CS 39006: Networks Lab
Assignment 2: Using TCP sockets
Submission Deadline: January 19, 2024, 2:00 PM

A Caesar Cipher is a simple encryption scheme in which each letter of a plaintext file is replaced by *k* letters ahead of it in the alphabet, where k is the key of the Cipher. So if *k* = 2, then A is replaced by C, B by D, …, X by Z, Y by A, and Z by B.

In this assignment, a client will send a plaintext file and a key *k* to an encryption server using a TCP socket. The server will encrypt the file using Caesar Cipher and send it back to the client using the same TCP socket.

Your task will be to write the two programs - one program corresponding to the server process and another program corresponding to the client process. The sever will be a concurrent TCP server.

The file is a text file of ***arbitrary size*** (> 0 bytes). You can assume (no need to check) that the file will contain only alphabets (lower or uppercase) and space,

The transfer of the contents of the file works using a communication protocol as follows.

1. The client reads the filename to be sent from the user (keyboard). If the file does not exist, an error message is printed and the user is asked to enter a filename again.
2. If a correct filename is entered, the user is asked to enter the key *k*.
3. The client establishes a connection to the server using the connect() call.
4. The client first sends the key *k* to the server, and then sends the file content, in that order. Since the file can be arbitrarily long, the client cannot send the entire file in a single send() call, and sends the file in small chunks using multiple send() calls until the entire file is transferred. The chunk size used by the client is not known to the server.
5. The server reads the file contents sent by the client and copies it into a temporary file in the current directory. The file should be named as *<IP.port of client>.txt* where *IP* is the IP address of the client and *port* is the port of the client. For example, if the client has IP address 10.200.10.20 and port 15000, the file should be named *10.200.10.20.15000.txt*.
6. After the complete file is received, the server encrypts the file read and stores the encrypted file in another file named *X.enc*, where *X* is the name of the original file. For example, for the above example, the encrypted file will be stored in a file named *10.200.10.20.15000.txt.enc*.
7. The server sends the file to the client, once again in chunks. The chunk size used by the server is not known to the client.
8. After the complete file is sent, the server closes the connection.

9. The client stores the file in the same directory as the original file with the filename *Y.enc*, where *Y* is the original filename of the file. For example, if the filename was *dummy.txt*, the encrypted file is named *dummy.txt.enc*.
10. The client then prints a message stating the file is encrypted, giving the filenames of the original and the encrypted file.
11. The whole process then repeats, i.e., the client waits for the next filename to be encrypted from the user.

You need to ensure the following in your code:

1. No buffer of size > 100 can be used in either client or server.
2. The client cannot make more than one pass over the file, nor can it find the size of the file in any other way. Same goes for the server. More strictly, you cannot use the size of the file anywhere in your code.
3. You cannot use fopen/fscanf/fprintf functions. You must use open/read/write functions to read/write from/to the file.

Submission Instruction:

You should write two C programs – file_server.c (contains the server program) and file_client.c (contains the client program). Keep these two files in a single compressed folder (zip or tar.gz) having the name <roll number>_Assignment2.zip or <roll number>_Assignment2.tar.gz. Upload this compressed folder at Moodle course page by the deadline.