



**Universidad Nacional
Autónoma de México**
Facultad de Ingeniería



Fundamentos de Programación (1122)

Laboratorios de computación
salas A y B

Profesor: M.I. Marco Antonio Martínez Quintana
Semestre 2021-1

Practica No. 6

Entorno de C

Editores, Compilación y Ejecución

Grupo: 1129

No. de Equipo de cómputo empleado: No aplica

No. de Lista o Brigada: No aplica

No. de Lista: 42

Nombre: Adolfo Román Jiménez

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.

Introducción:

Un programa puede entenderse como la forma de expresar la solución a un problema de manera que sea comprensible para el ordenador. En otras palabras, un programa es un conjunto ordenado de instrucciones que se dan a la computadora indicando el conjunto de operaciones o tareas que se desea llevar a cabo.

El lenguaje de programación C es un lenguaje de alto nivel que se puede caracterizar por los siguientes puntos:

- Es de propósito general, esto significa que puede ser usado tanto para el desarrollo de sistemas operativos como para programas científicos, programas de aplicación o programas de educación y juegos.
- Posee una alta transportabilidad ya que los programas escritos en C pueden ser llevados de un tipo de ordenador a otro y funcionarán. Los cambios que hay que realizar son mínimos.
- Como solo tiene 32 palabras reservadas se dice que es compacto.
- Los compiladores generan ejecutables más pequeños porque por norma general, los programas escritos en lenguaje C poseen menos líneas de código que otros lenguajes de programación de alto nivel.
- El lenguaje de programación C es un lenguaje imperativo o procedimental. Esto significa que indica secuencias de acciones con el fin de llegar a un objetivo. Generalmente los lenguajes imperativos o procedimentales contienen una parte declarativa. El lenguaje C da órdenes a la máquina.
- Es un lenguaje estructurado ya que los programas escritos con él se pueden organizar en módulos.

Su principal desventaja es que un programa puede estar escrito con tal brevedad y concisión que se vuelven casi ilegibles. Esto es debido a su versatilidad.

El compilador C trata las letras mayúsculas y minúsculas como caracteres diferentes.

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. Tanto el editor de texto como el compilador van de la mano con el sistema operativo y si posee o no interfaz gráfica por lo que son factores que se deben de tomar en cuenta a la hora de elegir el entorno para desarrollar programas en C.

Es importante señalar que no es lo mismo un editor de texto que un procesador de texto. El primero 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. A continuación, se presentan algunos de los editores más comunes.

Editor Visual Interface de GNU/Linux (VI)

El editor vi (visual interface) es el editor más común en cualquier distribución de sistemas operativos con núcleo basado en UNIX. Está disponible en línea de comandos y si el sistema operativo tiene entorno gráfico se puede acceder a él desde la terminal.

VI es un editor que puede resultar difícil de usar en un inicio. Aunque existen editores más intuitivos en su uso, en muchas ocasiones VI es el único disponible.

Para iniciar VI, debe tarse desde la línea de comandos:

