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

# Laboratorios de computación salas A y B

*Profesor:* M.I. Marco Antonio Martínez Quintana

*Asignatura:* Fundamentos de Programación

*Grupo:* 3

*No de Práctica(s):* 6. Entorno de C (editores, compilación y ejecución).

*Integrante(s):* Hernández Vázquez Daniela

*No. de Equipo de cómputo empleado:* No aplica

*No. de Lista o Brigada:* 26

*Semestre:* 1

*Fecha de entrega:* Lunes 9 de noviembre de 2020

*Observaciones:*

CALIFICACIÓN: _____

# Practica 6: 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.

## Actividades:

- Utilizando un editor de GNU/Linux, crear un archivo de texto
- Modificar/actualizar un archivo ya existente con un editor GNU/Linux.
- Crear, compilar y ejecutar un programa simple escrito en C en GNU/Linux
- En algún entorno de desarrollo de Windows, crear, compilar y ejecutar un programa simple escrito en C.

## Introducción

Un lenguaje de programación permite expresar una serie de instrucciones que podrán ser realizadas por una computadora. Unos de los lenguajes de programación mayormente difundidos es el lenguaje C.

Éste es muy utilizado ya que la forma de dar instrucciones es muy cercana a lo que un humano podría abstraer, es decir, las instrucciones no son tal cual las que una computadora podría entender, para ello se necesitaría conocer a fondo el microprocesador, el sistema operativo entre otros aspectos. Por esta razón, C es conocido como un lenguaje de alto nivel, esto significa a que las instrucciones podrían ser entendidas fácilmente por un humano. En contraparte, un lenguaje de bajo nivel, son instrucciones que son cercanas a lo que la máquina puede entender y difícilmente pueden ser comprendidas por una persona que no tenga conocimientos de la máquina en que operarán. Algunos autores consideran al lenguaje C como un lenguaje de mediano nivel, ya que no es totalmente transparente sino tiene elementos que tienen que ver con la arquitectura de la máquina a la hora de programar.

Otra característica de C, es que es muy poderoso en el aspecto de combinar características de un lenguaje de alto nivel (facilidad de programación), con uno de bajo nivel (manejo más preciso de una máquina); por lo que se han creado variantes que permiten programar miles de dispositivos electrónicos en el mundo con sus respectivos compiladores.

Un programa en C se elabora describiendo cada una de las instrucciones de acuerdo a las reglas definidas en este lenguaje en un archivo de texto para después ser procesadas en un compilador. Un compilador es un programa que toma como entrada un archivo de texto y tiene como salida un programa ejecutable, éste tiene instrucciones que poden ser procesadas por el hardware de la computadora en conjunto con el sistema operativo que corre sobre ella. Se tiene como ventaja que un programa escrito en lenguaje C, siguiendo siempre su estándar, puede correr en cualquier máquina siempre y cuando exista un compilador de C hecho para tal.

Para realizar un programa usando el lenguaje C, es necesario pensar primero en el sistema operativo que corre sobre la máquina y posteriormente, si este sistema cuenta

con interfaz gráfica o sólo posee línea de comandos. A veces, se puede pensar siempre en sólo usar sistemas operativos con interfaz gráfica dado a que su manejo es más sencillo, sin embargo, esta se encuentra limitada para operar toda la funcionalidad del sistema operativo además de que consume recursos de cómputo que pueden ser indispensables para equipos donde el rendimiento es imprescindible. Una vez que se han seleccionado estos elementos, se necesita buscar qué opciones de editores y compiladores están disponibles.

## 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 se 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 taclearse 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 deshacer el ultimo cambio.

