

Introduction to C

Natalia Bidart

FaMAF - UNC

27 de Abril de 2011

Intro

Source code (código fuente)

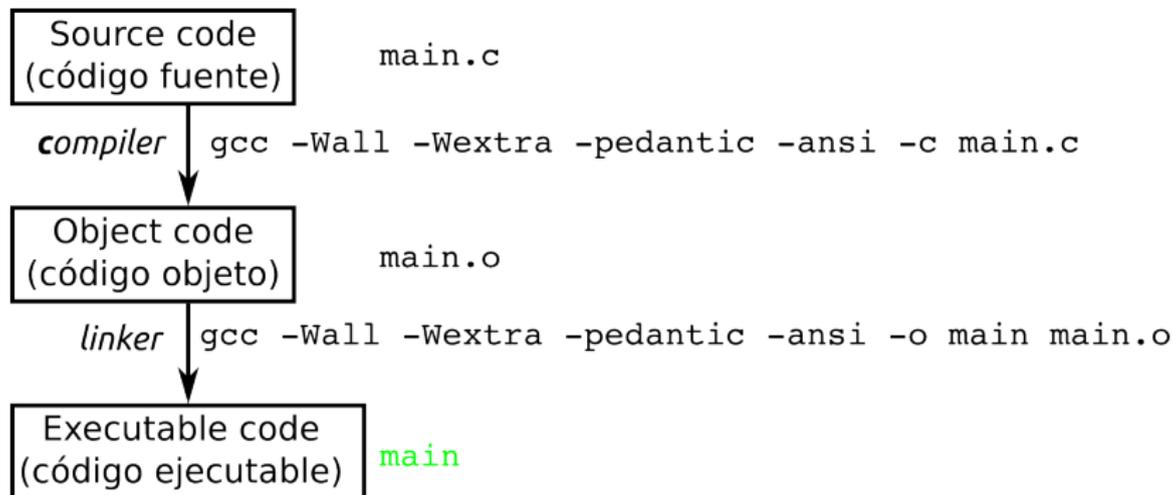
Compile

Link

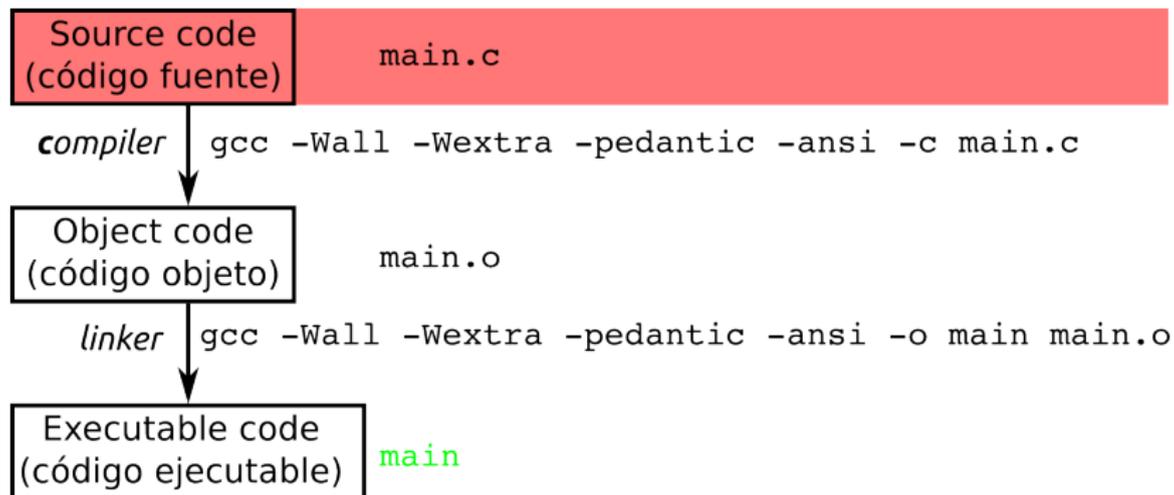
