Lab 6

HTTP request

[Compulsory]

Authors: Ragnar Nohre, moved to English Johannes Schmidt

This lab's goal is to exercise socket programming and to use the HTTP protocol on basic level.

In this lab you shall write a small HTTP-server that returns a page showing the client's *request*. We divide the task into two parts. Part 2 is a continuation of part 1.

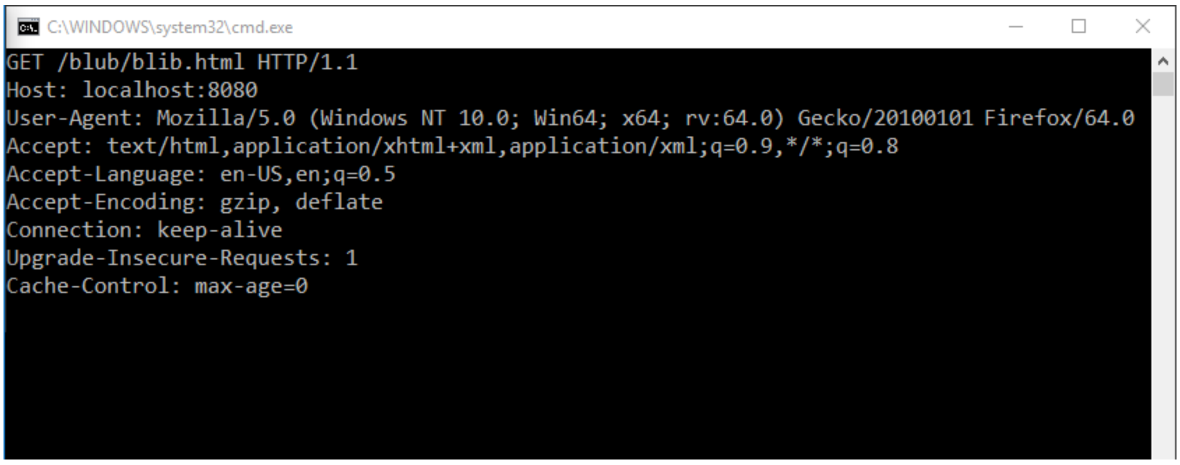
6.1 Part 1

In this part you shall write a webserver that prints the client's request in the command line window.

1. Write a TCP-server doing the following: As soon as a client connects and sends text, the server shall print that text to the terminal window and then shut the connection down. The server shall **continue to run** (and can thus repeat the latter procedure with new connecting clients). Try port 80 if possible, otherwise take another number. (Hint: your server receives a *byte-array*. Transform it to a string with the method `decode("ASCII")`.)

2. Open a webbrowser and surf to *localhost:YOURPORTNUMBER*. The webbrowser sends now a request to your server (provided your server is running). If it works correctly, the request made by your browser is printed in the terminal. Below is a screenshot of my server's output

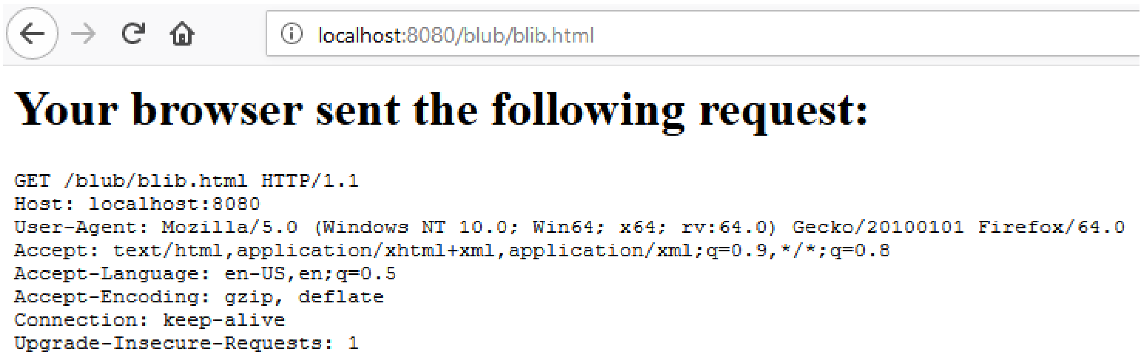
(running on port 8080) when I enter into the webbrowser the address *http://localhost:8080/blub/blib.html*. It is important that the output is *nice* with one request-rubric per line!



```
GET /blub/blib.html HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
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
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

6.2 Part 2

Instead of printing the client's request to the terminal window, now your server shall respond with an html-document that describes the request. The following screenshot shows how it could look like.



Your browser sent the following request:

```
GET /blub/blib.html HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
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
Upgrade-Insecure-Requests: 1
```

Hints:

1. The server's *response* needs to start with the line `HTTP/1.1 200 ok`. Then should follow several response-headers, then a *blank* line, and then the *html-document*. The following code fragment shows how you could generate the first three lines in the server's response.

```
sock.sendall(bytearray("HTTP/1.1 200 ok\n", "ASCII") )
sock.sendall(bytearray("\n", "ASCII") )
sock.sendall(bytearray("<html>\n", "ASCII") )
```

2. Use the html tag `<pre>` in order to preserve the request rubric's formatting.

6.3 Examination

Submit your file from part 2 on Canvas and present it to your lab assistant.