Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Operating Systems
Lab #3: Beginning C Programming
Session date: August 29th, 2018
Due date:September 4th, 2018

Goals: To program, compile and execute simple programs in C.
Suggested Reading:
The C programming Language by Brian W. Kernighan and Dennis M. Ritchie. Available at https://www.safaribooksonline.com/library/view/the-c-programming/9780133086249/ Learn C - interactive tutorial at https://www.learn-c.org/
Description:
5. Describe the difference between passing a parameter to a function by reference and passing it by value. (10 points)
When you pass a parameter as a value, the function creates a local copy of the variable, so you have 2 variables with the same data. But if you change the value from the caller, it will not be accessible from the callee, and if in the callee it modifies the value and you want to use the updated value from the caller, you need to manually assign the new value.
When you pass a parameter by reference it uses the same variable, it targets the same space in memory, so if it is modified either by caller or callee, the change will be visible to both.
Passing parameters by reference is better for performance, as it does not has to allocate new resources, but you need to have a better control of how info is modified on that variable.

User manual

To be able to use this programs in a Debian distribution, you need to previously have installed a C Compiler, e.g. gcc.
After that execute the following command, substituting the values between <> for each case.
$gcc <program_name.c> -o <executable_name>

program1_a01370139_a01373264.c
This program receives a number n as a command line argument and that sums all numbers between 0 and n, it print partial results.
$./program1_a01370139_a01373264 4

program2_a01370139_a01373264.c
This program displays the contents of the file given as an argument, stopping every 20 lines until a key is pressed.
$./program2_a01370139_a01373264 42.txt
program34_a01370139_a01373264.c

This program prints the lines of a file which contain a word given as the program argument. It can take “count” as an extra option to count the number of times the word appears and “invert” to print the results that do not contain the word given.
$./program34_a01370139_a01373264 42.txt answer
$./program34_a01370139_a01373264 42.txt answer count
$./program34_a01370139_a01373264 42.txt answer invert

Developer manual

First program

The first program is pretty easy to explain, basically we have to add every number between 0 and a number *n* given by the user, the team proposes a recursive function solution in which its base case will be whenever we reach 0 since this program starts in *n* and subtracts 1 every time it enters again to the function. In the main we can see that, in order for it to work it’ll need to receive a number from the command line, if more arguments are given are no number is entered, the code won’t execute and will alert the user.

Second program

In the main function of this code we can appreciate that it only needs the name of the file to be read, if it does not receive an argument or it receives more than expected won’t work and will alert the user. The first thing this program does is to check if the file name given by the user can be opened and if it contains elements. When we receive the file to be read, the program opens it and has a line counter that waits until it reaches 20 to act, whenever it reaches 20 the counter resets and the program waits for the user to press *enter* for the program to continue its execution and print more lines.

Third and fourth program

We decided to encapsulate both problems on a single code, and basically this depends on how many arguments are written on the command line, if it receives three arguments it means that it will execute the basic function which consists on making a simple grep out of a file, meaning that the command line will expect that the user gives a file to make the grep to and a word to be searched on this file. If the user enters another argument on the command line, the code expects that this argument is either *count* or *invert*. The count argument functions as the simple grep does, it searches on the lines of the file given a matching element that is similar to the word given by the user and then prints the lines that have the coincidence, with the added function that it tells you how many times it had a match throughout the file. And if the code receives *invert* as an argument, it will print every line that does not have a matching element with the word given by the user.