

CS 118

Project 1: Web Server Implementation using BSD Sockets

Katie Cai                      404740817

Baolinh Nguyen              104732121

## High Level Description

The web server we built is based off of the code provided by the TAs at the following link:

<http://metro.cs.ucla.edu/courses/cs118/project1-example/server.c>

The default port number is 7000.

After initializing and creating the socket, the server listens for incoming HTTP requests. Once an HTTP request is received, a child process is forked to handle that request. The role of the child process is to:

1. Dump the HTTP request it received

```
void processRequest(char* request)
{
    printf("%s", request);
}
```

The following source code dumps the HTTP request received

2. Parse the HTTP request message in order to serve the requested file if the file is found

```
DIR *dp;
struct dirent *ep;
dp = opendir("./");
int validFile = 0;
FILE* fp;
int extension = -1;
if (dp != NULL)
{
    while ((ep = readdir(dp)))
    {
        if (strcasecmp(ep->d_name, file_name) == 0) {
            fp = fopen(ep->d_name, "r");
            extension = getExtension(ep->d_name);
            if (fp == NULL)
            {
                fprintf(stderr, "Error opening file");
            }
            else
            {
                validFile = 1;
            }
        }
    }
    (void)closedir(dp);
}
```

Snippet of code used to search the current directory for the file in question

3. Send back the appropriate HTTP response message

```

if (validFile)
{
    if (write(new_sock_fd, found_header_buffer, strlen(found_header_buffer)) < 0)
        fprintf(stderr, "error writing");
}
else
{
    if (write(new_sock_fd, not_found_header_buffer, strlen(not_found_header_buffer)) < 0)
        fprintf(stderr, "error writing");
}
if (write(new_sock_fd, contentType, strlen(contentType)) < 0)
    fprintf(stderr, "error writing");
if (write(new_sock_fd, CRLF, strlen(CRLF)) < 0)
    fprintf(stderr, "error writing");
if (write(new_sock_fd, file_buffer, len) < 0)
    fprintf(stderr, "error writing");

```

How the server handles sending the HTTP response message and the file itself.

To process the HTTP request and parse through, we look for the name of the file, making sure that we correctly change the “%20” encoding of the space character and use that to step through the current directory, comparing the file name with files found in the directory. The server also makes sure to note the file extension, for use later in the HTTP response header. If the file is not found, the server makes sure to indicate that.

In order to properly send the file over, the server needs to place its contents in a buffer than can be sent. Otherwise, the server uses a combination of fopen, fseek, and fread in order to place the file contents into a buffer. In the case that the file requested is not present, the server sends a buffer that is pre-filled with a 404 message.

```

if (validFile)
{
    if (fseek(fp, 0, SEEK_END) != 0)
        fprintf(stderr, "error using fseek");

    len = ftell(fp);

    file_buffer = malloc(sizeof(char)*len+1);
    if (fseek(fp, 0, SEEK_SET) != 0)
        fprintf(stderr, "error using fseek");
    fread(file_buffer, sizeof(char), len, fp);
    file_buffer[len] = '\0';
}

```

Buffering the contents of the file into a file\_buffer we can then send

Once the file has been buffered, the server begins to send over the header response messages. The server first sends the appropriate HTTP status line and the content-type header line. Once the header lines have been sent, the server sends the actual file to the client.

## Difficulties we faced

One of the problems we had was that we weren't getting the response even though our code appeared to send the data through the socket. We noticed when debugging that what was supposed to be sent to the client was actually outputted again on the server side. After looking over our socket code again, we realized that it was because we didn't call `wait` in the parent process, so the parent process would terminate before the child process and the client therefore wouldn't receive the response. Once we added the `waitpid()` in the parent, the issue was resolved.

Another problem that we had was figuring out how to account for files with spaces. We had initially tested with Safari and thought that spaces would be encoded as follows: `my\ file.txt`. Because of that, our initial attempt used some unnecessarily meticulous string parsing and tokenizing to check for spaces, but we later realized that spaces are automatically converted into `%20`, so we checked for that instead.

### How to compile and run source code

To compile our code, the user simply needs to type `make`.

To run our webserver, the user simply needs to run the command `./webserver`. On the client side, in a browser, the user needs to search for files as follows: `localhost:7000/<filename>`.

