

# Informática II

El compilador de C del proyecto GNU (`gcc`, `g++`)

Gonzalo F. Pérez Paina



Universidad Tecnológica Nacional  
Facultad Regional Córdoba  
UTN-FRC

– 2018 –

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

- ▶ Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), etc.

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

- ▶ Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), etc.
- ▶ Puede compilar C++

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

- ▶ Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), etc.
- ▶ Puede compilar C++
- ▶ Realiza la optimización del código

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

- ▶ Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), etc.
- ▶ Puede compilar C++
- ▶ Realiza la optimización del código
- ▶ Genera información de depuración (debugging)

# El compilador de C del proyecto GNU

## Herramienta de compilación

Herramientas de compilación del proyecto GNU ([GNU Compiler Collection](#)):

- ▶ Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), etc.
- ▶ Puede compilar C++
- ▶ Realiza la optimización del código
- ▶ Genera información de depuración (debugging)
- ▶ Es un compilador cruzado (cross-compiler)

```
$ gcc --version
```

```
$ gcc -v
```

```
(--build, --host, --target)
```

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

## Compilación

```
> gcc hola.c
```

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

## Compilación

```
> gcc hola.c
```

(salida a.out)

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

## Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta?

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

## Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

`hola.c`

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

### Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

`hola.c`

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

### Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

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

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

## Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

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

(habilita todas las advertencias)

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

`hola.c`

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

`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 }
```

## Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

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

(habilita todas las advertencias)

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

`hola.c`

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

`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 }
```

### Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

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

(habilita todas las advertencias)

### Compilar y ejecutar

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

# El compilador de C del proyecto GNU

## Compilando - Algunos ejemplos

hola.c

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

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 }
```

### Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

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

(habilita todas las advertencias)

### 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=]
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, ¡%s!\n", nombre);
7 }
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, \u0021%s!\n", nombre);
7 }
```

hola\_fn.h

```
1 void hola(const char * nombre);
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, \u0021%s!\n", nombre);
7 }
```

hola\_fn.h

```
1 void hola(const char * nombre);
```

## Compilación

```
> gcc -Wall main.c hola_fn.c -o nuevohola
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, ¡%s!\n", nombre);
7 }
```

hola\_fn.h

```
1 void hola(const char * nombre);
```

## Compilación

```
> gcc -Wall main.c hola_fn.c -o nuevohola
```

Se puede compilar separadamente cada archivo fuente

```
> gcc -Wall -c main.c
> gcc -Wall -c hola_fn.c
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, ¡%s!\n", nombre);
7 }
```

hola\_fn.h

```
1 void hola(const char * nombre);
```

## Compilación

```
> gcc -Wall main.c hola_fn.c -o nuevohola
```

Se puede compilar separadamente cada archivo fuente

```
> gcc -Wall -c main.c
> gcc -Wall -c hola_fn.c
```

y unirlo con el linker

```
> gcc main.o hola_fn.o -o nuevohola
```

# El compilador de C del proyecto GNU

## Compilando - Ejemplos con varios archivos fuentes

main.c

```
1 #include "hola_fn.h"
2
3 int main(void)
4 {
5     hola("mundo");
6     return 0;
7 }
```

hola\_fn.c

```
1 #include <stdio.h>
2 #include "hola_fn.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, \u25b6%s!\n", nombre);
7 }
```

hola\_fn.h

```
1 void hola(const char * nombre);
```

## Compilación

```
> gcc -Wall main.c hola_fn.c -o nuevohola
```

Se puede compilar separadamente cada archivo fuente

```
> gcc -Wall -c main.c
> gcc -Wall -c hola_fn.c
```

y unirlo con el linker

```
> gcc main.o hola_fn.o -o nuevohola
```

Esto permite modificar un archivo fuente y recompilar solo el archivo modificado.