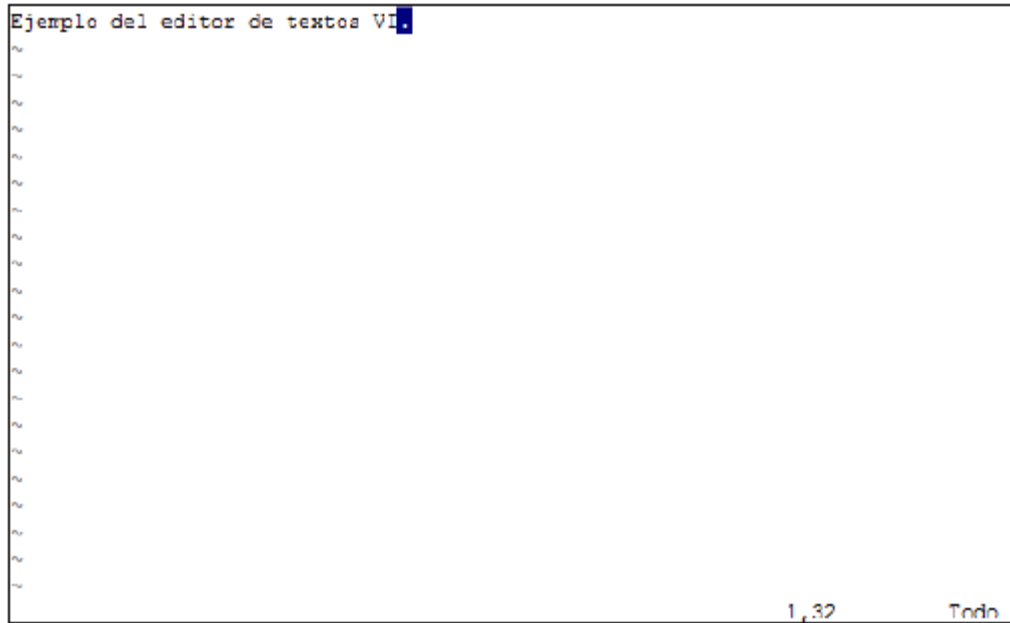
```
vi nombre_archivo[.ext]
```

Donde “nombre_archivo” puede ser el nombre del archivo a editar o el nombre de un archivo nuevo que se creará con VI. Es válido incluir la ruta donde se localiza o localizará el archivo. Existen más métodos de apertura para usuarios más avanzados. VI tiene tres modos de operación.

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. Algunos ejemplos son:

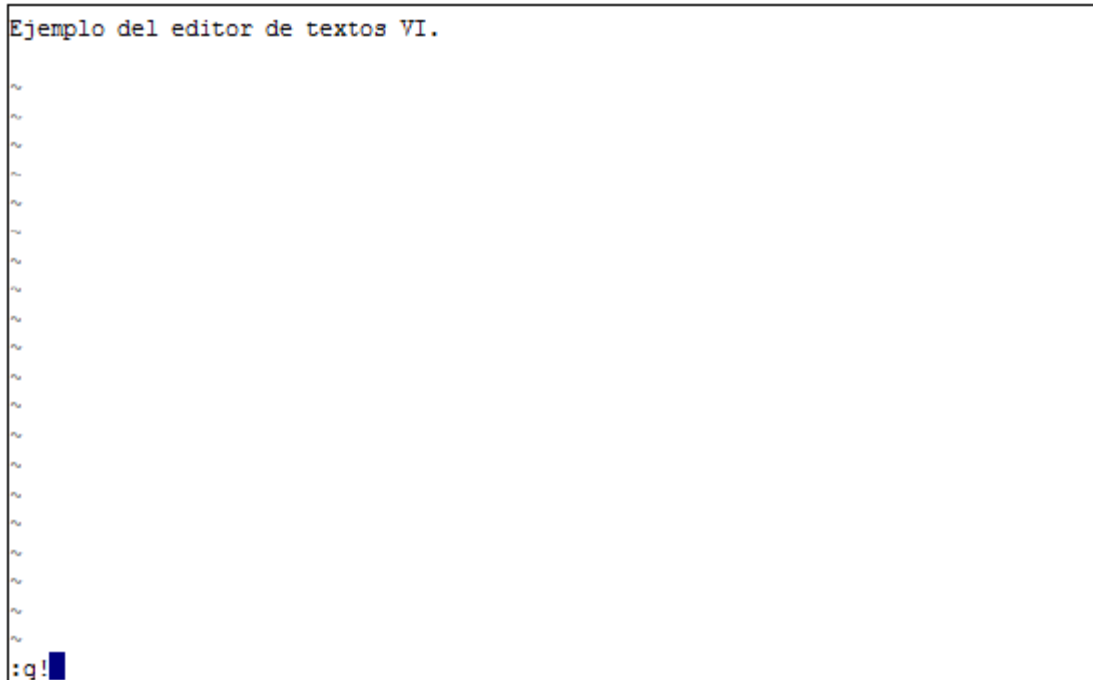
- ↑ o k mueve el cursor hacia arriba.
- ↓ o j mueve el cursor hacia abajo.
- ← o h mueve el cursor hacia la izquierda.
- → o l mueve el cursor hacia la derecha.
- 1G lleva el cursor al comienzo de la primera línea.
- G lleva el cursor al comienzo de la última línea.
- x borra el carácter marcado por 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 ultima linea

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 :. Algunos ejemplos son:

- /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. Se puede desplazar con las flechas del teclado y borrar con la tecla de retroceso o de suprimir.

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.
- coloca una línea arriba de la línea seleccionada por el cursor y pasa al modo insertar.
- 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.

