CS 118 Spring 2018

Project 1 Report:
Web Server Implementation using BSD Sockets

Shen Teng, 104758168

Goal

The goal of this project is to develop a Web server in C/C++ using BSD sockets. In addition, by doing this project, we took the chance to learn a little bit more about how the Web browser and server work behind the scene.

High-level Description of the Server's Design

As our TA explained in the discussion session, in order to establish a connection between the client and the server, the client and the server first have to create a socket. So, firstly, we used the socket programming APIs to create sockets. Then, for the server socket, we binded it with the local IP address and a port number specified by the user by getting the port number as the command line argument. Then we set the server socket to listen and wait until it accepts the connection coming from the client.

Once the browser connects with the server socket, the server will read from the client until one full http request is detected, then print out the request, and generate response for the client's request. Then, our server write the generated response to the socket and check if there was any error while writing to the socket. If there is no error, we made the server to finally end the connection by closing the socket.

We put the above general process of creating sockets and establishing connection part in our "lab1.cpp" file, and then made functions to take care of getting the filename, converting the file name into lower cases, taking care of space in the file name, getting the file extension, finding the file in the directory, and generating the response in the form of HTTP response in a header file called "lab1.h" and used them in the "lab1.cpp" file.

Difficulties and How We Solved Them

1. In order to handle special characters e.g. space, which will be url-encoded by the browser, we use other's solution.
   The file can be found here :
   http://read.pudn.com/downloads80/sourcecode/windows/file/309471/UriCodec.cpp__.htm
2. Since there are many string operation, we used C string at the beginning. It became tedious to spend time on managing memory. Later we switched to C++ and used string class.

Brief Manual on How to Compile and Run

With Makefile we included, all the user has to do is to type make command like this:

```
$ make
```

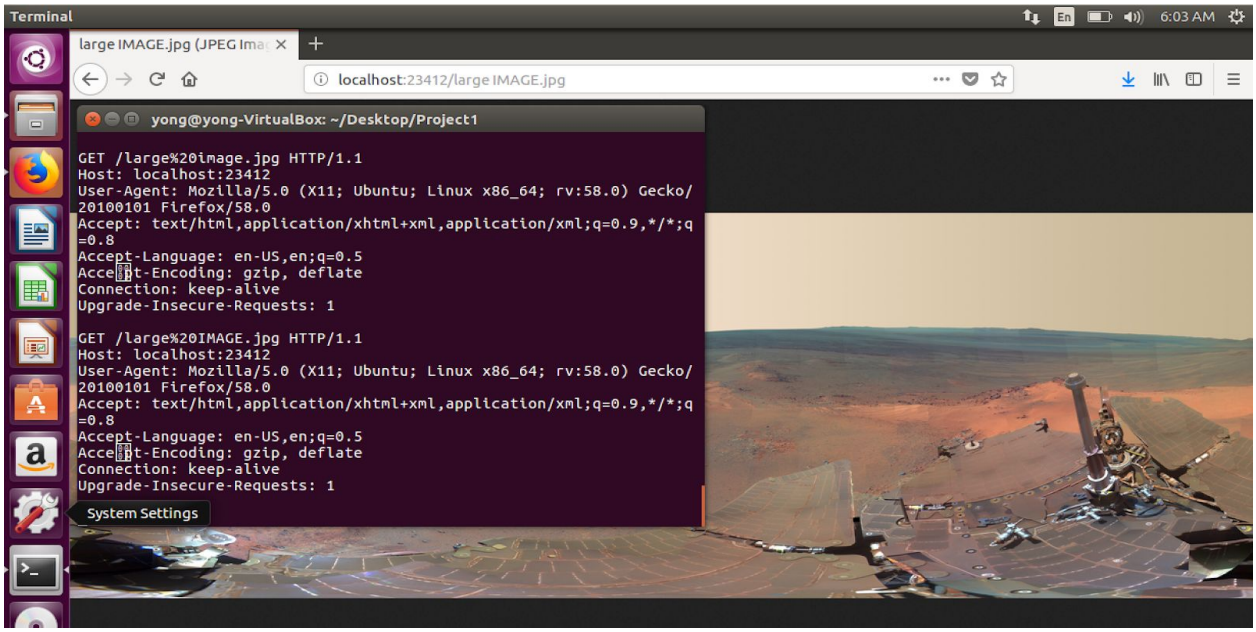
Makefile basically does the following:

```
g++ lab1.cpp -std=c++11 -O2
./a.out 23412
```

At first, with g++ compiler, the first command compiles our C++ file and generate an executable file “a.out”.
Then, it runs the executable file “a.out” with the port number 23412, which is a port number we chose.