# El compilador de C del proyecto GNU

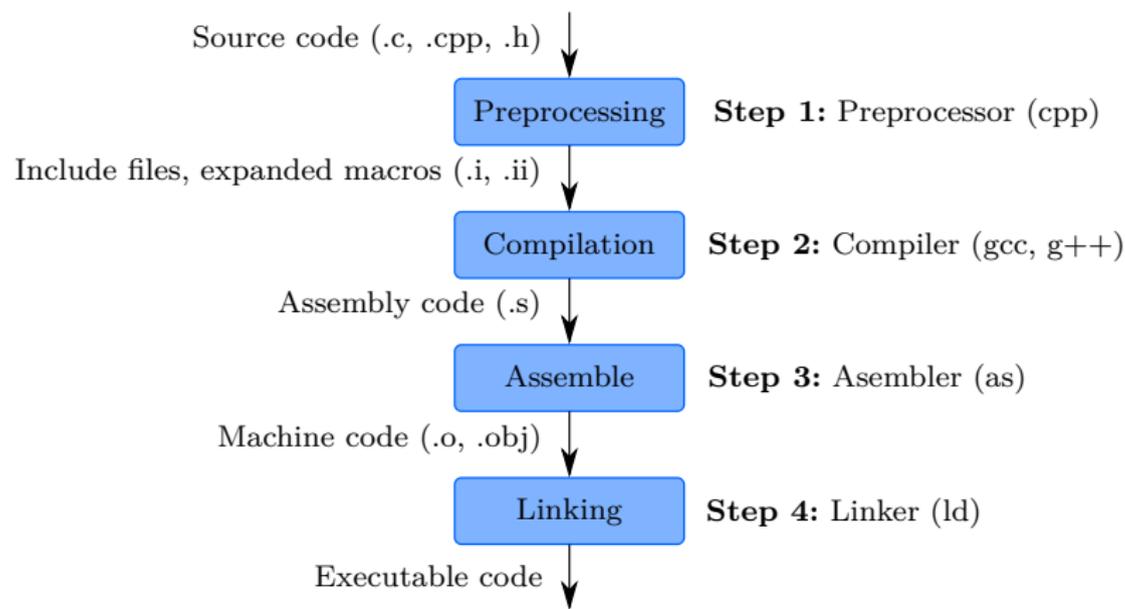
## Etapas de compilación

El proceso de compilación/construcción involucra **4 etapas** (preprocesamiento, compilación, ensamblado, y linkeo)

# El compilador de C del proyecto GNU

## Etapas de compilación

El proceso de compilación/construcción involucra **4 etapas** (preprocesamiento, compilación, ensamblado, y linkeo)



# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

test.c

---

```
1 #define TEST "Hola_mundo!"  
2 const char str[] = TEST;
```

---

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

test.c

---

```
1 #define TEST "Hola_mundo!"  
2 const char str[] = TEST;
```

---

```
> gcc -E test.c
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

test.c

```
1 #define TEST "Hola_mundo!"  
2 const char str[] = TEST;
```

```
> gcc -E test.c
```

```
# 1 "test.c"  
# 1 "<built-in>"  
# 1 "<command-line>"  
# 1 "/usr/include/stdc-predef.h" 1 3 4  
# 1 "<command-line>" 2  
# 1 "test.c"  
  
const char str[] = "Hola, Mundo!";
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

test.c

```
1 #define TEST "Hola_mundo!"  
2 const char str[] = TEST;
```

```
> gcc -E test.c
```

```
# 1 "test.c"  
# 1 "<built-in>"  
# 1 "<command-line>"  
# 1 "/usr/include/stdc-predef.h" 1 3 4  
# 1 "<command-line>" 2  
# 1 "test.c"  
  
const char str[] = "Hola, Mundo!";
```

- ▶ Opción `-E`: `gcc` ejecuta el preprocesador, muestra la salida, y termina sin compilar el código fuente resultante
- ▶ Además, el preprocesador inserta líneas de registro de archivo fuente y el nro. de línea en la forma `#num-de-linea "archivo fuente"`

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

hello.c

---

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

---

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

hello.c

---

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

---

```
> gcc -E hello.c -o hello.i
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

hello.c

---

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

---

```
> gcc -E hello.c -o hello.i
```

```
> cpp hello.c
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

hello.c

---

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

---

```
> gcc -E hello.c -o hello.i
```

```
> cpp hello.c
```

## Copiar el archivo fuente y modificarlo

---

```
1 /* Hello Linux */
2 #include <stdio.h>
3 #define STRING "Hello, Linux programming world!\n"
4
5 int main(void)
6 {
7     /* Using a macro to print a message */
8     printf(STRING);
9     return 0;
10 }
```

---

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Preprocesado

hello.c

---

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

---

```
> gcc -E hello.c -o hello.i
```

```
> cpp hello.c
```

## Copiar el archivo fuente y modificarlo

---

```
1 /* Hello Linux */
2 #include <stdio.h>
3 #define STRING "Hello, Linux programming world!\n"
4
5 int main(void)
6 {
7     /* Using a macro to print a message */
8     printf(STRING);
9     return 0;
10 }
```

---

**Observar:** macros, comentarios, archivos cabecera (includes)

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Compilación

Compilar continuando de la etapa anterior

```
> gcc -x cpp-output -c hello.i -o hello.o
```

La opción `-x` le indica que continúe desde la etapa indicada (man)

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Compilación

Compilar continuando de la etapa anterior

```
> gcc -x cpp-output -c hello.i -o hello.o
```

