

Informática II

Programación en C bajo GNU/Linux

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC


– 2019 –

Programar en lenguaje C

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```



Programar en lenguaje C

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

- ▶ ¿Qué representa el texto de arriba? 

Programar en lenguaje C



```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

- ▶ ¿Qué representa el texto de arriba? 
- ▶ ¿Qué hay que hacer para obtener los sig. en la terminal? 

```
Hola mundo.
```

Programar en lenguaje C

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

- ▶ ¿Qué representa el texto de arriba? 
- ▶ ¿Qué hay que hacer para obtener los sig. en la terminal? 

```
Hola mundo.
```

- ▶ ¿Qué herramienta utilizan?

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software.

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida
- ▶ Etc.

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida
- ▶ Etc.

¿Cuáles conocen?

IDE – Integrated Development Environment

Geany

The screenshot shows the Geany IDE interface. The main editor window displays the following C code:

```
1  /*  
2  * hello_world.c  
3  *  
4  * Copyright 2016 Enrico Tröger <enrico@geany.org>  
5  *  
6  * This program is free software; you can redistribute it and/or modify  
7  * it under the terms of the GNU General Public License as published by  
8  * the Free Software Foundation; either version 2 of the License, or  
9  * (at your option) any later version.  
10 *  
11 * This program is distributed in the hope that it will be useful,  
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of  
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
14 * GNU General Public License for more details.  
15 *  
16 * You should have received a copy of the GNU General Public License  
17 * along with this program; if not, write to the Free Software  
18 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,  
19 * MA 02110-1301, USA.  
20 */  
21  
22 #include <stdio.h>  
23  
24 int main(int argc, char **argv)  
25 {  
26     printf("Hello World\n");  
27     printf("%d\n", i);  
28  
29     return 0;  
30 }  
31  
32
```

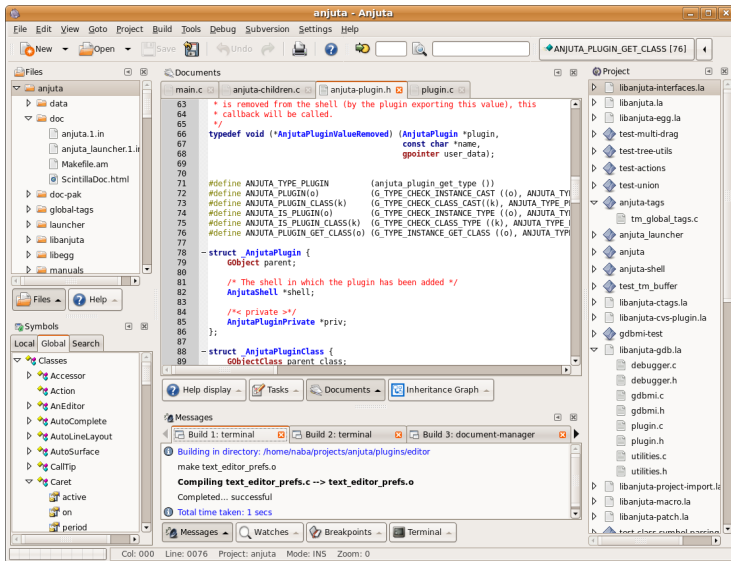
The left sidebar shows the Symbols pane with a tree view containing 'Functions' and 'main [24]'. The bottom status bar displays the following information:

| | |
|----------|--|
| Status | gcc -Wall -o "hello_world" "hello_world.c" (in directory: /home/user) |
| Compiler | hello_world.c: In function 'main': |
| Messages | hello_world.c:27:17: error: 'i' undeclared (first use in this function) |
| Scribble | printf("%d\n", i); ^ |
| Terminal | hello_world.c:27:17: note: each undeclared identifier is reported only once for each function it appears in Compilation failed. |

line: 23 / 32 col: 0 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: C scope: unknown

IDE – Integrated Development Environment

Anjuta



IDE – Integrated Development Environment

CodeLite

The screenshot shows the CodeLite IDE interface. The main editor window displays the following C++ code in `main.cpp`:

