COMPUTER NETWORKS

Assignment 1
HTTP Server & HTTP Client

Abdelazez Elsayed 38

HTTP Server

The server Consist of 3 Classes and a server.cpp source code

Server.cpp

A main program to run server , the server at first initializes its listening socket and looping till get a incoming connection and have a free thread, if so, it starts a thread and assign the client socket to it using `SocketHandler Class`

SocketHandler

A Class that takes the socket as an argument for constructor and then handles the incoming connection of the socket using `handle()` method

Class members

Cur_threads :

Atomic integer to count how many threads is taken

Handle()

Loops over socket to get the data from it, once the client has sent all the request it calls `HttpParser` and `HttpBuilder` , two classes to parse the incoming Http request and then build the right Http Response

```
sendall(SOCKET ConnectSocket,const char *buf, int *len)
```

A function that sends and makes sure that all the data is sent to client socket

HttpParser

A class to parse the incoming http request and get data from it

Class members

```
vector<char> parse(vector<char> input);
```

A method to parse the http request, the request is given as vector of chars

This method parses only one request and returns the remaining request to be parsed again (This situation happens only in pipelining,if not so, the returned is vector of size 0 i.e. there was only one request sent to this method and nothing remained after it)

```
map<string,string> header_map ;
```

A map has the incoming request header field and value

```
string method;
```

The request method , currently handling GET and POST only

```
string request_uri;
```

The request uri

```
vector<char>* body = nullptr;
```

The body of request if the request is POST

HttpBuilder

Class Members

```
void build();
```

```
A method to start building the response of Http
```

```
string to_string();
```

Returns the response as string

HTTP Client

The client works by reading an input file that has the commands line by line in it

The input file is given in the argv

The client consists of two files main.cpp and file_parser.h

File_parser.h

```
vector<vector<string>> parse(string file_path);
```

A method that reads the input file, parses it and returns vector of vector of string

Where vector[i][j] represents the command i and j goes from 0 to 3 , 0 the command type (client_get/client_post) , 1 file_path , 2 host_addr 3 port_num (if not exist in command it is set by default 80)

```
string create_http_command(string method,string file_path);
```

A function that takes the method (client_get/client_post) and file path and returns a string of corresponding HTTP request to make the command

```
pair<int,char*> get_body(string file_path);
```

A function that takes a file path and returns a pair of int and char*

The char* represent an array of bytes of that file encoded binary

The int member represents the size of the char*

Main.cpp

```
SOCKET get_connection(string host,string port)
```

A function that takes host and port and returns a socket with that address

, this function uses `make_socket(string host,string port)`

If the socket is opened , the difference is `make_connection()` always makes a new socket where `get_connection()` saves already opened sockets in a map and returns that socket if exist

```
int send_data(string http_command,SOCKET ConnectSocket)
```

A function that sends the http_command (Request) through ConnectSocket returns -1 if it fails uses `sendall` same as in Server SocketHandler function

```
string recieve_data(SOCKET ConnectSocket)
```

A function that returns the received data from a socket as a string

```
bool isReadyToReceive(SOCKET socket)
```

A function that tests if a socket returns a data within a time window

```
bool isFinished(char *buff,int iResult,int recvbuflen)
```

A function that tests if a buffer is finished , i.e. the full respond is sent

```
void closeAll()
```

A function that closes all opened connections

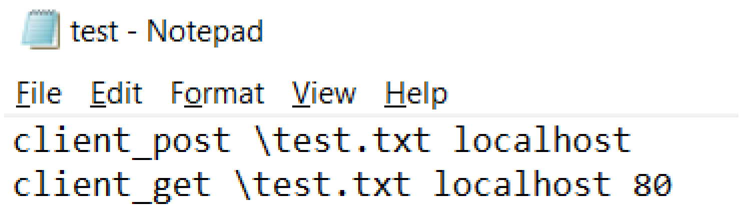
```
int  my_close_socket(string host,SOCKET ConnectSocket)
```

A function that closes the connection,

Note !