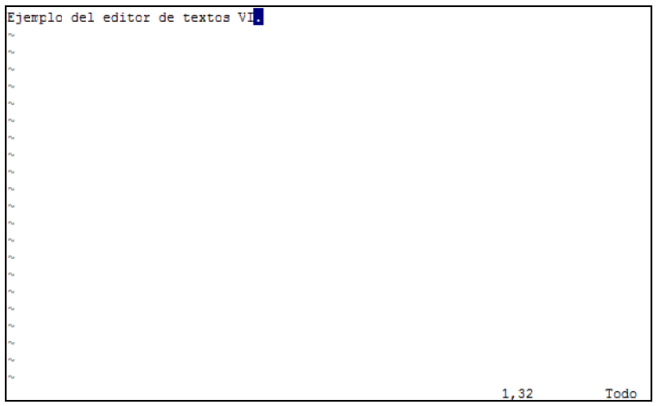



**Figura 1:** VI en modo comando.

### *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 **:**. 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.

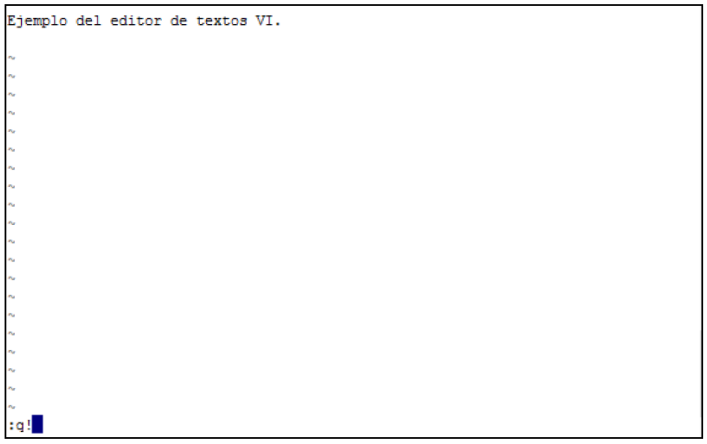



**Figura 2:** VI en modo de última línea.

## *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.
- 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.

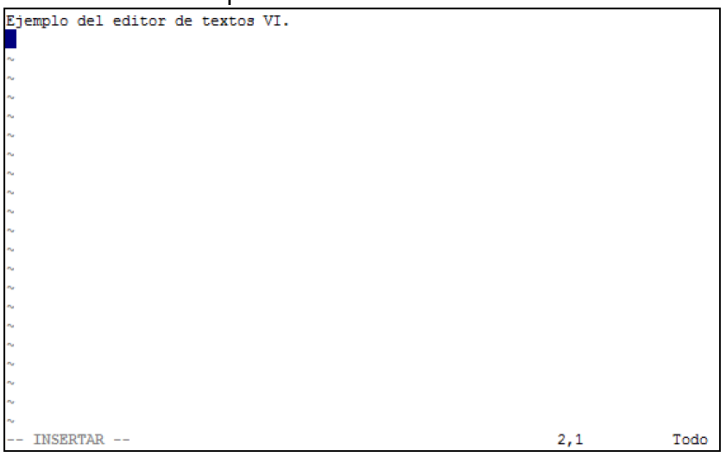



**Figura 3:** VI en modo de insertar.

## 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. No es necesario saber cómo se utiliza sino proporciona una interfaz que describe los comandos básicos.

NANO es un editor clon de otro editor llamado PICO.

Para iniciar NANO, debe taclearse desde la línea de comandos:

nano nombre_archivo[.ext]

Donde "nombre_archivo" puede ser el nombre del archivo a editar o el nombre de un archivo nuevo.

Una vez en el editor, en la parte inferior se pueden observar los comandos básicos. Si se presiona la tecla **F1** es posible visualizar la ayuda con la lista de todos comandos que existen.

Los atajos de teclado pueden corresponder a:

- **^** que es la tecla **Ctrl.**
- **M**- que es la tecla **Esc** o bien **Alt**.

