Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of School of Computer Science and Statistics
Department of Computer Science

3D3 – Project 1

WEB CLIENT SERVER

3D3 – Computer Networks

Name: Videept Kohli
Student Number:18335157
Date of Submission: 8/03/19
Lecturer: Ciaran McGoldrick

Aim:

In this project, I implemented a http client and a server before establishing a connection using them to send data including files and text. The client specifies the file to be sent using protocols defined and the server parses the request and then fetches the file before returning it as an html file. If the file does not exist, then we return an error code.

High level Design of Server and Client

- Web-Client
  - $./web-client [URL] [URL]... The program can take in two URLs at runtime which point to the files we need to extract using command line arguments using the format shown
  - We use the send () function to send the request to the server and the recv() function to receive the file from the server.
  - Client uses GET command to send the server the file which it requested.
  - We store the address of the clients using sin function using the client end socket we create. Then we use the bind() function to provide the IP address and port number to the socket.
  - We receive the data and then store the file in a buffer before outputting it to the terminal.

- Web-Server
  - The server checks whether the file exists and after checking whether it does it returns a response of 200 ok along with the directory of the file.
  - The default port number we have referred to in our program is 4000. Whenever port number is used it will connect to port 4000.
  - The server socket has a default address of “localhost” or it also connects to “127.0.0.1”.
  - Finally I store the address of the server side using the sin family function in a server socket created and then a bind() function to provide the IP Address and the port number.
  - I’m also using pthreads to perform multi-threading operations to allow multiple clients to run on a single server using the same port.
  - I’ve used the listen function to make the server start listening. Here, we mention the number of clients to be queued at the same time.
  - I’ve used the inet_ntop function to transform IPv4 and IPv6 addresses received from the client from binary to text form.
  - Once the client sends the request it’s decoded and then I’ve used response and request classes to parse the request received. The request includes the directory in which we could find our file.
  - After parsing the file is opened in the given directory. A Status code of 404 File not found is given if the file does not exist.
  - Otherwise it stores the contents of file and return the error status code 200 ok
  - Now the file is sent back to the client using the send() function which includes the response message and the file contents.
  - At the end we close the socket for the next client to get linked to it .

Problems I ran into:

- **Persistent Connections**- While my project is able to run non-persistent connections, I kept searching for how to make persistent connections work but to no avail. I tried to make it work using a while loop, but that was giving me a stack error. The server kept closing as soon as it got the file.

- **Vagrant**- I spent a lot of time in getting the environment working on Windows but after that didn't work out even after consulting instructors then I shifted to doing it on Linux in VirtualBox.

- **Imperfect Output**- While the file is getting sent to the client, after the output is printed I'm getting a few garbage characters which shouldn't be the case. I tried changing the size of the buffers but that didn't help.

- **MakeFile**- I had never made a MakeFile before so I struggled with the concept of one and had to watch a few videos to understand the basic concept behind one.

- **Implementing Multithreading**- While I was not able to getting multiple clients working together, I used the concept of pthreads and now I can get multiple clients open at the same time.

- **Socket Programming**- I've never done socket programming before and to make the server and client I had to go through a wide plethora of resources.

Library Files Required

Below shows the libraries that we used in the build of our web client server.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <iostream>
#include <fstream>
#include <strings.h>
#include <string>
#include <pthread.h>
#include <sstream>
```

Testing Code:

I've tested all the individual files to make sure that they all work independently of the overall design.

I tested the HTTP parser to ensure that messages from response/request were being parsed correctly and checked each individual parameter.

Web-client – I tested this with my web server and a sample input file to check whether the output file is as requested. It also received a message back from the server. The user types in the file he wants and gets a response message.

Web-server – I first checked this with a sample web client- Mozilla Firefox to check whether the address sent prints the required file. Then I checked it with the client and it continued to work.