Execution

Specifics

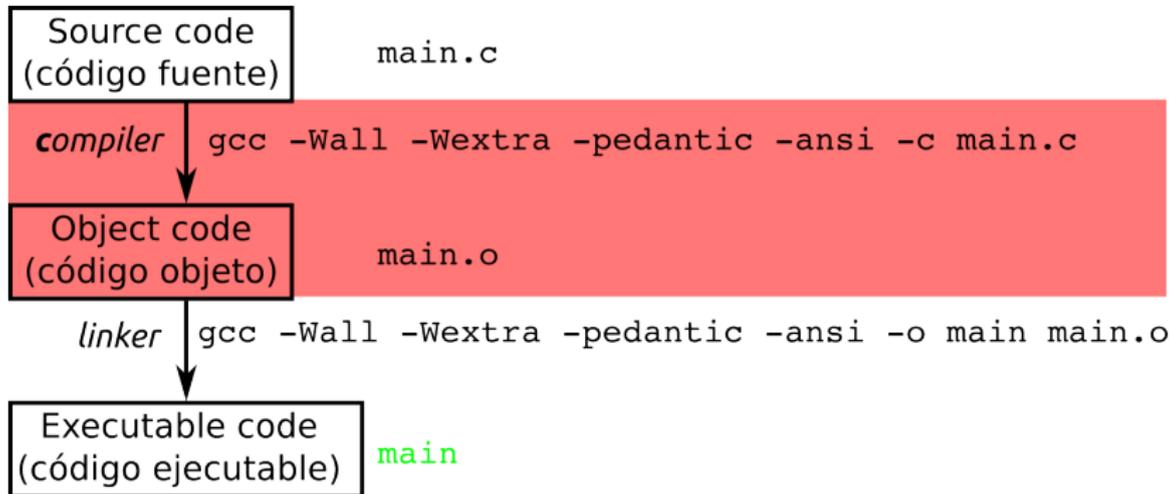
C code lifecycle (el ciclo de vida de código C)



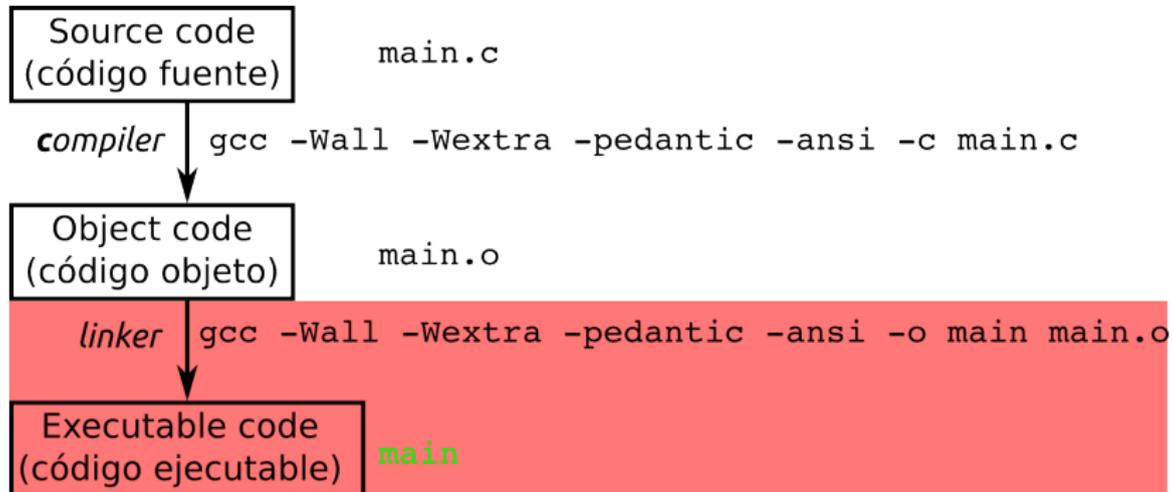
C code lifecycle (el ciclo de vida de código C)



