



Linux básico y GCC

Objetivos

- Introducir los conceptos más básicos del manejo de consola necesarios para la realización de las tareas básicas de programación.
- Introducir en el manejo del GCC como herramienta para la compilación de código.

Requisitos

- Pc con Linux
- Compilador GCC
- Editor de texto o IDE

Introducción al manejo de la consola GNU/Linux

- ¿Qué es la consola?

La consola o terminal (Shell) es un programa informático donde interactúa el usuario con el sistema operativo mediante una ventana que espera órdenes escritas por el usuario desde el teclado.

- ¿Por qué usar la consola?

La consola permite un mayor grado de funciones y configuración de lo que queremos hacer con una aplicación o acción en general respecto del entorno gráfico. "A grosso modo", puedes tener un mayor control sobre tu equipo.

En GNU/Linux la consola es algo necesario. Acciones para dar o quitar permisos, configurar e instalar drivers que no estén empaquetados y puedan ser ejecutados por un instalador, matar procesos de una manera más efectiva, ejercer como súper usuario cuando estás en una cuenta cualquiera del equipo y muchas acciones más que puedes descubrir a lo largo del manual.

- ¿Puede cualquier usuario usar la consola?

Bueno, esta pregunta hay que responderla con criterio. Cualquier usuario puede usar la consola siempre que sepa lo que está haciendo en ella, ya que si ejecutamos algún comando sin conocimiento y este resulta peligroso para nuestro sistema, podríamos dejar nuestro sistema inutilizable, borrar archivos necesarios, etc.

- ¿Qué conocimientos previos son necesarios?

Los conocimientos previos más básicos son los comandos que hay en la consola. Es imposible saberlos todos de memoria, pero si es recomendable que los más usados se conozcan muy bien. A la hora de hacer configuraciones, instalaciones, modificaciones, etc. si es necesario que se tenga noción de que archivo es, su importancia en Linux, guardar una copia del archivo.

Los comandos al escribirlos en pantalla se ejecutan en la carpeta actual donde se esté ubicado, por tanto, si se quiere realizar un acción sobre otra carpeta basta con poner la ruta después del comando

Rutas

Secuencia de directorios anidados separados con el carácter slash (/) con un archivo o directorio al final.

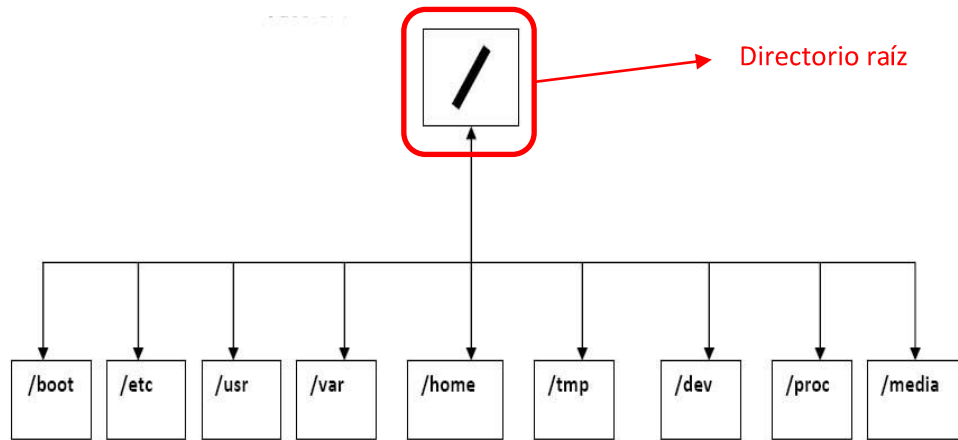


Figura 1. Estructura de directorios en linux

Directorios especiales:

- / Directorio raíz
- ./ Directorio actual
- ../ Directorio padre del directorio en el cual me encuentro ubicado

Existen 2 tipos de rutas:

- **Rutas Absolutas:** Rutas vistas desde el directorio raíz.
- **Rutas Relativas:** Rutas vistas desde un directorio en particular.

