Programming Assignment 1
Introduction to Socket Programming in C/C++

Name : Toka Alaa ahmed
ID: 14

- **Code overall organization:**

server:

my_server: contains the main function which check if the command to run the server is correct then create object of server class and call its start_server(port) function.

Server: contains the following:

- start_server:
  1- creates new socket and bind it.
  2- listens on the socket.
  3- accepts an incoming connection on a bound socket.
  4- create new server_thread(newsockfd, new_timeout) object and call start_thread() from it.

- Vector of working server_thread objects

Server_thread: contains the following:

- start_thread: creates new thread to execute serveClient()

- serveClient:
  1-initialize buffer with MAX_BUF_SIZE.
  2- call recv() and print the message received.

client:

my_client: contains the main function which check if the command to run the server is correct then create object of Client class and call its start_client(port) function.

Client: contains the following:

- start_client:
  1-parse the commands in the input file.
  2-if connection is not open with the server then create new socket and connect to server.
  3- sen requests messages to the server.

- socketisOpen: check if there is an opened connection with the server. If exists return key = IP + PortNo.

- create_Socket:
  1- create new socket.
  2- connect to the server.
  3- add “key” to “connections” vector.

- send_request: send http request msg to the server.

Parser: contains the following:

- parse_commands(file):
  1- check if file path exists.
  2- read commands from the file and convert it to requests objects.

- get_request: return vector of requests objects.

◆ Data structure used:

Request class:

- contains the following data:
  string type = GET or POST
  string protocol = http version
  string url = url path
  string serverPort = server port no
  map<string, string> headers = headers lines
  string data

-contains the following functions:

```
string getData();
void setData(string data);
map<string, string> getHeaders();
bool hasHeader(string key);
void setHeaders(map<string, string> headers);
void addHeader(string key, string value);
string getProtocol();
void setProtocol(string protocol);
string getType();
void setType(string type);
string getUrl();
void setUrl(string url);
string get_request();
string getHeaderByKey(string key);
void setPort(string serverPort);
string getPort();
```

- Connections vector: vector<string> of key = “IP:PortNo.” Of opening connections.

Sample run:

client:

```
Connection is Open at 127.0.0.1:5000
toka@toka-Aspire-E1-570:~/Downloads/my_client/bin/Debug$ ./Network1 localhost 50
01
Enter Input File path to read commands:
/home/toka/Downloads/my_client/commands
new Connection is established 127.0.0.1:5000
Connection is Open at 127.0.0.1:5000
Connection is Open at 127.0.0.1:5000
toka@toka-Aspire-E1-570:~/Downloads/my_client/bin/Debug$
```

server:

```
toka@toka-Aspire-E1-570:~/Downloads/my_server/bin/Debug$ ./my_server 5000
server is running on port: 5000
GET /abc.html HTTP/1.1
host : localhost

GET /index.html HTTP/1.1
host : localhost

GET /pic.jpg HTTP/1.1
host : localhost
```

commands file:

commands

```
GET /abc.html localhost 5000
GET /index.html localhost 5000
GET /pic.jpg localhost 5000
```