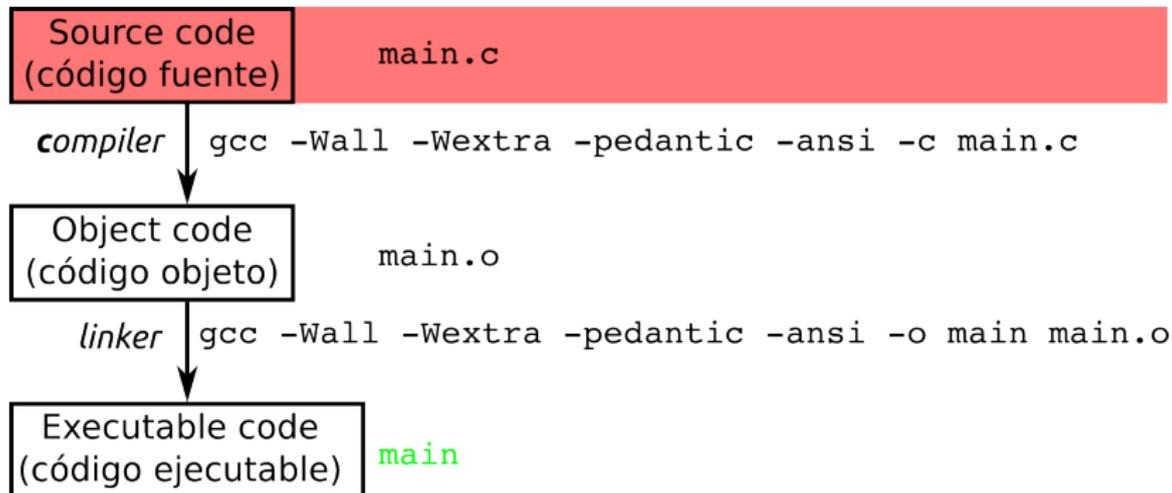
C code lifecycle (el ciclo de vida de código C)



C code lifecycle (el ciclo de vida de código C)

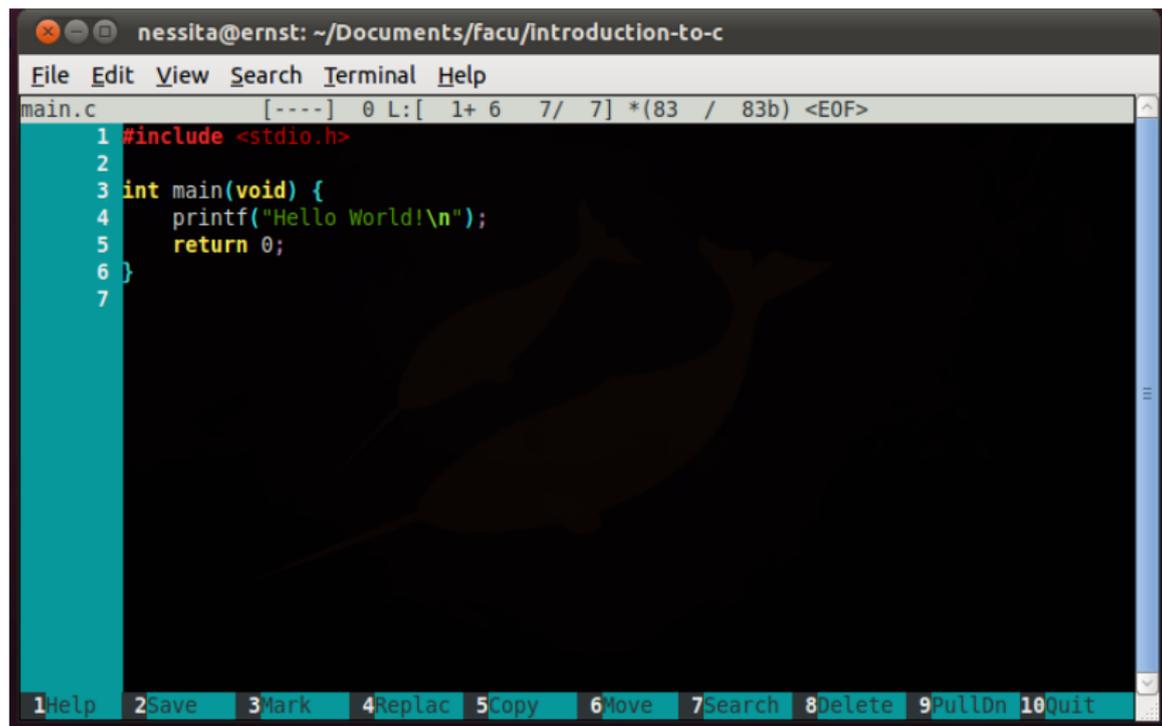


Source code (código fuente)



