CS 118: Computer Network Fundamentals

Project 1: Web Server Implementation using BSD Sockets

Simran Chawla - 604687055
Dinkar Khattar – 204818138

High-level Overview of Server Design:

Our server is designed in C language. We separated everything possible into different helper functions at the head of the file. We first set up the socket connection using the given port number and boilerplate server socket code. We then listen on the socket for requests and receive requests from the client (browser) in a loop by calling accept(2) and then recv(2). We print the received http request to the console as per the spec, and check if the request is valid. We ignore any request for ‘favicon.ico’. We convert it into lowercase with the help of a helper function lostr(). The code then handles spaces, also with a helper function spaceHandler(), which takes in the filename and converts the user’s input to have spaces if they were entered, by replacing all %20’s with the space character.

In order to open the file the user is requesting, we have a function to retrieve the path, then we use basename() to get the filename. The extension of the filename is then found and assigned to the variable fileType using the function findExtension() that matches the substring in the filename after ‘.’ to it’s MIME type. If the extension is not found, we set it to “application/octet-stream” which is for any binary. The file is loaded, using the function we created called loadFile(). Load file ensures the file-matching is case-insensitive by calling lowerFiles(). This helper function scans through each file in the directory, converting each to lowercase then comparing to the inputted filename. The function then outputs the original filename if there is a match.

To send back the http response, we create appropriate headers. If the file is not found, we send back a simple html file preceded by its headers (HTTP/1.1 404 Not Found, Connection,
Server, Content-Type, Content-length) that writes “ERROR 404 File not found” to the browser. Else, we send back the file using sendfile() preceded by appropriate headers (HTTP/1.1 200 OK, Connection, Server, Content-Type, Content-length). Finally, we close the current socket opened for the file, and start listening for another request.

Difficulties:

Our largest difficulties occurred in attempting to load the non-html files onto the browser. This was difficult, as normal files cannot be loaded with the same syntax as html, as these files are not meant to be displayed as-is on a web browser. We began attempting to place the image files into html files within the C code, however the files were unable to be read as html. We realized that in order for the images and gifs to be able to appear in the browser, they would have to be converted and transferred as binary files. We used the sendfile() function to accomplish this. The ordering of the headers in the http response ended up also being an issue.

Another challenge was figuring out when to use sizeof() over strlen() in C when dealing with char arrays. In certain areas, we found the function to produce an undesirable result, and in other places it worked fine. We were able to debug and decide where each function was appropriate.

We overcame each of our difficulties with thorough debugging and checking values of our variables throughout the execution with the help of gdb.

Manual:

To run our code –

1. Open the terminal and cd into our project folder
2. Type ‘make’ into the terminal

3. In the terminal, type: ./server 4000 (or any other port number greater than 1024).
4. In your browser's URL box, type: localhost:4000/filename

Sample Outputs:

Explanation of Requests:

GET /filename HTTP/1.1
	→ GET request for the path /filename with an HTTP protocol version of 1.1
Host: localhost:4000
	→ specifies name of host (localhost) and port number (4000)
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
	→ provides browser information (OS and version)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
	→ browser information (valid file types)
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
	→ browser information (text encoding)
Connection: keep-alive
	→ keeps the connection persistent
Upgrade-Insecure-Requests: 1
	→ sends signal to express preference for an encrypted and authenticated response

Explanation of Responses:

HTTP/1.1 200 OK
	→ verifies that the connection is successful
Connection: close
	→ closes connection after receiving response
Server: Web Server in C
	→ provides the language and type of program
Content-Type: content_type
	→ provides the type of content to be displayed on the browser (ex: text/html)
Content-length: n
	→ provides length of the content file

HTTP Request 1 – Successful HTML Page Request

```
GET /index.html HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 1

```
HTTP/1.1 200 OK
```

```
Connection: close
Server: Web Server in C
Content-Type: text/html
Content-length: 139
```

HTTP Request 2 – Successful JPEG Request from Directory

```
GET /img.jpeg HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 2

```
HTTP/1.1 200 OK
Connection: close
Server: Web Server in C
Content-Type: image/jpeg
Content-length: 74924
```

HTTP Request 3 – Successful GIF Request from Directory

```
GET /try.gif HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 3

```
HTTP/1.1 200 OK
Connection: close
Server: Web Server in C
Content-Type: image/gif
Content-length: 32126
```

HTTP Request 4 – Successful TXT Request from Directory

```
GET /test.txt HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 4

```
HTTP/1.1 200 OK
Connection: close
Server: Web Server in C
Content-Type: text/plain
Content-length: 6
```

HTTP Request 5 – Successful no extension Request from Directory

```
GET /test HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 5

```
HTTP/1.1 200 OK
Connection: close
Server: Web Server in C
Content-Type: application/octet-stream
Content-length: 12
```

HTTP Request 6 – Request of File that Does Not Exist

```
GET /dkldsjfk.html HTTP/1.1
Host: localhost:4000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
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
```

HTTP Response 6

```
HTTP/1.1 404 Not Found
Connection: close
Server: Web Server in C
Content-Type: text/html
Content-length: 33
```