***Argumentos de la línea de comandos***

Los argumentos de la línea de comandos es algo nuevo para nosotros. Tal como su nombre lo dice: es darle la facultad a un archivo ejecutable, evidentemente ya compilado; que, justamente antes de ponerlo en marcha en nuestra línea de comandos *(./file)*, se le pase un argumento. De ahí el nombre: “Argumentos de la línea de comandos” (similar a cuando le pasamos argumentos a comandos de la Shell).

Esto se hace, en principio, si se le pasa un argumento inicial a la función principal de tipo “main” de nuestro archivo originario, que aún resulta ser de tipo .c, y no se deja “vacia” *(void)* al ser declarada como: int main (void).

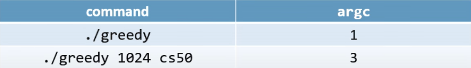
Entonces, dicho lo anterior, no podemos usar “int main(void)” si queremos recopilar otros datos en la línea de comandos cuando el usuario ejecute el programa.

Para recopilar estos argumentos de la línea de comandos del usuario, cambie su declaración de “main” para que se vea así: *int main****(int argc, string argv[])*** *{ ... (es preciso recalcar que dejamos la apertura del corchete). Hemos pasado entonces dos parametros, argumentos o entradas a “main” para poder trabajar con“argumentos de la línea de comandos”:*

***1. int argc y 2. string argv[] (un entero llamado: “argc” y una matriz de cadenas llamada: “argv”, respectivamente).***

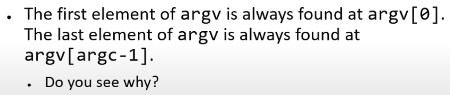
*Estos dos argumentos,* ***argc y argv****, le permiten saber qué datos el usuario ha proporcionado en la línea de comandos (en caso que éste desee pasarle un argumento, o varios, al fichero ejecutable justo en el momento antes de ser ejecutado: “./file argument1 argument2...”).* ***Entonces, argc y argv, le dan la faculad al usuario de poner a correr datos extras en la línea de comando (adicional al archivo ejecutable propiamente), los interpreta y ejecuta: los argumentos especificados en cuestión.***

*Propiamente,* ***argc,*** *como número entero que es, lo que hace es contar o almacenar el número de argumentos de la línea de comandos que el usuario escribió cuando el programa fue ejecutado.*



***nota: Por cada espacio, cualquier texto agregado al archivo ejecutable será contado como un argumento adicional.***

*Por otro lado,* ***argv,*** *es el argumento vector. Propiamente un vector o matriz de cadenas, almacena cadenas porque está declarado de la siguiente forma: “string argv[]”. Entonces, lo que hace este argumento es, primero, señalar cuántos elementos (cadenas de texto) tendrá su programa (que también es un argumento, una cadena de texto en sí mismo, la cadena de texto: “./greedy”).* ***Dentro de [] se señala una cadena de texto por elemento (contando desde 0, donde 0 equivale al elemento o cadena de texto número 1),******que se refiere a los elementos o argumentos (en últimas cadenas de texto) que el usuario realmente ha escrito en la línea de comandos cuando se ejecutó el programa; por ejemplo, “./greedy 1024 cs50”. Entonces, bajo ese mismo orden de ideas, el argumento “argv” quedaría así: string argv [2]:*** *serían tres cadena de texto y que no lo confunda la cadena de texto “1024”, no es un número entero, es una cadena de texto conformada por cuatro caracteres porque así fue declarada inicialmente con el argumento “string argv []”, correspondería entonces a la cadena de texto: argv [1].*



*Estos son nombramientos convencionales, pero no necesariamente deben llamarse así: “argc o argv”. Sólo se les ha pasado ese nombre para darle un nombre de variable a los tipos de datos fundamentales que permiten pasarle argumentos extras a un archivo ejecutable: el número entero y la matriz de cadena.*

***HAY UNA MEJOR EXPLICACIÓN DE ESTE TEMA, YA POR MEDIO DE UN EJERCICIO, LLAMADO: “9. PROBLEM SET B. cifrado de julio cesar.c” en la carpeta de Harvard/Week 2, Arrays/Ejercicios Harvard.***

**dato de interés: hay una función que permite convertir números en cadenas de textos.**

**dato de interés: ejecutando la siguiente línea de comando: “echo $?” podemos saber que valor devuelve *return.***