The simplest program ever

(el programa más simple de todos)

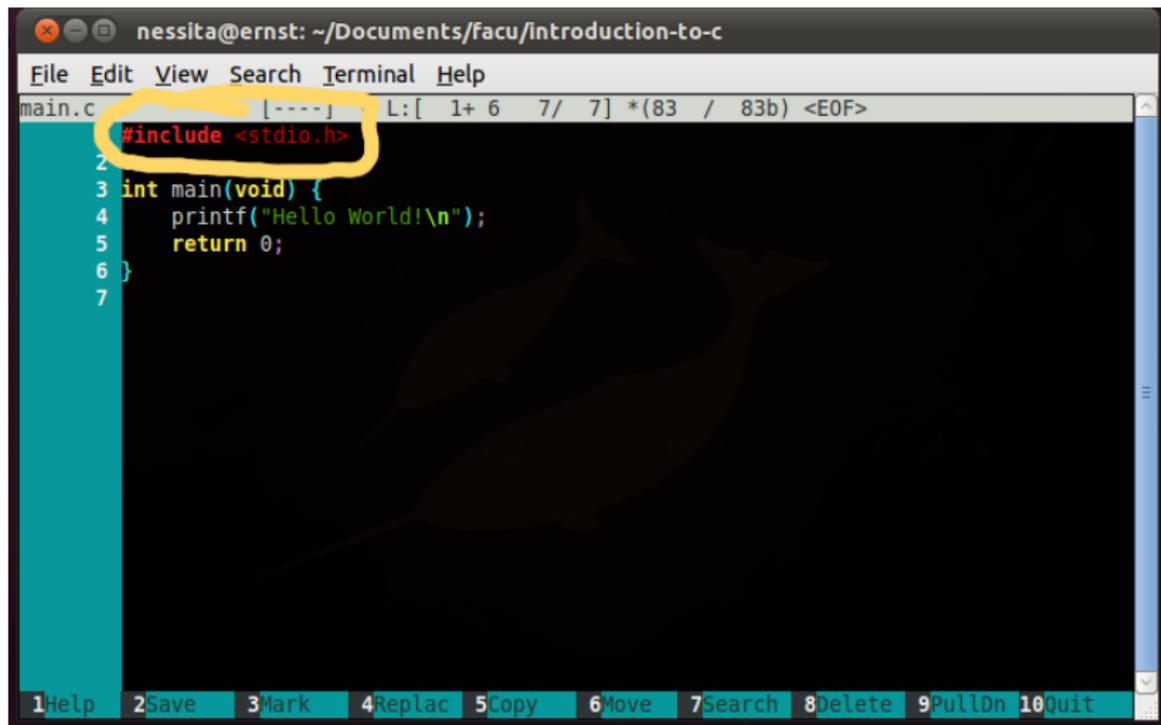


The image shows a terminal window with a dark background and light-colored text. The window title is "nessita@ernst: ~/Documents/facu/introduction-to-c". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The file name "main.c" is visible in the top left. The code is as follows:

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

At the bottom of the terminal, there is a status bar with the following items: 1Help, 2Save, 3Mark, 4Replac, 5Copy, 6Move, 7Search, 8Delete, 9PullOn, 10Quit.

Including libraries (incluyendo bibliotecas)

