Programming Assignment 1

Introduction to Socket Programming in C/C++

Names :

Mohamed Ahmed Abd-Twab (57)
Mohamed Samir shaaban (60)
Mohamed Mahmoud Adel (79)

Problem Statement:

- we are required to implement client communicate with server using Persistent HTTP requests (TCP connection) and provide GET and POST requests .

- we are required to implement TXT , HTML , image , video(optional) in HTTP requests.

- we are required to implement application in C/C++.

<u>How to run code using commands :</u>

- first define directory in client path called files and create text file called input.txt contains all requests :

  - GET file_name host_name port_number

- Go to server path and type :

```
g++ server.cpp -o s.out
sudo ./s.out
```

- Go to client path and type :

```
g++ client.cpp -o c.out
./c.out localhost 80
```

- you can see requests and responses in terminal or can see in each directory.

<u>Code Organization :</u>

<u>client side:</u>

- open input file and read all requests in string array then close it .
- call **open_connection()** function to open connection with server via certain socket .
- loop on requests array and serve request by request.
- if request is **GET :**
  - send request include get method , file name , version , host name , connection to be keep alive .
  - Wait response from server
    - ➢ if response is 404 so no file and go to next connection.
    - ➢ If response is 200 OK so parse data from response and create file by this data using **store_to_directory** function.
- if requests is **POST** :
  - first try to open file in client if file is not exit print error and go to next request.
  - If file exit send request include post method , file name , version , host name , connection to be keep alive
  - Wait **OK** response from server .
  - If receive OK Send data to server.
  - Go to next request.
- each time send and receive on same socket no need to initialize new one.

server side:

- first initialize socket to listen and bind on it called sockfd.
- call function to wait and accept connection on this socket and initialize new socket to send and receive data on it.
- after connection waiting to receive requests from client .
- if receive **GET** requests :
  - call function **parse_get** : Try to open file mention in this request if if file does not exit send 404
    then wait for another requests from client
  - file exit get data from it and send 202 OK and read data from file then append it to response after finish send response to client and wait for another requests from client.

- if receive **POST** requests :
  - call function **parse_post** :
    - send OK response to client and wait to receive data from client.
    - Receive data from client and create new file with same name mentioned in request body by same data received.
    - Finish and wait for another requests.
- No need to establish connection each time with client , we open connection with client once and send , receive on it
- if connection close for any reason server wait to reconnect with any coming connection from same client or another one
- if server finish all requests from client close connection from this client then wait for reconnect from same client or another one.

Notes:

- server serve one client each time : open connection with one client and serve all requests from it then close this connection and wait for connection from another client or same one.

- we create 3 folder in client and server each folder contain certain extension TXT , image and html , so if send request with file name only expect to see it in related extension folder .

- if define path the folder of path should by in related extension if post or get requests.

- we open connection in client with host name and port from terminal commands.

major functions

client side :

- void store_to_dirctory(string response , string file_name) :
    * parse response and get data from it
    * open file and store data in it.
- string open_file_get_data(string file_name) :
    * open file and get data from it
    * include data in HTTP message.

server side :

- void Parse_Post(string request , int new_fd):
    * parse post request and send ok and receive data then store it.
- void Parse_Get(string request , int new_fd):
    * parse get requests and send data to client.

both sides :

- const vector<string> explode(const string& s, const char& c)
    * using in split
- int sendall(int sockfd, char* buf, int *len):
    * to handle big files in send HTTP so try make partial send by sending partial data using send function.
- string recv_timeout(int sockfd , int timeout):
    * to handle receive big HTTP so try to receive data in multiple chunks by checking by checking socket and use recv function once find data to be received on socket using while loop to be sure we received all data .

data structures:

- ordinary data structures like :

  vector<string> to split in it.
  Streaming to read and write in filename
  fcntl : to non-blocking socket