|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | García Morales Karina |
| *Asignatura:* | Fundamentos de programación |
| *Grupo:* | 1121 |
| *No de Práctica(s):* | Práctica no. 6 |
| *Integrante(s):* | Espinosa Carrillo Lis Andrés |
|  |  |
| *No. de Equipo de cómputo empleado* | Equipo no. 36 |
| *Semestre:* | 2019-1 |
| *Fecha de entrega:* | 3 de Octubre de 2018 |
| *Obervaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Guía práctica de estudio 06:**

**Entorno de C (editores, compilación y ejecución)**

**Objetivo:**

Conocer y usar los ambientes y herramientas para el desarrollo y ejecución de programas en Lenguaje C, como editores y compiladores en diversos sistemas operativos.

**Desarrollo:**

Un lenguaje de programación nos permite expresar una serie de instrucciones que podrán ser ejecutadas por una computadora. El lenguaje que utilizaremos es el lenguaje de programación C.

Utilizamos este lenguaje de programación ya que su forma de dar las instrucciones es bien parecida a lo que un ser humano puede abstraer, esto hace que el lenguaje de programación C sea un lenguaje de alto nivel, esto se significa que las instrucciones pueden ser entendidas fácilmente por el ser humano.

**Editores de C**

Un programa en C debe ser escrito en un editor de texto para después generar un programa ejecutable en la computadora por medio de un compilador, no es lo mismo un editor de texto que un procesador de texto.

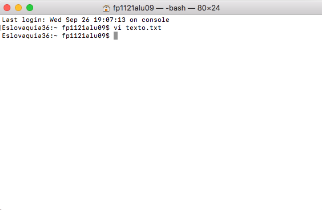
Un editor de texto edita un texto plano que puede tener muchas utilidades como guardar una configuración, tener escrito un programa, etc., y será interpretado hasta que se haga una lectura de éste.

Un procesador de texto permite dar formato al texto, a la hoja donde está escrito, incrustar imágenes, etc., su salida puede ser un archivo de texto plano que contiene etiquetas que señalan el formato que se le dio al texto o algo un poco más complejo.

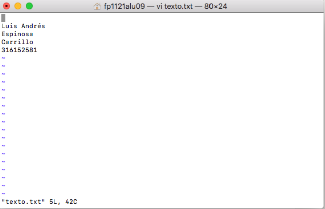
**Editor Visual Interface de GNU/Linux (VI)**

Este es el editor más común en cualquier distribución se sistemas operativos con núcleo basado en UNIX.

Comenzamos creando un archivo con el comando ***vi texto.txt***



Luego seguimos a editar el archivo metiéndonos a él y con la letra ***i*** comenzamos a meter texto al archivo, en el cuál introducimos nuestros nombres, apellidos y números de cuenta.



**Modo comando**

Es el modo por defecto de VI cuando se abre. Las teclas presionadas ejecutan diversas acciones predeterminadas y no se puede editar el texto libremente. Los comandos son sensitivos a las mayúsculas y a las minúsculas.

Después de haber editado el archivo, presionamos la tecla ***esc*** para poder probar los siguientes comandos.

* **↑** o k mueve el cursor hacia arriba.
* **↓** o j mueve el cursor hacia abajo.
* ← o h mueve el cursor hacia la izquierda.
* → o l mueve el cursor hacia la derecha.
* **1G** lleva el cursor al comienzo de la primera línea.
* **G** lleva el cursor al comienzo de la última línea.
* **x** borra el carácter marcado por el cursor.
* **dd** borra o corta la línea donde está el cursor.
* **ndd** donde n es la cantidad de líneas que se borrarán o cortarán después del cursor.
* **D** borra o corta desde la posición de cursor hasta el final de la línea.
* **dw** borra o corta desde la posición del cursor hasta el final de una palabra.
* **yy** copia la línea donde está el cursor.
* **p** pega un contenido copiado o borrado.
* **u** deshace el último cambio.

**Modo de última línea**

Se puede acceder a él desde el modo de última línea. Es muy similar al modo comando, pero los comandos no tendrán efecto hasta que se presiona la tecla ***Enter*** además de que se visualizará el comando en la última línea del editor. Es posible cancelar el comando con la tecla Esc. Los comandos de última línea se caracterizan porque inician con /, ? o :.

* **/texto** donde la cadena texto será buscada hacia delante de donde se encuentra el cursor.
* **?texto** donde la cadena texto será buscada hacia atrás de donde se encuentra el cursor.
* **:q** para salir de VI sin haber editado el texto desde la última vez que se guardó.
* **:q!** para salir de VI sin guardar los cambios.
* **:w** para guardar los cambios sin salir de VI.
* **:w** archivo para realizar la orden “guardar como”, siendo archivo el nombre donde se guardará el documento.
* **:wq** guarda los cambios y sale de VI.

