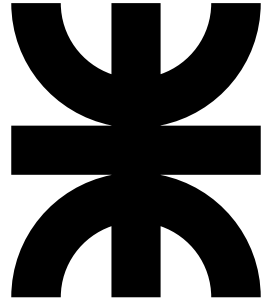
Informática II

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

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2019 –

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

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

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

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

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

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

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

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

El compilador de C del proyecto GNU

Herramientas de compilación del proyecto GNU (GNU Compiler Collection):

- Maneja varios dialectos de C: ANSI-C, tradicional (Kernighan & Ritchie), extensiones GNU, 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`)

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilación

```
> gcc hola.c
```

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilación

```
> gcc hola.c
```

(salida `a.out`)

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

Compilación

```
> gcc hola.c
```

(salida `a.out`) Cómo se ejecuta?

Cómo cambiar el nombre a binario?

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

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

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
```

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

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)

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

mal.c

```c
#include <stdio.h>

int main(void)
{
  printf("Dos y dos son %f\n", 4);
  return 0;
}
```

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)

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

mal.c

```c
#include <stdio.h>

int main(void)
{
  printf("Dos y dos son %f\n", 4);
  return 0;
}
```

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
```

Compilando – Algunos ejemplos

hola.c

```c
#include <stdio.h>

int main(void)
{
  printf("Hola, mundo!\n");
  return 0;
}
```

mal.c

```c
#include <stdio.h>

int main(void)
{
  printf("Dos y dos son %f\n", 4);
  return 0;
}
```

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=]
```

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

hola_fn.h

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

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

hola_fn.h

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

Compilación

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

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

hola_fn.h

```c
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
```

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

hola_fn.h

```c
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 (enlazador)

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

Compilando – Ejemplos con varios archivos fuentes

main.c

```c
#include "hola_fn.h"

int main(void)
{
  hola("mundo");
  return 0;
}
```

hola_fn.c

```c
#include <stdio.h>
#include "hola_fn.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

hola_fn.h

```c
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 (enlazador)

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

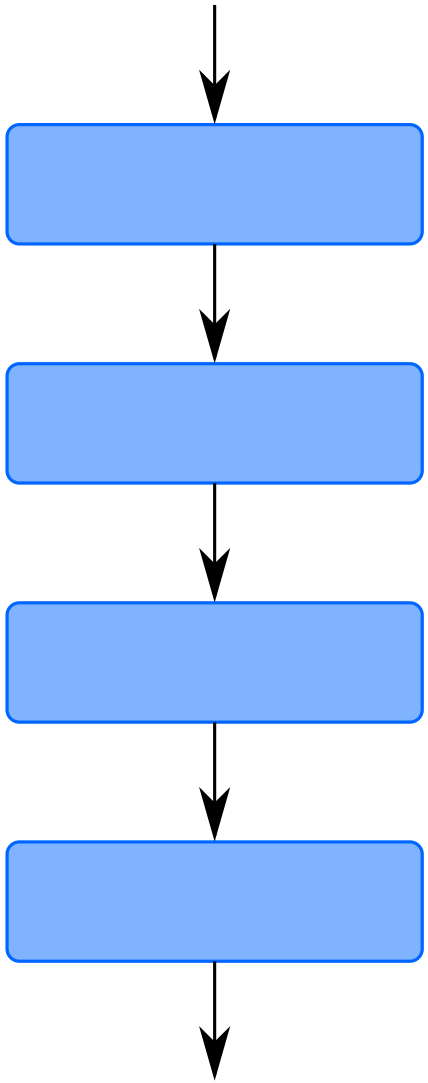
Esto permite modifica un archivo fuente y recompilar solo ese archivo.

Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).