Ejemplo del editor NANO.

```

^G Ver ayuda ^O Guardar ^R L Fichero ^Y PÃ¡g Ant ^K CortarTxt ^C Pos act a
^J Justificar ^W Buscar ^V PÃ¡g Sig ^U UnCut Text ^T OrtografÃ-a

nano help text

The nano editor is designed to emulate the functionality and ease-of-use of
the UW Pico text editor. There are four main sections of the editor. The top
line shows the program version, the current filename being edited, and whether
or not the file has been modified. Next is the main editor window showing the
file being edited. The status line is the third line from the bottom and
shows important messages. The bottom two lines show the most commonly used
shortcuts in the editor.

The notation for shortcuts is as follows: Control-key sequences are notated
with a caret (^) symbol and can be entered either by using the Control (Ctrl)
key or pressing the Escape (Esc) key twice. Escape-key sequences are notated
with the Meta (M) symbol and can be entered using either the Esc, Alt, or Meta
key depending on your keyboard setup. Also, pressing Esc twice and then
typing a three-digit decimal number from 000 to 255 will enter the character
with the corresponding value. The following keystrokes are available in the
main editor window. Alternative keys are shown in parentheses:

^V PÃ¡g Sig ^N LÃ-nea siguiente M-/ Ãltima lÃ-nea
```

GEDIT

Es un editor de texto de entorno gráfico para sistemas basados en UNIX y se encuentra por defecto en el entorno de GNOME.

Permite todas las funciones básicas que se pueden encontrar en la mayoría de los editores como copiar, pegar, guardar, etcétera, con las mismas combinaciones de teclas que ya son conocidas por

los usuarios. También cuenta con menús en la parte superior que permite acceder a todas las funciones del editor por lo que no requiere conocimientos avanzados para poderlo usar.

A diferencia de los editores de línea de comandos, la mayoría de los editores con interfaz gráfica tienen funciones como enumerar cada línea, sangría automática (en los lenguajes de programación se suele dejar tabulaciones antes de iniciar cada línea que ayudan a jerarquizar un programa codificado con el fin de aumentar la legibilidad y en algunos

cuantos son de vital importancia), corrector ortográfico y coloreado sintáctico (colorear los distintos componentes sintácticos de ciertos lenguajes de programación con la finalidad de facilitar el desarrollo con elementos visuales).

Notepad

Es un editor de texto plano de entorno gráfico disponible en todas las ediciones de Windows. A diferencia de GEDIT es muy limitado en funcionalidad, la cual es más parecida a un editor de línea de comandos, de hecho, es la evolución de uno de ellos, EDIT que era el editor para MS-DOS. A pesar de lo anterior, es conveniente usarlo cuando no existen herramientas adicionales para editar archivos de texto plano.

Notepad++

Es un editor de texto plano diseñado para ejecutarse en entorno gráfico con sistema operativo Windows; es de código libre. Este tiene gran variedad de funciones que ayudan a los desarrolladores escribir programas de manera eficaz, como el autocompletado, corrector ortográfico, coloreado sintáctico, edición múltiple de archivos, resaltado de paréntesis, etcétera. Soporta gran variedad de lenguajes de programación y permite instalar aditamentos para aumentar su funcionalidad.

GitHub atom

Es otro editor de texto de reciente aparición y de gran versatilidad para programadores porque tiene varias funciones además de soporte para distintos lenguajes de programación.

Tiene licencia MIT, está hecho para correr en entornos gráficos en sistemas operativos Linux, Windows y Mac OS X.

La mayoría de usuarios prefieren usar Notepad++ como su editor base a pesar de que Atom puede ser una competencia directa para éste. Atom además está diseñado para usuarios experimentados mientras que Notepad++ es tanto para iniciantes como avanzados.

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.

Recordando, un programa en C es universal, por lo que cada una de las instrucciones que lo conforman debe poder entenderlas muchos de los equipos en el mercado, aunque su naturaleza sea distinta. Por ello, un compilador depende del equipo, porque es un traductor que transforma ese lenguaje universal a un programa ejecutable que sólo puede correr ese equipo.

Un programa en C, tampoco puede ser escrito de manera arbitraria sino respetando una serie de reglas para que el compilador pueda entenderlas y realizar su función. Un estándar muy común es ANSI C y existen diferentes extensiones como ISOC99 y GNU

C que representan mejoras para el estándar original. Realizar un programa en dicho estándar garantiza que puede correr en cualquier máquina siempre y cuando exista un compilador hecho para ella. A veces, el programador no sigue un estándar o lo desconoce, usando características no estándar que, a la hora de usar el mismo programa para otra máquina no funciona teniendo que realizar adaptaciones que se reflejan en costos. Por ejemplo, es muy común empezar a desarrollar programas en C en plataforma Windows en procesadores x86 usando características propias. Al trasladar, por alguna necesidad, dicho programa a plataforma Linux con procesador ARM, el programa no funcionará porque no se siguió el estándar que garantiza universalidad para el lenguaje C.

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.