### Sample Outputs

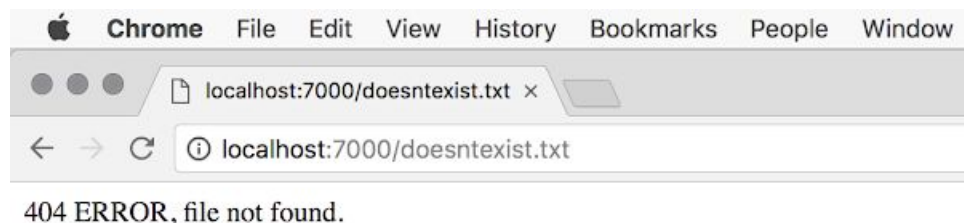
An file that does not exist in the current directory:

GET request for the file that does not exist:

```
ls-169-232-126-8:WebServer katiecai$ ./webserver 7000
GET /doesntexist.txt HTTP/1.1
Host: localhost:7000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

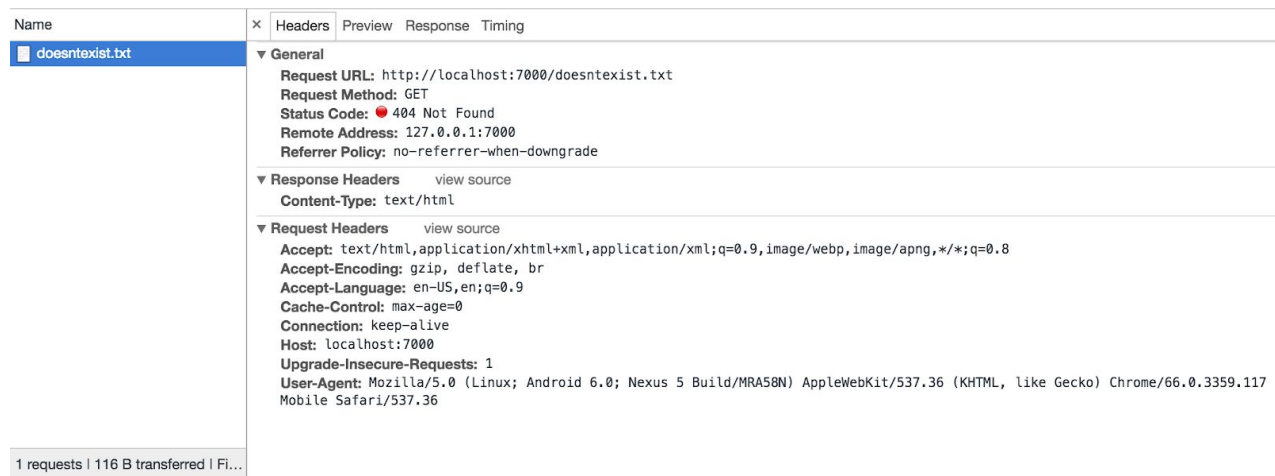
GET /favicon.ico HTTP/1.1
Host: localhost:7000
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://localhost:7000/doesntexist.txt
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

The 404 message displayed on the browser side:



We a 404 error message in the browser because doesntexist.txt isn't in the current directory.

Headers for the response:



The screenshot shows the Chrome DevTools Network tab. The left sidebar lists a single request named 'doesntexist.txt'. The main panel is divided into three sections: General, Response Headers, and Request Headers. The General section shows the Request URL as 'http://localhost:7000/doesntexist.txt', the Request Method as 'GET', the Status Code as '404 Not Found', the Remote Address as '127.0.0.1:7000', and the Referrer Policy as 'no-referrer-when-downgrade'. The Response Headers section shows 'Content-Type: text/html'. The Request Headers section shows various headers including 'Accept', 'Accept-Encoding', 'Accept-Language', 'Cache-Control', 'Connection', 'Host', 'Upgrade-Insecure-Requests', and 'User-Agent'. The bottom status bar indicates '1 requests | 116 B transferred | Fi...'.

Name	Headers	Preview	Response	Timing