```
1 #include <iostream>
2
3 int main(int argc, char **argv)
4 {
5     std::string str;
6     str.append("hello");
7     str.append("\n");
8     str.append("world");
9     std::cout << str.c_str() << std::endl;
10    return 0;
11 }
12
```

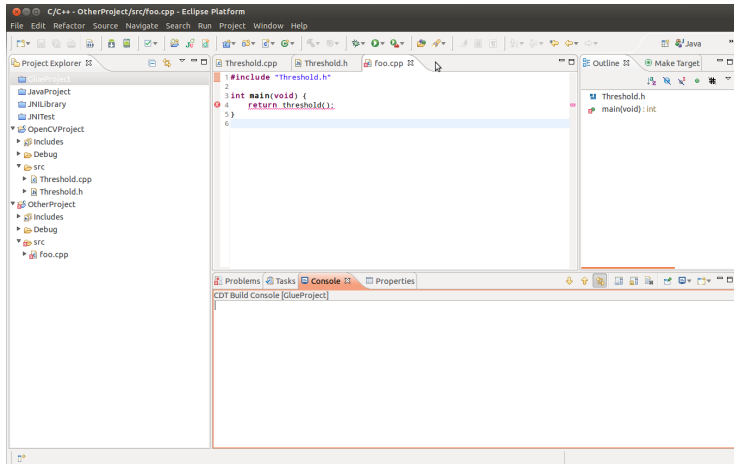
The right sidebar contains three panels:

- Breakpoints:** Shows a single breakpoint at line 9 of `/Users/eran/...ang1/main.cpp`.
- Threads:** Shows one thread with ID 24208, stopped at `step over` in the `main` function.
- Callstack:** Shows the current call stack with `main` at the top, located at `/Users/...`.

The status bar at the bottom indicates: `Done`, `Ln 11, Col 1, Pos 194, Normal bookmark`, and `Env: Default, Dbg: Default`.

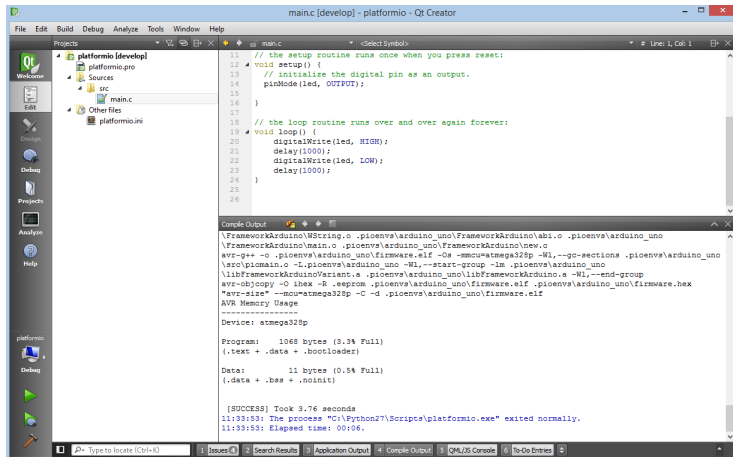
IDE – Integrated Development Environment

Eclipse CDT



IDE – Integrated Development Environment

Qt Creator



The screenshot displays the Qt Creator IDE interface. The main window is titled "main.c [develop] - platformio - Qt Creator". The interface includes a menu bar (File, Edit, Build, Debug, Analyze, Tools, Window, Help), a toolbar, and a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The central area is divided into three panes:

- Project Explorer:** Shows the project structure for "platformio [develop]". It includes "platformio.pro", "Sources" (with a sub-folder "src" containing "main.c"), and "Other files" (containing "platformio.ini").
- Code Editor:** Displays the source code for "main.c". The code is as follows:

```
11 // the setup routine runs once when you press reset:
12 void setup() {
13     // initialize the digital pin as an output.
14     pinMode(led, OUTPUT);
15 }
16
17
18 // the loop routine runs over and over again forever:
19 void loop() {
20     digitalWrite(led, HIGH);
21     delay(1000);
22     digitalWrite(led, LOW);
23     delay(1000);
24 }
25
26
```
- Compile Output:** Shows the compilation and execution results:

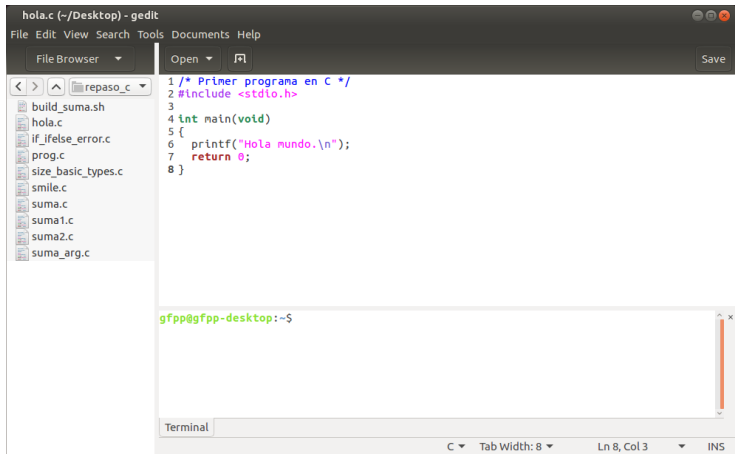
```
\FrameworkArduino\WString.o .pioenvs\arduino_uno\FrameworkArduino\abi.o .pioenvs\arduino_uno
\FrameworkArduino\main.o .pioenvs\arduino_uno\FrameworkArduino\new.o
avr-g++ -o .pioenvs\arduino_uno\firmware.elf -Os -mmcu=atmega328p -Wl,--gc-sections .pioenvs\arduino_uno
\src\main.o -L.pioenvs\arduino_uno -Wl,--start-group -lm .pioenvs\arduino_uno
\lib\FrameworkArduino\variant.a .pioenvs\arduino_uno\lib\FrameworkArduino.a -Wl,--end-group
avr-objcopy -O ihex -R .eeprom .pioenvs\arduino_uno\firmware.elf .pioenvs\arduino_uno\firmware.hex
"avr-size" --mcu=atmega328p -C -d .pioenvs\arduino_uno\firmware.elf
AVR Memory Usage
-----
Device: atmega328p
Program: 1068 bytes (3.3% Full)
(.text + .data + .bootloader)
Data: 11 bytes (0.5% Full)
(.data + .bss + .noinit)