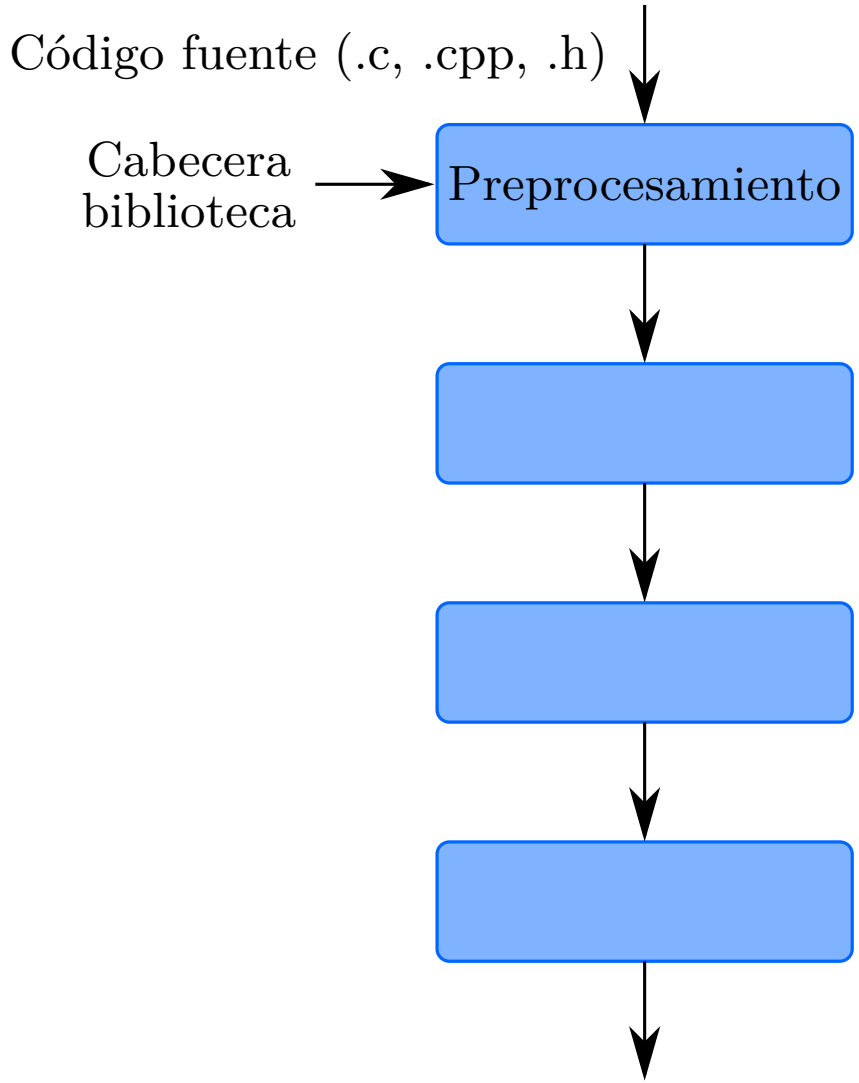
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



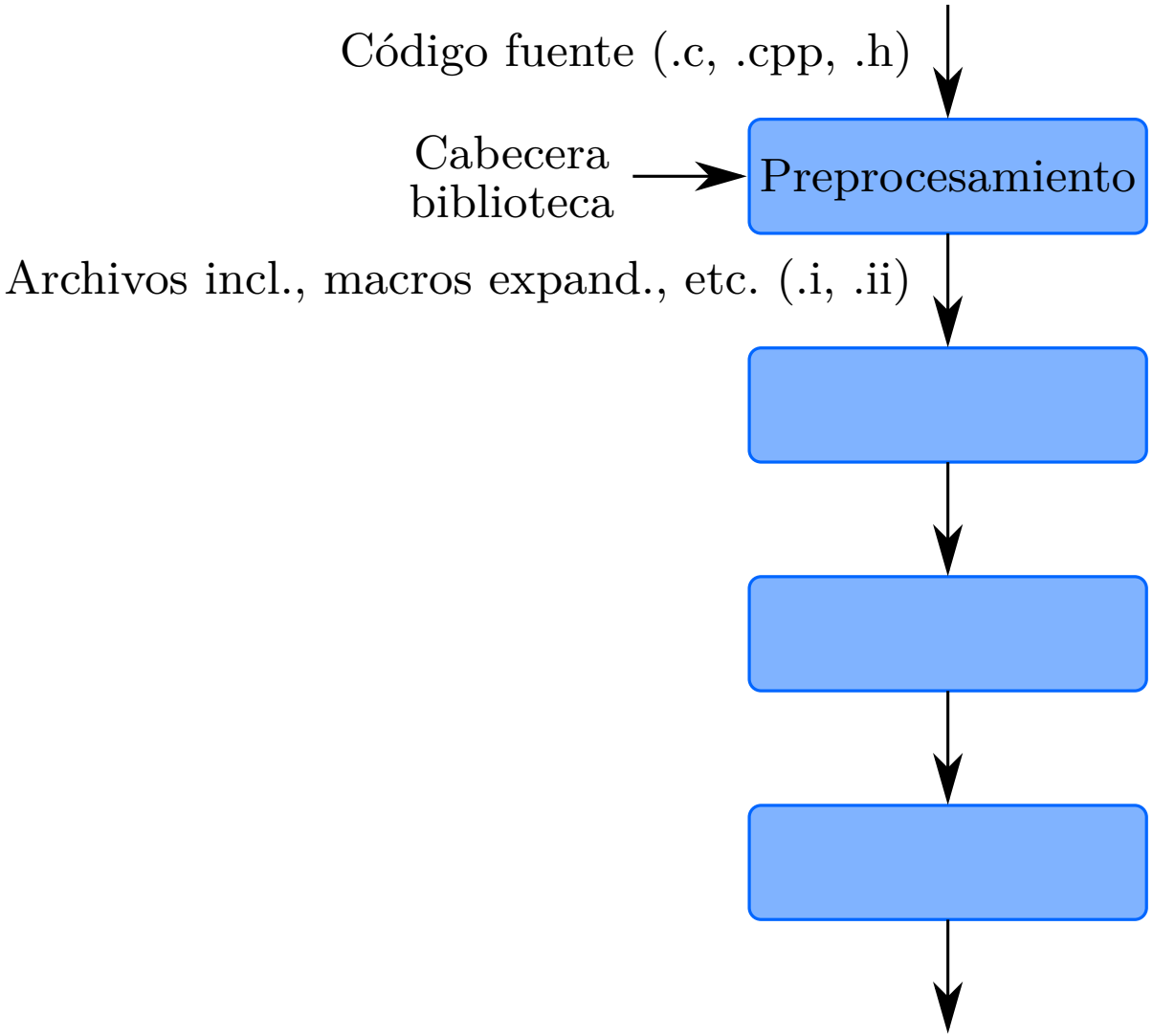
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



