CS 118 Computer Network Fundamentals

Project 1: Web Server Implementation

Daniel Chen 605006027 and Winston Lau 504934155

High Level Description

Our server's design is one that listens for TCP connections through a socket which is an abstraction for an IP address:port number pair, establishes a connection, recieves a request from a client, and responds to the request. Upon recieving the request, our program parses through it to check the request line, making sure the request is of type 'GET'. It then saves the url path and the version into variables to be used again later. After saving those values, our program parses the url to save the extension and checks to make sure it is a valid extension (jpeg,jpg,txt,html,htm,png,gif). For part A of the assignment, we just dumped out the exact request message. For part B, we then checked the file to see if it was a valid file in the same directory as the server. Then the server prepares the response message, getting information for multiple headers like date, connection, last-modified, content-length, and content-type and finally, sends the data that was requested to the client. If it can't find the file requested, it responds with a basic 404 Error.

General Assumptions: We set the maximum URL length to be 2048 bytes. The idea was taken from https://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url-in-different-browsers. We also only accept 'GET' requests.

Difficulties

An initial difficulty that we faced was just writing too much complex code. We thought to save the values of all the headers in the request message and parsed through the request for all the headers which was just quite complex and prone to boundary errors and verbose coding. We eventually decided that we probably don't need the headers and got rid of that whole segment. Ultimately, a lot of time was just lost due to our lack of understanding of what we needed.

Another difficulty we faced was testing. When trying to see if the data was being transferred properly, our server and client connected quickly, but the content and 404 Error never displayed which caused us a lot of confusion as to what was wrong. We eventually realized that it was because we forgot to call the function that actually responded to the request.

One other difficulty for us was relearning C. It has been quite a while since we have programmed in it and forgot a lot of the basics that we used to be familiar with. This led to a lot of reading and rereading of man pages.

We also struggled with dealing with the '/' front of the url as we did not even know that it was part of the url. This caused us much confusion as we couldn't figure out why a file that was in the same directory as the server could not be found.

Finally, it was difficult for us to decided how to structure the code. We started off modulating it in that we had a socket section which was only concerned with creating and establishing the connection, but were unsure how else to modulate it.

Manual

You must first compile our code by typing `make`. The Makefile provided will do the compiling for you.

To start the server, simply type `./webserver`.

Make sure that the files that you want to see displayed are in the same directory as the server. Our server supports html, htm, txt, jpg, jpeg, png, and gif files only.

To actually request a file, go into a browser and type localhost:5861/FILENAME and your file should load. 5861 is just the port number that we decided to use.

To shutdown the server, use C-c (cntl and c). Otherwise, it will keep waiting for more connections. Our Server supports 5 clients.

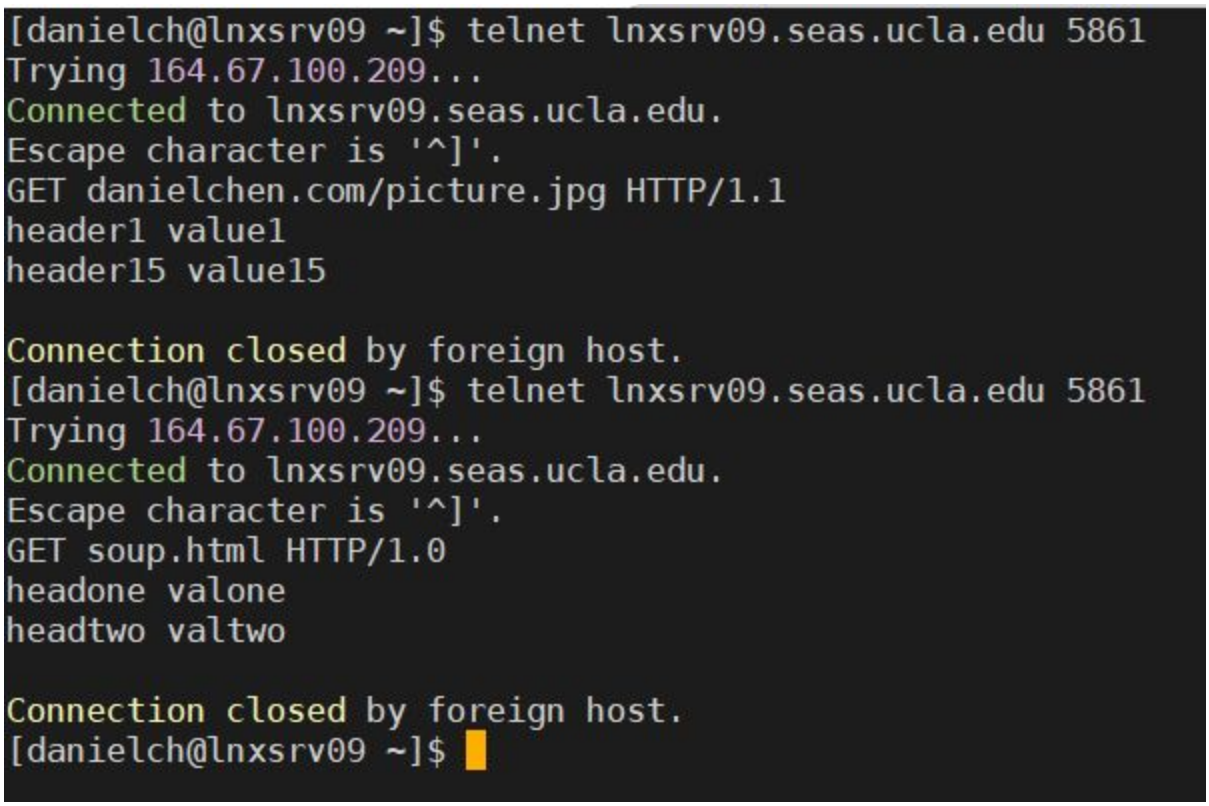
To clean up your directory of the object code for the server