doesntexist.txt	<p><b>General</b></p> <p>Request URL: http://localhost:7000/doesntexist.txt Request Method: GET Status Code: 404 Not Found Remote Address: 127.0.0.1:7000 Referrer Policy: no-referrer-when-downgrade</p> <p><b>Response Headers</b> <a href="#">view source</a></p> <p>Content-Type: text/html</p> <p><b>Request Headers</b> <a href="#">view source</a></p> <p>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9 Cache-Control: max-age=0 Connection: keep-alive Host: localhost:7000 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Mobile Safari/537.36</p>			

1 requests | 116 B transferred | Fi...

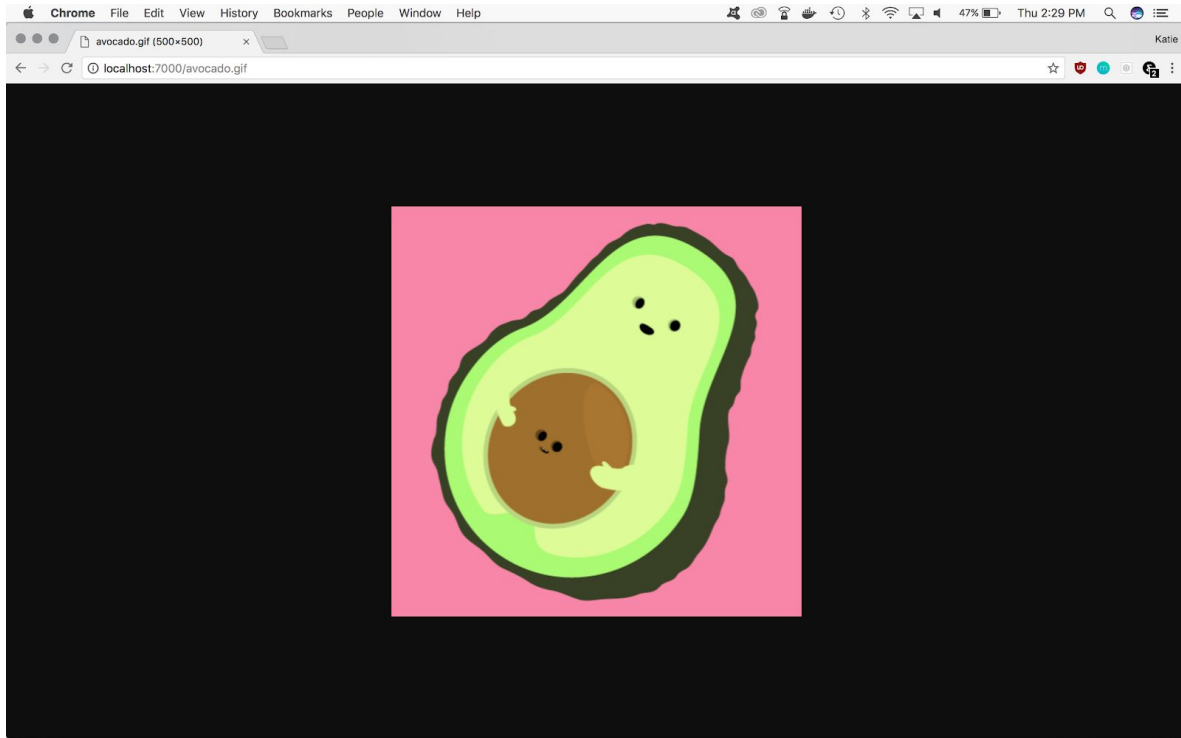
Request for a file in the current directory:

Output of GET request for avocado.gif:

```
GET / HTTP/1.1
Host: localhost:7000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

GET /avocado.gif HTTP/1.1
Host: localhost:7000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

The image displayed on the browser side:



Since avocado.gif is in the current directory, we are able to display it in the browser.

Headers sent in the response for avocado.gif:

Name	Headers	Preview	Response	Timing
avocado.gif	<div><div>General</div><div>Request URL: http://localhost:7000/avocado.gif</div><div>Request Method: GET</div><div>Status Code: 200 OK</div><div>Remote Address: 127.0.0.1:7000</div><div>Referrer Policy: no-referrer-when-downgrade</div></div> <div><div>Response Headers</div><div>Content-Type: image/gif</div></div> <div><div>Request Headers</div><div>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8</div><div>Accept-Encoding: gzip, deflate, br</div><div>Accept-Language: en-US,en;q=0.9</div><div>Cache-Control: max-age=0</div><div>Connection: keep-alive</div><div>Host: localhost:7000</div><div>Upgrade-Insecure-Requests: 1</div><div>User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Mobile Safari/537.36</div></div>			