La opción `-x` le indica que continúe desde la etapa indicada (man)

```
> file hello.o
hello.o: ELF 64-bit LSB relocatable, x86-64,
version 1 (SYSV), not stripped
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida `hello`)

```
> gcc hello.o -o hello
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida hello)

```
> gcc hello.o -o hello
```

```
> file hello
```

```
hello: ELF 64-bit LSB executable, x86-64, version 1  
(SYSV), dynamically linked, interpreter /lib64/ld-  
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]  
=620d3c9fadabd53755c0a647c0a43a172481a6f8, not stripped
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida hello)

```
> gcc hello.o -o hello
```

```
> file hello
```

```
hello: ELF 64-bit LSB executable, x86-64, version 1  
(SYSV), dynamically linked, interpreter /lib64/ld-  
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]  
=620d3c9fadabd53755c0a647c0a43a172481a6f8, not stripped
```

```
> ldd hello
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida hello)

```
> gcc hello.o -o hello
```

```
> file hello
```

```
hello: ELF 64-bit LSB executable, x86-64, version 1  
(SYSV), dynamically linked, interpreter /lib64/ld-  
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]  
=620d3c9fadabd53755c0a647c0a43a172481a6f8, not stripped
```

```
> ldd hello
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

(objdump -x, readelf -d)

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida hello)

```
> gcc hello.o -o hello
```

```
> file hello
```

```
hello: ELF 64-bit LSB executable, x86-64, version 1  
(SYSV), dynamically linked, interpreter /lib64/ld-  
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]  
=620d3c9fadabd53755c0a647c0a43a172481a6f8, not stripped
```

```
> ldd hello
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

(objdump -x, readelf -d)

Flag `-save-temps` genera archivos intermedios

```
> gcc -save-temps hello.c -o hello
```

# El compilador de C del proyecto GNU

## Etapas de compilación paso a paso –Linkeo

Linkeo con biblioteca estándar (salida hello)

```
> gcc hello.o -o hello
```

```
> file hello
```

```
hello: ELF 64-bit LSB executable, x86-64, version 1  
(SYSV), dynamically linked, interpreter /lib64/ld-  
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]  
=620d3c9fadabd53755c0a647c0a43a172481a6f8, not stripped
```

```
> ldd hello
```

```
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

(objdump -x, readelf -d)

Flag `-save-temps` genera archivos intermedios

```
> gcc -save-temps hello.c -o hello
```

(Flags `-std=c90`, `-Wall`, `-Werror`)



# El compilador de C del proyecto GNU

## El preprocesador –Macros y compilación condicional

dtest.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5 #ifdef TEST
6     printf("Modo_test\n");
7 #endif
8     printf("Ejecutando...\n");
9     return 0;
10 }
```

---

# El compilador de C del proyecto GNU

## El preprocesador –Macros y compilación condicional

dtest.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5 #ifdef TEST
6     printf("Modo_test\n");
7 #endif
8     printf("Ejecutando...\n");
9     return 0;
10 }
```

---

```
> gcc -Wall dtest.c
> ./a.out
Ejecutando...
```

# El compilador de C del proyecto GNU

## El preprocesador –Macros y compilación condicional

dtest.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5 #ifdef TEST
6     printf("Modo_test\n");
7 #endif
8     printf("Ejecutando...\n");
9     return 0;
10 }
```

---

```
> gcc -Wall dtest.c
> ./a.out
Ejecutando...
```

Macro definida desde la línea de comandos

```
> gcc -Wall -DTEST dtest.c
> ./a.out
Modo test
Ejecutando...
```

# El compilador de C del proyecto GNU

## El preprocesador –Macros y compilación condicional

dtest.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5 #ifdef TEST
6     printf("Modo_test\n");
7 #endif
8     printf("Ejecutando...\n");
9     return 0;
10 }
```

---

```
> gcc -Wall dtest.c
> ./a.out
Ejecutando...
```

Macro definida desde la línea de comandos

```
> gcc -Wall -DTEST dtest.c
> ./a.out
Modo test
Ejecutando...
```

(Macros predefinidas: `cpp -dM /dev/null`)

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor

dtestval.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("El valor de NUM es %d\n", NUM);
6     return 0;
7 }
```

---

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor

dtestval.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("El valor de NUM es %d\n", NUM);
6     return 0;
7 }
```

```
> gcc -Wall -DNUM=100 dtestval.c
> ./a.out
El valor de NUM es 100
```

(-DNAME=VALUE)

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor

dtestval.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("El valor de NUM es %d\n", NUM);
6     return 0;
7 }
```

```
> gcc -Wall -DNUM=100 dtestval.c
> ./a.out
El valor de NUM es 100
```

(-DNAME=VALUE)

```
> gcc -Wall -DNUM="2+2" dtestval.c
> ./a.out
El valor de NUM es 4
```

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor (cont.)

dtestval1.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Diez veces NUM es %d\n", 10 * (NUM));
6     return 0;
7 }
```

---

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor (cont.)

dtestval1.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Diez veces NUM es %d\n", 10 * (NUM));
6     return 0;
7 }
```

---

```
> gcc -Wall -DNUM="2+2" dtestval1.c
> ./a.out
El valor de NUM es 40
```

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor (cont.)

dtestval1.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Diez veces NUM es %d\n", 10 * (NUM));
6     return 0;
7 }
```

```
> gcc -Wall -DNUM="2+2" dtestval1.c
> ./a.out
El valor de NUM es 40
```

¿Qué valor se imprime si no se ponen los paréntesis en la macro?

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor (cont.)

dtestval1.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Diez veces NUM es %d\n", 10 * (NUM));
6     return 0;
7 }
```

```
> gcc -Wall -DNUM="2+2" dtestval1.c
> ./a.out
El valor de NUM es 40
```

¿Qué valor se imprime si no se ponen los paréntesis en la macro?

Valor por defecto de una macro

```
> gcc -Wall -DNUM dtestval1.c
> ./a.out
El valor de NUM es 1
```

# El compilador de C del proyecto GNU

## El preprocesador –Macros con valor (cont.)

dtestval1.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Diez veces NUM es %d\n", 10 * (NUM));
6     return 0;
7 }
```

```
> gcc -Wall -DNUM="2+2" dtestval1.c
> ./a.out
El valor de NUM es 40
```

¿Qué valor se imprime si no se ponen los paréntesis en la macro?

Valor por defecto de una macro

```
> gcc -Wall -DNUM dtestval1.c
> ./a.out
El valor de NUM es 1
```

Una macro vacía `-DNUM=""` queda definida (`#ifdef`) pero no se expande a nada.



# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas

### Biblioteca

Colección de ficheros objetos precompilados que pueden ser enlazados dentro de un programas.

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas

### Biblioteca

Colección de ficheros objetos precompilados que pueden ser enlazados dentro de un programas.

**Estática:** Archivos especiales con la extensión `.a` (ejemplo: `libm.a`)

**Dinámica:** Archivos especiales con la extensión `.so` (ejemplo: `libm.so`)

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas

### Biblioteca

Colección de ficheros objetos precompilados que pueden ser enlazados dentro de un programas.

**Estática:** Archivos especiales con la extensión `.a` (ejemplo: `libm.a`)

**Dinámica:** Archivos especiales con la extensión `.so` (ejemplo: `libm.so`)

- ▶ Se encuentran normalmente en directorios
  - ▶ `/usr/lib` o `/lib`
  - ▶ `/usr/lib64` o `/lib64`
- ▶ o en directorios específicos de la arquitectura
  - ▶ `/usr/lib/i386-linux-gnu/`
  - ▶ `/usr/lib/x86_64-linux-gnu`

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas

### Biblioteca

Colección de ficheros objetos precompilados que pueden ser enlazados dentro de un programas.

**Estática:** Archivos especiales con la extensión `.a` (ejemplo: `libm.a`)

**Dinámica:** Archivos especiales con la extensión `.so` (ejemplo: `libm.so`)

- ▶ Se encuentran normalmente en directorios
  - ▶ `/usr/lib` o `/lib`
  - ▶ `/usr/lib64` o `/lib64`
- ▶ o en directorios específicos de la arquitectura
  - ▶ `/usr/lib/i386-linux-gnu/`
  - ▶ `/usr/lib/x86_64-linux-gnu`

Las declaraciones de los [prototipos de funciones](#) de funciones de biblioteca se encuentran en archivos de cabecera `.h`

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

---

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raiz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

---

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

---

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raíz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

---

```
> gcc -Wall calc.c -o calc
/tmp/cceIc3rZ.o: In function 'main':
calc.c:(.text+0x23): undefined reference to 'sqrt'
collect2: error: ld returned 1 exit status
```

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raíz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

```
> gcc -Wall calc.c -o calc
/tmp/cceIc3rZ.o: In function 'main':
calc.c:(.text+0x23): undefined reference to 'sqrt'
collect2: error: ld returned 1 exit status
```

▶ /tmp/cceIc3rZ.o

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raíz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

