.html wget_mallman.html
sha1sum proj2_mallman.html wget_mallman.html
3b0f29f80161201916345ec42fafcf4607899a4c  my_proj_mallman.html
3b0f29f80161201916345ec42fafcf4607899a4c  wget_mallman.html

./proj2 -R -o testing.html -u http://www.icir.org/mallman/doesnt-exist
RSP: HTTP/1.1 404 Not Found
RSP: Date: Thu, 30 Aug 2018 17:36:22 GMT
RSP: Server: Apache/2.4.34 (Fedora)
RSP: Content-Length: 296
RSP: Connection: close
RSP: Content-Type: text/html; charset=iso-8859-1
web server sent an unsupported return code: 404
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".
   (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 input files and reference output on the class web page by the end of the day on September 10. (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 class 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!*

# EECS 425: Computer Networks
# Project #2 Extensions
Due: October 1, 11:59 PM

Graduate students will be responsible for writing the basic web client described above, as well as two extensions described below.

## Following Redirects

When the "-f" 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 -R -u http://www.case.edu/ -o case.html
RSP: HTTP/1.1 301 Moved Permanently
RSP: Date: Thu, 30 Aug 2018 18:19:37 GMT
RSP: Server: Apache
RSP: Location: http://case.edu/
RSP: Content-Length: 224
RSP: Connection: close
RSP: Content-Type: text/html; charset=iso-8859-1
web server sent an unsupported return code: 301
```

In this case, the desired web page is not at the URL give on the command line ("http://www.case.edu/"). 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 "-f" 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 "-r" or "-R" options are given the client should print every request and response header encountered in the order it is encountered. The output file given with "-o" will contain the contents of the ultimate (last) response. An example:

```
./proj2 -r -R -u http://www.case.edu/ -o case.html -f
REQ: GET / HTTP/1.0
REQ: Host: www.case.edu
REQ: User-Agent: CWRU EECS 325 Client 1.0
RSP: HTTP/1.1 301 Moved Permanently
RSP: Date: Thu, 30 Aug 2018 18:19:37 GMT
RSP: Server: Apache
RSP: Location: http://case.edu/
RSP: Content-Length: 224
RSP: Connection: close
RSP: Content-Type: text/html; charset=iso-8859-1
REQ: GET / HTTP/1.0
REQ: Host: case.edu
REQ: User-Agent: CWRU EECS 325 Client 1.0
RSP: HTTP/1.1 200 OK
RSP: Date: Thu, 30 Aug 2018 18:26:23 GMT
RSP: Server: Apache
RSP: Set-Cookie: www-case-edu=129.22.108.4.1535653583055626; path=/; expires=Fri, 30-Aug-19 18:26:23 (
RSP: Accept-Ranges: bytes
RSP: Vary: User-Agent
RSP: Connection: close
RSP: Content-Type: text/html
```

```
wget -O wget_case.html http://case.edu/
[...]
ls -l case.html wget_case.html
-rw-------  1 mallman  staff  35938 Aug 30 14:26 case.html
-rw-------  1 mallman  staff  35938 Aug 30 14:27 wget_case.html
sha1sum case.html wget_case.html
c4b45eee9974352df42a9e89e591e1ec68ac3604  case.html
c4b45eee9974352df42a9e89e591e1ec68ac3604  wget_case.html
```

## Chunked Encoding

One of the mechanisms that was added in HTTP version 1.1, but not included in HTTP version 1.0 is "chunked encoding". When the "-c" option is given on the command line the web client will correctly download and store HTTP responses that arrive using this encoding strategy. When using "-c", the requests will advertise the client as "HTTP/1.1" instead of "HTTP/1.0". (Note: the client will still use "HTTP/1.0" in the absence of the "-c" option.) Further, the URL's content will not come as a stream of bytes, but as a series of chunks, which include non-content information about their size. You will need to research the details of chunked encoding before you complete this portion of the project. Example:

```
wget -O wget_case.html http://case.edu/
[...]
./proj2 -u http://case.edu/ -o case-1.0.html -r
REQ: GET / HTTP/1.0
REQ: Host: case.edu
REQ: User-Agent: CWRU EECS 325 Client 1.0
./proj2 -u http://case.edu/ -o case-1.1.html -c -r
REQ: GET / HTTP/1.1
REQ: Host: case.edu
REQ: User-Agent: CWRU EECS 325 Client 1.0
sha1sum *html
c4b45eee9974352df42a9e89e591e1ec68ac3604  case-1.0.html
c4b45eee9974352df42a9e89e591e1ec68ac3604  case-1.1.html
c4b45eee9974352df42a9e89e591e1ec68ac3604  wget_case.html
```