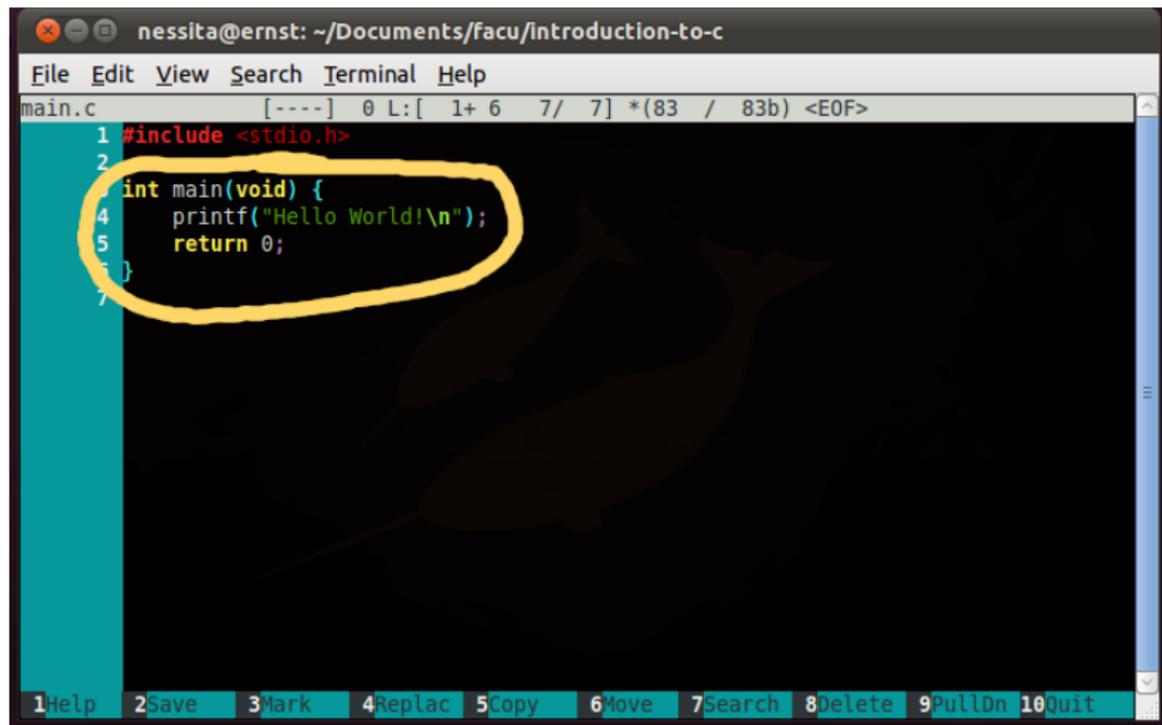


The image shows a terminal window with a dark background and a light-colored title bar. The title bar contains the text "nessita@ernst: ~/Documents/facu/introduction-to-c". Below the title bar is a menu bar with the items "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal displays the following code:

```
main.c [----] L:[ 1+ 6 7/ 7] *(83 / 83b) <EOF>
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

The line `#include <stdio.h>` is highlighted with a yellow circle. At the bottom of the terminal window, there is a status bar with the following text: "1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9pullOn 10Quit".

The entry point (el punto de entrada)

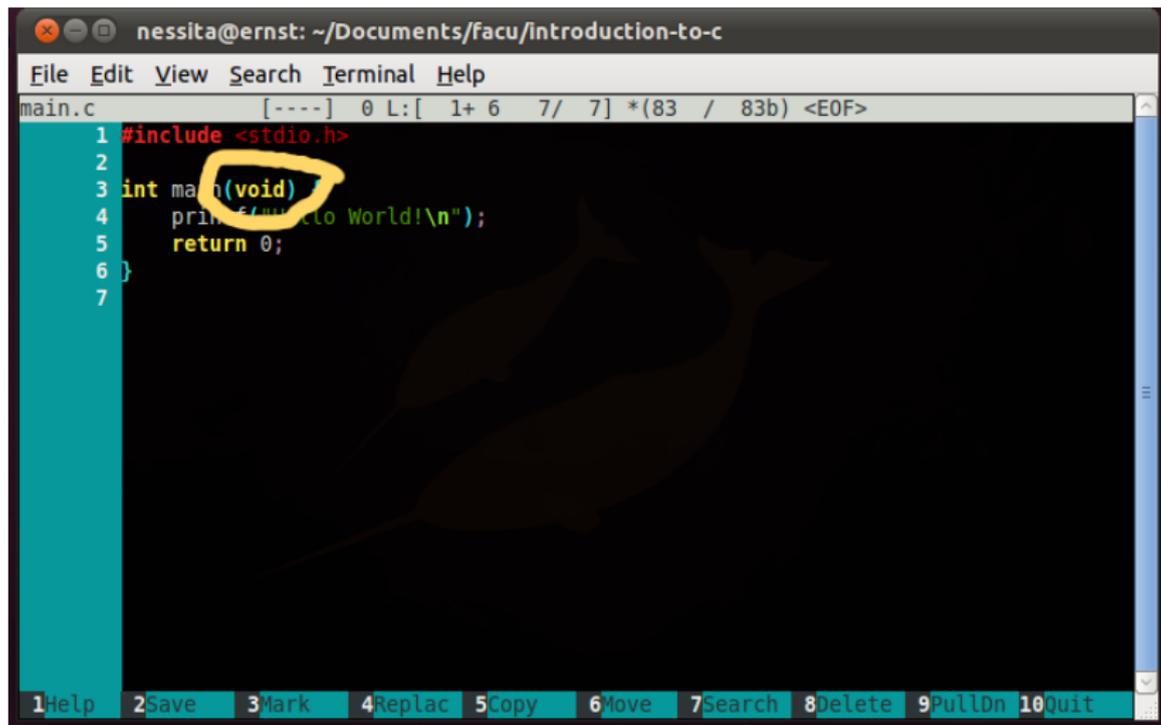


The image shows a terminal window with a text editor. The window title is "nessita@ernst: ~/Documents/facu/introduction-to-c". The editor has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The file name is "main.c". The code is as follows:

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

The code block from line 3 to line 6 is circled in yellow. At the bottom of the terminal, there is a status bar with numbered shortcuts: 1Help, 2Save, 3Mark, 4Replac, 5Copy, 6Move, 7Search, 8Delete, 9pullOn, 10Quit.

Explicitly saying no args
(explícitamente no aceptando argumentos)