Es importante señalar que un solo error puede desencadenar muchos otros y al corregirlo los demás dejarán de ser errores. También, en muchas ocasiones el error no se encuentra en la línea que el compilador señala sino en líneas anteriores, lo que señala es la línea en que el compilador ya no encontró estructura el programa codificado ya que éste no tiene criterio propio sino se trata de un programa de computadora.

Cuando el compilador señala un error no cabe más que invocar algún editor de texto, revisar cuidadosamente el programa y corregir. Se debe verificar la coherencia total del programa para evitar tener que volver a repetir este paso de manera continua.

A veces el compilador arroja advertencias durante el proceso, se generará el archivo ejecutable, pero puede tener problemas a la hora de ejecución por lo que es mejor investigar de qué tratan o porqué se generaron.

A continuación, se presentan dos compiladores que pueden ser de utilidad.

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.

Existe también una versión modificada que puede correr y crear programas para plataformas Windows en un paquete llamado MinGW (Minimalist GNU for Windows).

Al compilar un programa en C el compilador genera diversos archivos intermedios que corresponden a las distintas fases que realiza. Éstas no son de interés por el momento

y son eliminadas una vez obtenido el archivo ejecutable. GCC tiene diferentes opciones de ejecución para usuarios más avanzados.

Suponiendo que se tiene un programa escrito en C y se le llamó *calculadora.c* la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:

```
gcc calculadora.c
```

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*):

```
gcc calculadora.c -o calculadora.out
```

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se está usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro *-l* seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

```
gcc calculadora.c -o calculadora -lnombre_libreria
```

LCC

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

Para poder hacer uso de LCC debe haber sido instalado previamente y agregado al PATH del sistema (es la ruta que sigue el sistema operativo para encontrar la ubicación de un ejecutable al ser llamado desde el símbolo de sistema). Se tiene que agregar al PATH por lo menos *lcc.exe* y *lclnk.exe* localizados por defecto en *C:\lcc\bin*.

A diferencia de GCC, la compilación consiste en dos pasos, el primero genera un archivo objeto y el segundo a partir de éste genera el programa ejecutable. Existen opciones adicionales para usuarios avanzados a la hora de invocar al compilador.

Si se tiene un programa llamado *calculadora.c* se debe establecer primero la ruta donde se encuentra el archivo y luego generar el archivo objeto *calculadora.obj* con el siguiente comando:

```
lcc calculadora.c
```

Después se tiene que generar el programa ejecutable *calculadora.exe* por medio de:

```
lcclnk calculadora.obj
```

TCC

Es un compilador de uso libre diseñado para ejecutarse en sistemas operativos Windows.

Para poder hacer uso de TCC debe haber sido instalado previamente y agregado al PATH del sistema al igual que LCC.

Se tiene también como en GCC y LCC diferentes opciones para usuarios avanzados.

Si se tiene un programa llamado *calculadora.c* se tiene que establecer primero la ruta donde se encuentra el archivo. El siguiente comando permite generar el programa ejecutable y ejecutarlo:

```
tcc -run calculadora.c
```

Otros compiladores e IDE

Un compilador simplemente es un programa que tiene como entrada un archivo de texto con el programa codificado en C y cuya salida es el programa ejecutable y la salida para marcar errores o advertencias.

Una IDE significa entorno de desarrollo integrado por sus siglas en inglés y combina un editor de textos con un compilador además de varias herramientas que facilitan la programación haciendo todo lo mencionado en esta práctica invisible para el programador. Incluso la ejecución es más sencilla desde una IDE.

Un IDE famoso diseñado para MS-DOS fue Borland Turbo C. LCC, al instalarse, incluye un IDE llamado wedit en el que basta escribir el programa en su editor y compilar y ejecutar el programa con la tecla F9.

En la siguiente lista se presentan varios compiladores e IDE para varios sistemas operativos y arquitecturas. Algunos compiladores soportan otros lenguajes además de C.

- GCC
- Embarcadero (Borland)
- Ch
- Code::Blocks,
- Cygwin
- Dev-C++
- DJGPP
- TCC
- EMX/RSXNT
- LCC para Windows de 32 o 64 bits
- Microsoft Visual C/C++
- MinGW para Windows de 32 o 64 bits
- Miracle C
- Orange C
- Pelles C
- Watcom

EJECUCIÓN

La ejecución es la etapa que sigue después de haber compilado el programa. Una vez compilado el programa, se puede distribuir para equipos que ejecuten el mismo sistema operativo y tengan la misma plataforma de hardware (tipo de procesador, set de instrucciones y arquitectura en general). Los pasos para realizar la ejecución dependen del sistema operativo y del entorno.