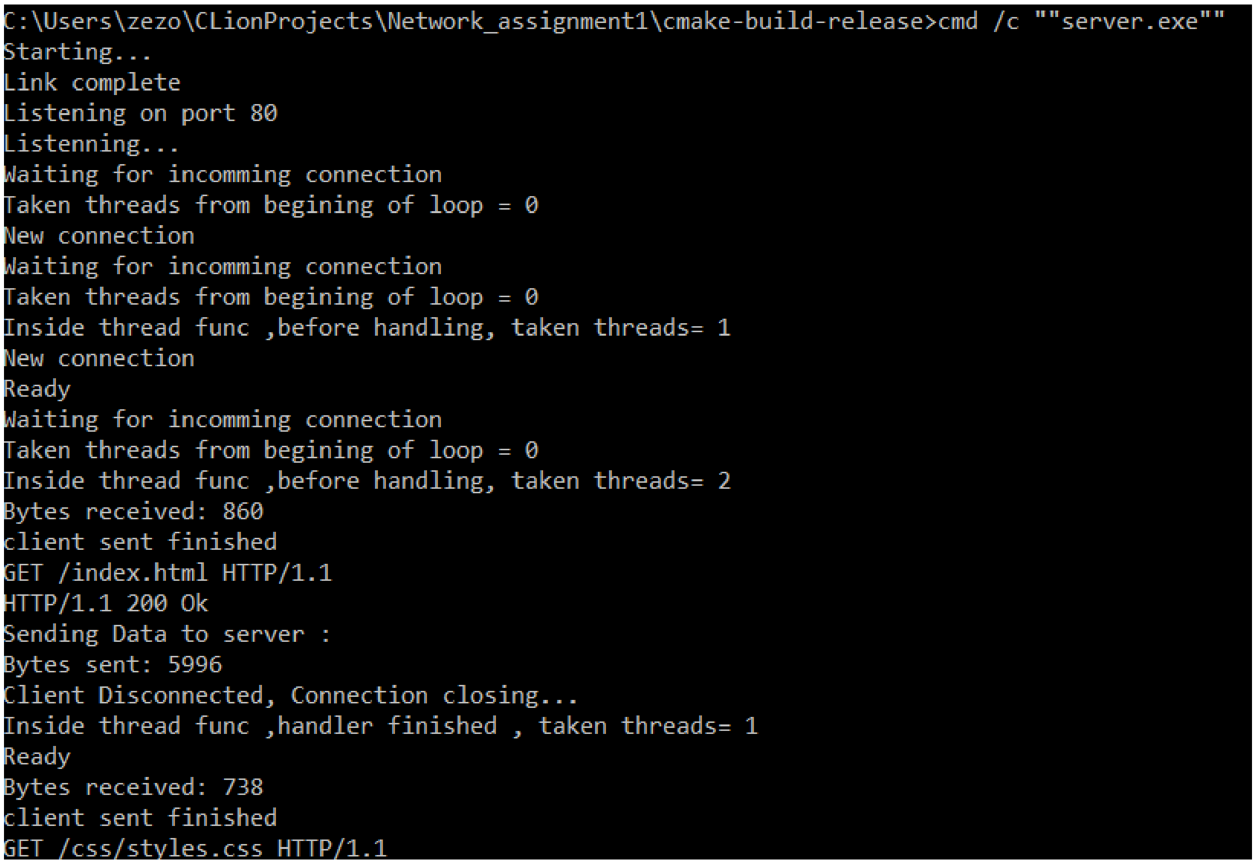
Client.exe takes as an argument the path of input file of commands