```
nessita@ernst: ~/Documents/facu/introduction-to-c
File Edit View Search Terminal Help
main.c [----] 0 L:[ 1+ 6 7/ 7] *(83 / 83b) <EOF>
1 #include <stdio.h>
2
3 int main(void)
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

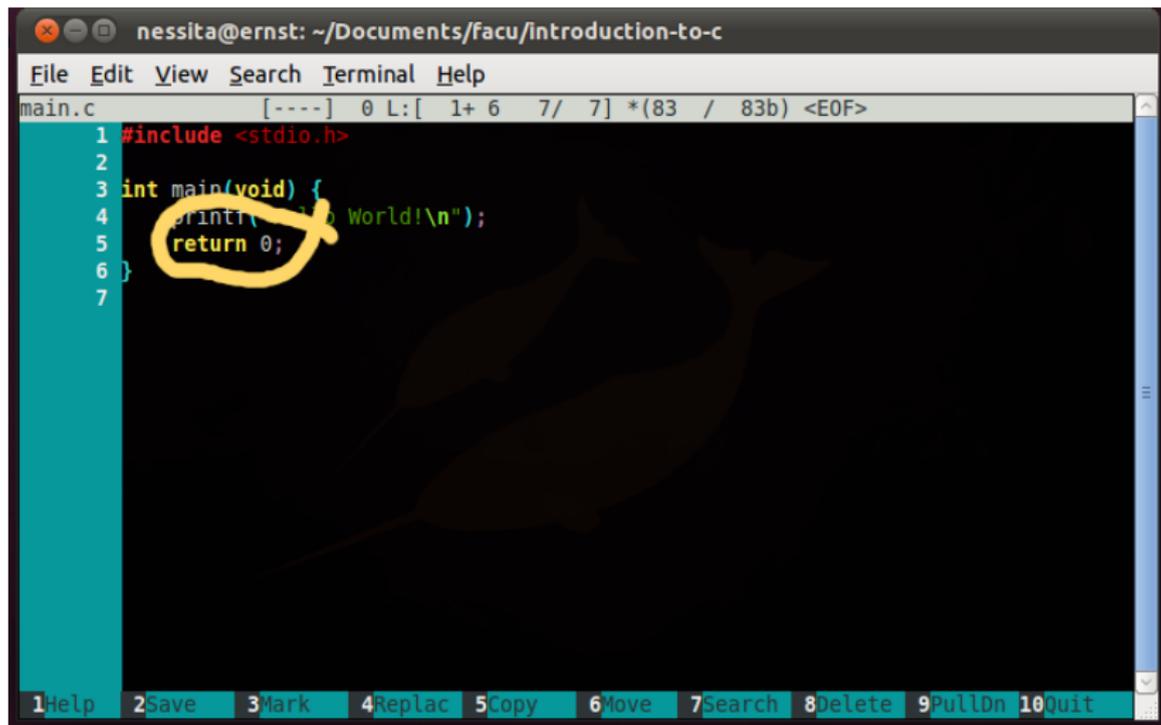
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullOn 10Quit