En Windows se puede ejecutar el programa haciendo doble clic sobre el programa ya compilado, pero recomienda exhaustivamente que se haga desde símbolo de sistema. Como el programa realizado es para línea de comandos, si se ejecuta en entorno gráfico en Windows el programa sólo se abrirá y se cerrará sin poder ver los resultados de la ejecución, aunque el programa haga sus funciones. Existen métodos para evitar que el programa se cierre, pero suelen no corresponder a ANSI C por lo que se desaconseja. Es mejor ejecutar el programa en el símbolo de sistema porque, aunque el programa finalice su ejecución, los resultados continuarán siendo visibles en la consola.

Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es *calculadora.out*, para ejecutar debe teclearse en línea de comandos:

```
./calculadora.out
```

Y en Windows, teniendo un programa llamado *calculadora.exe* debe teclearse en símbolo de sistema:

```
calculadora.exe
```

En ambos casos localizándose previamente en la ruta donde se encuentra el ejecutable. En Windows a veces puede omitirse mencionar *.exe*.

Si el programa realizado necesita tener una entrada de información por medio de argumentos, éstos se colocan así:

```
calculadora argumento1 argumento2
```

En un inicio, cualquier programa escrito en C sólo funcionará en modo línea de comandos, ya que para que tenga una interfaz gráfica se requiere de conocimientos avanzados sobre el lenguaje y del sistema operativo para el que se diseña el programa.

Es difícil realizar un programa con interfaz gráfica universal y que respete ANSI C, ya que el entorno depende del sistema operativo y las herramientas que provee.

Muchos errores no se reflejarán en el compilador porque el programa está correctamente escrito de acuerdo a lo que ANSI C señala, pero lo que se programó puede ser erróneo y tener resultados distintos a los deseados. Por ello, en la fase de ejecución deben hacerse diversas pruebas para verificar que el programa hace lo que debería.

Aunque el programa aparente funcionar, deben hacerse pruebas exhaustivas para verificar la integridad del programa. Por ejemplo, si el programa realiza una división, es necesario evaluar que el divisor no sea cero, o bien, que los números que se manejan no rebasen el valor máximo que el equipo soporta, entre muchos otros detalles que pueden presentarse.

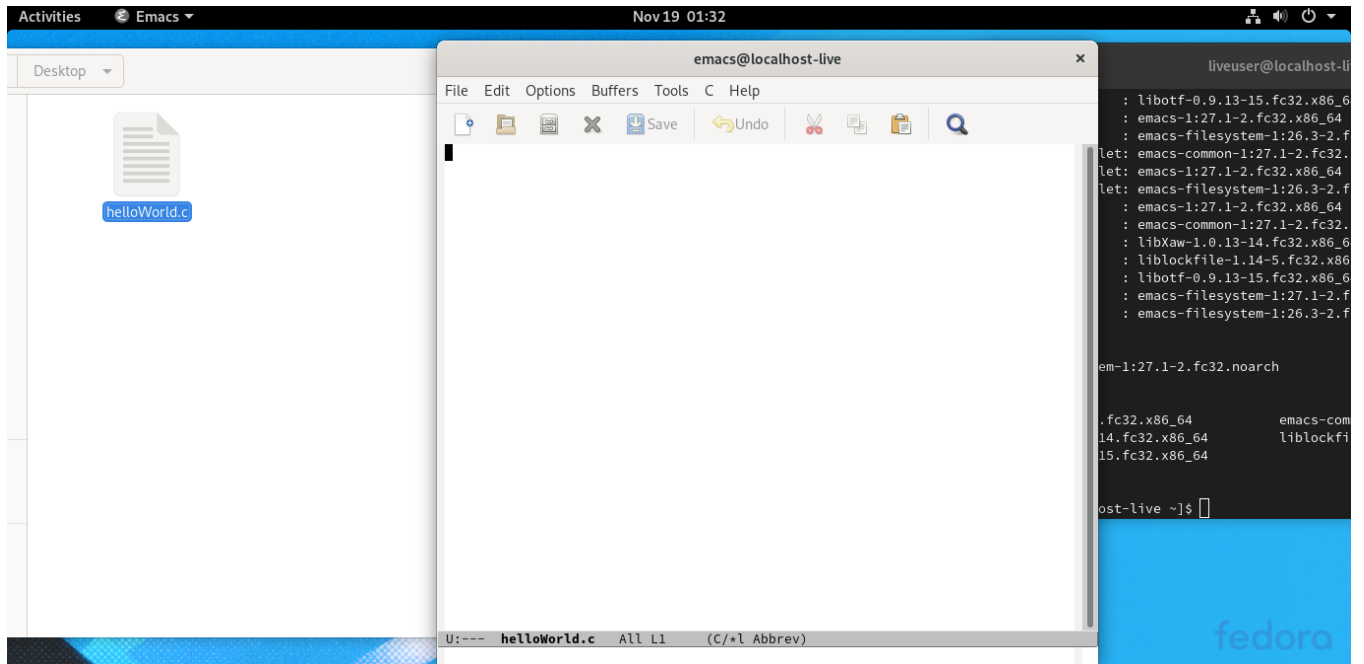
Es muy común que se construya un programa conforme a lo que ANSI C determina y se pruebe en un equipo personal que, por lo general, es de altas prestaciones. A la hora de instalar ese programa en un equipo especializado que tiene prestaciones limitadas, el programa puede no funcionar ya que no se optimizó. Por ejemplo, se crea un programa que al ejecutarse ocupa 600 MB en memoria principal en un equipo con 4 GB y se ejecutará sin problemas. No obstante, este programa tiene que correr en un equipo con 512 MB de memoria principal. El sistema operativo y otros procesos están ocupando ya una parte de esta memoria, aunque el sistema operativo haga lo posible por ejecutar el programa, será con un rendimiento pobre o simplemente el sistema se colgará.

