EX2 – HTTP client

Goals:

The purpose of this project is two-fold:

1. give students hands-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 an 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 the user's command line input, sends the request to a Web server, receives the reply from the server, and displays the reply message on the 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 the 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 HTTP1.1 request is:

GET /index.html HTTP/1.1

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/1.1\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 an options "-r" and a required argument <URL>.

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

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

The default request is GET request.

Option "-r" along with its argument <n pr1=value1 pr2=value2 …> specify that the request has n parameters, and 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 "/".

You should handle 3XX responses with Location header, but only those where the URL is with HTTP and not HTTPS. If you see such response, you should create another request with the URL in the location.

In client.c, you need to:

1. Parse the <URL> given in the command line. If there is a port, you should verify that it is a positive number under 2^16.
   The parsing is the easy part, don't spend time on it.
   A suggested logic:
   a. If you see '-', then look for r, if you don't see it, print the Usage message and exit.
      If you see -r, look for a number n, if there is no number, print the Usage message and exit. After the number n, you should look for str=str, there can be spaces (in this case, spaces are not allowed within one parameter, only between parameters), after reading n arguments, go back to stage a. There can be 0 arguments when n=0.

   b. If there is no '-', then this is your URL.
      i. Check that it begins with http://, otherwise, print the Usage message and exit.
      ii. Read the domain name until you see either ':', '/' or end-of-string.

1. If you see :, look for a positive number, which is less than 2^16, if not print the Usage message and exit.
2. If you see '/', look for a path (can be also without a path).

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.
7. If the response is 3XX with a Location header that contains URL, and the protocol in the URL is HTTP go back to 1.b.ii. Otherwise, do nothing.

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/1.1

In case you need to send another request, you should open new TCP connection.

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 [-r n < pr1=value1 pr2=value2 …>] <URL>"

Enter a new line after each error message.

Examples:

1. ./client http://httpbin.org/get

2. Request:
   GET /get HTTP/1.1
   Host: www.httpbin.com

3. ./client -r 3 addr=jecrusalem tel=02-6655443 age=23 http://httpbin.org/anything
   Request:
   GET /anything?addr=jerusalem&tel=02-6655443&age=23 HTTP/1.1
   Host: www.httpbin.com

4. Command line errors example:
   a. ./client -r addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2
      Has to be a number after -r
   b. ./client -r 3 addr=jerusalem tel=02-6655443 age23 http://www.ptsv2.com/t/ex2
      The third parameter age23 is not of the right format: name=value
   c. ./client -r 3 addr=jerusalem 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 has too few parameters.
   d. ./client -r 2 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2
      Too many parameters
   e. ./client http://blala
      This is not a 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 –o client client.c

client is the executable file.

What to submit:

You should submit a tar file with client.c and README.

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.

You can play with http://httpbin.org/ , it has many options to play with your code.