**Modo insertar**

Este modo permite insertar texto. Las teclas presionadas ya no harán una acción como en el modo comando sino será el contenido que formará el texto del documento.

Para ingresar al modo insertar existen varios comandos:

* **i** pasa al modo insertar poniendo el texto a la izquierda del cursor.
* **a** pasa al modo insertar poniendo el texto a la derecha del cursor.
* **A** pasa al modo insertar colocando el texto al final de la línea donde el cursor se encuentra.
* **I** pasa al modo insertar colocando el texto al principio de la línea donde el cursor se encuentra.
* **O** coloca una línea arriba de la línea seleccionada por el cursor y pasa al modo insertar.
* **o** coloca una línea debajo de la seleccionada por el cursor y pasa al modo insertar.

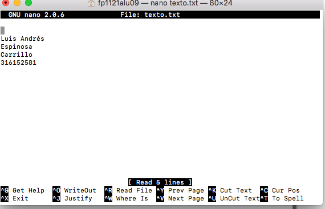
Para salir del modo insertar debe presionarse la tecla ***Esc***. Para verificar que se encuentra en modo insertar es se puede ver -- insertar -- en la última línea del editor.

**GNU NANO**

Es un editor de texto disponible para sistemas operativos basados en UNIX en línea de comandos. Se puede acceder en un entorno gráfico desde la aplicación de terminal. Este editor es mucho más intuitivo que VI, aunque menos potente.

Utilizamos el editor GNU NANO y los siguientes comandos:

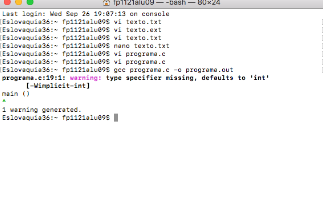
* **F1** es posible visualizar la ayuda con la lista de todos comandos que existen.
* **^** que es la tecla **Ctrl.**
* **M-** que es la tecla **Esc** o bien **Al**



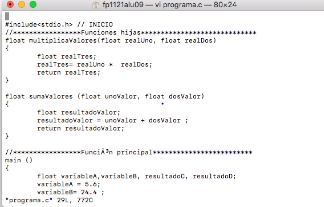
**GEDIT**

Es un editor de texto de entorno gráfico para sistemas basados en UNIX; sin embargo, no pudimos utilizar este editor ya que no lo teníamos en nuestros equipos.

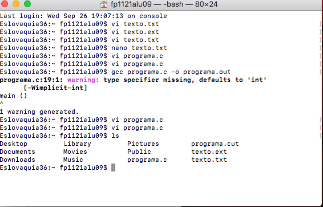
Después copiamos un programa a la terminal y verificamos que se compilara correctamente. Al principio no compiló porque le faltaba la palabra int.



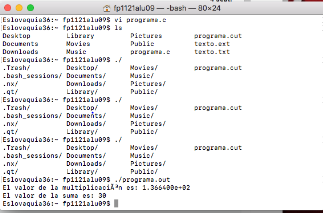
Sin embargo, después de agregársela, el programa pudo ser compilado sin ningún problema.



Utilizamos el comando ***ls*** para verificar que realmente nuestros archivos fueron creados.



Guardamos el programa que habíamos copiado anteriormente con el cambio que le hicimos en un formato **.out** para después ejecutarlo y comprobar que funcionaba



**Compiladores**

Una vez codificado un programa en C en algún editor de texto, éste debe ser leído por un programa que produzca un archivo ejecutable. A este programa se le conoce como compilador y depende totalmente del hardware de la computadora y el sistema operativo que corre sobre ella.

Es muy común cometer algún error al elaborar un programa en C como son faltas a la sintaxis que indica el estándar, usar elementos que no se habían declarado, utilizar funciones de una biblioteca sin haberla especificado, entre muchos otros que se irán conociendo en un futuro. La mayoría de estos errores provocan que el compilador no pueda generar el programa ejecutable y muestra en la línea de comandos de qué error se trata y en qué línea pudo haberse producido.

**GCC (GNU Compiler Collection)**

Es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX. Entre sus compiladores existe el que sirve para programas escritos en C. Se encuentra por defecto en diversas distribuciones de Linux. El compilador trabaja en línea de comandos.

***gcc calculadora.c -o calculadora.out***

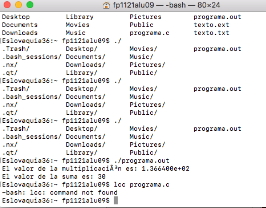
Esto creará un archivo a.out (en Windows a.exe) que es el programa ejecutable resultado de la compilación. Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro -o de gcc, por ejemplo, para que se llame calculadora.out (en Windows calculadora.exe)

Gracias a este compilador fue como pudimos compilar el programa anterior.

**LCC**

Es un compilador similar a GCC de uso libre diseñado para ejecutarse en sistemas operativos Windows, sean de 64 bits o 32 bits.

