Project 2: HTTP Proxy

Project Description

HTTP GET requests are used to request data from the specified source. For example, if we want to access a webpage at `http://www.foo.com/bar.html`, our browser will send something similar to the following HTTP GET request.

```
GET /bar.html HTTP/1.1
Host: www.foo.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:28.0) Gecko/20100101 Firefox/28.0
Accept:text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

In the above example, our browser is the client, and `Host: www.foo.com` is the server host name. The client is requesting server resource located at `/bar.html`. Usually, HTTP servers run on port 80. So, by default, the server port number is 80. If, for example, the HTTP server runs on port 8080, then the URL would be `http://www.foo.com:8080/bar.html`, and the `Host` field in the HTTP request will also contain the port number: `www.foo.com:8080`.

The HTTP server responds with HTTP response. A valid HTTP response includes: (1) the status line, (2) the response header, and (3) the entity body. For example, the text below shows an example HTTP response. Notice that this response header includes a field: `Content-Length`, which tells the client how much data to expect.

```
HTTP/1.1 200 OK
Date: Thu, 02 Apr 2015 01:51:49 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Tue, 10 Feb 2015 17:56:15 GMT
ETag: "56638402b-a85-50ebf9a5ecdc0"
Accept-Ranges: bytes
Content-Length: 2693
Content-Type: text/html; charset=ISO-8859-1
```

```
... ... <content of the bar.html>
```

An **HTTP proxy** is an application that resides between the HTTP client and server. The HTTP proxy relays the client's HTTP requests to the HTTP server, and forwards the server's response back to the client. Proxies are often used when the client cannot directly connect with the HTTP server. If a client is configured to use an HTTP proxy, the HTTP GET request it sends out will not only include path to the resource at the server, but also the server host name. For example, the request would look like:

```
GET http://www.foo.com/bar.html HTTP/1.1
Host: www.foo.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:28.0) Gecko/20100101 Firefox/28.0
Accept:text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

To request the resource for the client, the HTTP proxy starts a new TCP connection with the server host `www.foo.com` at port number `80` and ask for file `bar.html`. When the HTTP proxy relays this request from the client, it will have to remove the host name, `http://www.foo.com` from the request line of the GET request, so that the request line specifies only the local resource path at the destination server. When the HTTP response arrives at the proxy, the proxy will forward it to the client. If the proxy is correctly implemented, a client web-browser should be able to display the requested resource.

You need to write the proxy program. Your HTTP proxy should support a subset of the HTTP standard. It only needs to forward HTTP GET requests and should only support basic portions of the standard. That is, you do not need to support persistent connections, request pipelining or other, advanced portions of the standard. Your proxy must only correctly handle HTTP requests where a single request/response pair is sent over a single TCP connection. Specifically, your proxy should operate as follows:

- It should create a TCP server socket to listen for incoming TCP connections on an unused port, and output the host name and port number the proxy is running on.

- When a new connection request comes in, the proxy accepts the connection, establishing a TCP connection with the client.

- The proxy reads HTTP request sent from the client and prepares an HTTP request to send to the HTTP server.

- The proxy starts a new connection with the HTTP server, and sends its prepared HTTP request to the server.

- The proxy reads the HTTP response and forwards the HTTP response (the status line, the response header, and the entity body).

- The proxy closes the TCP connection with the server.

- The proxy sends the server's response back to the client via its TCP connection with the client. This TCP connection with the client should have remained open during its communication with the server.

- The proxy closes the connection socket to the client.

FAQ

*** What is in the folder?**

utils.c	functions used by the proxy server
proxy.c	contains source code of the proxy
utils.h	header file of utils.c
makefile	make file to build proxy
proxy	proxy executable

*** What do we have to do?**

You have to supply the missing code in the files 'utils.c' and 'proxy.c'. The location of missing code is

marked with "TODO:". Your task is to fill in the missing code.

*** How do I test my programs?**

You first download a resource located at a URL using the wget command without the proxy. This is the correct resource. Then you download the same resource at the same URL with your proxy. Use the `diff` command to check if the two downloaded resources matches.

Suppose your HTTP proxy is started on `eecs325-VirtualBox` port `47590`. You can run the following Bash shell command

```
$> bash
$> export http_proxy=http://127.0.0.1:47590 && wget
http://www.foo.com/bar.html
```

To download without the HTTP proxy, use the following command:

```
$> export http_proxy="" && wget
http://www.foo.com/bar.html
```

Note:

- You MUST replace http://www.foo.com/bar.html with a valid URL.
- Since this is only an HTTP proxy, not an HTTPS proxy, your URL MUST use HTTP, not HTTPS.
- You may also run your HTTP proxy on your own machine and use Wireshark for debugging.
- Your web browser also has a cache. A second request for the same URL may be directly served by your browser cache without going through the proxy. Therefore, you are recommended to use wget for debugging and testing.

REMINDERS

- Submit your code to Canvas in a tarball file by running command in terminal:
  `zip -r [you-case-id]-proj-2.zip project-2`
- If your code does not **compile** or could not **run**, you will not get credits.
- Document your code (by inserting comments in your code)
- DUE: 11:59 pm, Tuesday, May 7th

Appendix

Below is a list of URLs you can use for testing your proxy implementation.

```
http://engineering.case.edu/eecs
http://engineering.case.edu/eecs/about
http://engineering.case.edu/eecs/node/190
http://engineering.case.edu
http://engineering.case.edu/eecs/node/306
http://engineering.case.edu/eecs/research
http://engineering.case.edu/eecs/sites/engineering.case
.edu.eecs/files/images/header-research.jpg
https://engineering.case.edu/eecs/seminars
```