

CS118: Computer Network Fundamentals

Project 1: Web Server Implementation using BSD Sockets

Yuhai Li 104844760

Chiao Lu 204848946

Description

The server first creates and initializes a socket. Then it stops and wait for new connections. Once a new connection is built, it will start to process the request as described in the project manual. After that, the server will send back a HTTP response and wait for a new connection.

To fulfill the requirements, the server first prints the HTTP request on the console. And then it extracts the requested filename and compares it with all files in the server root folder case-insensitively. If the name matches any file in the folder, the server creates and returns a HTTP response with the file content. Otherwise, the server just returns an error 404 message to notify the user that the file is not found.

Problems and Solutions

GCC and Make Not Installed

I was unable to compile and run the program using `make`. After debugging for several hours, I realized that I do not have `make` and `gcc` installed on my Ubuntu server.

To solve the problem, I ran the command `sudo apt-get install make` and `sudo apt-get install gcc`, which solved the problem.

Couldn't Successfully Transfer Large Files

Transferring large binary files (>100MB) seems to be OK. However, after using `diff` to compare the transferred file and the original file, we noticed that the two files were not identical.

The cause of this issue is unexpected. It turns out that HTTP Response/Request does not end a line using `\n`; instead, we needed to use `\r\n` (a carriage return plus a line feed) to indicate the end of a line. By changing all `\n` to `\r\n`, we were able to correctly transfer large files and pass the `diff` test.

Unable to Reuse the Same Port

When testing the program, it was necessary to constantly stop and rerun the program using different parameters. While doing so, we discovered a strange behavior: even if we completely terminate the program before running it again (by issuing a `killall` command), we were still unable to use the same port as we used before.

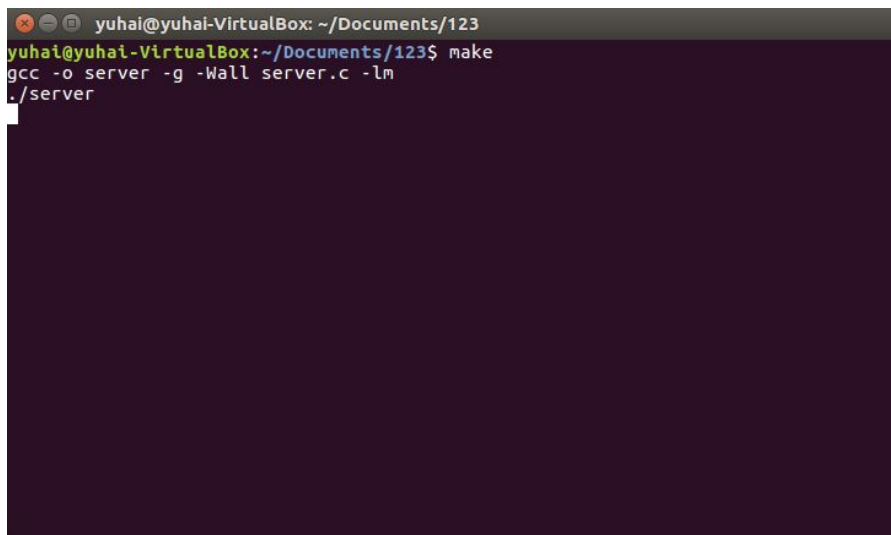
To solve the problem, we called the function `setsockopt()` before calling `bind()`. By doing so, we tell the program to reuse the port that we had used previously.

Manual

- Compile and run: `make`
- Clean the executable: `make clean`
- Test URL: <http://localhost:8888/><filename>
- Kill program: `Ctrl + c`

Sample outputs

1. The server program compiles and runs normally.

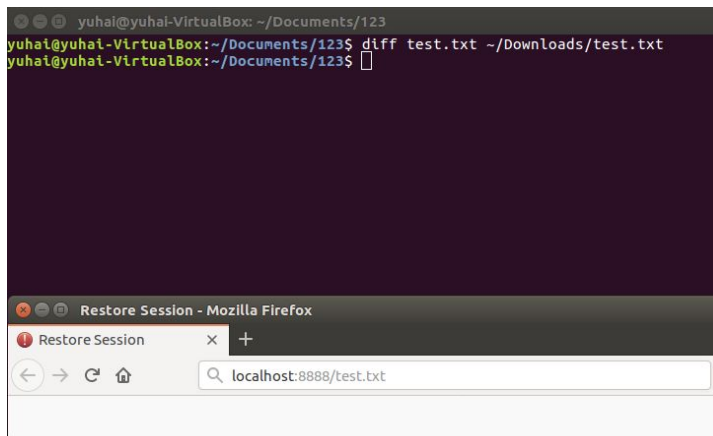


```
yuhai@yuhai-VirtualBox: ~/Documents/123
yuhai@yuhai-VirtualBox:~/Documents/123$ make
gcc -o server -g -Wall server.c -lm
./server
```