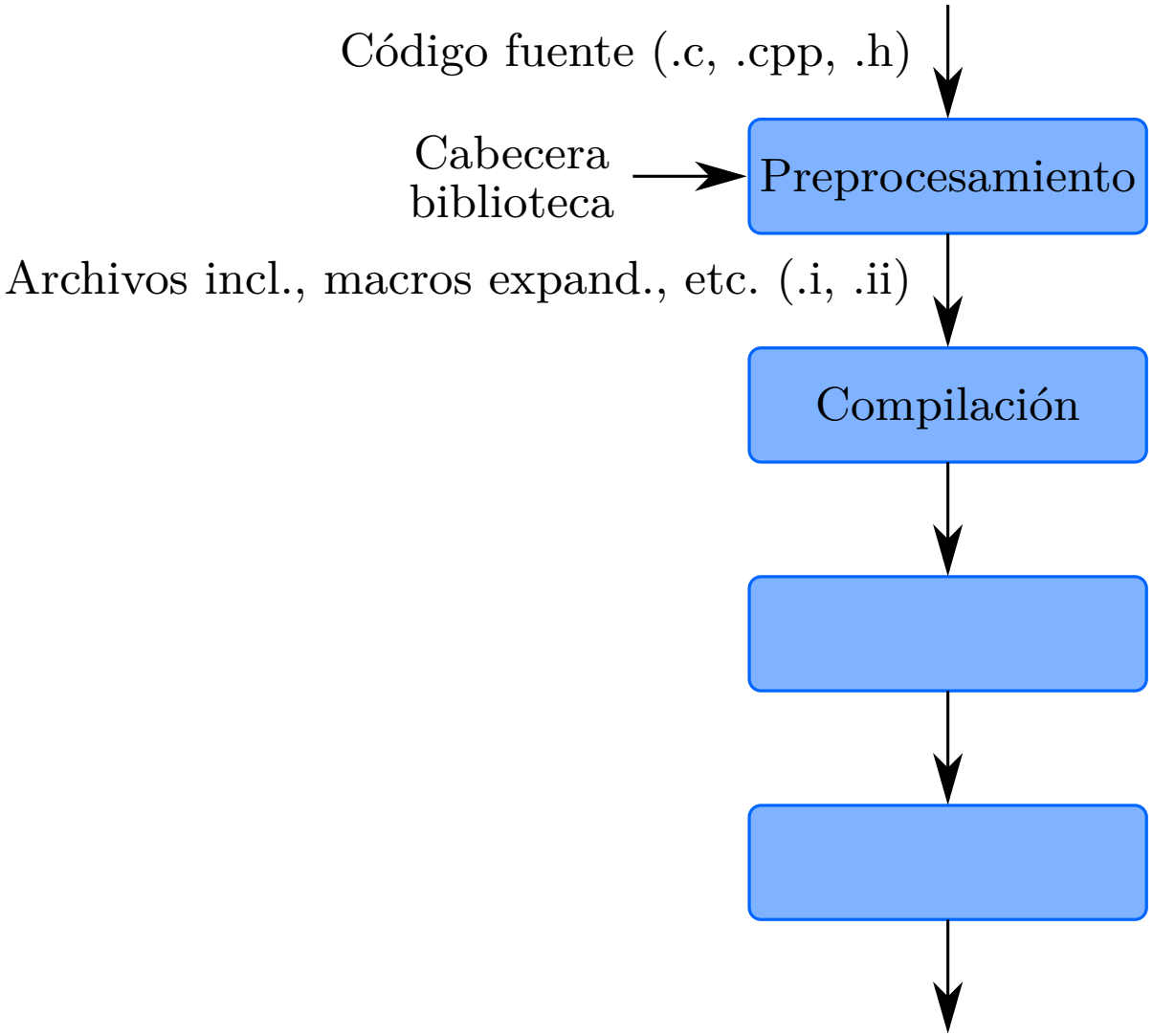
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