Interacting with the user (interactuando con el usuario)

```
nessita@ernst: ~/Documents/facu/introduction-to-c
File Edit View Search Terminal Help
main.c [----] 0 L:[ 1+ 6 7/ 7] *(83 / 83b) <EOF>
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9pullOn 10Quit

Returning values (devolviendo valores como resultado)

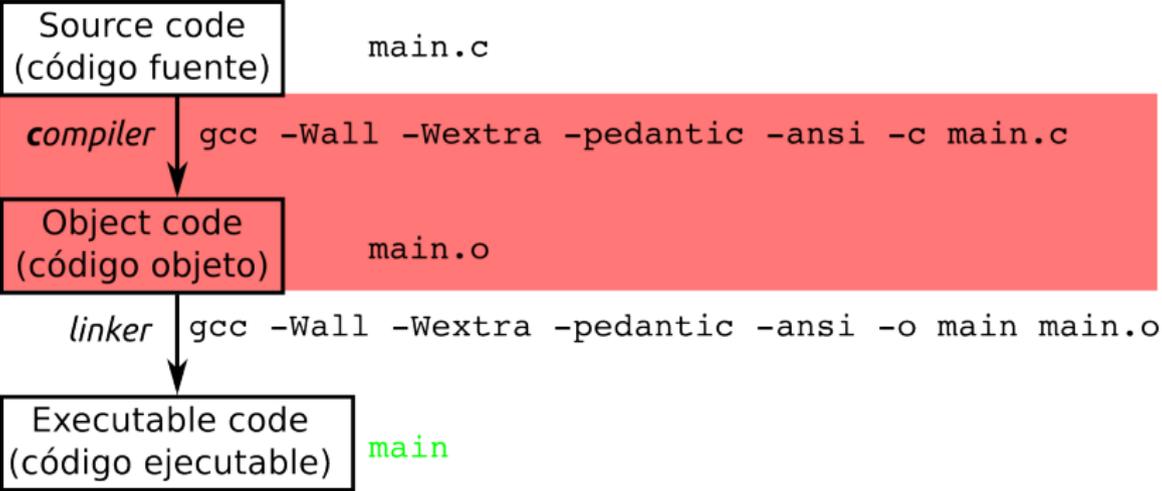


The image shows a terminal window with a dark background and a light blue sidebar on the left. The window title is "nessita@ernst: ~/Documents/facu/introduction-to-c". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows a C program in "main.c" with the following code:

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World!\n");
5     return 0;
6 }
7
```

The line "return 0;" on line 5 is circled in yellow. At the bottom of the terminal, there is a status bar with the following items: 1Help, 2Save, 3Mark, 4Replac, 5Copy, 6Move, 7Search, 8Delete, 9PullOn, 10Quit.

Compile



Compile

```
$ gcc -Wall -Wextra -pedantic -ansi -c main.c
```

`gcc` The C compiler (el compilador de C)

`-Wall` All the warnings (todas las advertencias)

`-Wextra` Extra warnings (advertencias extras)

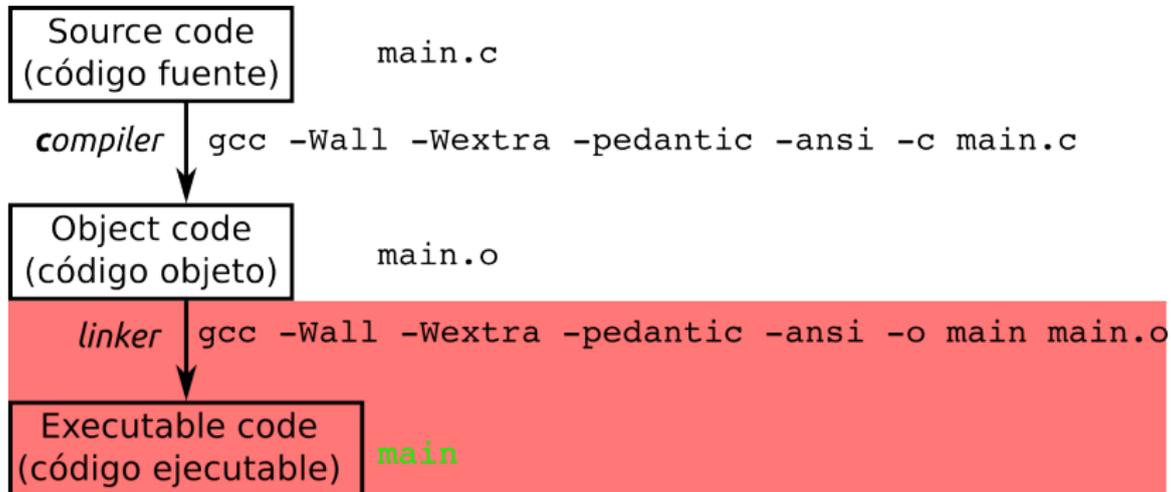
`-pedantic` The strict warnings (advertencias estrictas)