[SUCCESS] Took 3.76 seconds
11:33:53: The process "C:\Python27\Scripts\platformio.exe" exited normally.
11:33:53: Elapsed time: 00:06.
```

At the bottom, a status bar shows navigation shortcuts: 1 Issues, 2 Search Results, 3 Application Output, 4 Compile Output, 5 QML/JS Console, 6 To-Do Entries.

IDE – Integrated Development Environment

gedit

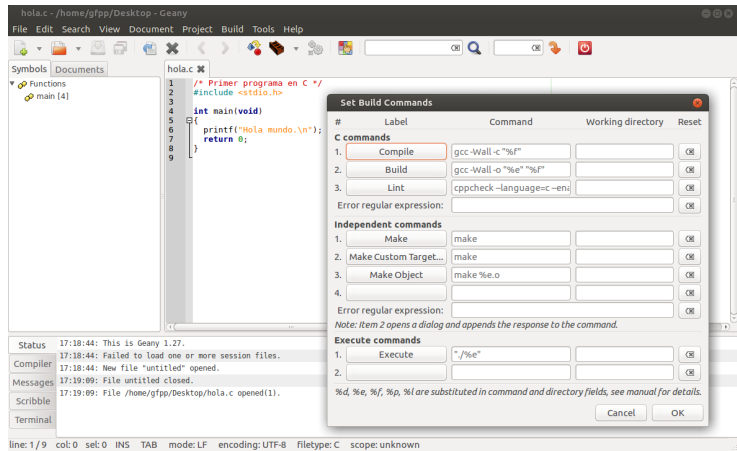


The screenshot shows the gedit IDE window titled "hola.c (~/Desktop) - gedit". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains a File Browser dropdown, an Open button with a folder icon, and a Save button. The left sidebar shows a file browser for the "repaso_c" directory, listing files: build_suma.sh, hola.c, if_ifelse_error.c, prog.c, size_basic_types.c, smile.c, suma.c, suma1.c, suma2.c, and suma_arg.c. The main editor area displays the following C code:

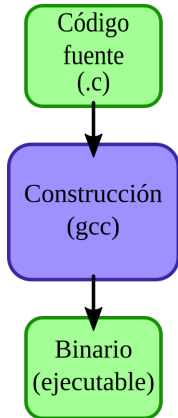
```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Below the editor is a terminal window with the prompt "gfpp@gfpp-desktop:~\$". At the bottom of the window, there are status indicators: "C", "Tab Width: 8", "Ln 8, Col 3", and "INS".

Ejemplo de configuración del IDE Geany

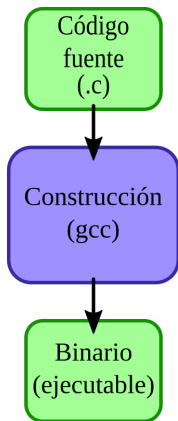


Construcción de un programa



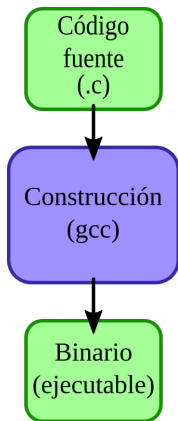
Construcción de un programa

1. Escribir el código fuente



```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola_mundo.\n");
7     return 0;
8 }
```

Construcción de un programa

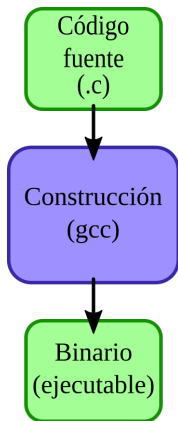


1. Escribir el código fuente

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola_mundo.\n");
7     return 0;
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

Construcción de un programa



1. Escribir el código fuente

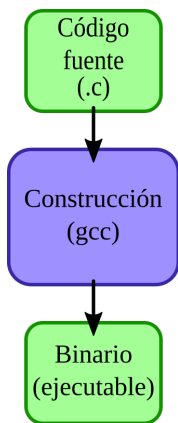
```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola_mundo.\n");  
7     return 0;  
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

3. Construir el programa

```
> gcc hola.c
```

Construcción de un programa



1. Escribir el código fuente

```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola mundo.\n");  
7     return 0;  
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

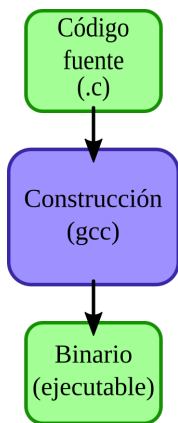
3. Construir el programa

```
> gcc hola.c
```

4. Ejecutar el programa

```
> ./a.out  
Hola mundo.
```


Construcción de un programa



1. Escribir el código fuente

```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola_mundo.\n");  
7     return 0;  
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

3. Construir el programa

```
> gcc hola.c
```

4. Ejecutar el programa

```
> ./a.out  
Hola mundo.
```

En la cátedra se utilizará este procedimiento de compilación

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out)

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Introducción a gcc

hola.c