Es muy importante tratar de que un programa sea íntegro y optimizado para garantizar su correcto funcionamiento y en ocasiones, el prestigio de su autor.

Actividades:

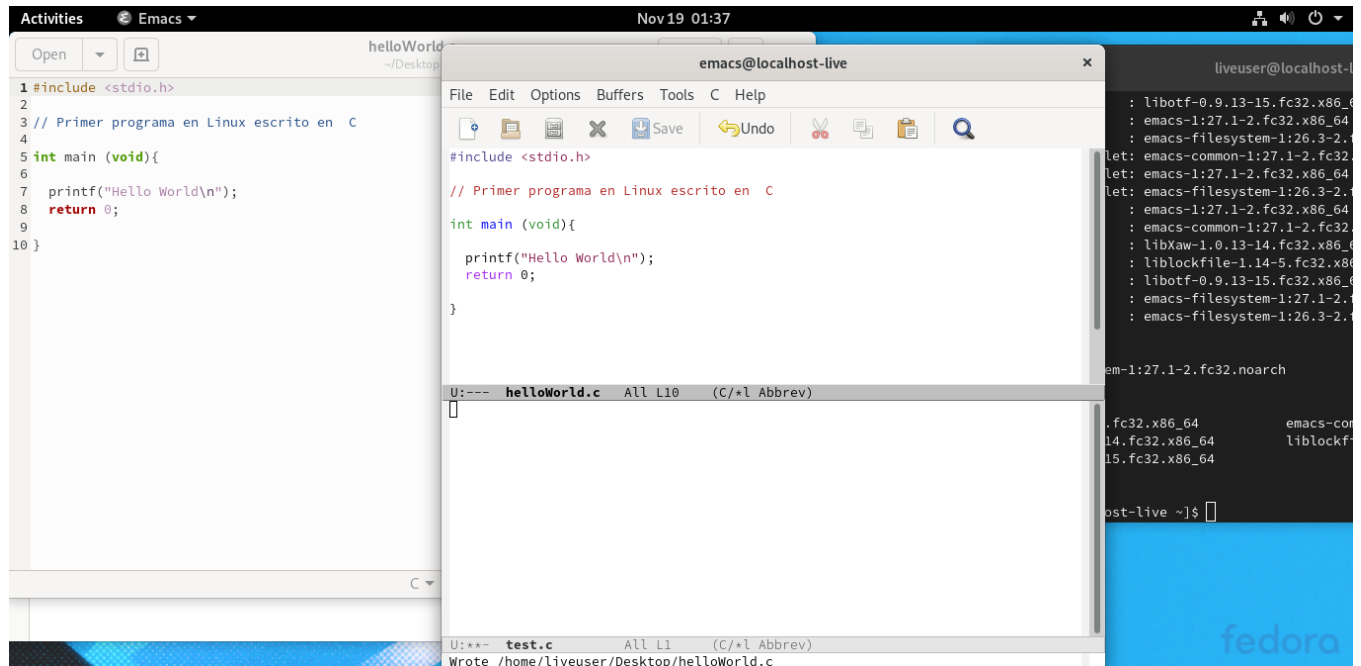
- Utilizando un editor de GNU/Linux, crear un archivo de texto