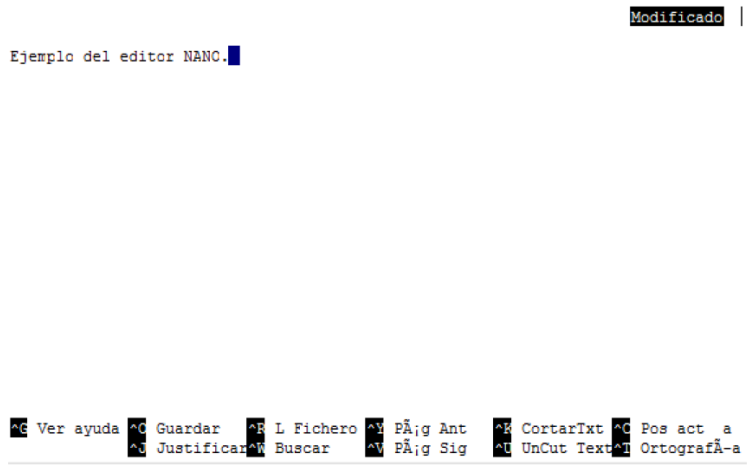



**Figura 4:** El editor de texto NANO.

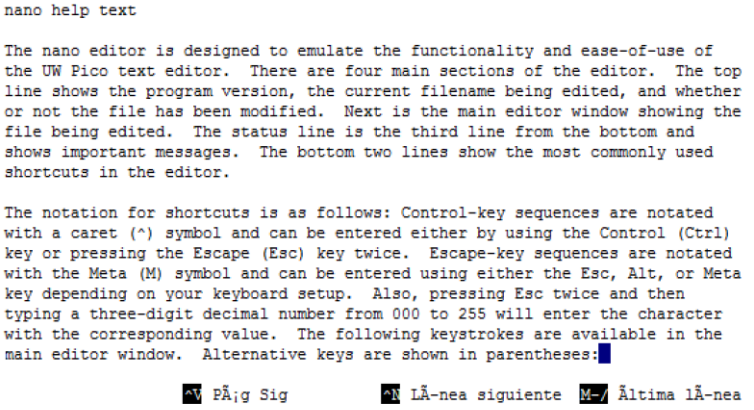



**Figura 5:** Editor de texto NANO. Ayuda del Editor.

### 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. Éste 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.

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

## Ejecución

La ejecución es la etapa que sigue después de haber compilado el programa. Uva 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:

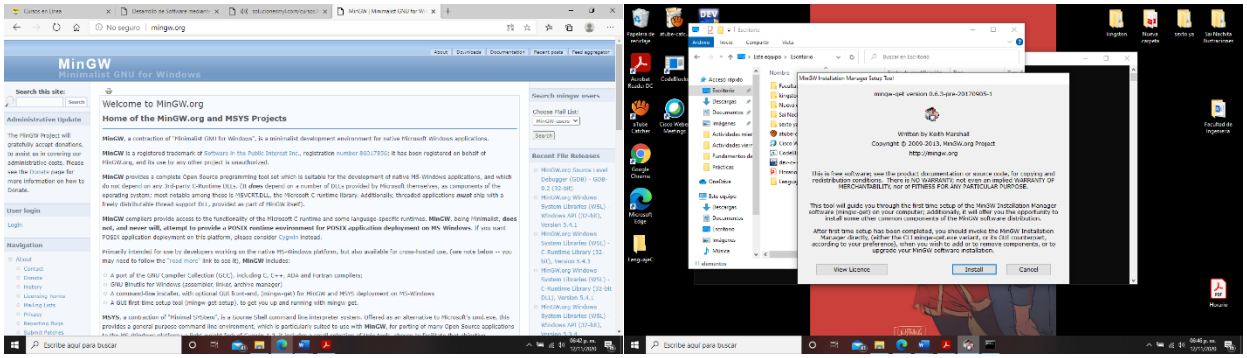
```
calculadora.exe
```

En 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.
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:

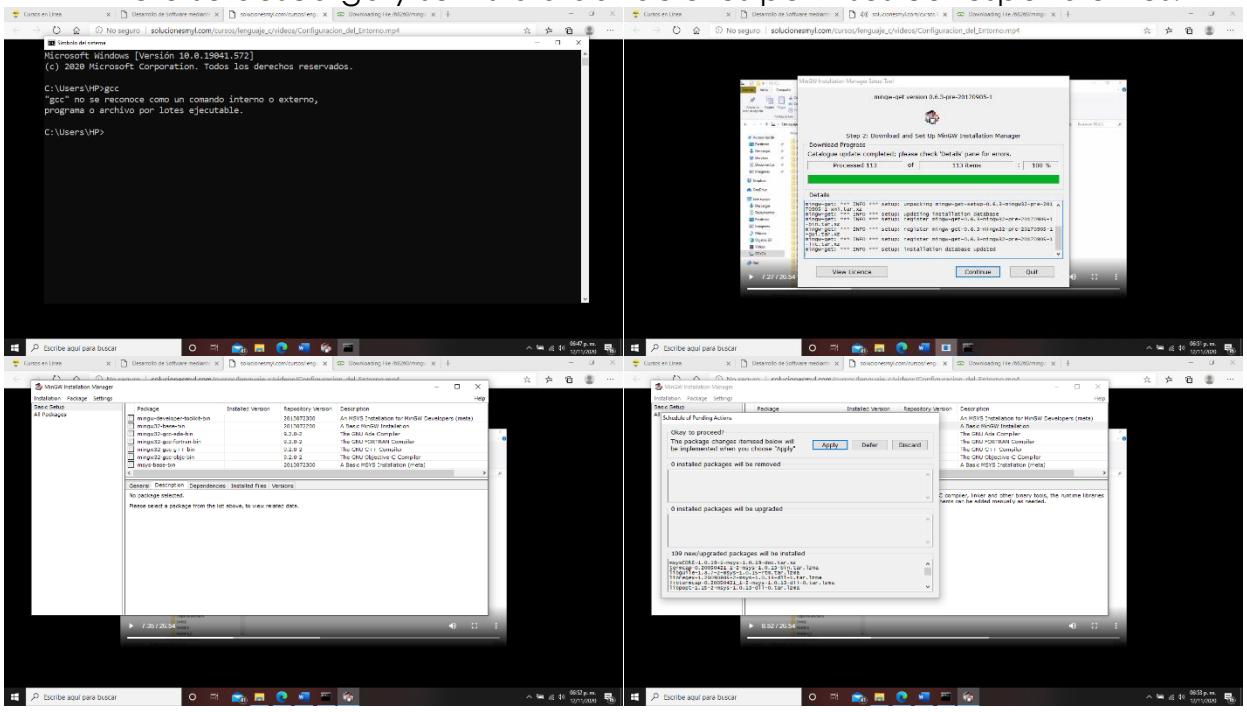
```
calculadora argumento1 argumento2
```

