**PRACTICAL LAB 07 – IT4062E**

**Exercise 1.** Use TCP socket and multiplexing technique with the select() function to write a network application to encode and decode files (Ceasar algorithm). The protocol is described as followed:

* The format of the message format:

|  |  |  |
| --- | --- | --- |
| Opcode | Length | Payload |

* Opcode (1 byte): Operation code:

0: Encode

1: Decode

2: Transfer data of a file

3: Error notification

* Length (2 bytes): size of data in Payload field (number of bytes)
* Payload: Transferring data.
* If Opcode equals 0 or 1, Payload field contains the key value
* If Opcode equals 2 and Length is greater than 0, Payload field contains data of the file
* If Opcode equals 2 and Length equals 0, data transmission is completed
* If Opcode equals 3, Payload is blank (no data)
* Actions/Activities of the protocol:
* Step 1: Client sends the request message of encoding/decoding with the key value to server
* Step 2: Client sends data of the file to server. If the transmission is completed, client send the message with Opcode = 2 and Length = 0
* Step 3: Server stores file data to a temporary file. Server will execute the encoding/decoding operation as requested by client. If there is any error, send the message with Opcode = 3
* Step 4: Server send the encoded/decoded file to client. If the transmission is completed, client send the message with Opcode = 2 and Length = 0
* Step 5: Server deletes the temporary file and the encoded/decoded file
* *Note: To simplify the protocol implementation, assume that user only sends one request and waits for the results at a time.*

**Requirements:**

* Server:
* Start the server with its port number specified by command line parameters:

$./server Port\_Number (Example: $./server 5500)

* Use a shared folder to store files sent by clients. Note: users can send file with the same name without any problems.
* Client:
* Start the client with the server IP address and port specified by command line parameters:

$./client IP\_Addr Port\_Number (Example: $./client 10.0.0.1 5500)

* Has a menu to select encoding/decoding functions, then user can type the key and file name from keyboard.
* Receive the resulted file from server

***Note: Use Makefile to create the binary files with the names of files: server and client***

**Exercise 2.** Use TCP socket and multiplexing technique with the poll() function to build a networking application for logging in/out for users.

* Start the server with its port number specified by command line parameters:

$./server Port\_Number (Example: $./server 5500)

* Start the client with the server IP address and port specified by command line parameters:

$./client IP\_Addr Port\_Number (Example: $./client 10.0.0.1 5500)

* Requirements:
* Each client terminal can only login a single account
* Each account can login in multiple terminals
* If the number of failing login-attempts of an account reaches 5 times, server will lock that account
* User accounts are stored in a text file named account.txt, each user information is stored in a line (see the example)

UserID Password Status

Where Status is 0 if locked, and status is 1 if unlocked

***Note: Use Makefile to create the binary files with the names of files: server and client***