Sample Outputs

Part A

> First I established a connection by typing telnet lnxsrv09.seas.ucla.edu 5861, then I typed a request message "GET danielchen.com/picture.jpg HTTP/1.1" with random headers and values (header1 value1 and header15 value15) which gets dumped to the console properly. GET soup.html HTTP/1.0 also gets dumped to the console properly.

Client:



Server:

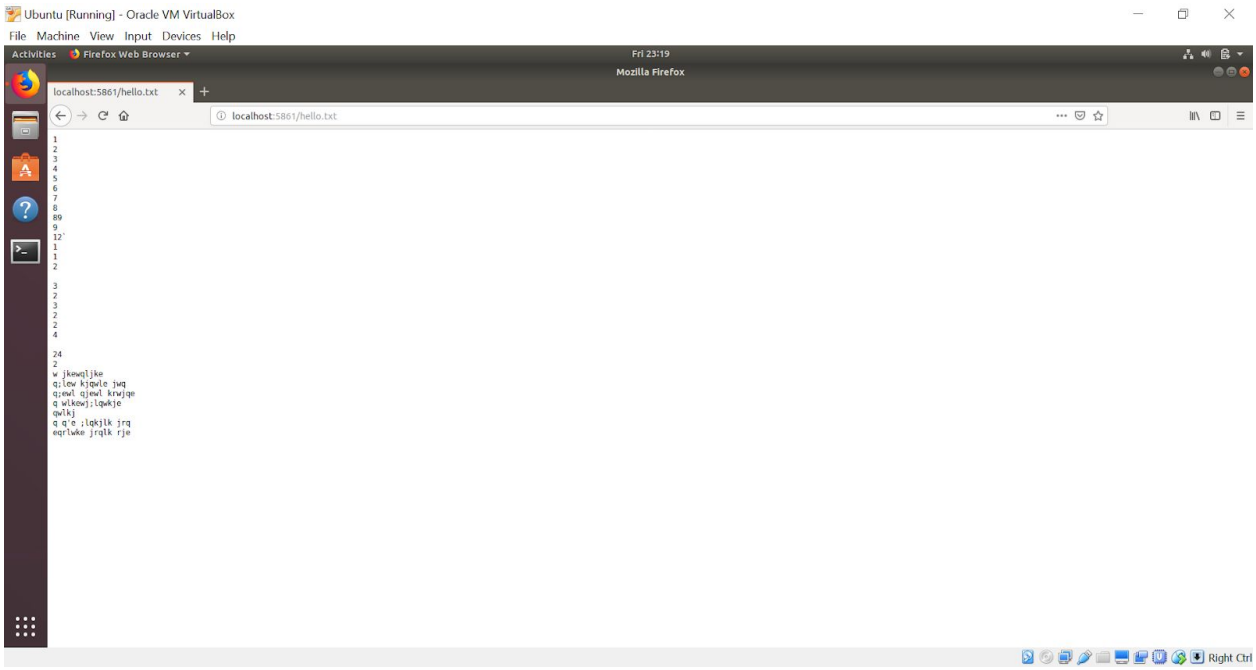
```
[danielch@lnxsrv09 ~/118CS]$ ./webserver
Connection made to: 164.67.100.209
GET danielchen.com/picture.jpg HTTP/1.1
header1 value1
header15 value15

danielchen.com/picture.jpgConnection made to: 164.67.100.209
GET danielchen.com/picture.jpg HTTP/1.1
header1 value1
header15 value15
GET soup.html HTTP/1.0
headone valone
headtwo valtwo
```

Part B

> For this part, we made a file called hello.txt and we put it in the same directory as the server. We moved our server into a virtual machine and ran it in the machine. We connected using Mozilla Firefox and successfully accessed our file.

Client:



Server:

```
danielch@danielch-VirtualBox:~/Documents/118CS/1lab$ ./webserver
Connection made to: 127.0.0.1
GET /hello.txt HTTP/1.1
Host: localhost:5861
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
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
If-Modified-Since: Sat, 27 Apr 2019 05:55:27 GMT
```

> We also tested out our 404 Error message by typing a random file name that is not in the same directory as the server.

Client:

Server:

```
danielch@danielch-VirtualBox:~/Documents/118CS/1lab$ ./webserver
Connection made to: 127.0.0.1
GET /wrong.txt HTTP/1.1
Host: localhost:5861
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
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
```

> Lastly, we tested a file with an extension that our server does not support

Server:

```
danielch@danielch-VirtualBox:~/Documents/118CS/1lab$ ./webserver
Connection made to: 127.0.0.1
GET /wrong.bogustxt HTTP/1.1
Host: localhost:5861
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
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


URL has invalid extension
```