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.

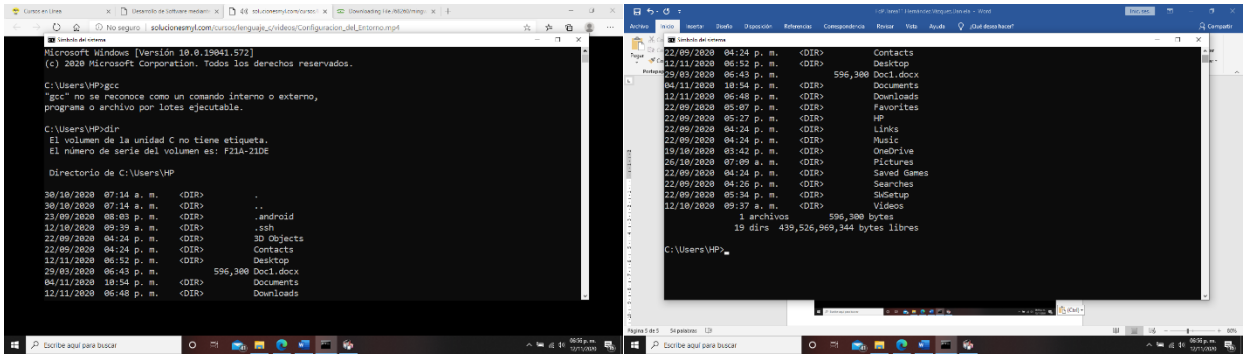
*A continuación, mando el procedimiento de instalación y ejecución del compilador gcc.*

