BSD Socket Web Server

Project 1 CS118

Zhouyang Xue

Yang Li

Description

For this project, we implemented the a naive web server with BSD socket.

Our program can be sliced into three major parts that work with each other, to allow users access html/txt files and images on our web server.

The three parts are:

1. Socket connection setup.
2. Deliver/serve the requested file.
3. Issue the response message and print it out on the console.

For socket setup, we used IPv4 protocol, which we are more familiar with and easier in terms of built-in function complexity. In the end, our server also create child processes for each visitor, so that multiple browsers can visit the server at the same time. We also have a child\_handler() function to deal with certain occasions in child processes.

Setup variables:

sock\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

To deliver the requested files, we first parse the filename from the request message into a string.

Then we check the ‘\0’s position to determine whether a requested file is issued. If not, the server response with the “404 Not found” message and write on the console “No file specified” (which is mainly for debugging purpose). Note that we also have a space\_edit() function to help parsing. Replacing %20 with “ “.

After parsing the filename, we used normal C read and write system calls to pass the file through the socket.

Finally, for the response part, we check for and represent the file type using an indicator, and thus return the response message based on different file types.

Challenges

In this lab, we have several challenges. The first is that when we set up the connections

We find that the accept function fails sometimes. So instead of using accept once, we write a while loop and keep accept inside, so that our server will wait for client to connect with it.

The second challenge is that when debugging, our result is always 404 not found. We used xcode to debug and find that is caused by the misused function when we read the file length twice.

The last challenge is how to handle the space in the file we want to find. When I type space in my browser, i find that it will be automatically converted into %20. So I write a replace space function simply to replace %20 with space, so that our new server works well the file name contains space character.

Manual

The Makefile contains three targets.

* default: compile webserver.c into webserver.

The executable webserver takes in one single parameter, which is the port number.

* remove: delete the executable webserver.
* dist: make the tarball file.

Comments

Some sample output messages on console:

> localhost:3004

HTTP REQUEST MESSAGE:

GET / HTTP/1.1

Host: localhost:3004

Connection: keep-alive

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36

Upgrade-Insecure-Requests: 1

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,zh-CN;q=0.8,zh;q=0.7,zh-TW;q=0.6

File Not Specified

> locahost:3004/image.jpg

HTTP REQUEST MESSAGE:

GET /image.jpg HTTP/1.1

Host: localhost:3004

Connection: keep-alive

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36

Upgrade-Insecure-Requests: 1

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,zh-CN;q=0.8,zh;q=0.7,zh-TW;q=0.6

HTTP RESPONSE MESSAGE:

HTTP/1.1 200 OK

Connection: close

Date: Sun, 04 Feb 2018 01:45:52 UTC

Server: ZXLY/1.0

Last-Modified: Sat, 03 Feb 2018 21:09:48 UTC

14679Type: image/jpg

"image.jpg" delivered!

Test case:

In order to test our code we first set the local host to be a random number. We use our browser to do the test. We fist type localhost:port name and to check if our console prints out the message. After that we put several files with different size and different formats including html and jpg into the same folders as the server. We type the file we want to find. Some files we create them with space in the name in order to check if our server can handle the space. We check the correctness by looking at the console and also the browser to check if the image and html file can be loaded.