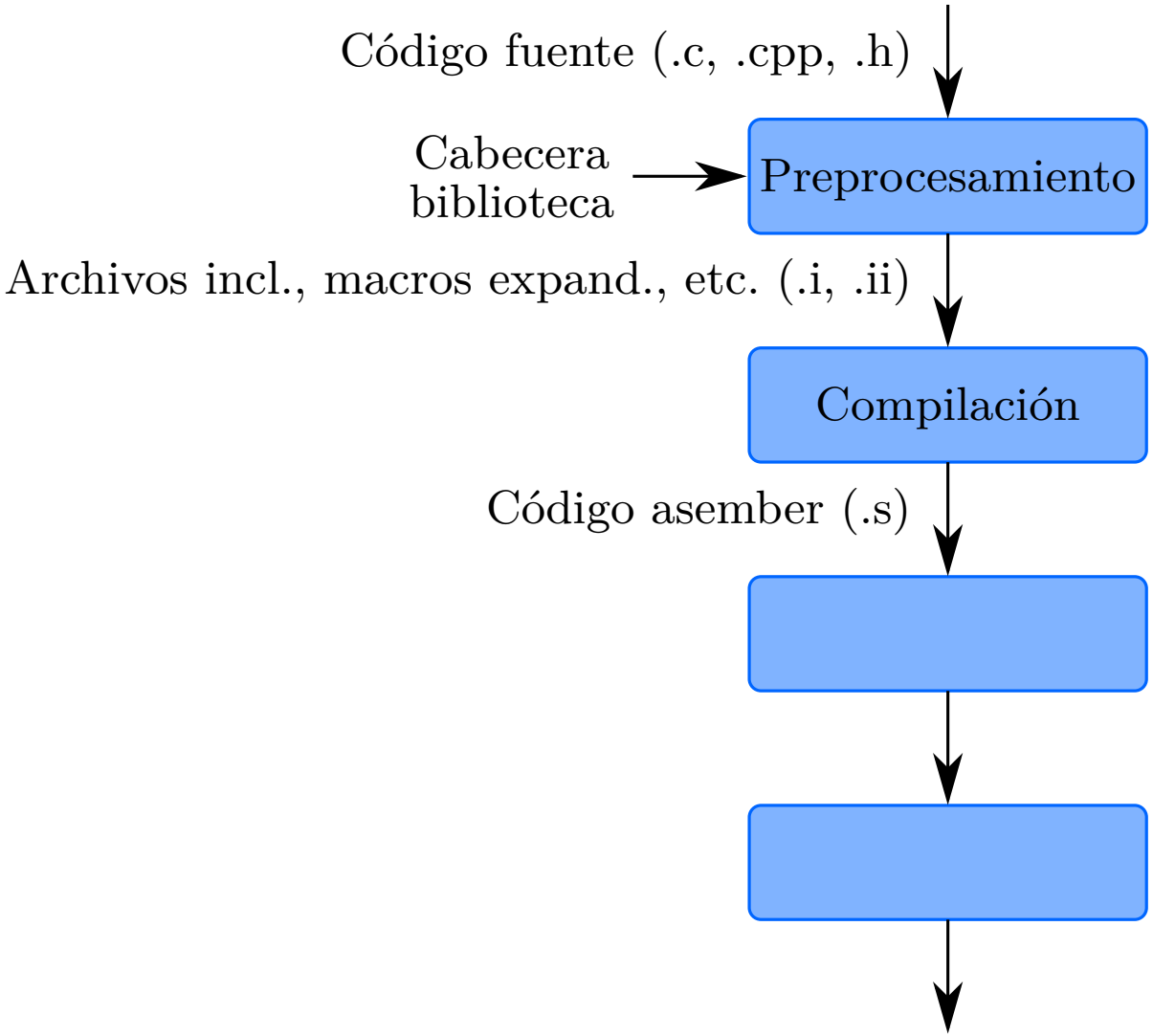
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