```
> gcc -Wall calc.c -o calc
/tmp/cceIc3rZ.o: In function 'main':
calc.c:(.text+0x23): undefined reference to 'sqrt'
collect2: error: ld returned 1 exit status
```

- ▶ /tmp/cceIc3rZ.o
- ▶ undefined reference (.text+0x23)

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raíz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

```
> gcc -Wall calc.c -o calc
/tmp/cceIc3rZ.o: In function 'main':
calc.c:(.text+0x23): undefined reference to 'sqrt'
collect2: error: ld returned 1 exit status
```

- ▶ /tmp/cceIc3rZ.o
- ▶ undefined reference (.text+0x23)
- ▶ ld returned 1

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo

calc.c

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(void)
5 {
6     double x = 2.0;
7     double y = sqrt(x);
8     printf("La raiz cuadrada de %f es %f\n", x, y);
9     return 0;
10 }
```

```
> gcc -Wall calc.c -o calc
/tmp/cceIc3rZ.o: In function 'main':
calc.c:(.text+0x23): undefined reference to 'sqrt'
collect2: error: ld returned 1 exit status
```

- ▶ /tmp/cceIc3rZ.o
- ▶ undefined reference (.text+0x23)
- ▶ ld returned 1
- ▶ Borrar el include a math.h y re-compilar (gcc -c)

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

(mezcla bibliotecas estática y dinámica)

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

(mezcla bibliotecas estática y dinámica)

Corrección

```
> gcc -static -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
```

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

(mezcla bibliotecas estática y dinámica)

Corrección

```
> gcc -static -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
```

### Enlazado con biblioteca dinámica

```
> gcc -Wall calc.c -lm -o calc
```

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

(mezcla bibliotecas estática y dinámica)

Corrección

```
> gcc -static -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
```

### Enlazado con biblioteca dinámica

```
> gcc -Wall calc.c -lm -o calc
```

(-lNAME enlaza contra la biblioteca 'libNAME.so')

# El compilador de C del proyecto GNU

## Enlazando con bibliotecas externas –Ejemplo (cont.)

### Enlazado con biblioteca estática

```
> gcc -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
/usr/lib/x86_64-linux-gnu/libm.a(s_rint.o): In function '__rint':
(.text+0x9): undefined reference to '_dl_x86_cpu_features'
collect2: error: ld returned 1 exit status
```

(mezcla bibliotecas estática y dinámica)

Corrección

```
> gcc -static -Wall calc.c /usr/lib/x86_64-linux-gnu/libm.a -o calc
```

### Enlazado con biblioteca dinámica

```
> gcc -Wall calc.c -lm -o calc
```

(-lNAME enlaza contra la biblioteca 'libNAME.so')

- ▶ Probar comando `ldd` y `file` con ambas aplicaciones.
- ▶ Ver tamaños de los binario.

# El compilador de C del proyecto GNU

## Archivos de cabecera de bibliotecas

badconv.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     double x = strtod("123", NULL);
6     printf("El valor es %f\n", x);
7     return 0;
8 }
```

---

# El compilador de C del proyecto GNU

## Archivos de cabecera de bibliotecas

badconv.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     double x = strtod("123", NULL);
6     printf("El valor es %f\n", x);
7     return 0;
8 }
```

## Compilar

```
gcc -Wall badconv.c -o badconv
badconv.c: In function 'main':
badconv.c:5:14: warning: implicit declaration of function
'strtod' [-Wimplicit-function-declaration]
    double x = strtod("123", NULL);
                ^
```

# El compilador de C del proyecto GNU

## Archivos de cabecera de bibliotecas

badconv.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     double x = strtod("123", NULL);
6     printf("El valor es %f\n", x);
7     return 0;
8 }
```

## Compilar

```
gcc -Wall badconv.c -o badconv
badconv.c: In function 'main':
badconv.c:5:14: warning: implicit declaration of function
'strtod' [-Wimplicit-function-declaration]
    double x = strtod("123", NULL);
                ^
```

## Ejecutar

```
> ./badconv
El valor es 0.000000
```

# El compilador de C del proyecto GNU

## Archivos de cabecera de bibliotecas

badconv.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     double x = strtod("123", NULL);
6     printf("El valor es %f\n", x);
7     return 0;
8 }
```

## Compilar

```
gcc -Wall badconv.c -o badconv
badconv.c: In function 'main':
badconv.c:5:14: warning: implicit declaration of function
'strtod' [-Wimplicit-function-declaration]
    double x = strtod("123", NULL);
                ^
```

## Ejecutar

```
> ./badconv
El valor es 0.000000
```

Agregar archivo de cabecera y probar nuevamente.



# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

Qué significa?

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR: FILE.h: No such file or directory**

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR:** /usr/bin/ld: cannot find library

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR: FILE.h: No such file or directory**

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR: /usr/bin/ld: cannot find library**

Qué significa? No se encuentra la biblioteca compartida al enlazar

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR: FILE.h: No such file or directory**

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR: /usr/bin/ld: cannot find library**

Qué significa? No se encuentra la biblioteca compartida al enlazar

Por defecto, gcc busca los archivos en los siguientes directorios

- ▶ Archivos de cabecera

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR:** /usr/bin/ld: cannot find library

Qué significa? No se encuentra la biblioteca compartida al enlazar

Por defecto, gcc busca los archivos en los siguientes directorios

- ▶ Archivos de cabecera
  - ▶ /usr/include
  - ▶ /usr/local/include

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR: FILE.h: No such file or directory**

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR: /usr/bin/ld: cannot find library**

Qué significa? No se encuentra la biblioteca compartida al enlazar

Por defecto, gcc busca los archivos en los siguientes directorios

- ▶ Archivos de cabecera
  - ▶ /usr/include
  - ▶ /usr/local/include
- ▶ Bibliotecas

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR:** /usr/bin/ld: cannot find library

Qué significa? No se encuentra la biblioteca compartida al enlazar

Por defecto, gcc busca los archivos en los siguientes directorios

- ▶ Archivos de cabecera
  - ▶ /usr/include
  - ▶ /usr/local/include
- ▶ Bibliotecas
  - ▶ /usr/lib
  - ▶ /usr/local/lib

# El compilador de C del proyecto GNU

## Otras opciones de compilación

**ERROR:** FILE.h: No such file or directory

Qué significa? No se encuentra un archivo de cabecera de una biblioteca

**ERROR:** /usr/bin/ld: cannot find library

Qué significa? No se encuentra la biblioteca compartida al enlazar

Por defecto, gcc busca los archivos en los siguientes directorios

- ▶ Archivos de cabecera
  - ▶ /usr/include
  - ▶ /usr/local/include
- ▶ Bibliotecas
  - ▶ /usr/lib
  - ▶ /usr/local/lib

Opciones de compilación para agregar rutas: -I y -L. Ejemplos:

```
> gcc -I/usr/include/newlib source.c
```

```
> gcc -L/usr/X11/lib -lX11 source.c
```



# El compilador de C del proyecto GNU

## Estándares del lenguaje C

`gcc` compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

`gcc` compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

`gcc` compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

`gcc` compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones
- ▶ vectores de tamaño variable

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

`gcc` compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones
- ▶ vectores de tamaño variable
- ▶ estructuras vacías, etc.

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

gcc compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones
- ▶ vectores de tamaño variable
- ▶ estructuras vacías, etc.

Opciones más comunes:

- ▶ `-ansi`: en C es equivalente a `-std=c90`.

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

gcc compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones
- ▶ vectores de tamaño variable
- ▶ estructuras vacías, etc.

Opciones más comunes:

- ▶ `-ansi`: en C es equivalente a `-std=c90`.
- ▶ `-pedantic`: genera advertencias si se utiliza alguna extensión de GNU.

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

gcc compila por defecto el dialecto de GNU del lenguaje C, llamado GNU C.

Este dialecto incorpora el estándar oficial ANSI/ISO con varias extensiones<sup>1</sup> útiles para sistemas GNU, por ejemplo:

- ▶ funciones definidas dentro de funciones
- ▶ vectores de tamaño variable
- ▶ estructuras vacías, etc.

Opciones más comunes:

- ▶ `-ansi`: en C es equivalente a `-std=c90`.
- ▶ `-pedantic`: genera advertencias si se utiliza alguna extensión de GNU.
- ▶ `-std`: puede ser `-std=c90`, `-std=c99`, etc.

---

<sup>1</sup><https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html>

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

ansi.c

---

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     const char asm[] = "6502";
6     printf("La cadena asm es: '%s'\n", asm);
7     return 0;
8 }
```

---

Ejemplo de programa incompatible con la extensión de GNU C.

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

ansi.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     const char asm[] = "6502";
6     printf("La cadena asm es: '%s'\n", asm);
7     return 0;
8 }
```

Ejemplo de programa incompatible con la extensión de GNU C.