Ejemplos:

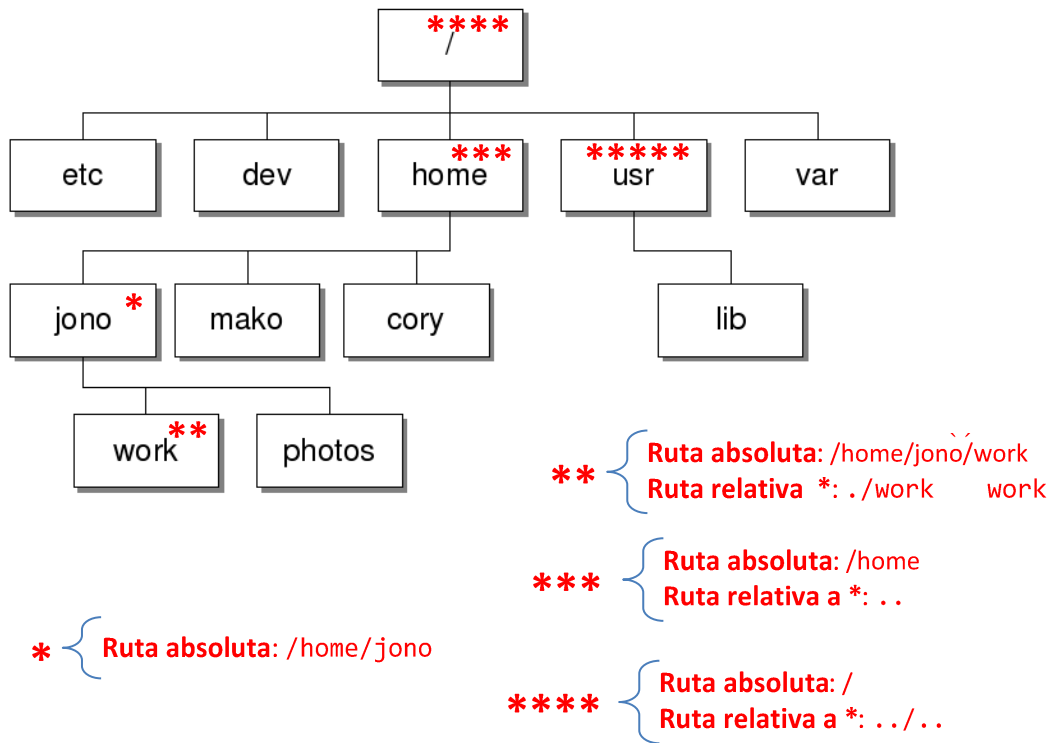


Figura 2. Rutas absolutas y relativas

1. ¿Cuál es la ruta absoluta de *home*?
2. ¿Cuál es la ruta de *home* relativa a *work*?
3. Si estoy ubicado en el directorio *home*, ¿Cuál es la ruta absoluta y relativa para ubicarse en *photos*?
4. Si estoy ubicado en el directorio *jono*, ¿Cuál es la ruta absoluta y relativa para ubicarse en *photos*?
5. Si estoy ubicado en el directorio *jono*, ¿Cuál es la ruta absoluta y relativa para ubicarse en *lib*?

Comandos básicos de GNU/Linux

Comandos de Linux	
man	Manual de comandos
pwd	Ubicación actual
cd	Cambiar de directorio
ls	Listado de archivos y directorios
clear	Limpiar pantalla
mkdir	Crear directorio
rm	Borrar directorio

Ver archivos y directorio

El comando “ls” lista los archivos de un directorio, en orden alfanumerico, exceptuando los archivos que empiezan con el carácter “.” (archivos ocultos). El directorio dir es opcional y cuando no aparece los archivos listados son los del directorio actual.

- \$ ls <opciones> <dir> Lista archivos del directorio dir, las opciones son opcionales
- \$ ls Lista los archivos del directorio actual
- \$ ls /ruta/dir Lista los archivos de un directorio específico

Ambos comandos pueden ser modificados para mostrar cosas específicas, las opciones más usadas son:

- \$ ls -a Listar todos los archivos y carpetas incluyendo ocultos
- \$ ls -l Listar las propiedades de los archivos
- \$ ls -t Listar ordenando por fecha de modificación
- \$ ls -m Listar en una sola línea y separados por comas

Para mayor información puede consultar el manual del comando: **man ls**.

Ejemplo:

#Listando archivos y directorios del directorio raíz: /home/
usuario@nombrePC:~\$ ls /home/

Cambiar de directorio

Para cambiar de directorio al navegar entre nuestros archivos por medio de la terminal se puede hacer uso del comando "**cd**". El cambio de directorio sólo se llevará a cabo si el directorio especificado existe, si no es así, nos quedaremos en el mismo directorio desde el que se invoco el comando.

Si el cambio de directorio se ha dado con éxito, el nombre del directorio al cual accedimos se mostrará en la terminal. Es usual usar el comando `pwd` después del comando `cd` para verificar el directorio actual.

Algunas de las opciones disponibles para el comando `cd` son:

- `cd <dir>` Ir al directorio `dir`
- `cd -` Ir al directorio anterior
- `cd ..` Ir al directorio padre
- `cd ~` Ir a la carpeta "home"

Para mayor información puede consultar el manual del comando: **man cd**.

Ejemplo:

#Entrar a un directorio

```
usuario@nombrePC:~$ cd ruta/NombreDirectorio
```

Crear directorios

Para crear directorios en GNU/Linux, existe un comando simple a la par que útil, se trata de "`mkdir`" no tiene mayor ciencia que el escribir "`mkdir`" más el nombre de la carpeta a crear.

Para mayor información puede consultar el manual del comando: **man mkdir**.

Ejemplo:

#Crear un directorio

```
usuario@nombrePC:~$ mkdir NombreDirectorio
```

Borrar directorios y archivos

Si se quiere borrar un directorio en Linux, se puede hacer uso del comando "**rm**" la sintaxis es simple, "**rm**" más el nombre del fichero/carpeta a eliminar.

Algunas de las opciones disponibles para el comando **rm** son:

- **rm <opc><dir/fichero/dir>**
- **rm -r** Para un borrado recursivo
- **rm -f** Para un borrado forzado sin pedir autorización para cada archivo
- **rm -i** Para pedir confirmación por cada archivo borrado

Para mayor información puede consultar el manual del comando: **man rm**.

Ejemplos:

#Borrar un directorio vacio

usuario@nombrePC:~\$ rm NombreDirectorio

#Borrar todo el contenido de un directorio

usuario@nombrePC:~\$ rm -r NombreDirectorio

#Borrar un archivo especifico

usuario@nombrePC:~\$ rm -r NombreDirectorio/NombreArchivo.extension

Copiar directorios y archivos

Para copiar directorios y archivos se puede usar el comando "**cp**", este comando viene de la palabra "copy" en Inglés que significa "copiar"

Para mayor información puede consultar el manual del comando: **man cp**.

Ejemplos:

#Realizar la copia de un archivo y dejar la copia en el mismo directorio que el original.

usuario@nombrePC:~\$ cp ArchivoOriginal ArchivoCopia

#Para realizar lo mismo pero con directorios y de forma recursiva

usuario@nombrePC:~\$ cp -r CarpetaOriginal/ CarpetaCopia/

#Se puede especificar que la copia se ponga en otro lugar distinto al de origen

usuario@nombrePC:~\$ cp ArchivoOriginal /rutacopia/ArchivoCopia

#Obviamente se puede hacer lo mismo con carpetas

usuario@nombrePC:~\$ cp -r CarpetaOriginal /rutacopia/CarpetaCopia

Mover/Renombrar archivos y directorios

Mover archivos y directorios bajo terminal equivale a cortar y pegar en modo gráfico, renombrar archivos y directorios equivale a dar click en "Cambiar nombre" en entorno gráfico, nosotros podemos lograr estas dos cosas con un sólo comando, para lograrlo, utilizaremos el comando "mv" este comando no tiene mayor ciencia que teclear "**mv**" origen destino, funciona tanto con archivos como con carpetas.

Para mayor información puede consultar el manual del comando: **man mv**.

Ejemplos:

#Mover archivo a un directorio específico

usuario@nombrePC:~\$ mv ArchivoOrigen /LugarDeDestino/ArchivoDestino

#Renombrar una carpeta y dejarla en el mismo lugar

usuario@nombrePC:~\$ mv NombreOriginal NombreNuevo

Buscar archivos y directorio

Si deseamos buscar algo, ya sea archivo o directorio, podemos recurrir al comando "find", este comando tiene diversos modificadores, por lo general la búsqueda mediante terminal es más rápida y consume menos recursos que la búsqueda mediante una aplicación gráfica, a continuación se explican 2 opciones del comando:

Buscar por nombre: find /lugar_busqueda/ -name nombre_archivo

Buscar por tamaño: find /lugar_busqueda/ -size tamañokb

Para mayor información puede consultar el manual del comando: **man find**.

Ejemplos:

#Búsqueda por nombre

usuario@nombrePC:~\$ find /home/usuario/ -name Archivo.tar.gz

#Búsqueda por tamaño


```
usuario@nombrePC:~$ find /home/usuario/ -size +500
```

#NOTA: Lo que hace el último ejemplo es buscar archivos de más de 500 KB

Limpiar la terminal

Después de usar un buen tiempo la terminal, es probable que nos encontremos confundidos de tanto texto que hay y que no necesitamos, para limpiar la ventana podemos hacer uso del comando "**clear**", se trata de un comando sencillo a la par que útil

Ejemplo:

#Limpiar la terminal

```
usuario@nombrePC:~$ clear
```

O puedes usar **ctrl+l**

Para las siguientes preguntas asuma que se encuentra ubicado en la ruta */home*

6. ¿Cuáles son los comandos para ir y crear el directorio *Italy* dentro de *photos*?
7. ¿Cuál es el comando para crear el directorio *Spain* dentro de *photos* permaneciendo en *home*?
8. ¿Cuál es el comando (o secuencia de comandos) para crear 2 directorios llamados *dir1* y *dir2* dentro de *work*?
9. Como se elimina el directorio *dir1* asumiendo que este no esta vacío?
10. ¿Cuál es el comando (o conjunto de comandos) para listar el contenido del directorio *jono* con sus propiedades y archivos ocultos?

El compilador GCC

GCC es un compilador rápido, muy flexible, y riguroso con el estándar de C ANSI. Como ejemplo de sus múltiples virtudes, diremos que gcc puede funcionar como compilador cruzado para un gran número de arquitecturas distintas. GCC no proporciona un entorno IDEs, es solo una herramienta más a utilizar en el proceso. GCC se encarga de realizar (o encargar el trabajo a otras utilidades) el preprocesado del código, la compilación, y el enlazado. Dicho de otra manera, nosotros proporcionamos a gcc nuestro código fuente en C, y él nos devuelve un archivo binario compilado para nuestra arquitectura.

Manejo del GCC¹

La sintaxis de gcc es la siguiente:

```
$ gcc [options] file.c
```

Para compilar el programa holamundo.c:

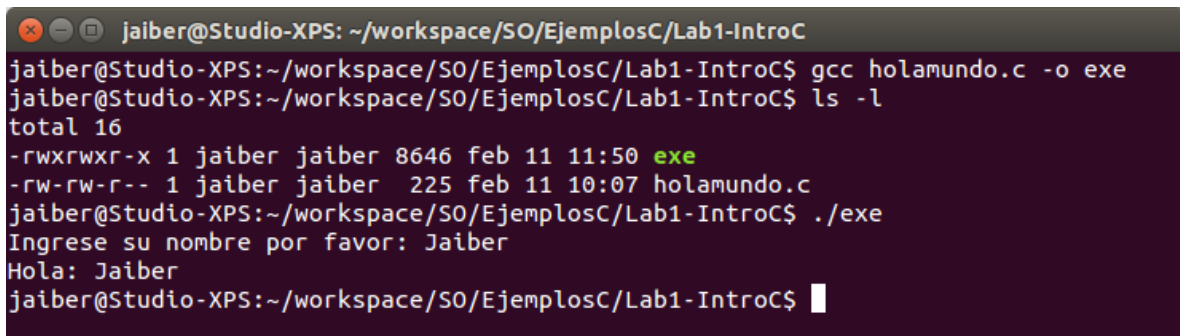
```
$ gcc holamundo.c -o ejecutable
```

Luego verificamos que se cree el ejecutable exitosamente

```
$ ls -l
```

Nos muestra un archivo ejecutable, que es el archivo ejecutable resultado de la compilación. Por ultimo ejecutamos el programa

```
$ ./exe
```



```
jaiber@Studio-XPS: ~/workspace/SO/EjemplosC/Lab1-IntroC
jaiber@Studio-XPS:~/workspace/SO/EjemplosC/Lab1-IntroC$ gcc holamundo.c -o exe
jaiber@Studio-XPS:~/workspace/SO/EjemplosC/Lab1-IntroC$ ls -l
total 16
-rwxrwxr-x 1 jaiber jaiber 8646 feb 11 11:50 exe
-rw-rw-r-- 1 jaiber jaiber 225 feb 11 10:07 holamundo.c
jaiber@Studio-XPS:~/workspace/SO/EjemplosC/Lab1-IntroC$ ./exe
Ingrese su nombre por favor: Jaiber
Hola: Jaiber
jaiber@Studio-XPS:~/workspace/SO/EjemplosC/Lab1-IntroC$
```

Figura 3. Ejemplo del manejo de gcc

¹ Para mayor información:

https://www.mhe.es/universidad/informatica/8448198441/archivos/apendice_general_1.pdf

Si el código fuente tiene errores al compilar el programa:

```
$gcc holamundo.c
holamundo.c: In function `main':
holamundo.c:7: `a' undeclared (first use in this function)
holamundo.c:7: (Each undeclared identifier is reported only once
holamundo.c:7: for each function it appears in.)
holamundo.c:7: parse error before `return'
```

Como vemos gcc nos proporciona el fichero y la línea en la que ha detectado el error. El formato de la salida de error es reconocido por la mayoría de los editores. Obviamente, cuando gcc genera algún error, no se crea archivo ejecutable como resultado.

Warnings y errores

- **Error:** fallo al analizar el código C que impide la generación de un ejecutable final.
- **Warning:** advertencia del compilador al analizar el código C que no impide la generación de un ejecutable final.

Al compilar nos podemos encontrar con advertencias como las siguientes:

```
$ gcc holamundo.c
holamundo.c: In function `main':
holamundo.c:6: warning: control reaches end of non-void function
```

A pesar del warning, gcc ha compilado un fichero ejecutable. Si se presenta un error, no se genera el ejecutable

Opciones más comunes

A continuación mostramos algunas de las opciones más habituales al usar gcc:

-help

Indica a gcc que muestre su salida de ayuda (muy reducida).

-o <file>

El archivo ejecutable generado por gcc es por defecto a.out. Mediante este modificador, le especificamos el nombre del ejecutable.

-Wall

No omite la detección de ningún warning. Por defecto, gcc omite una colección de warnings "poco importantes".

-g

Incluye en el binario información necesaria para utilizar un depurador posteriormente.

-O <nivel>

Indica a gcc que utilice optimizaciones en el código. Los niveles posibles van desde 0 (no optimizar) hasta 3 (optimización máxima). Utilizar el optimizador aumenta el tiempo de compilación, pero suele generar ejecutables más rápidos.

-E

Sólo realiza la fase del preprocesador, no compila, ni ensambla, ni enlaza.

-S

Preprocesa y compila, pero no ensambla ni enlaza.

-c

Preprocesa, compila y ensambla, pero no enlaza.

-I <dir>

Especifica un directorio adicional donde gcc debe buscar los archivos de cabecera indicados en el código fuente.

-L <dir>

Especifica un directorio adicional donde gcc debe buscar las librerías necesarias en el proceso de enlazado.

-l<library>

Especifica el nombre de una librería adicional que deberá ser utilizada en el proceso de enlazado.

La colección completa de modificadores a utilizar con gcc se encuentra en su página de manual, **man gcc**.

Compilando el primer programa

La siguiente gráfica muestra el flujo de trabajo para crear un programa en c:

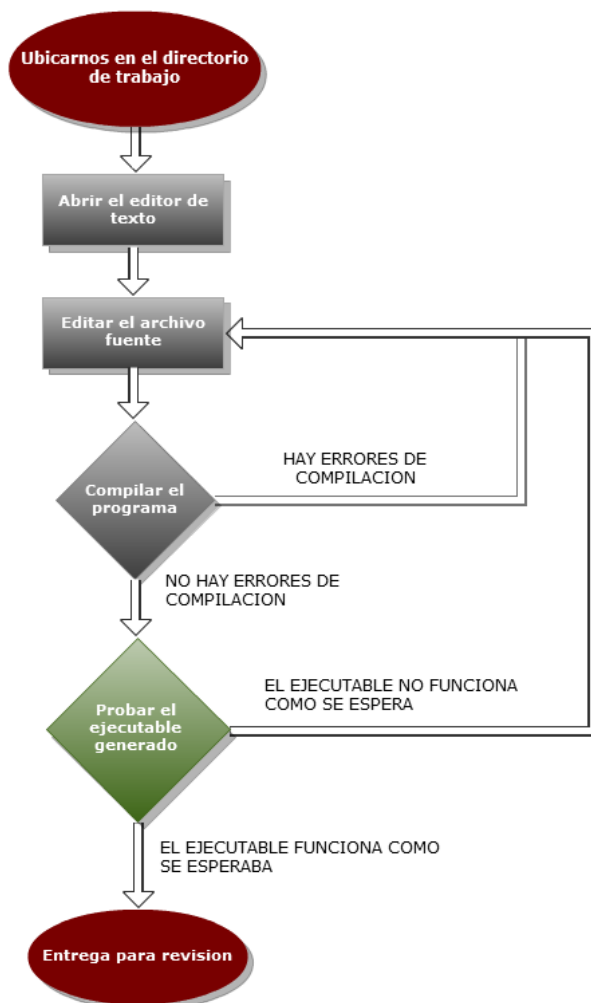
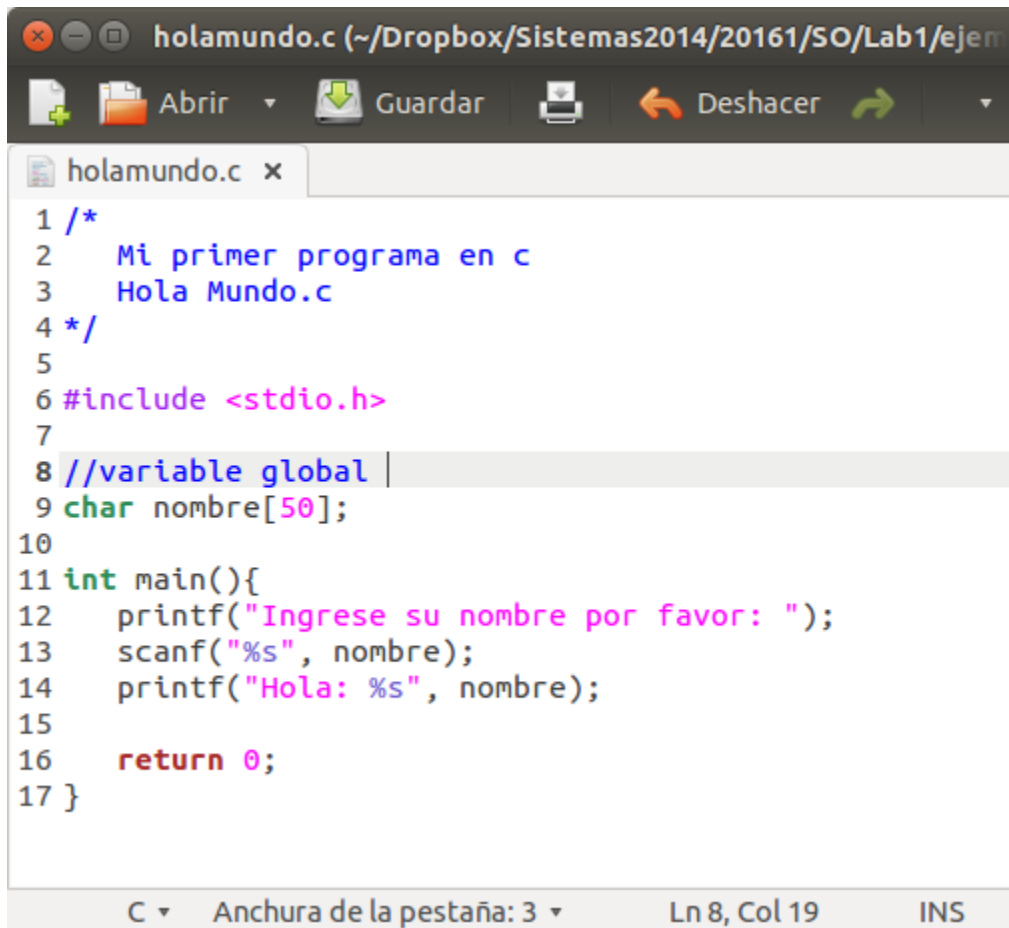


Figura 4. Flujo para crear un programa.

11. Siga los siguientes pasos para compilar nuestro pirmer programa:

- Abra un editor de texto plano (gedit, sublime text o cualquier otro)
- Una vez abierto el editor, se codifica el programa (ver figura 5) y se guarda en la ruta `/home/<user>/SO/labs`, el nombre de este debe tener extensión `.c`.
- Después de que el programa ha sido codificado y guardado se verifica por seguridad que si se encuentre en el directorio de trabajo, use el comando `ls`.
- Compile el programa usando GCC y lo parámetros adecuados para generar un ejecutable llamado `myexe` (Ver figura 3).

- e. Si hay errores, volver al código y corregirlos. De lo contrario ejecute el programa.



The image shows a code editor window titled 'holamundo.c (~/.Dropbox/Sistemas2014/20161/SO/Lab1/ejem)'. The editor has a menu bar with 'Abrir', 'Guardar', and 'Deshacer'. The code is as follows:

```
1 /*
2     Mi primer programa en c
3     Hola Mundo.c
4 */
5
6 #include <stdio.h>
7
8 //variable global |
9 char nombre[50];
10
11 int main(){
12     printf("Ingrese su nombre por favor: ");
13     scanf("%s", nombre);
14     printf("Hola: %s", nombre);
15
16     return 0;
17 }
```

The status bar at the bottom shows 'C', 'Anchura de la pestaña: 3', 'Ln 8, Col 19', and 'INS'.

Figura 5. Código fuente del hola mundo.

12. ¿Qué sucede si se ejecuta el comando `gcc` sin la opción `-o`?