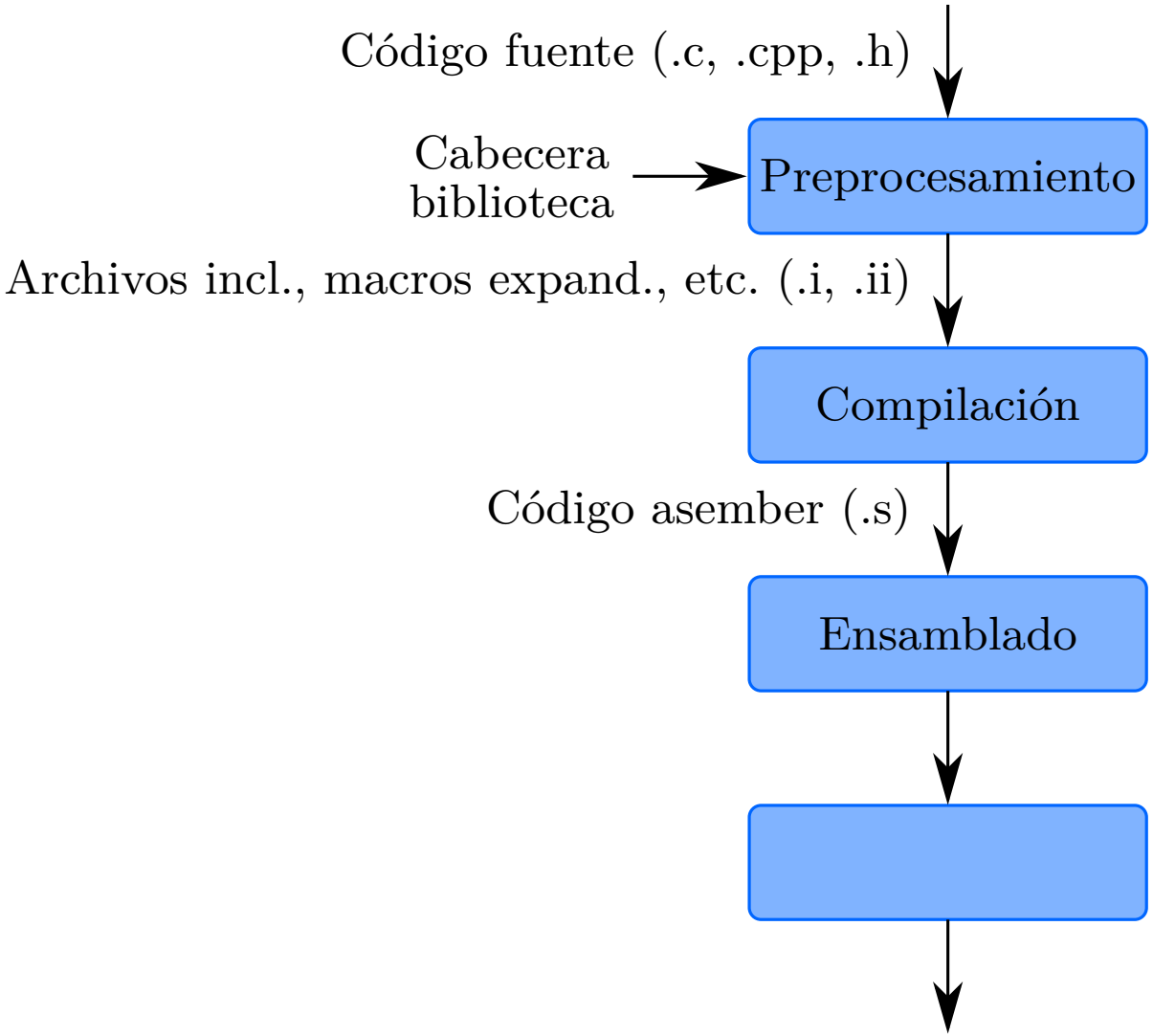
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