Compilar

```
> gcc -Wall ansi.c
ansi.c: In function 'main':
ansi.c:5:14: error: expected identifier or '(' before 'asm'
  const char asm[] = "6502";
           ^
ansi.c:6:38: error: expected expression before 'asm'
  printf("La cadena asm es: '%s'\n", asm);
                                     ^
```

# El compilador de C del proyecto GNU

## Estándares del lenguaje C

ansi.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     const char asm[] = "6502";
6     printf("La cadena asm es: '%s'\n", asm);
7     return 0;
8 }
```

Ejemplo de programa incompatible con la extensión de GNU C.

Compilar

```
> gcc -Wall ansi.c
ansi.c: In function 'main':
ansi.c:5:14: error: expected identifier or '(' before 'asm'
    const char asm[] = "6502";
           ^
ansi.c:6:38: error: expected expression before 'asm'
    printf("La cadena asm es: '%s'\n", asm);
                                   ^
```

```
> gcc -Wall -ansi ansi.c
> ./a.out
La cadena asm es: '6502'
```

# El compilador de C del proyecto GNU

## Estándares del lenguaje C –ANSI/ISO estricto

gnuarray.c

---

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int i, n = argc;
6     double x[n];
7
8     for(i = 0; i < n; i++)
9         x[i] = i;
10
11     return 0;
12 }
```

---

Ejemplo de programa con array de tamaño variable de la extensión de GNU C.

# El compilador de C del proyecto GNU

## Estándares del lenguaje C –ANSI/ISO estricto

gnuarray.c

---

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int i, n = argc;
6     double x[n];
7
8     for(i = 0; i < n; i++)
9         x[i] = i;
10
11     return 0;
12 }
```

---

Ejemplo de programa con array de tamaño variable de la extensión de GNU C.

```
> gcc -Wall -ansi gnuarray.c
```

# El compilador de C del proyecto GNU

## Estándares del lenguaje C –ANSI/ISO estricto

gnuarray.c

---

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int i, n = argc;
6     double x[n];
7
8     for(i = 0; i < n; i++)
9         x[i] = i;
10
11     return 0;
12 }
```

---

Ejemplo de programa con array de tamaño variable de la extensión de GNU C.

```
> gcc -Wall -ansi gnuarray.c
```

```
> gcc -Wall -ansi -pedantic gnuarray.c
gnuarray.c: In function 'main':
gnuarray.c:6:3: warning: ISO C90 forbids variable length
array 'x' [-Wvla]
```



# El compilador de C del proyecto GNU

## Construcción de bibliotecas

Se creará una biblioteca pequeña `libhola` que contiene dos funciones `hola` y `adios`.

# El compilador de C del proyecto GNU

## Construcción de bibliotecas

Se creará una biblioteca pequeña `libhola` que contiene dos funciones `hola` y `adios`.

`hola_fn.c`

---

```
1 #include <stdio.h>
2 #include "hola.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, \u25a1%s!\n", nombre);
7 }
```

---

# El compilador de C del proyecto GNU

## Construcción de bibliotecas

Se creará una biblioteca pequeña `libhola` que contiene dos funciones `hola` y `adios`.

### `hola_fn.c`

---

```
1 #include <stdio.h>
2 #include "hola.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, \u0026#92;s!\n", nombre);
7 }
```

---

### `adios_fn.c`

---

```
1 #include <stdio.h>
2 #include "hola.h"
3
4 void adios(void)
5 {
6     printf("Adios!\n");
7 }
```

---

# El compilador de C del proyecto GNU

## Construcción de bibliotecas

Se creará una biblioteca pequeña `libhola` que contiene dos funciones `hola` y `adios`.

### `hola_fn.c`

---

```
1 #include <stdio.h>
2 #include "hola.h"
3
4 void hola(const char * nombre)
5 {
6     printf("Hola, %s!\n", nombre);
7 }
```

---

### `adios_fn.c`

---

```
1 #include <stdio.h>
2 #include "hola.h"
3
4 void adios(void)
5 {
6     printf("Adios!\n");
7 }
```

---

### `hola.h`

---

```
1 #ifndef _HOLA_H
2 #define _HOLA_H
3
4 void hola(const char * );
5 void adios(void);
6
7 #endif
```

---

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Estática

Compilar

```
> gcc -Wall -c hola_fn.c  
> gcc -Wall -c adios_fn.c
```

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Estática

Compilar

```
> gcc -Wall -c hola_fn.c  
> gcc -Wall -c adios_fn.c
```

Combinar los archivos y generar la biblioteca

```
> ar cr libhola.a hola_fn.o adios_fn.o
```

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Estática

### Compilar

```
> gcc -Wall -c hola_fn.c  
> gcc -Wall -c adios_fn.c
```

### Combinar los archivos y generar la biblioteca

```
> ar cr libhola.a hola_fn.o adios_fn.o
```

- ▶ **c**: crear el archivo (\*.a)
- ▶ **r**: insertar un miembro (con reemplazo)

