# **CS241 Lab 3**

# **Winter 2020**

# **file I/O**

**Objective:**

This lab is designed to help students gain hands-on experience with file I/O and command line input. The use of **diff** utility will also be introduced.

**Description**

You are going to implemented an improved Caesar cipher for better security in this lab. Caesar cipher is an old encryption technique in which each letter in the original text is “shifted” a fixed number of places down to the alphabet. Here, the fixed number is referred to as the **key** for the cipher. For example, if the key is 4, then the mapping between the plain text and cipher text alphabets will be as follows:

**Plain Text:** A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

**Cipher Text:** E F G H I J K L M N O P Q R S T U V W X Y Z A B C D

Caesar cipher can be automatically broken by using the known letter frequencies of English or simply trying out all of the 25 keys (brute force). A simple yet effective way of improvement over the Caesar cipher is to use several numbers, instead of one, as the key. With multiple numbers as the key, different shifts can be made in the plain text at different positions. For example, we could shift the first input character by 5, the second by 14, the third by 17, and the fourth by 10. Then we repeat this pattern: shifting the fifth input character by 5, the sixth by 14, and so on, until we run out of characters in the plain text. Apparently, the more numbers the key has, the more difficult it is to break the cipher.

In the above example, the four numbers are 5, 14, 17, and 10, which are the numerical equivalents of the string ``**FORK**'' in a 26-letter alphabet consisting of the letters A-Z. In this lab, you will modify a Caesar cipher program that uses one number as its key and make it use four numbers, given as a string of four characters, like “FORK”, as its key.

To complete this lab, first, download and modify the given program to allow a string of four letters, instead of one integer number, as a command line argument for the key. Then, convert the input string into a group of four integer numbers and store them in an integer array. Finally, declare an integer variable, say *n*, to keep track of the number of characters to be processed, and use *n* % 4 to determine the index of the number to be used for encryption/decryption. In addition, you may find a way to make your program scalable; that is, allowing the key to be extended to any length. You may replace 4 in the code with the length of the input string and create a dynamically allocated array to hold the corresponding integer numbers for the key.

More specifically, what you need to do are as follows:

* Make a directory for this lab. For example: **mkdir /cis241/lab3**. And then go to this folder.
* Use the command below to download the source code file **cipher.c**, **makefile** and data file **data.txt** from the instructor’s web page to your current directory.

wget [http://cis.gvsu.edu/~wangx/teaching/cis241/w20/lab03/{cipher.c,makefile,data.txt}](http://cis.gvsu.edu/~wangx/teaching/cis241/w20/lab03/%7bcipher.c,makefile,data.txt%7d)

* Run the make command to build an executable and run it automatically. Check and see if it works as expected.

make

make test

diff -s data.txt data.bak

* Open **cipher.c** in a text editor, read the code carefully (make sure you fully understand it), and modify it as described above.
* Open **makefile** in a text editor and modify the **test** section accordingly. In addition, add the following two sections:
  + **compare:** compare the original plaintext data with the recovered (decrypted) plaintext data.
  + **clean:** remove unused files.
* Run the make command to build an executable and run it automatically. Check and see if it works as expected.

Show your results, source code to your instructor when you have done all the above successfully. If you cannot demo in class, please turn in your code, including every file, to Blackboard by due time.

**The following may be helpful:**

* The function strlen(str) in <string.h> library will return the length of string str. For example:

len = strlen (str); // if str = “FORKAB”, then len = 6

* The following functions are defined in **ctype.h** file:

**int isupper (int x);** /\* returns non-zero integer if x is an upper-case letter, 0 otherwise \*/

**int islower(int x);** /\* returns non-zero integer if x is a lower case letter, 0 otherwise \*/