Archivo creado con Emacs llamado helloWorld.c



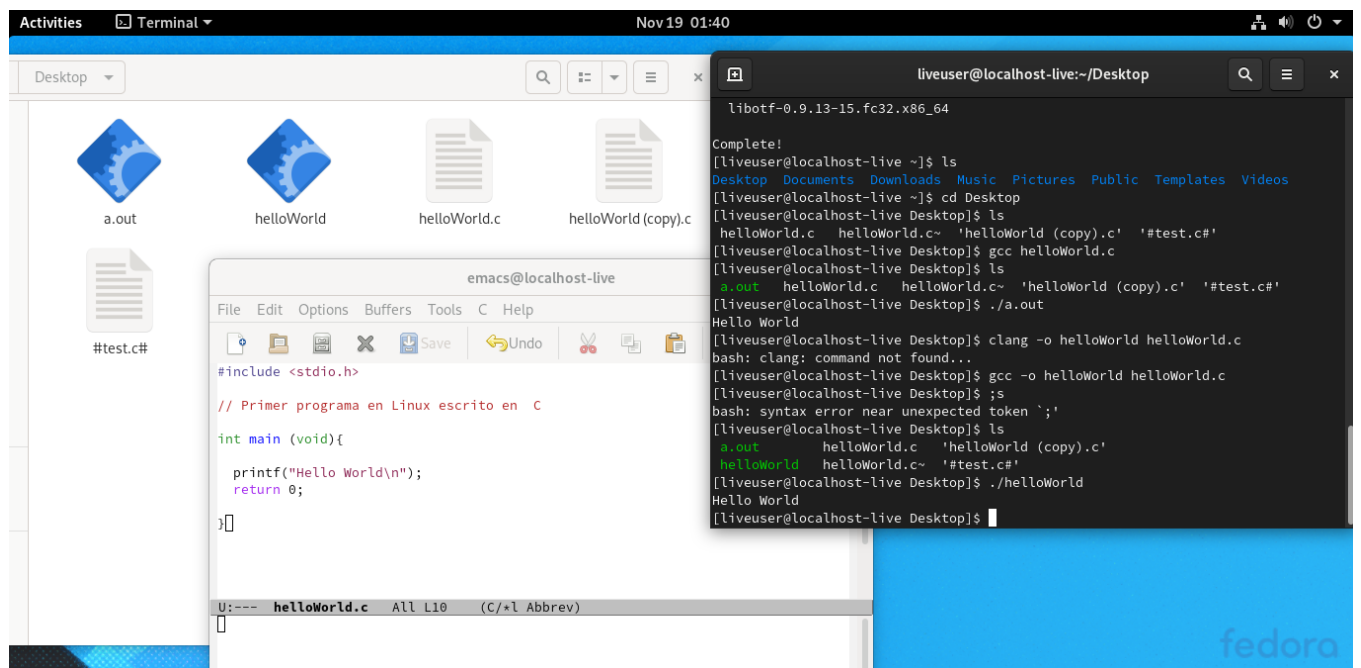
- Modificar/actualizar un archivo ya existente con un editor GNU/Linux.

Archivo helloWorld.c creado con Emacs, modificado.



