

EX2 – HTTP client

Goals:

The purpose of this project is two-fold:

1. give students hand-on experience with socket programming
2. help students better understand application-level protocols by implementing a well-known protocol, HTTP.

In this programming assignment you will write HTTP client. Students are not required to implement the full HTTP specification, but only a very limited subset of it.

You will implement the following:

An HTTP client that constructs an HTTP request based on user's command line input, sends the request to a Web server, receives the reply from the server, and displays the reply message on screen. You should support only IPv4 connections.

Background:

What is HTTP? HTTP stands for Hyper Text Transfer Protocol and is used for communication among web clients and servers. HTTP has a simple stateless client/server paradigm. The web client initiates a conversation by opening a connection to the server. Once a connection is set up, the client sends an HTTP request to server. Upon receiving the HTTP request from the client, the server sends an HTTP response back to the client. An HTTP request consists of two parts: a header and a body. In this project, the basic HTTP request from a client doesn't contain a body. The first line of any request header should be:

Method Request-URI Version. An example HTTP1.0 request is:

```
GET /index.html HTTP/1.0
```

```
Host: www.jce.ac.il
```

The request header and body are separated by two sets of carriage return and line feed (`\r\n`). Since we do not need the body, the end of a header marks the end of a request. Using a C char string, the example request above should be: `"GET /index.html HTTP/1.0\r\nHost: www.jce.ac.il\r\n\r\n"`.

What is a URL?

Uniform Resource Locators (URLs) are formatted strings that identify resources in the web: documents, images, downloadable files, electronic mailboxes, etc. It generally has the format:

Protocol://Host[:port]/Filepath.

In this project, when a port is not specified, the default HTTP port number of 80 is used. For example, a file called "foo.html" on HTTP server "www.yoyo.com" in directory "/pub/files" corresponds to this URL: `http://www.yoyo.com/pub/files/foo.html`. The default HTTP network port is 80; if an HTTP server resides on a different network port (say, port 1234), then the URL becomes: `http://www.yoyo.com:1234/pub/files/foo.html`.

Program Description and What You Need to Do:

You will write the program `client.c`.

The Client

The client takes two options "-p" and "-r" and a required argument <URL>.

Command line usage: `client [-p <text>] [-r n <pr1=value1 pr2=value2 ...>] <URL>`. The flags and the url can come at any order, the only limitation is that the parameters should come right after the flag -r and the text should come right after the flag -p.

<URL> specifies the URL of the object that the client is requesting from server. The URL format is `http://hostname[:port]/filepath`.

The default request is GET request. Option "-p" along with its argument <text> specifies that this is a POST request. **You should use POST method in your HTTP request instead of GET**, you should also add a Content-length header and request body with the text when "-p" is specified in the command line. The text after the -p flag will not contain space or special characters.

Option "-r" along with its argument **<n pr1=value1 pr2=value2 ...>** specify that the request has **n parameters**, each of the parameters format is 'name'='value' separated by space. The parameters should appear after the path, for example, if there are 2 parameters, the format will be: `/path?pr1=value1&pr2=value2`

You can assume that the url has to start with <http://>

If the url has no path, the path in the request is "/".

In `client.c`, you need to:

1. Parse the <URL> given in the command line.
2. Construct an HTTP request based on the options specified in the command line
3. Connect to the server
4. Send the HTTP request to the server
5. Receive an HTTP response
6. Display the response on the screen.

After constructing the http request and before you send it to the server, print it to stdout in the following format:

```
printf("HTTP request =\n%s\nLEN = %d\n", request, strlen(request));
```

where request holds your constructed request.

After getting the response from the server and printing it to stdout, print the following message:

```
printf("\n Total received response bytes: %d\n",size);
```

where size is the number of characters in the response.

Your client should close connection after getting the file. **You should use HTTP/1.0.**

Error handling:

1. In any case of a failure in one of the system calls, use `perror(<sys_call>)` and exit the program (for errors on `gethostbyname` call `herror` instead).
2. In any case of wrong command usage, print "Usage: client [-p <text>] [-r **n** <pr1=value1 pr2=value2 ...>] <URL>"

Enter new line after each error message.

Examples:

1. `./client http://www.ptsv2.com/t/ex2`
Request:
GET </t/ex2> HTTP/1.0
Host: www.ptsv2.com
2. `./client -r 3 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2`
Request:
GET </t/ex2?addr=jecrusalem&tel=02-6655443&age=23> HTTP/1.0
Host: www.ptsv2.com
3. `./client -p blabla http://www.ptsv2.com/t/ex2`
Request:
POST </t/ex2> HTTP/1.0
Host: www.ptsv2.com
Content-length:6

blala
4. `./client -p blabla -r 3 addr=jerusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2`
Request:
POST </t/ex2?addr=jerusalem&tel=02-6655443&age=23> HTTP/1.0

Host: www.ptsv2.com

Content-length:6

blala

5. `./client -r 3 addr=jerusalem tel=02-6655443 age=23 -p blabla http://www.ptsv2.com/t/ex2`
The request is same as before.

6. **Command line errors example:**

- a. `./client -p -r 3 addr=jecrusalem tel=02-6655443 age=23 blabla http://www.ptsv2.com/t/ex2`
-r will be considered as the text of -p flag and the number 3 will yield the error.
- b. `./client -p blabla -r addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2`
Has to be number after -r
- c. `./client -p blabla -r 3 addr=jecrusalem tel=02-6655443 age23 http://www.ptsv2.com/t/ex2`
Third parameter age23 is not of the right format: name=value
- d. `./client -p blabla -r 3 addr=jecrusalem tel=02-6655443 http://www.ptsv2.com/t/ex2`
Either the url will be considered as the 3rd parameter and it is not of the right format or too few parameters, depends on the order you check the command line.
- e. `./client -p blabla -r 2 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2`
Too many parameters
- f. `./client http://blala`
This is not usage error, you will fail when trying to get the IP address for that host.

Useful function:

Strchr, Strstr, strcat

Compile the client:

`gcc -Wall client.c -o client`

client is the executable file.

What to submit:

You should submit a tar file with client.c and README. Find README instructions in the course web-site.

Test the client:

You can use the client to connect to any HTTP server. You should try different URLs and options to make sure that the client works correctly.

In order to test parameters and post request, you can use the server at www.ptsv2.com, first go to that server, type your unique string in the 'lookup' box and press lookup. You'll be redirected to a page www.ptsv2.com/t/your-string. This page is now your server and it shows the requests it gets. In order to test your requests, send them to www.ptsv2.com/t/your-string/post and you'll be able to see the requests details in the www.ptsv2.com/t/your-string page.