Web Server

Mandatory assignment Part 2

The Mandatory assignment in the course, means that it needs to be both handed-in and approved in order for the student to be able to do the exam, or he/she will miss an exam attempt otherwise.

Objective

The main objective of this assignment is to learn the basics of socket programming for TCP connections in Python (creating a socket, binding it to a specific address and port, receive an HTTP packet..), by developing a web server to which web browsers or other kind of clients can send requests to.
Why a web server? They are one of the most targeted public faces of an organization. You will learn in this education how they can be attacked and how to make them more secure, but before that, it's important to understand how they work.

Description of the mandatory task

Develop a web server that handles one HTTP request at a time (no multi-threading allowed).
The web server should:

1. Accept and parse the HTTP request.
2. Get the requested file from the server's file system. If no specific file is requested, the file `index.html` should be used.
3. Create a valid HTTP response message consisting of the requested file preceded by the corresponding header lines.
4. Send the response to the client. If the file requested by the client is not present in the server's file system, then the server should send an HTTP "404 Not Found" message back to the client (remember also in this case, to precede it by the corresponding header lines). If you wish, you can make the server to send in this case an HTML file prepared to be used to inform the user about the file not existing in the system.
   a. You should implement the response codes 200, 400 and 404.
5. Keep a log file of all HTTP requests received. Try to do the logging in apache format. An example of that can be found in the end of this document.
6. You are NOT allowed to use the `http.server or similar` library. Instead you should build it using what you learned about TCP sockets.

How to run and test the server?

Create two HTML files, one of them being "index.html", and the other one, for instance, being "test.html". Place these two files in the same directory that the server is in, and then run the server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26). Try the following if possible: from another computer, open a browser and enter the corresponding URL. For example:

```
http://128.238.251.26:6789/test.html
```

Note the use of the port number after the colon. This number needs to be replaced with whatever port your server is listening from.

What you should see

Following the same example as above, if you enter the URL `http://128.238.251.26:6789/test.html`, the content of the file "test.html" should be displayed in the browser.
If you enter the URL `http://128.238.251.26:6789` , you should then see the content of the file "index.html".
And finally, if you enter, for instance, the URL `http://128.238.251.26:6789/blabla.html` where "blabla.html" is the name of a file that doesn't exist in the server (it could be anything: an HTML file, an image..), then you should get a "404 Not Found".

Note that if you omit the port part in the URL, that is you type `http://128.238.251.26/`, then any standard browser will always assume port 80 by default and you will be able to connect to your web server only if the server is indeed listening at port 80.

Some hints or things to keep in mind

Have a look at the format of HTTP response (see next page) and remember to add the special characters `\r\n` wherever they required by the HTTP protocol, including after the body of the message.
Remember to handle the corresponding Python exception that is raised when the server tries to open and read a file that doesn't exist in the file system.

Mandatory task requirements

This project can be done in **groups of 3-4 people**. **YOU SHOULD STILL HAND IN INDIVIDUALLY.**
The programming language should be **Python**.
There is no requirement for using a specific development environment, but you are encouraged to use Kali Linux as OS.
You will probably find many examples of web servers online. Still, try to do your own, it's not difficult, you'll learn more and it's way funnier!

About the hand in

The hand-in should be done through Fronter. A special hand-in folder is already available for that purpose.

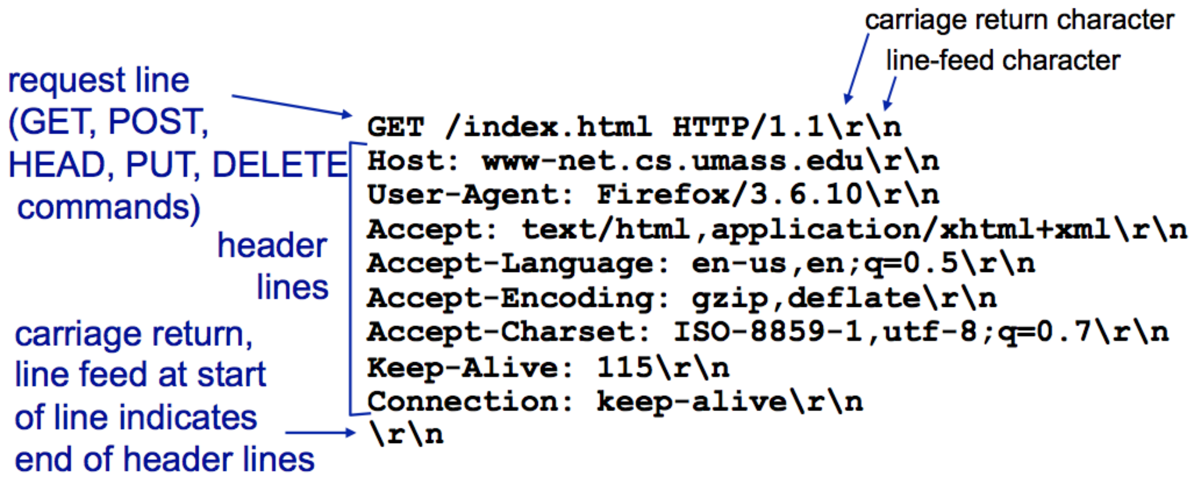
You should hand in a ZIP file containing ONLY the following material:

1. The server source code in Python (file with extension `.py`)
2. The screenshots of your client browser, showing that it actually receives the files it requests to the web server.
3. The log file of HTTP requests received at the server side.

If you have any doubts, contact the course teachers.

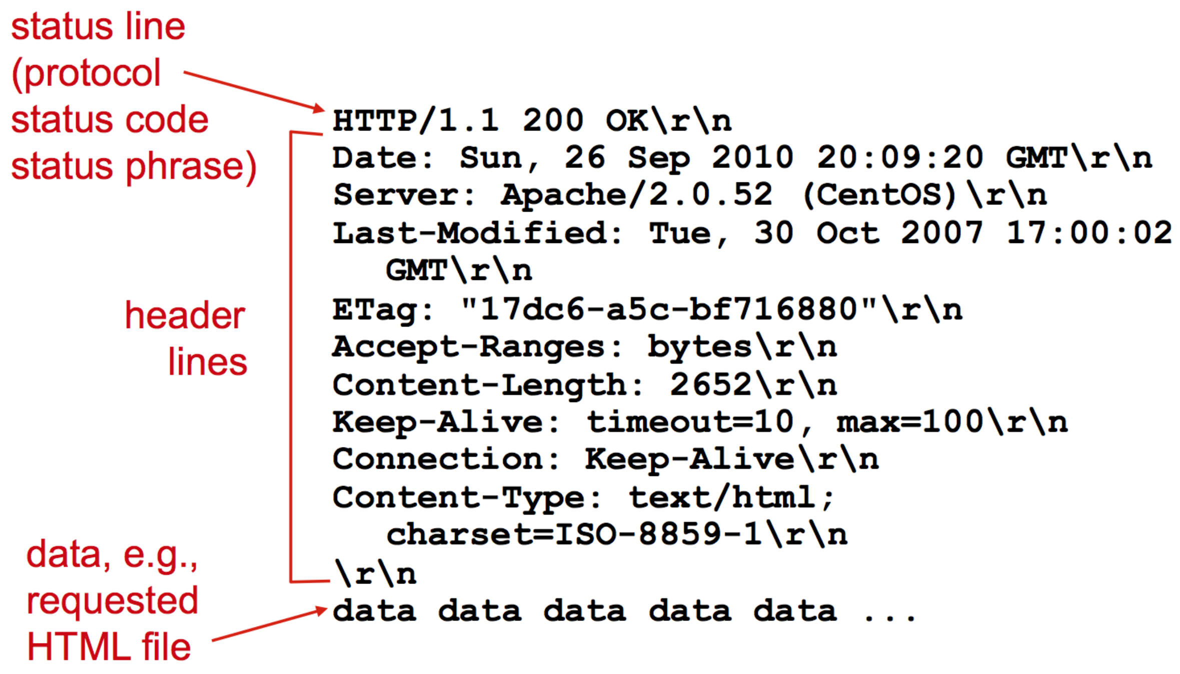
Dany & Sebastian
[daka@ek.dk] [sebi@ek.dk]

HTTP request message


carriage return character
line-feed character
request line (GET, POST, HEAD, PUT, DELETE commands)
header lines
carriage return, line feed at start of line indicates end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP response message



```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html;
   charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response codes

Appears in first line in server-to-client response message.

Some of them:

200 OK

request succeeded, requested object later in this message

400 Bad Request

generic error code, request message not understood by server

404 Not Found

requested document not found on this server

Apache log format

Your logfile should contain information on the http request/response. You can in general specify your own logging, but we would like you to have the following:

```
REQUESTORS_IP - - [DATE_AND_TIME] “HTTP_METHOD REQUESTED_SOURCE_PATH_FILE
HTTP_VERSION” HTTP_STATUSCODE SIZE_OF_RESPONSE_FILE
```

Here is an example of a logfile entry:

```
212.130.228.71 - - [10/Jan/2022:14:22:02 +0000] "GET /index.html HTTP/1.1" 200 17
```