COM SCI 118 Spring 2019 Computer Network Fundamentals

Project1: Web Server Implementation using BSD sockets

Linh Huynh, *704953181*
Zixuan Wang, *304546929*

High Level Description

The web server program ***server.c*** is written in C. The initialization and socket connection codes are written based on the code skeleton provided by the TA. The default port is **8000**. The server runs forever until the user type crtl-C. Once the connection is made, the server will wait for requests from client. Upon receiving requests, the server creates a child process to process while it keeps listening on new clients.

The main three steps (functions) that the server processes requests from client and sends response:

void handle_requests(int fd)

The function takes the socket number and parses the request from the client. The information required to handle the request are the HTTP method, the file name and HTTP protocol version. If the method is GET, the function will call function *handle_get()* to get the file. Otherwise, it will send a "400 Bad Request" to client since the web server program only supports GET method.

void handle_get(int fd, char* filename, char* protocol)

This function takes the socket number, the file name of the file requested by client, and the protocol. It opens the file and loads data into a buffer; then it sends the response to the client. If no file is found, then it sends a "404 Not Found" message to clients.

void send_response(int fd, char *header, char* content_type, char* body, int content_length)

This function writes the response to the client through the given socket number with the given header, content type, content length and the body of message.

Some helper function to handle request:

FILE* open_file(char* filename)

This function opens the file requested by the client from the file name, ignoring the case of the file name. It returns the file descriptor of the file if the file exists, otherwise it returns NULL.

char* get_file_format(char* filename)

This function extracts the file extension from the file name. It returns the media type (MIME) of the file. The server can support plain text files encoded in UTF-8 (*.html, *.htm, and *.txt), static image files (*.jpg, *.jpeg, and *.png), and animated image files (*.gif). If the file name does not contain a file extension, the default type of unknown binary file (application/octet-stream) is returned.

void to_lower_case(char* s)

This function makes the given string into lower-case.

void remove_carriage_return(char* s)

This function removes the carriage return '\r' character at the end of a string.

void default_index(int fd, char* protocol)

This function displays a HTML "Hello World" page on client if no file name is given.

void resp_404(int fd, char* protocol)

This function displays a HTML "404 Not found" page if no file is found.

void bad_request(int fd, char* protocol)

This function displays a HTML “400 Bad request” page if a method other than GET is use or when the protocol is invalid.

The main function:

int main()

The function creates a new socket and binds the created socket to an address consisted of a port number on a host machine. Then it gets ready to listen from clients for connections and keeps accepting connections from clients. Codes of setting up sockets connection for interprocess communication in this function are generally provided by the TA.

Difficulties

1. **Process file names that contain spaces**

We handle the case where file names contain spaces by utilizing strtok(3) function. The strtok() function breaks a string into a sequence of zero or more non-empty tokens. The function is called on the header of the HTTP request from client to extract the method, the filename, and the HTTP protocol. Since the header order is fixed, we can go through each token one-by-one to get the information we need. The first token is the method. The following tokens which are not “HTTP” token are the file name. The rest is the HTTP protocol.

2. **Handle file names in a case-insensitive fashion**

We handle this by using the <dirent.h> header which defines the data type that contains the information of the files inside a directory. First, we get the directory absolute path with getcwd(3) function. Then, we open the directory with opendir(3) function and read all the regular files (DT_REG) in the directory with readdir(3). Lastly, we use strcasecmp(3) which compares two strings ignoring the case to check if the given file name matches any file name from the directory. If it does, we open the file and load its data into a buffer.

