Hacettepe University
Computer Science and Engineering Department

Name and Surname	**:** Necati Berk ÖZGÜR
Identity Number	**:** 21785229
Course	**:** BBM203
Experiment	**:** Programming Assignment II
Subject	**:** A Simplified Network Modeling & Simulation
Date Due	**: 25**/11/2018
Advisors	**:** R. A. Selim YILMAZ
e-mail	**:** b21785229@cs.hacettepe.edu.tr

2. Software Using Documentation

2.1. Software Usage

To run this software properly, user should provide 3 input files: one with clients and their info, one with route info between these clients and one with commands. Program processes these input files and prompts proper outputs regarding the commands given.

2.2. Provided Possibilities

-Program can provide client to client communication over other clients securely by its structure which prevents information leakage to unauthorized clients.
-Program splits given message into chunks of supported length.
-Prompts the status of the frames(which carries necessary information) to screen during transfer process.
-Logs the process to frame and also to client whenever a message moves over/to/from a client.
-User is able to see all queue status of all clients.
-User is able to see a given frame at a given queue.

2.3 Error Messages

Program gives an error when:
-There is no frame to show. (SHOW_FRAME_INFO)
    -User should enter a valid frame number.
-There is no queue to show. (SHOW_Q_INFO)
    -User should enter a valid queue.
-There is no message to send in outgoing queue of given client. (SEND <client>)
    -User should enter a valid client name.
-There is no log to show. (PRINT_LOG <client>)
    -User should enter a valid client name to see its logs.

3.Software Design Notes

3.1. Description of the program

3.1.1. Problem

A system should be designed to communicate between client of a given network. This system should provide this communication safely.

3.1.2.Solution

The ideal solution for this problem is sending necessary information directly from sender to receiver. But this is impossible since all clients are cannot communicate between. Instead, we can send this information over the other clients between sender and receiver. But in this case, some security drawbacks come up. These problems can be achieved by comparing client information with given information in data to be sent. Also this data can be covered with a kind of shell which is modeled as a **stack** structure in this assignment. These stacks should be sent and received by clients to the message **queues** (incoming and outgoing) of these clients. Queue structure is preferred to send and receive messages in correct order.

3.2. System Chart

INPUT	PROGRAMS	OUTPUT
clients.dat *routing.dat* *commands.dat* *sender/rec. #port* *chunk size*	*HUBBMNET*	*Output is prompted to screen.*

3.3. Main Data Structures

"Queue and Stack" are the key data structures of this experiment. Besides, **`malloc`** for dynamic memory allocation to prevent memory maluse and **`struct`** structrue are also used to implement various data types(logs, clients, queues, stacks, routes, layers, carried data)

3.4. Algorithm

1. Make initialization:
   1.1. Open input file(s).
      With built in C function **`fopen()`**.
   1.2 Generate proper Client Map and assign Routes accordingly.
      "**`clients`**" array stores these client info and their routes.
   1.3 Allocate memory needed to carry the message and obtain the communication task between clients.

2. Read commands.

   2.1 Read given Network Status Check commands and process each.

   -*MESSAGE<space>sender_ID<space>receiver_ID*: Initialize given message, split it into frames accordingly given chunk size. Assign sender ID and receiver ID. Allocate and initialize queues for send operation.
   -*SHOW_FRAME_INFO<space>client_ID<space>which_queue<space>frame_number*: Show requested frame, print its information.
   -S*HOW_Q_INFO<space>client_ID<space>which_queue:* Show current status of given queue.
   -*SEND<space>client_ID* : Send prepared message to its receiver.
   -*PRINT_LOG<space>client_ID*: Shows logs on this

3. Print proper outputs to screen.

4. Close files and free pointers allocated.
   `Fclose(FILE*), free(`*`ptr`*`).`

3.5. Specaial Design Properties

In this design, "***`stack`***" *and* "***`queue`***" structure is mainly used to implement data flow between clients. Also another key feature ***`send`*** function is used as a recursive function. By the recursive ***`send`*** function, code became more readable and clean.

By the ***malloc*** function which is a built in C function, datas given by user could be stored in variable sized arrays.

4. Software Testing Notes

4.1 Bugs and Software Reliability

By the boundary checking, and auto-sizing required stacks and queues user is not going to face with such errors like “segmentation fault” caused by “invalid access”.
But program is not protected against routing files which may cause infinite send request loops.

4.2 Software Extendibility and Upgradability

By modular construction(functions, type definitions etc.) of C language, the code is easily upgradable and reuseable.

REFERENCES

Pressman, 1987 (Supplied by ftp of Hacettepe University Dep. of Computer Science and Engineering)
The C Programming Language, Ritchie – Kernighan, 1988, Prentice Hall
Fundamentals of Data Structures in C, Horowitz – Sahni, 2008, Silicon Press