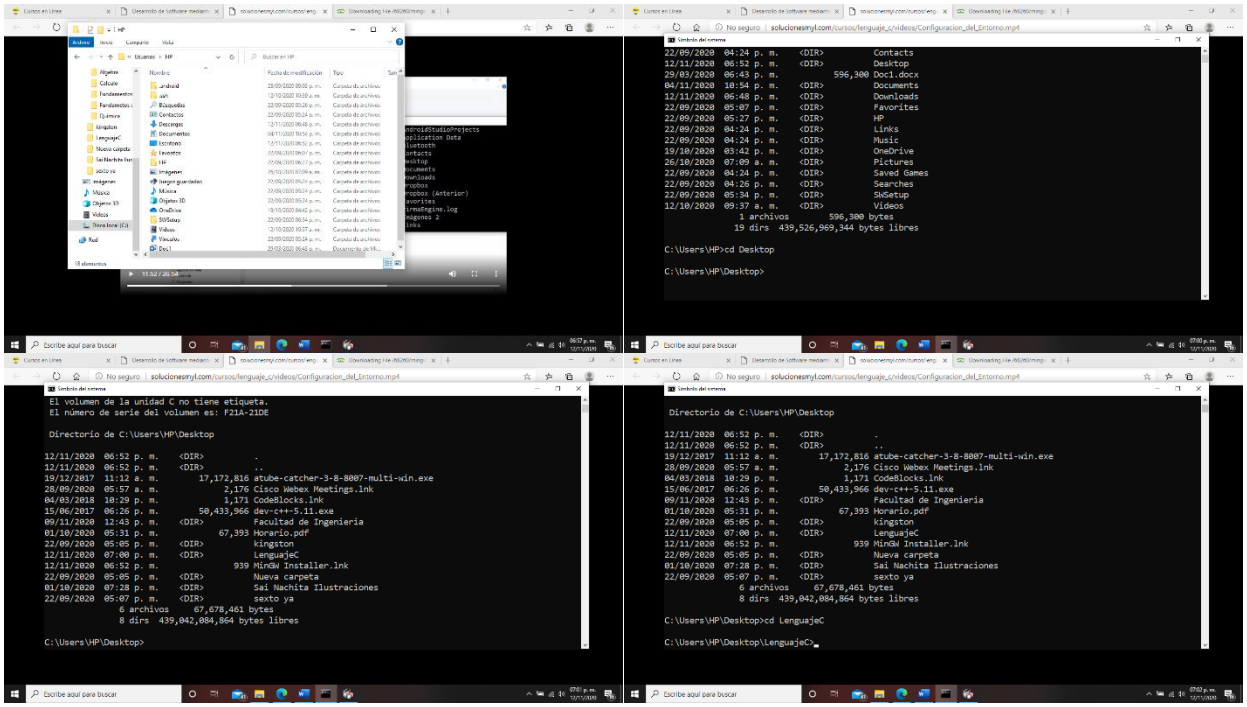



Primero se descarga y se instala dándole los permisos correspondientes.




Se eligen las preferencias de instalación y después de eso se entra a la terminal o también llamada símbolo del sistema para verificar que se ha instalado correctamente, su instalación es algo tardada, por lo tanto, vemos 3 comandos muy necesarios.