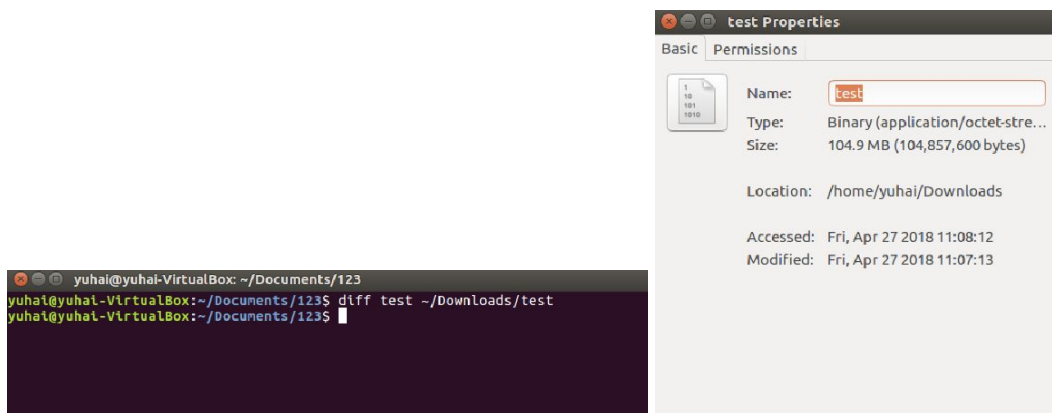
-
2. The server can print out the correct HTTP request messages to the console.

```
yuhai@yuhai-VirtualBox: ~/Documents/123
yuhai@yuhai-VirtualBox:~/Documents/123$ make
gcc -o server -g -Wall server.c -lm
./server
GET /test.txt HTTP/1.1
Host: localhost:8888
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
```

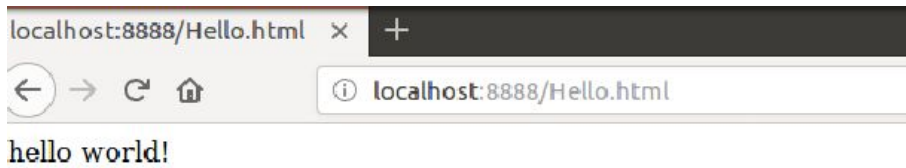
-
-
3. The server program transmits a small binary file (up to 512 bytes) correctly.



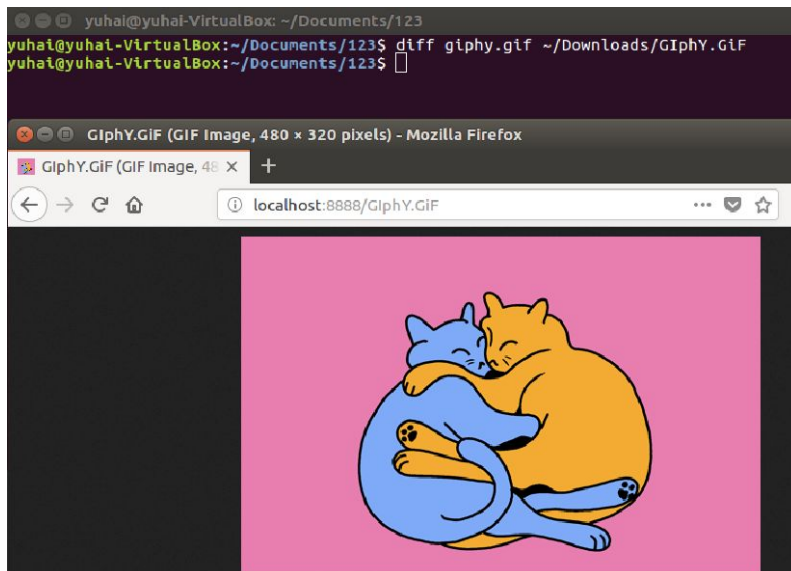
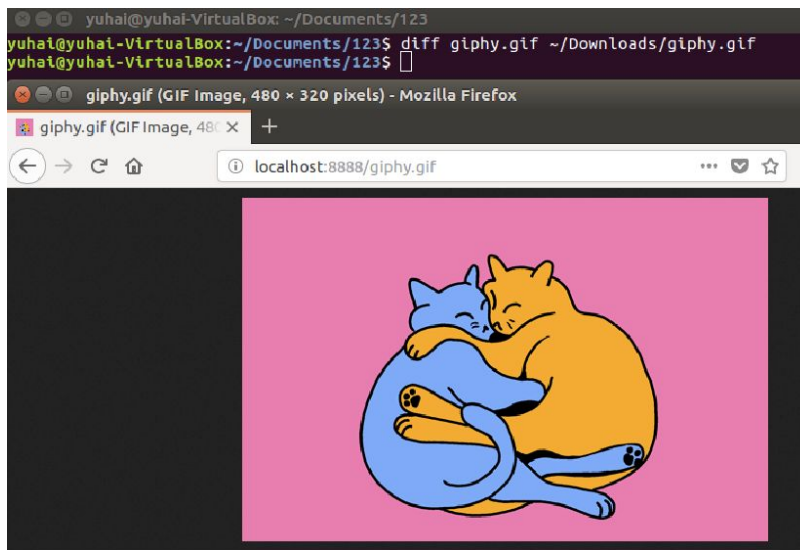
-
-
-
4. The server program transmits a large binary file (up to 100 MB) correctly.



-
5. The server program serves an HTML file correctly and it can show in the browser.



6. The server program serves an image file correctly and it can show in the browser.



-
7. The server program serves a file which file name contains space correctly in the browser.

