Report Lab 2 – by Aabhaas Gupta (1215181812)

Methodology:

I had certain things in mind before\while making the program:

- I wanted the program to be procedural and NOT object oriented. I find procedural flow to be a more programmatically-natural flow, and object-oriented flow to be a more humanly-natural flow.
- I wanted to make the request a sort of an object (which was obviously mentioned in the report).
- I wanted to have simple functions which would take a request as a parameter and then serve the request. For example, openFile(request), readFile(request), closeFile(request) etc.
- I wanted the program to be modular. Because of which the limitations in my current program can be overcome by simply editing as less as one function.
- I believe that presentation plays a big role, and so I have tried to make the clients and servers humanly understandable on the terminal.
- I wanted to go extremely low level with the help of C, specifically with pointers. (Also, I personally like scanf and printf more than cin and cout!!).
- I made a hyphen-separated string from the request object at the time of sending it to the server from the client, and called this serialization. Similarly, at server side, I converted this hyphen-separated string back into request object, and called it deserialization.
- I have altered the request structure from what it previously was by adding some fields to it. For example, operation type, file name etc.
- I would have personally preferred using a SQL-Connector and SQL instead of reading and writing to space separated text files. This is only for the lock table and client table and structures which are used internally.
- I wanted to initially use multithreading and a heartbeat system which could detect a client failure and clean up the tables and open files (if any) on the server.

Structures used in the program are:

- Request Structure (struct request)
- Locktable File (space separated)
- Clienttable File (space separated)
- Hyphen-separated strings (for passing between the client and server)

Limitations:

There are a 2 limitations and 3 restrictions to the program.

The **limitations** are:

- A file cannot be opened by two clients, even for reading.
- I could not implement a system where the client waits for sometime and if the response is not received then query the server again. Because of this, although I had implemented the server side to randomly drop requests and only send the saved response if the request was computed but never sent, It can never be really seen in the program.

The **restrictions** are:

- Client has to have the script file named in the following manner:
  - script<client number>.txt
- The Files at the server have to be named in the following manner:
  - f<number>.txt // any number file which is less than 20 would do.
- The number of files which can be read had to be predefined in my system, so I chose 20 as the maximum number of files which can be operated on. That is, f0.txt, f1.txt to f19.txt.