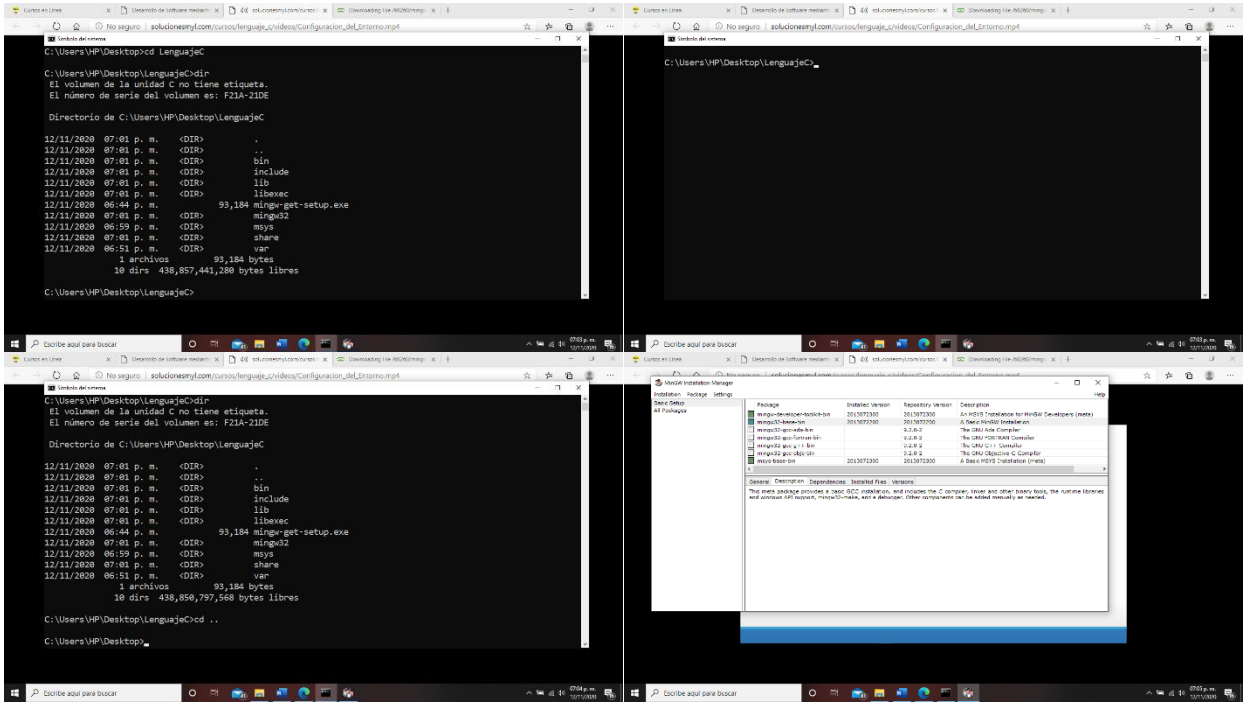


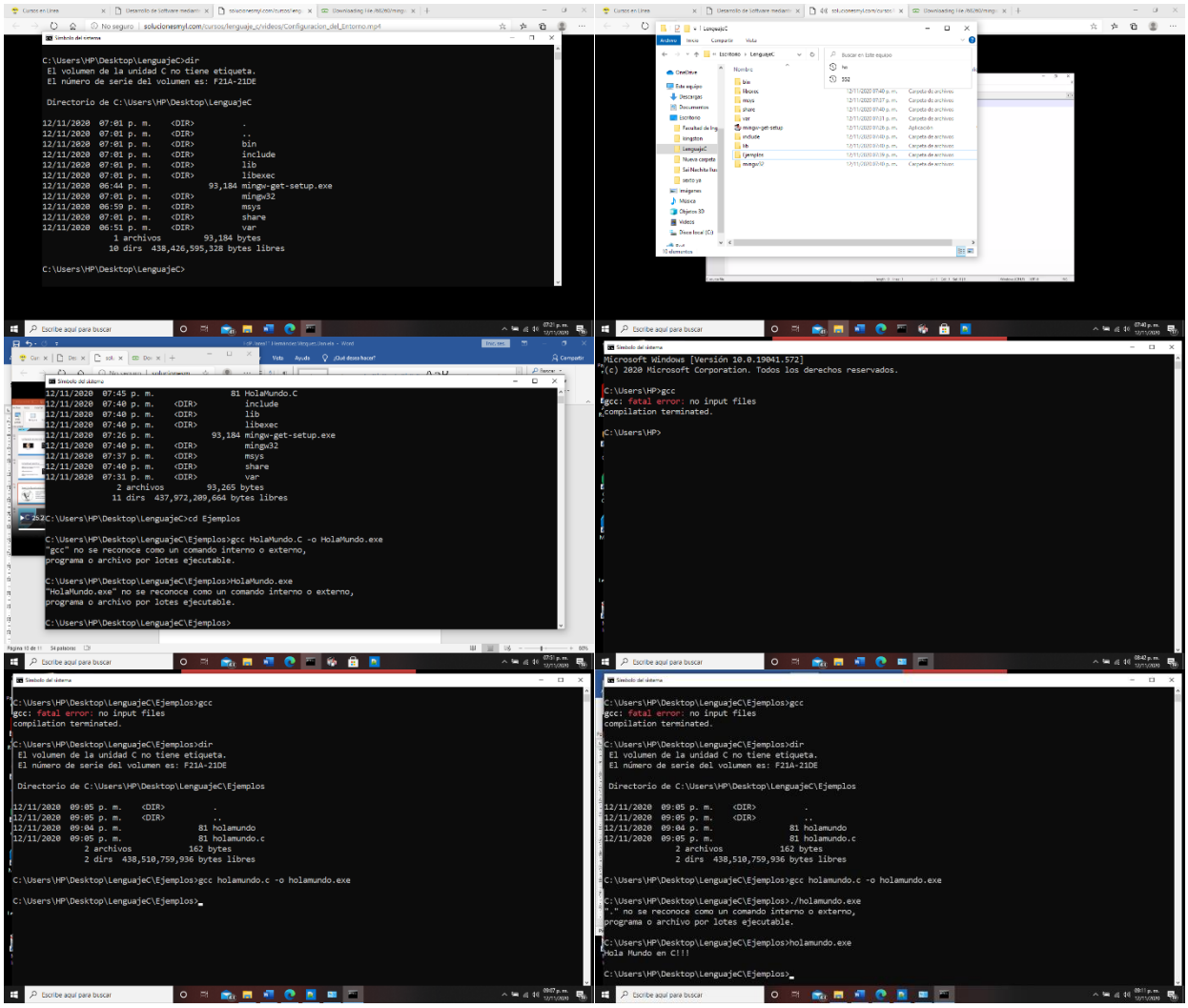


Dir: el cual lista los elementos dentro de nuestras carpetas.

Cd: cambio de directorio

Cls: el cual limpia nuestra pantalla dejándonos solo con la última línea registrada.






## Conclusiones:

Ahora desde una terminal nos es posible hacer una impresión dentro de la pantalla, los comandos vistos son muy necesarios a la hora de revidar todos los elementos que se encuentran dentro de nuestro equipo y de nuestras carpetas, de este modo es fácil ejecutar, De no tener bien configurado el programa es posible que cunado introduzcamos gcc no lo reconozca como comando ejecutable, si esto nos ocurre puede ser porque el sistema no está buscando dentro de los lugares adecuados y se tendrían que modificar algunos aspectos como yo en este caso tuve que editar path para que fuera capaz de reconocer o dentro de cualquier carpeta.

## 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/