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Estática

### Compilar

```
> gcc -Wall -c hola_fn.c  
> gcc -Wall -c adios_fn.c
```

### Combinar los archivos y generar la biblioteca

```
> ar cr libhola.a hola_fn.o adios_fn.o
```

- ▶ c: crear el archivo (\*.a)
- ▶ r: insertar un miembro (con reemplazo)

### Listar los archivos objetos de la biblioteca

```
> ar t libhola.a  
hola_fn.o  
adios_fn.o
```

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Estática

### Compilar

```
> gcc -Wall -c hola_fn.c  
> gcc -Wall -c adios_fn.c
```

### Combinar los archivos y generar la biblioteca

```
> ar cr libhola.a hola_fn.o adios_fn.o
```

- ▶ **c**: crear el archivo (\*.a)
- ▶ **r**: insertar un miembro (con reemplazo)

### Listar los archivos objetos de la biblioteca

```
> ar t libhola.a  
hola_fn.o  
adios_fn.o
```

(Ver comando `nm`)

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Dinámica

Compilar

```
> gcc -Wall -c -fpic hola_fn.c  
> gcc -Wall -c -fpic adios_fn.c
```

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Dinámica

### Compilar

```
> gcc -Wall -c -fpic hola_fn.c  
> gcc -Wall -c -fpic adios_fn.c
```

- ▶ **-fpic**: genera código independiente de la posición (position independent code)

# El compilador de C del proyecto GNU

## Construcción de bibliotecas –Dinámica

### Compilar

```
> gcc -Wall -c -fpic hola_fn.c  
> gcc -Wall -c -fpic adios_fn.c
```

- ▶ **-fpic**: genera código independiente de la posición (position independent code)

### Combinar los archivos y generar la biblioteca

```
> gcc -shared hola_fn.o adios_fn.o -o libhola.so
```



# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola `<nombre>`!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar `<nombre>` por su nombre particular.

# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola `<nombre>`!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar `<nombre>` por su nombre particular.
- ▶ Generar un binario compilando los 3 archivos fuentes juntos.

# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola `<nombre>`!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar `<nombre>` por su nombre particular.
- ▶ Generar un binario compilando los 3 archivos fuentes juntos.
- ▶ Crear una biblioteca estática `libhola.a` utilizando los fuentes `hola_fn.c` y `adios_fn.c`

# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola <nombre>!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar <nombre> por su nombre particular.
- ▶ Generar un binario compilando los 3 archivos fuentes juntos.
- ▶ Crear una biblioteca estática `libhola.a` utilizando los fuentes `hola_fn.c` y `adios_fn.c`
- ▶ Generar un binario utilizando la biblioteca estática `libhola.a`
  1. Compilar únicamente `saludo.c` con la opción `-c`.
  2. Generar un binario enlazando el archivo objeto `saludo.o` con la biblioteca.

# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola <nombre>!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar <nombre> por su nombre particular.
- ▶ Generar un binario compilando los 3 archivos fuentes juntos.
- ▶ Crear una biblioteca estática `libhola.a` utilizando los fuentes `hola_fn.c` y `adios_fn.c`
- ▶ Generar un binario utilizando la biblioteca estática `libhola.a`
  1. Compilar únicamente `saludo.c` con la opción `-c`.
  2. Generar un binario enlazando el archivo objeto `saludo.o` con la biblioteca.
- ▶ Aplicar los mismos pasos moviendo el archivo de cabecera `hola.h` a un directorio `header` (utilizar flag `-I`).

# El compilador de C del proyecto GNU

## Ejercicios

- ▶ Escribir un programa (`saludo.c`) que muestre en pantalla el saludo  
Hola <nombre>!  
Adios!  
utilizando las funciones `hola` y `adios` implementadas anteriormente.  
Reemplazar <nombre> por su nombre particular.
- ▶ Generar un binario compilando los 3 archivos fuentes juntos.
- ▶ Crear una biblioteca estática `libhola.a` utilizando los fuentes `hola_fn.c` y `adios_fn.c`
- ▶ Generar un binario utilizando la biblioteca estática `libhola.a`
  1. Compilar únicamente `saludo.c` con la opción `-c`.
  2. Generar un binario enlazando el archivo objeto `saludo.o` con la biblioteca.
- ▶ Aplicar los mismos pasos moviendo el archivo de cabecera `hola.h` a un directorio `header` (utilizar flag `-I`).
- ▶ Aplicar los mismos pasos moviendo el archivo de biblioteca `libhola.a` a un directorio `lib` (utilizar flag `-L`).



# El compilador de C del proyecto GNU

## Un pequeño makefile