```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola mundo.\n");  
7     return 0;  
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

```
> gcc -Wall hola.c -o hola
```


Introducción a gcc

hola.c

```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola_mundo.\n");  
7     return 0;  
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

```
> gcc -Wall hola.c -o hola
```

(habilita todas las advertencias/warnings)

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Error de compilación (-Wall)

```
mal.c:5:10: warning: format '%f'
expects argument of type 'double',
but argument 2 has type 'int'
[-Wformat=]
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4.); // CORREGIDO
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Error de compilación (-Wall)

```
mal.c:5:10: warning: format '%f'
expects argument of type 'double',
but argument 2 has type 'int'
[-Wformat=]
```


Actividad práctica

1. Reproducir el ejemplo anterior (*Hola mundo!*) en GNU/Linux
 - ▶ Editar el archivo `hola.c` con `gedit`
 - ▶ Abrir una terminal (p.e. `gnome-terminal`) e ir al directorio donde se encuentra el archivo fuente (comando `cd`)
 - ▶ Compilar el programa con `gcc`
 - ▶ Ejecutar

Actividad práctica

1. Reproducir el ejemplo anterior (*Hola mundo!*) en GNU/Linux
 - ▶ Editar el archivo `hola.c` con `gedit`
 - ▶ Abrir una terminal (p.e. `gnome-terminal`) e ir al directorio donde se encuentra el archivo fuente (comando `cd`)
 - ▶ Compilar el programa con `gcc`
 - ▶ Ejecutar
2. Configurar `gedit` para poder programar en C
 - ▶ Habilitar la numeración de líneas y el resaltado de sintaxis
 - ▶ Habilitar la terminal embebida

Actividad práctica

1. Reproducir el ejemplo anterior (*Hola mundo!*) en GNU/Linux
 - ▶ Editar el archivo `hola.c` con `gedit`
 - ▶ Abrir una terminal (p.e. `gnome-terminal`) e ir al directorio donde se encuentra el archivo fuente (comando `cd`)
 - ▶ Compilar el programa con `gcc`
 - ▶ Ejecutar
2. Configurar `gedit` para poder programar en C
 - ▶ Habilitar la numeración de líneas y el resaltado de sintaxis
 - ▶ Habilitar la terminal embebida
3. Descargar, configurar y probar algunos de los IDE antes mencionados

Actividad práctica

1. Reproducir el ejemplo anterior (*Hola mundo!*) en GNU/Linux
 - ▶ Editar el archivo `hola.c` con `gedit`
 - ▶ Abrir una terminal (p.e. `gnome-terminal`) e ir al directorio donde se encuentra el archivo fuente (comando `cd`)
 - ▶ Compilar el programa con `gcc`
 - ▶ Ejecutar
2. Configurar `gedit` para poder programar en C
 - ▶ Habilitar la numeración de líneas y el resaltado de sintaxis
 - ▶ Habilitar la terminal embebida
3. Descargar, configurar y probar algunos de los IDE antes mencionados
4. Codificar un programa para sumar dos números enteros con interacción con el usuario

Actividad práctica

1. Reproducir el ejemplo anterior (*Hola mundo!*) en GNU/Linux
 - ▶ Editar el archivo `hola.c` con `gedit`
 - ▶ Abrir una terminal (p.e. `gnome-terminal`) e ir al directorio donde se encuentra el archivo fuente (comando `cd`)
 - ▶ Compilar el programa con `gcc`
 - ▶ Ejecutar
2. Configurar `gedit` para poder programar en C
 - ▶ Habilitar la numeración de líneas y el resaltado de sintaxis
 - ▶ Habilitar la terminal embebida
3. Descargar, configurar y probar algunos de los IDE antes mencionados
4. Codificar un programa para sumar dos números enteros con interacción con el usuario
5. Codificar un programa para sumar dos números enteros sin interacción con el usuario