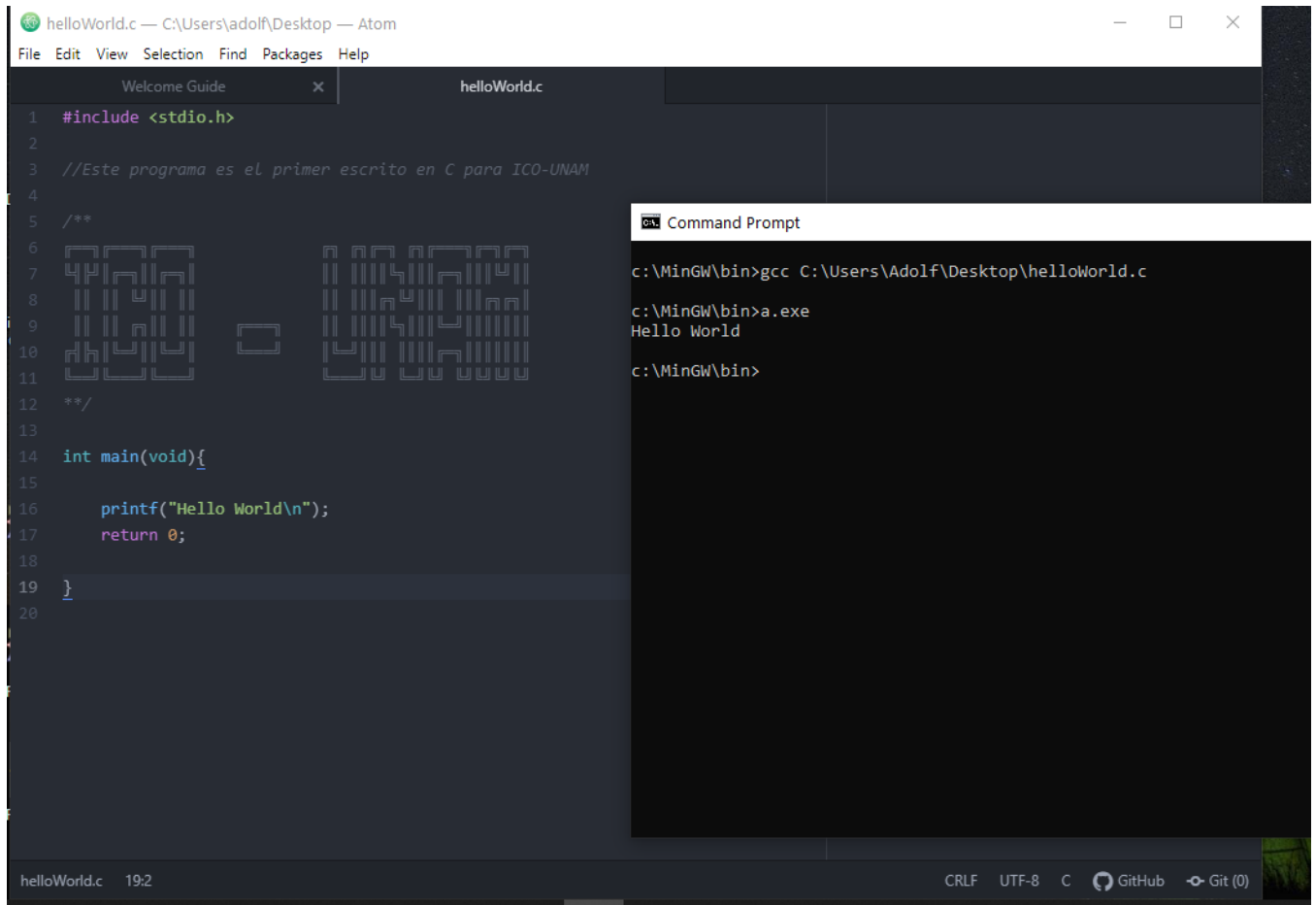
- Crear, compilar y ejecutar un programa simple escrito en C en GNU/Linux

Programa simple helloWorld.c creado y compilado en Terminal



* En algún entorno de desarrollo de Windows, crear, compilar y ejecutar un programa simple escrito en C.

Programa helloWorld.c escrito en Atom para Windows y compilado en Command Prompt



The image shows a screenshot of the Atom text editor and a Windows Command Prompt window. The Atom editor is open to a file named `helloWorld.c` located at `C:\Users\adolof\Desktop`. The code in the editor is as follows:

```
1 #include <stdio.h>
2
3 //Este programa es el primer escrito en C para ICO-UNAM
4
5 /**
6  4P1a1a      a a a a a a a a
7  II II II II  II II II II II II
8  II II II II  II II II II II II
9  II II II II  II II II II II II
10 II II II II  II II II II II II
11 II II II II  II II II II II II
12 **/
13
14 int main(void){
15
16     printf("Hello World\n");
17     return 0;
18 }
19
20
```

The Command Prompt window shows the following commands and output:

```
c:\MinGW\bin>gcc C:\Users\Adolf\Desktop\helloWorld.c
c:\MinGW\bin>a.exe
Hello World
c:\MinGW\bin>
```

The status bar at the bottom of the Atom editor indicates the file is `helloWorld.c`, line 19, column 2. It also shows settings for CRLF line endings, UTF-8 encoding, and C language mode, along with GitHub and Git integration icons.

CONCLUSIÓN:

En esta practica aprendimos a usar y conocimos algunas herramientas para escribir programas que los entornos de Linux contienen como lo fue Emacs en donde creamos el programa helloworld.c

También aprendimos a compilar estos programas ya sea en el entorno de Linux o Windows y para Windows usamos aprendimos a instalar el gcc con block stacks y aprendí a ejecutar el comando para compilar "gcc" presente en los dos sistemas así como a ejecutar programas desde cada una de la terminal y del CMD

Referencias:

- Facultad de ingeniería – UNAM, Manual de prácticas del laboratorio de Fundamentos de programación: http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17_FP.pdf
- Cursos en Línea. Soluciones MyL.: http://solucionesmyl.com/cursos/lenguaje_c/scanf.html
- Documentacion oficial MINGW: <http://mingw-w64.org/doku.php/documentation>