Networks Lab#1

Names :

- Omnia Saad El Dawy (19)
- Amira Nasrullah Mohammad (21)

<u>HTTP Web Server :</u>

Overall Organization :

- Open a socket using socket system call and set it to TCP
- bind the socket to an address
- while true: do
  - Listen for connections
  - Accept new connection from incoming client and delegate it to worker process.
  - Parse HTTP/1.0 request
  - Ensure well-formed request (return error otherwise)
  - Determine if target file exists and if permissions are set properl(return error otherwise)
  - Transmit contents of file to connect (reads from the file and writes on the socket)
  - Close the connection
- end while.

Class Server :

- **Data Structure :**
  - Arrays
  - Vectors

- **Functions :**

  - **readTheFile(FileName)**
    - This method takes as input the file name and returns a string connects the data of the file
    - check for the type if image to read it binary
    - if the file is text or html read it line by line and append to a string to return
    - if the file is image read it as binary and append it to a string stream
    - append the header of the response
    - return the string

Process VS Threads :-

- we used process as the process is used for a heavyweight task.
- To make clients better isolated and more robust, like with most servers, go with sockets. When one thread crashes badly, it usually takes down the entire process, including other threads working in that process.

HTTP Web Client

Overall Organization :

while more GET operation exists in the file do

- Create a TCP connection with the server
  - a socket is created through call to socket()
    - function an UN-named socket inside the kernel which takes domain/family as its first argument. For Internet family of IPv4 addresses we use AF_INET.
    - And the second argument 'SOCK_STREAM' specifies that the transport layer protocol that we want should be reliable .
    - The third argument is generally left zero to let the kernel decide the default protocol to use for this connection. For connection oriented reliable connections, the default protocol used is TCP.
    - and returns an integer known as socket descriptor .
- Wait for permission from the server
  - Set Information like IP address of the remote host and its port is bundled up in a structure and a call to function connect() is made which tries to connect this socket with the socket (IP address and port) of the remote host :
    - It takes three arguments, the socket file descriptor, the address of the host to which it wants to connect (including the port number), and the size of this address
- Send next GET requests to the server
  - send request to server by send(sockfd, buffer, strlen(buffer), 0)
    - It takes four arguments, socket descriptor as first argument.
    - Second argument : data to be sent through the socket (our GET request) .
    - And third argument : Length of this data as third argument.
    - And the fourth argument : flag its default value = 0.
- Receives data from the server
  - receive bytes from server until number of bytes returned = -1 so there is no more data by recv() function which takes four arguments :
    - socket descriptor as first argument.
    - Second argument : buffer to receive data from serve.
    - And third argument : Length of this buffer as third argument.
    - And the fourth argument : flag its default value = 0.
    - And return number of bytes received.
- Close the connection
  - By close() which takes file descriptor.
- Then write data received to file .

end while

Class Client :

Data Structures used :

- Arrays.
- Vectors.

Functions :

- split(string) : splits the string over space and return vector of strings.
- send_request() :
  - open a socket in the AF_INET domain.
  - Connect to the server.
  - Send request
  - loop until the requested files received.
- Main function :
  - Read requests from file .
  - send each request.
  - Take the received file and save it in the client directory.

Multi-threaded-Web-Server-and-Web-Client

HTTP web (Server and Client) in C++

First run the Server by :

```
./server.out portnumber
```

To add requests to the server add it to requests.txt

Then run the Client by :

```
./client.out server_ip (eg: 127.0.0.1) portnum
```