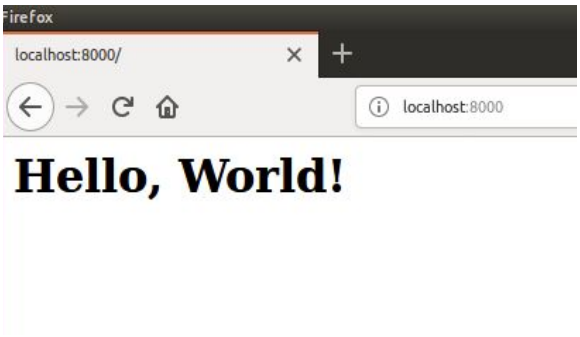
Manual

To compile the program, type **make** in the terminal
To start the server, type **make run** or **./server**
To remove the executable, type **make clean**
To connect to the server from a browser, use the URL http://localhost:8000/<filename>

Sample Output

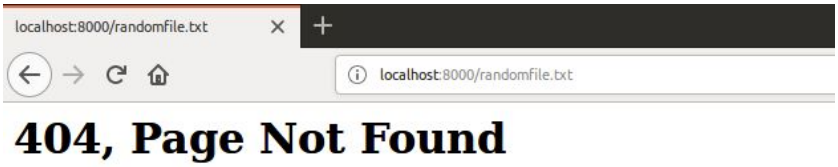
1. Connect to server and does not request any file

```
Server: Got connection from 127.0.0.1
GET / HTTP/1.1
Host: localhost:8000
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
```



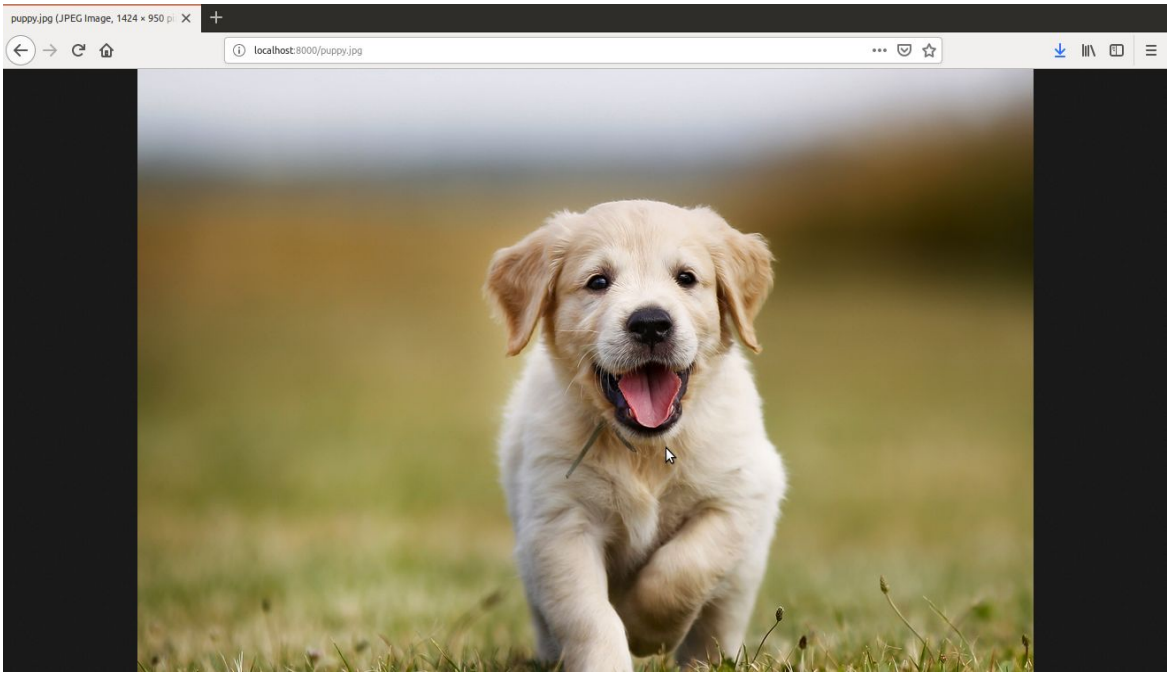
2. Connect to server and request a file that does not exist

```
Server: Got connection from 127.0.0.1
GET /randomfile.txt HTTP/1.1
Host: localhost:8000
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
```



3. Connect to server and request an image file

```
Server: Got connection from 127.0.0.1
GET /puppy.jpg HTTP/1.1
Host: localhost:8000
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
```



4. Connect to server and request a HTML file that contains spaces in file name

```
Server: Got connection from 127.0.0.1
GET /Walt%20Disney.html HTTP/1.1
Host: localhost:8000
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
```