Sample file be like



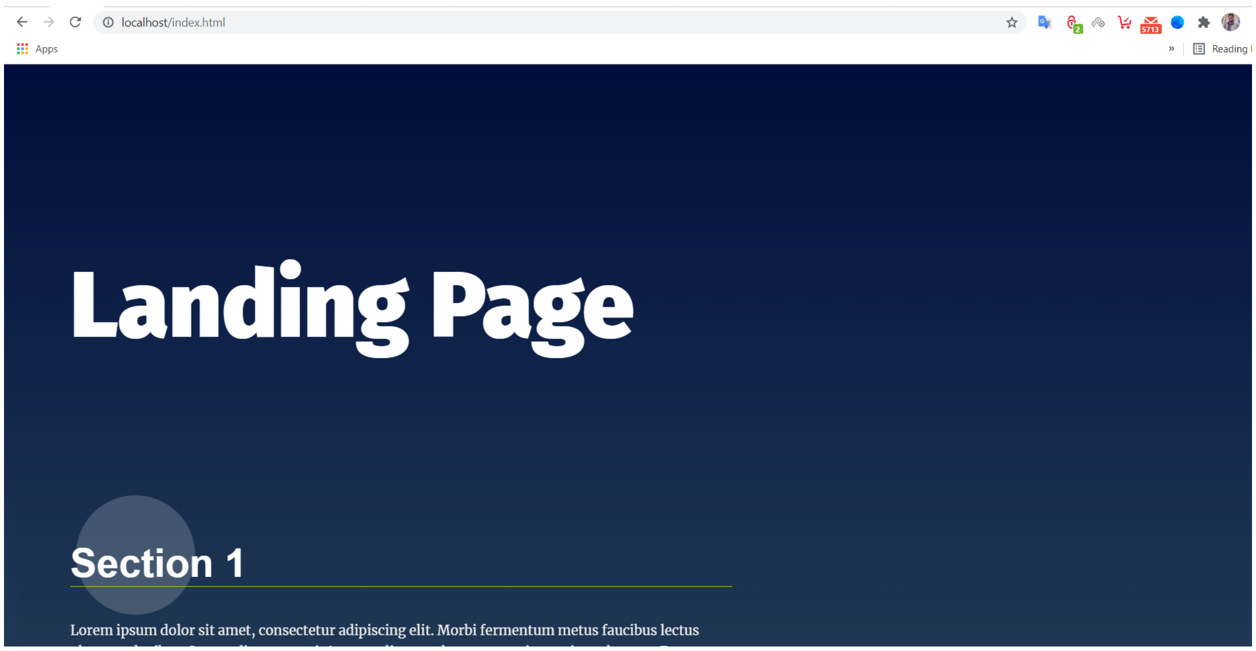
```
client_post \test.txt localhost
client_get \test.txt localhost 80
```

Sample runs



```
C:\Users\zezo\CLionProjects\Network_assignment1\cmake-build-release>cmd /c ""server.exe""
Starting...
Link complete
Listening on port 80
Listenning...
Waiting for incomming connection
Taken threads from begining of loop = 0
New connection
Waiting for incomming connection
Taken threads from begining of loop = 0
Inside thread func ,before handling, taken threads= 1
New connection
Ready
Waiting for incomming connection
Taken threads from begining of loop = 0
Inside thread func ,before handling, taken threads= 2
Bytes received: 860
client sent finished
GET /index.html HTTP/1.1
HTTP/1.1 200 Ok
Sending Data to server :
Bytes sent: 5996
Client Disconnected, Connection closing...
Inside thread func ,handler finished , taken threads= 1
Ready
Bytes received: 738
client sent finished
GET /css/styles.css HTTP/1.1
```

Browser usage


localhost/index.html
Apps
Reading l
Landing Page
Section 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi fermentum metus faucibus lectus

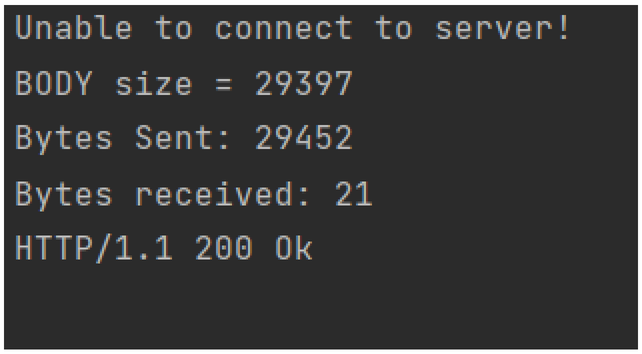
Client Sample Run

For commands

client_post \hold\tt.png 129.15.152.1 800

client_post \hold\tt.png localhost 80

Result :



```
Unable to connect to server!
BODY size = 29397
Bytes Sent: 29452
Bytes received: 21
HTTP/1.1 200 Ok
```