Multi-threaded HTTP Server

Task

Expected to write a small web server in c that supports a subset of the HTTP 1.0 specifications. The server;

- should be able to handle simultaneous requests
- implement the HTTP methods GET and HEAD
- handle and respond to invalid requests.

You should be able to demonstrate that your Web server is capable of delivering your home page to a Web browser. You should implement version 1.0 of HTTP, as defined in RFC 1945, where separate HTTP requests are sent for each component of the Web page. The server will be able to handle multiple simultaneous service requests in parallel. This means that the Web server is multi-threaded. In the main thread, the server listens to a fixed port. When it receives a TCP connection request, it sets up a TCP connection through another port and services the request in a separate thread.

To simplify this programming task, we will develop the code in two stages. In the first stage, you can write a multi-threaded server that simply displays the contents of the HTTP request message that it receives. After this program is running properly, you can add the code required to generate an appropriate response.

As you are developing the code, you can test your server from a Web browser, such as Chrome, Safari, Firefox web browser as http client application. But remember that you are not serving through the standard port 80, so you need to specify the port number within the URL that you give to your browser. For example, if your machine's name is **testhost.mydomain.com** your server is listening to port **6789**, and you want to retrieve the file **index.html**, then you would specify the following URL within the browser:

http:// testhost.mydomain.com:6789/index.html

If you omit ":6789", the browser will assume port 80 which most likely will not have a server listening on it.

When the server encounters an error, it sends a response message with the appropriate HTML source so that the error information is displayed in the browser window.

The Figure-1 shows the basic communications between server and client.

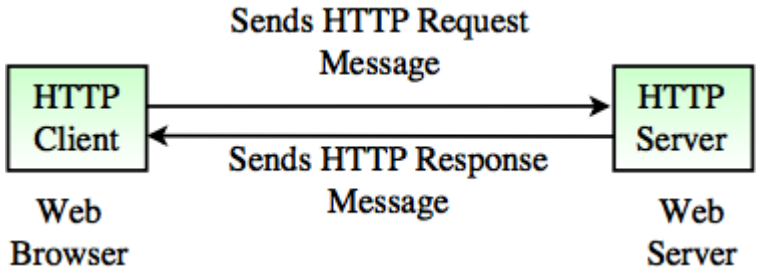
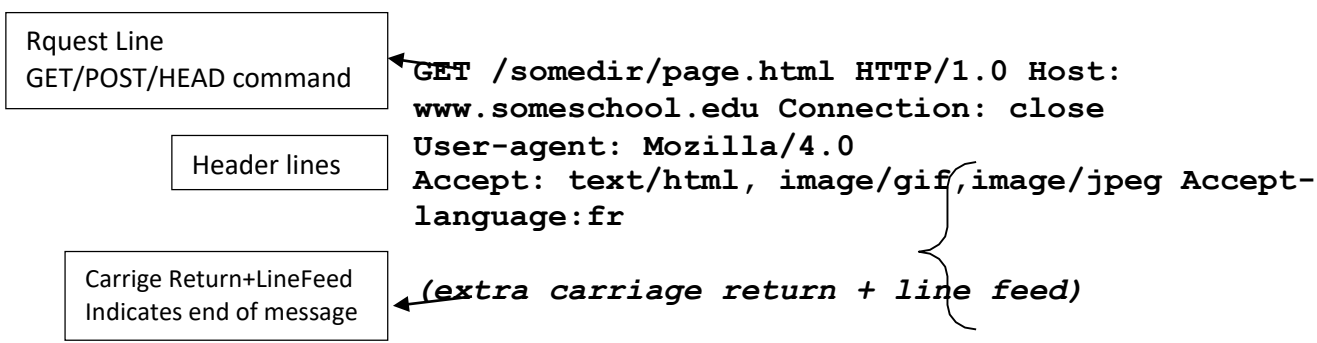


Figure-1: Http server and client communication

A typical Http message sent from server to client or vice versa, is as follows:



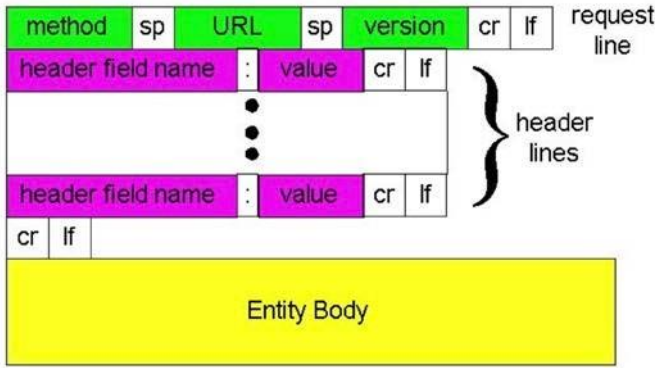


Figure-2: General format of Http request message.

```
GET puppies.html HTTP/1.1
Host: www.puppyshelter.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

puppyId=12345&name=Fido+Simpson
```

Figure-3: An example of Http request message.