Sin embargo, intentamos compilar el programa con LCC pero no se pudo ya que no lo tenían nuestros equipos**.**



**Ejercicios de tarea**

* **Ejercicio 1: Realiza un programa que muestre la suma de los 20 primeros números pares**.

|  |  |  |
| --- | --- | --- |
| **Análisis** | **Diagrama de flujo** | **Programa** |
| **DE:** Los primeros 20 números pares.  **DS:** La suma de los primeros 20 números pares.  **RESTRICCIONES:** Sólo considerar los primeros 20 números pares. |  |  |

* **Ejercicio 2: Realizar un programa que lea x cantidad de números, hasta que su suma sea mayor a 200. Imprimir al final, la cantidad de números leídos, su suma y promedio.**

|  |  |  |
| --- | --- | --- |
| **Análisis** | **Diagrama de flujo** | **Programa** |
| **DE:** X cantidad de números.  **DS:** Imprimir cantidad de números leídos, suma y promedio.  **RESTRICCIONES:** Hasta que la suma sea mayor a 200. |  |  |

* **Ejercicio 3: Realizar un programa que obtenga la siguiente numeración y la siguiente posición. (Utiliza ciclo for).**

**1**

**232**

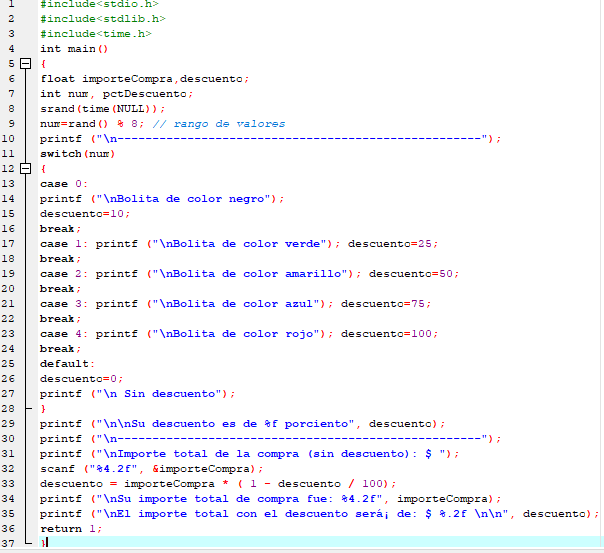
**34543**

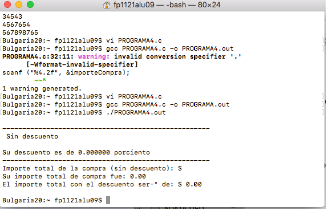
**4567654**

**567898765**

|  |  |  |
| --- | --- | --- |
| **Análisis** | **Diagrama de flujo** | **Programa** |
| **De:** La numeración dada y su orden.  **Ds:** Obtener programa que obtenga la numeración dada en ese mismo orden.  **Restricciones:** Tiene que darnos esa numeración en ese mismo orden. |  |  |

**Ejercicio 4: Describe que hace el siguiente programa, coméntalo y realiza su diagrama de flujo y pruebas de escritorio.**





Este programa sirve para generar un descuento en el producto dependiendo que bola salga al azar, cada bola tiene un cierto porcentaje de descuento.

**Diagrama de flujo**



**Conclusiones**

Durante el desarrollo de la práctica no hubo ningún problema al entender y utilizar los comandos ni tampoco al compilar y ejecutar el programa, lo único que se me dificultó fue el realizar los programas de los ejercicios de tarea para que en realidad pudieran funcionar correctamente.

**Bibliografía**

* Dr. Pedro Alberto Enríquez Palma. Editor VI. Consulta: septiembre de 2016. Disponible en: http://www.unirioja.es/cu/enriquez/docencia/Quimica/vi.pdf
* Francisconi Hugo Adrian. Nano. Consulta: septiembre de 2016. Disponible en: http://francisconi.org/linux/comandos/nano
* G2 Crowd. ATOM vs. Notepad++. Consulta: septiembre de 2016. Disponible en: https://www.g2crowd.com/compare/atom-vs-notepad
* Gerald Pfeifer (GCC team). GCC, the GNU Compiler Collection. Consulta: septiembre de 2015. Disponible en: https://gcc.gnu.org
* MinGW.org. MinGW - Minimalist GNU for Windows. Consulta: septiembre de 2015. Disponible en: http://www.mingw.org
* White-Hat Hacking. Uso de gcc bajo Linux. Consulta: septiembre de 2016. Disponible en: https://whitehathacking.wordpress.com/2010/10/31/uso-de-gcc-bajo-linux/
* Willus.org. Win32/64 C/C++ Compilers Page. Consulta: septiembre de 2016. Disponible en: http://www.willus.com/ccomp.shtml
* Fabrice Bellard. Tiny C Compiler. Consulta: septiembre de 2015. Disponible en: http://bellard.org/tcc/