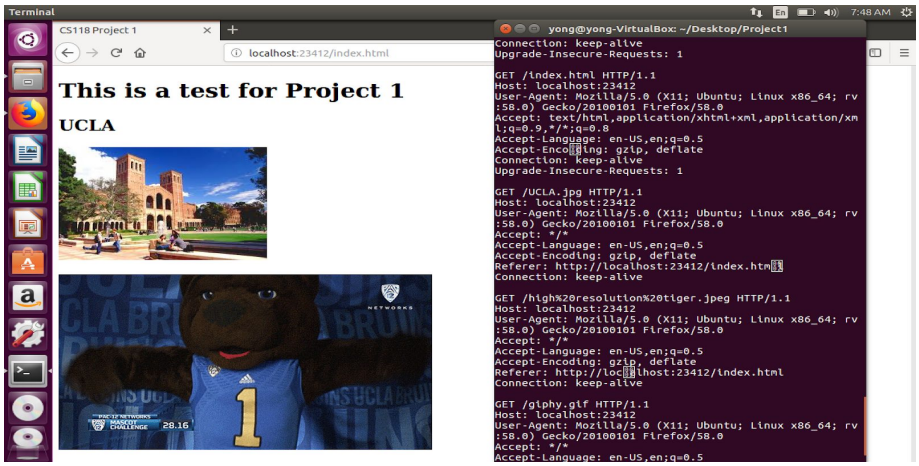
Sample Outputs

Large image .jpg file(~ 90MB):

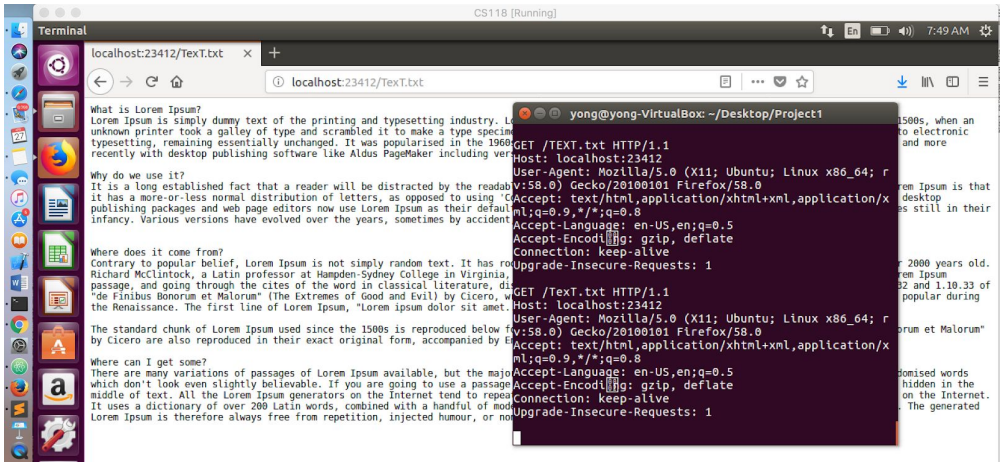


As you can see on the terminal, our server takes care of the space in the file name and correctly sends the file to the client. We also tried to test by using uppercase and lowercase in the filename and our server properly finds the file even if the client use uppercases for lowercases or lowercases for uppercases in the file name.

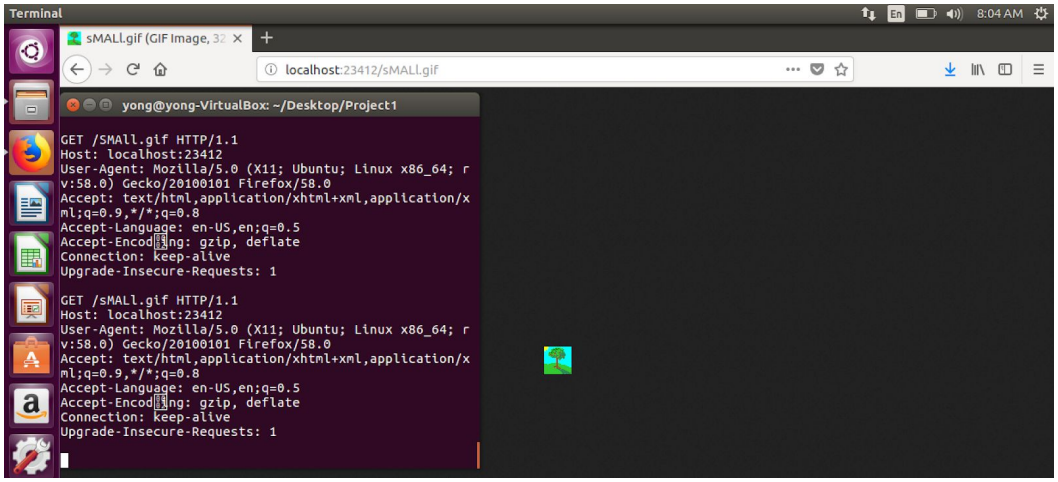
HTML file:



Text file (again also checking for case-insensitiveness in the file name):



Small binary file (~230 bytes):



When file does not exist: