



# Linux básico y GCC

## *Objetivos*

- Comprender los conceptos básicos sobre el manejo de la consola (o terminal) en Linux, necesarios para la realización de las tareas básicas de programación.
- Abordar el manejo del GCC como herramienta para la compilación de código en lenguaje de programación C.

## *Requisitos*

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

## 1. 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?

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

## 2. Rutas

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

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.

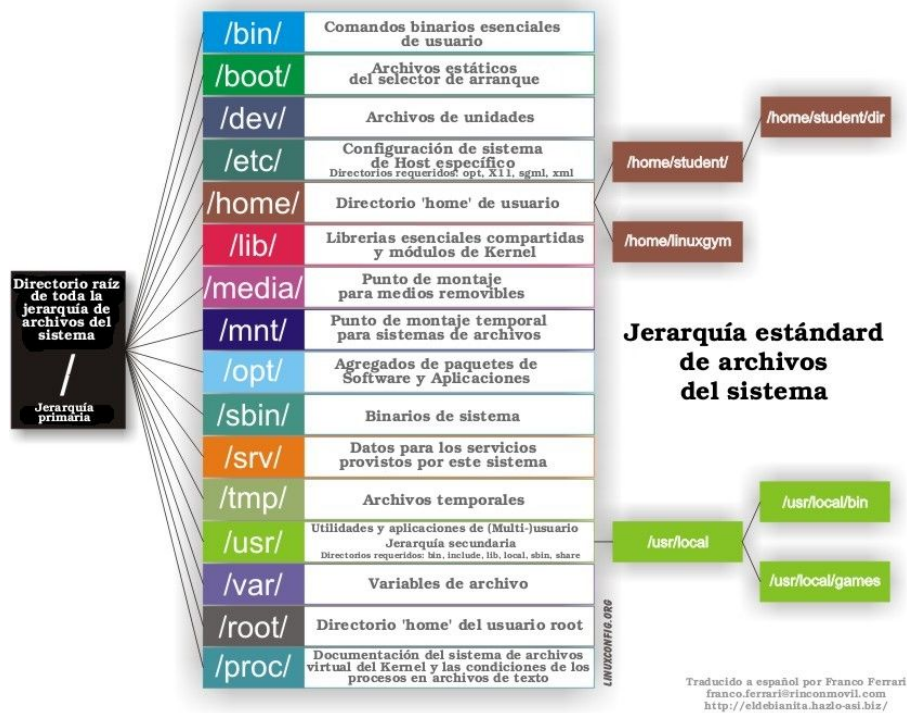


Figura 1. Estructura de directorios en linux

Ejemplos:

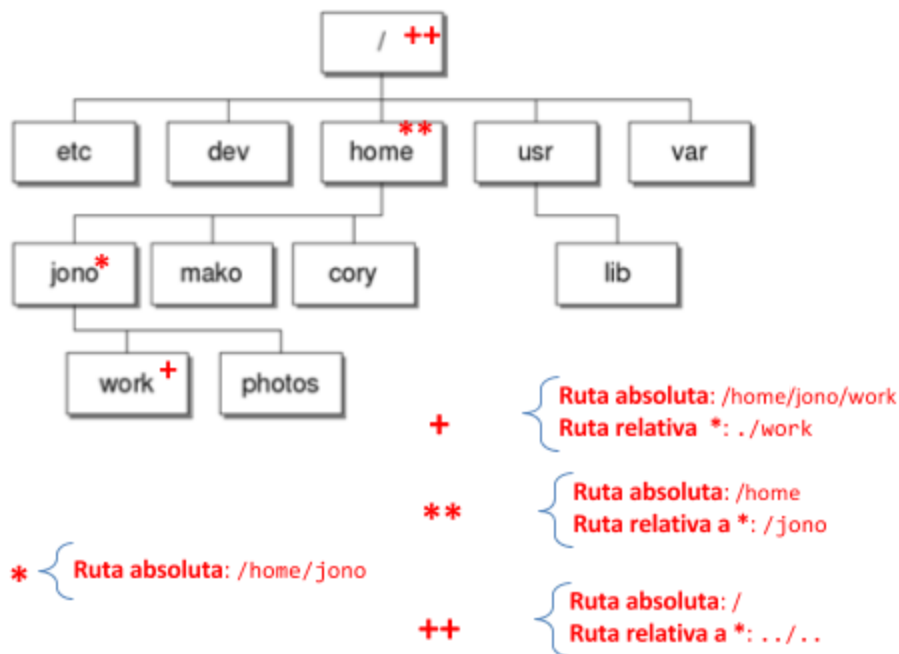


Figura 2. Rutas absolutas y relativas

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*?

### 3. Comandos Básicos de GNU/Linux

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

#### 3.1. Ver Archivos y Directorios

El comando “**ls**” lista los archivos de un directorio, en orden alfanumérico, 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.

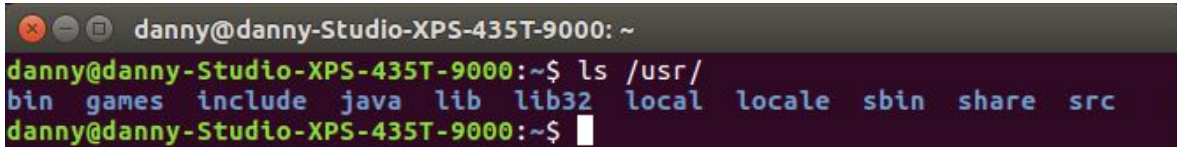
- \$ **ls** <opciones> <dir>
  - Lista archivos del directorio *dir*, las opciones son opcionales
- \$ **ls**
  - Lista los archivos del directorio actual
- \$ **ls** /ruta/dir
  - Lista los archivos de un directorio específico

Ambos comandos pueden ser modificados para mostrar información específica, 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 sólo línea y separados por comas

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

La Figura 3 muestra un ejemplo del uso del comando `ls`, listando archivos y directorios del directorio `usr`.



```
danny@danny-Studio-XPS-435T-9000: ~  
danny@danny-Studio-XPS-435T-9000:~$ ls /usr/  
bin games include java lib lib32 local locale sbin share src  
danny@danny-Studio-XPS-435T-9000:~$
```

Figura 3. Uso del comando `ls`

## 3.2. Cambiar el Directorio de Trabajo

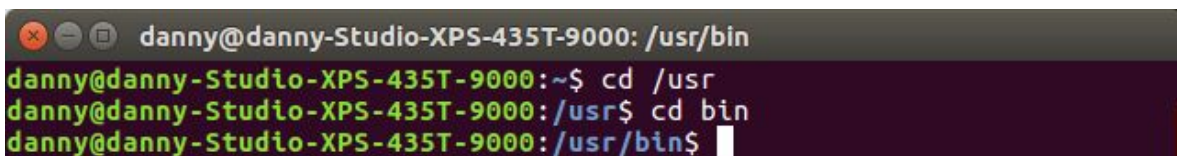
El comando “`cd`” permite cambiar el directorio de trabajo (working directory) al navegar entre nuestros archivos por medio de la terminal. El cambio de directorio de trabajo sólo se lleva a cabo si existe el directorio solicitado, si no es así, el sistema mantiene el directorio de trabajo actual.

Si el cambio de directorio de trabajo se realiza con éxito, el nombre del nuevo directorio de trabajo se muestra en el prompt de la terminal. Con frecuencia se usa 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"

La Figura 4 muestra un ejemplo del uso del comando `cd`. Para mayor información puede consultar el manual del comando: `man cd`.



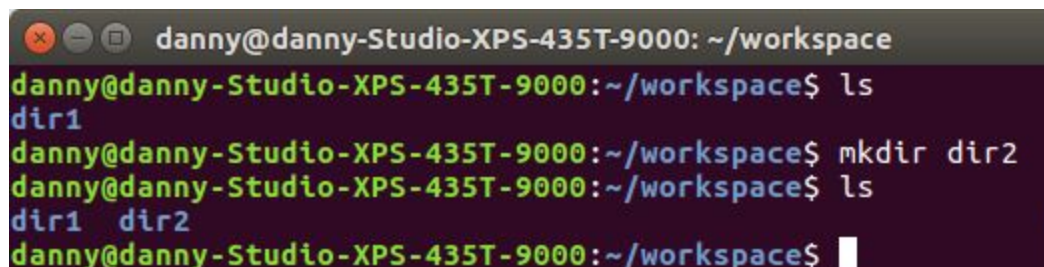
```
danny@danny-Studio-XPS-435T-9000: /usr/bin  
danny@danny-Studio-XPS-435T-9000:~$ cd /usr  
danny@danny-Studio-XPS-435T-9000:/usr$ cd bin  
danny@danny-Studio-XPS-435T-9000:/usr/bin$
```

Figura 4. Uso del comando `cd`

### 3.3. Crear Directorios

El comando "mkdir" permite crear directorios en un sistema Linux. Su modo de uso es muy simple, solo se requiere ingresar en la terminal "mkdir" seguido por el nombre de la carpeta a crear.

La Figura 5 muestra un ejemplo del uso del comando mkdir. Para mayor información puede consultar el manual del comando: `man mkdir`.



```
danny@danny-Studio-XPS-435T-9000: ~/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1
danny@danny-Studio-XPS-435T-9000:~/workspace$ mkdir dir2
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2
danny@danny-Studio-XPS-435T-9000:~/workspace$
```

Figura 5. Uso del comando mkdir

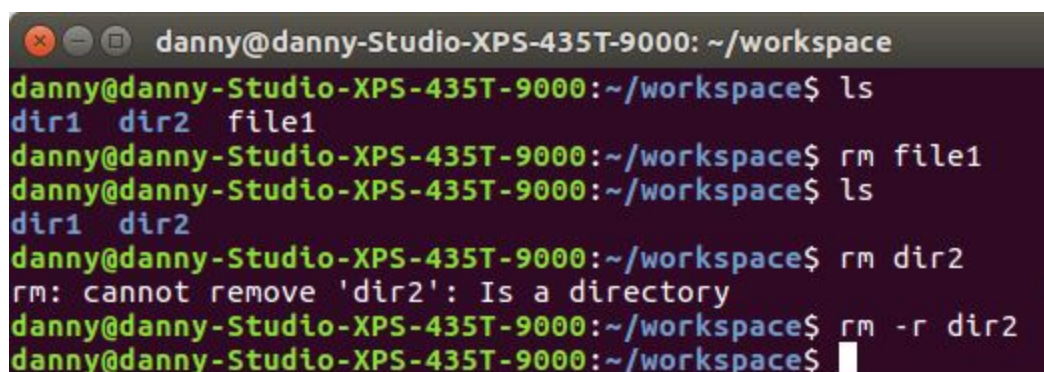
### 3.4. Borrar Archivos y Directorios

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

La Figura 6 muestra un ejemplo del uso del comando rm. Para mayor información puede consultar el manual del comando: `man rm`.



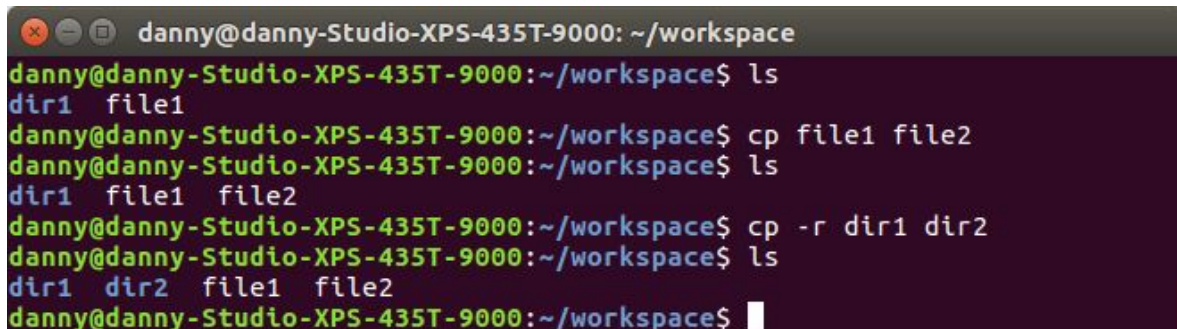
```
danny@danny-Studio-XPS-435T-9000: ~/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2 file1
danny@danny-Studio-XPS-435T-9000:~/workspace$ rm file1
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2
danny@danny-Studio-XPS-435T-9000:~/workspace$ rm dir2
rm: cannot remove 'dir2': Is a directory
danny@danny-Studio-XPS-435T-9000:~/workspace$ rm -r dir2
danny@danny-Studio-XPS-435T-9000:~/workspace$
```

Figura 6. Uso del comando rm

### 3.5. Copiar Archivos y Directorios

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

La Figura 7 muestra un ejemplo del uso del comando cp. Para mayor información puede consultar el manual del comando: man cp.

A terminal window with a dark background and light-colored text. The prompt is 'danny@danny-Studio-XPS-435T-9000: ~/workspace'. The commands and their outputs are: 'ls' shows 'dir1 file1'; 'cp file1 file2' is executed; 'ls' shows 'dir1 file1 file2'; 'cp -r dir1 dir2' is executed; 'ls' shows 'dir1 dir2 file1 file2'.

```
danny@danny-Studio-XPS-435T-9000: ~/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 file1
danny@danny-Studio-XPS-435T-9000:~/workspace$ cp file1 file2
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 file1 file2
danny@danny-Studio-XPS-435T-9000:~/workspace$ cp -r dir1 dir2
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2 file1 file2
danny@danny-Studio-XPS-435T-9000:~/workspace$
```

Figura 7. Uso del comando 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 /ruta/ArchivoCopia

**#Obviamente se puede hacer lo mismo con carpetas**

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

### 3.6. Mover/Renombrar Archivos y Directorios

Mover archivos y directorios en la 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 el comando "mv". Para usar este comando solo se requiere ingresar "mv origen destino", funciona igual con archivos y con carpetas.

La Figura 8 muestra un ejemplo del uso del comando mv. Para mayor información puede consultar el manual del comando: man mv.



```
danny@danny-Studio-XPS-435T-9000: ~/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2 file1 file2
danny@danny-Studio-XPS-435T-9000:~/workspace$ mv file1 file1_c
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1 dir2 file1_c file2
danny@danny-Studio-XPS-435T-9000:~/workspace$ mv dir1 dir1_c
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1_c dir2 file1_c file2
danny@danny-Studio-XPS-435T-9000:~/workspace$ mv file2 dir2/file2_mv
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls
dir1_c dir2 file1_c
danny@danny-Studio-XPS-435T-9000:~/workspace$ ls dir2/
file2_mv
danny@danny-Studio-XPS-435T-9000:~/workspace$
```

Figura 8. Uso del comando 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

### 3.7. Buscar Archivos y Directorios

El comando `find` es usado para buscar archivos o directorios en el sistema de archivos del computador. 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 dos opciones del comando:

Buscar por nombre: `find /lugar_busqueda/ -name nombre_archivo`

Buscar por tamaño: `find /lugar_busqueda/ -size tamañokb`

La Figura 9 muestra un ejemplo del uso del comando `find`. Para mayor información puede consultar el manual del comando: `man find`.

```
danny@danny-Studio-XPS-435T-9000: ~/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ pwd
/home/danny/workspace
danny@danny-Studio-XPS-435T-9000:~/workspace$ find /home/danny/workspace/ -name file2_mv
/home/danny/workspace/dir2/file2_mv
danny@danny-Studio-XPS-435T-9000:~/workspace$
```

Figura 9. Uso del comando find

Ejemplos:

#Búsqueda por nombre

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



#Busqueda por tamaño

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

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

### 3.8. Limpiar la Terminal

Después de usar un buen tiempo la terminal, es probable que nos encontremos confundidos por el texto que se encuentra desplegado en la terminal. Para limpiar la ventana podemos hacer uso del comando "clear".

Ejemplo:

#Limpiar la terminal

usuario@nombrePC:~\$ clear

# también se puede usar ctrl+l

### 3.9. Ejercicios de Autoevaluación

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 vacio? .
10. ¿Cuál es el comando (o conjunto de comandos ) para listar el contenido del directorio *jono* con sus propiedades y archivos ocultos ?

## 4. 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 de desarrollo (IDE), es solo una herramienta que se utiliza en el proceso de creación de un programa. GCC se encarga de realizar el preprocesado del código, la compilación, y el enlazado. Dicho de otra manera, nosotros proporcionamos a GCC nuestro código fuente en el lenguaje de programación C, y él nos devuelve un archivo binario compilado para nuestra arquitectura.

### 4.1. Manejo del GCC<sup>1</sup>

Vamos a compilar nuestro primer programa en lenguaje C, el programa Hola Mundo C.

La sintaxis en general para realizar una compilación de un código (un sólo archivo) usando el gcc es la siguiente:

```
$ gcc [opciones] <archivo.c>
```

Para compilar el programa `hola_mundo.c` y obtener un ejecutable llamado `hola`, se puede utilizar el siguiente comando:

```
$ gcc -o hola hola_mundo.c
```

Para verificar que el ejecutable se ha creado exitosamente, podemos hacer uso del comando `ls`.

```
$ ls -l
```

Por último, para ejecutar el programa `hola` se utiliza el siguiente comando:

```
$ ./ejecutable
```

```
# (punto)(slash)+<nombre_del_ejecutable>
```

---

<sup>1</sup> Para mayor información:

[https://www.mhe.es/universidad/informatica/8448198441/archivos/apendice\\_general\\_1.pdf](https://www.mhe.es/universidad/informatica/8448198441/archivos/apendice_general_1.pdf)

```

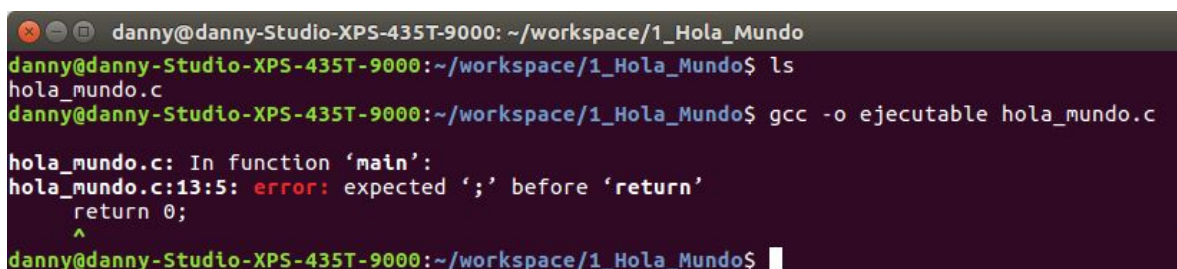
1  /* Programa Hola Mundo
2  * Esto es un comentario de varias líneas
3  * debe empezar con los caracteres (slash)(asterisco) y
4  * finalizar con (asterisco)(slash)
5  */
6
7  #include <stdio.h>
8
9  int main()
10 {
11     // Comentario de una sola línea
12     printf( "Hola mundo.\n" )
13     return 0;
14 }

```

Código 1. Programa Hola Mundo en C (para corregir)

Realice los siguientes pasos:

- Abra un editor de texto plano (gedit, sublime text o cualquier otro).
- Una vez abierto el editor, copie el contenido del programa (Código 1) y guárdelo en una ruta adecuada (por ejemplo, /home/<user>/SO/labs), el nombre del archivo debe tener extensión .c
- Compile el programa usando GCC y los parámetros adecuados para generar un ejecutable llamado ejecutable (ver Figura 10).
- Si hay errores, volver al código y corregirlos. De lo contrario ejecute el programa.



```

danny@danny-Studio-XPS-435T-9000: ~/workspace/1_Hola_Mundo
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$ ls
hola_mundo.c
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$ gcc -o ejecutable hola_mundo.c

hola_mundo.c: In function 'main':
hola_mundo.c:13:5: error: expected ';' before 'return'
    return 0;
    ^
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$

```

Figura 10. Ejemplo del manejo de gcc (con errores en el programa)

Si el código fuente tiene errores al compilar el programa (como en el caso del Código 1), gcc proporciona el nombre del archivo 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.

Para corregir el error del Código 1 sólo se necesita incluir un ; (punto y coma) al final de la línea 12). La Figura 11 muestra el resultado de la compilación y ejecución del código corregido.

```
danny@danny-Studio-XPS-435T-9000: ~/workspace/1_Hola_Mundo
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$ gcc -o ejecutable hola_mundo.c
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$ ls
ejecutable  hola_mundo.c
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$ ./ejecutable
Hola mundo.
danny@danny-Studio-XPS-435T-9000:~/workspace/1_Hola_Mundo$
```

Figura 11. Ejemplo del manejo de gcc (compilación sin errores)

La Figura 12 muestra el proceso generar para la creación y compilación de un programa en C, usando el programa GCC.

En [este enlace](#) puede visualizar la ejecución del Código 1 on-line.

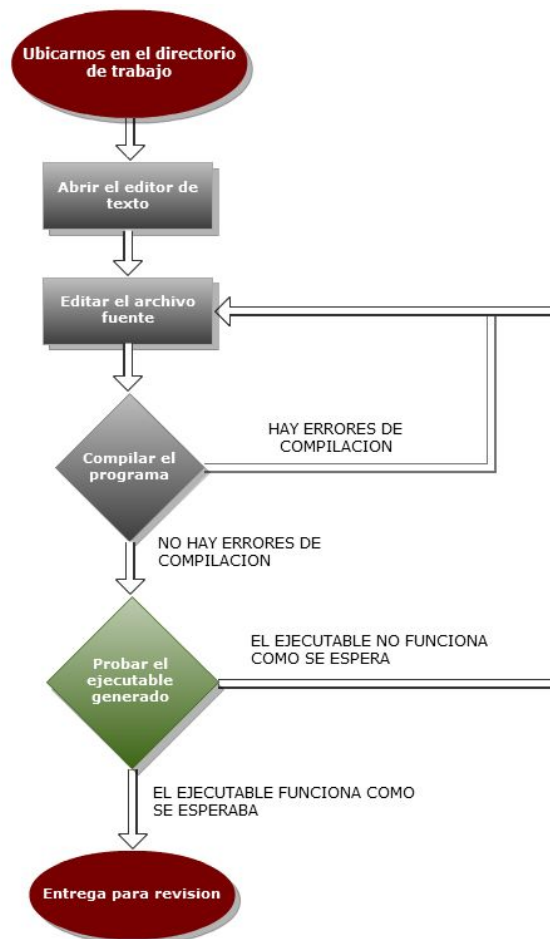


Figura 12. Flujo para crear un programa.

## 4.2. Advertencias y Errores

Cuando algo no está bien en el código C que queremos compilar, el compilador puede generar errores y advertencias (*warnings*). Cuando un error sucede al analizar el código C que se desea compilar, se impide la generación de un ejecutable final. Cuando sucede una advertencia del compilador, al analizar un código C, ésta no impide la generación de un ejecutable final.

## 4.3. Opciones más Comunes

A continuación mostramos algunas de las opciones más habituales al usar el programa 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`.