`-ansi` Standard C code (código C estándar)

`-c` Compile (compilar)

`main.c` One or more source code files (uno o más archivos de código fuente)

Link



Link

```
$ gcc -Wall -Wextra -pedantic -ansi -o main main.o
```

`gcc` The C linker (el linker de C)

`-Wall` All the warnings (todas las advertencias)

`-Wextra` Extra warnings (advertencias extras)

`-pedantic` The strict warnings (advertencias estrictas)

`-ansi` Standard C code (código C estándar)

`-o` Output executable filename, “a.out” by default (nombre del archivo de salida ejecutable, “a.out” por defecto)

`main.o` One or more object code files (uno o más archivos de código objeto)

Execution (ejecución)

```
nessita@ernst:~/Documents/facu/introduction-to-c$ ./main
Hello World!
nessita@ernst:~/Documents/facu/introduction-to-c$
```

Variable declaration (declaración de variables)

```
<type> <variable_name>;
```

```
int i;  
char c;  
double f;  
int *p;
```

- ▶ El estándar C90 requiere que las declaraciones de variables no se mezclen con código.
- ▶ Builtin types (los tipos del lenguaje):
 - ▶ int, unsigned int, long int, etc.
 - ▶ double
 - ▶ char
 - ▶ void
 - ▶ pointers (*)

Assignment (asignación)

=

```
int i, j;  
char c;  
i = 10;  
j = 500;  
c = 'a';  
/* esto es un comentario */
```

- ▶ Se puede declarar variables y asignarles un valor al mismo tiempo (siempre y cuando el valor sea conocido) con:

```
int i = 10, j = 500;  
char c = 'a';
```

Data Input (entrada de datos)

scanf

```
int i = 0;  
conversions = scanf("%d", &i);
```

- ▶ `stdin` representa la entrada estándar, lo que se ingresa por teclado.
- ▶ El primer argumento es lo que llamamos el “especificador de conversión”, que indica qué conversión hay que aplicarle a lo leído de la entrada estándar.
- ▶ El segundo argumento es un lugar en memoria para almacenar lo leído (ojo con los tamaños!).
- ▶ El resultado `conversions` indica cuántas conversiones exitosas se hicieron.

Data Output (salida de datos)

printf

```
int i = 0;  
printed = printf("Lo leído es: %d\n", i);
```

- ▶ stdout representa la salida estándar, lo que se lee en la pantalla.
- ▶ El primer argumento es el texto a mostrar en la salida estándar, puede incluir especificadores de conversión (iguales que los que usa scanf).
- ▶ Éstos indican qué conversión hay que aplicarle a los parámetros subsiguientes.
- ▶ El resultado `printed` indica cuántos caracteres fueron exitosamente impresos en la pantalla.

Man pages (páginas de manual)

```
$ man scanf
```

```
$ man 3 printf
```

Paquetes necesarios:

- ▶ manpages-dev
- ▶ manpages-es (traducciones al español)
- ▶ glibc-doc (para info, que es otro comando para consultar manuales)

Operaciones básicas

- + Sum (suma)
- Substraction (resta)
- * Multiplication (multiplicación)
- / Float division (división flotante)
- % Module (módulo)

Tarea

Escribir un programa que pida al usuario un entero, y muestre por pantalla el cuadrado del entero ingresado.