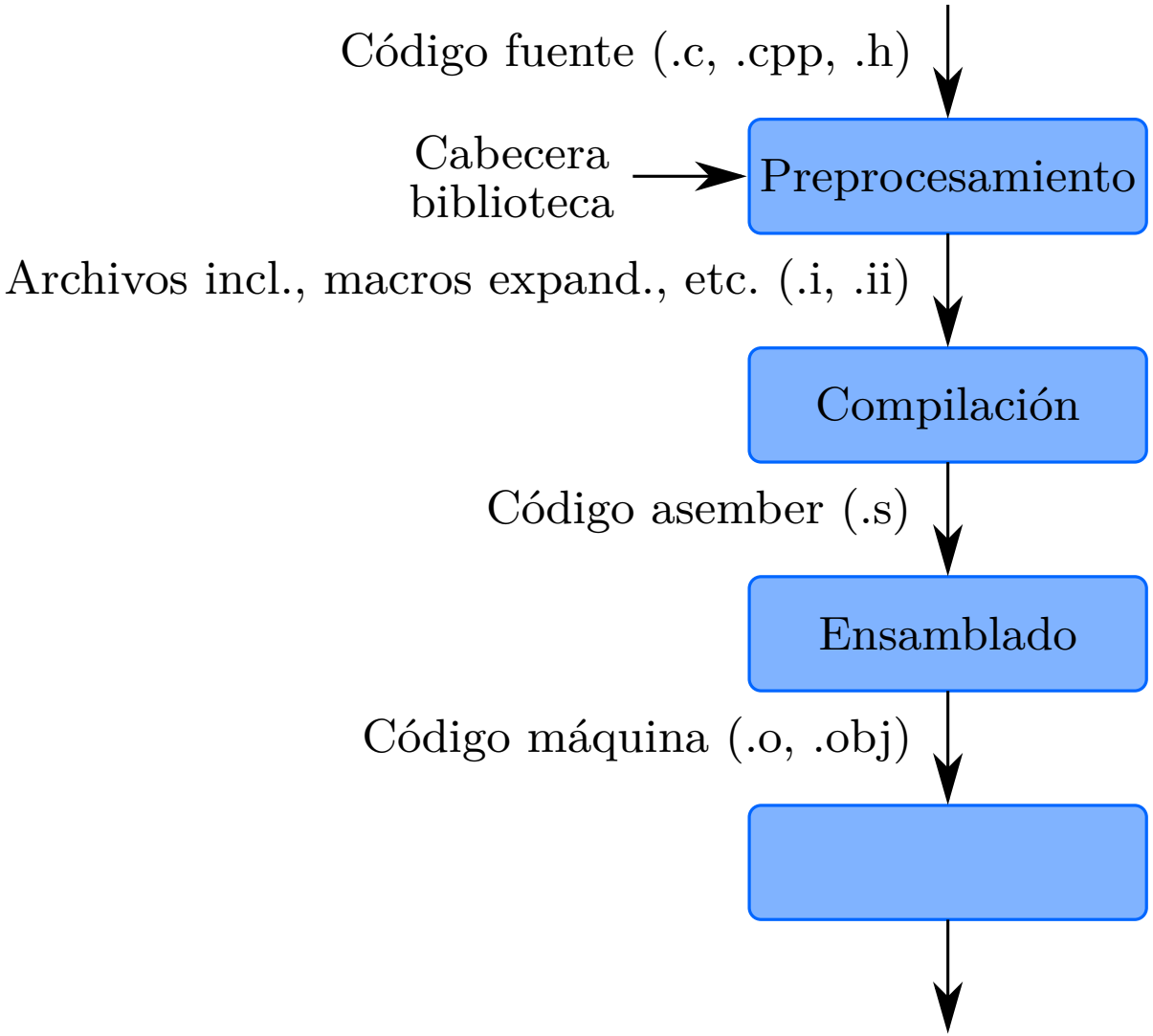
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



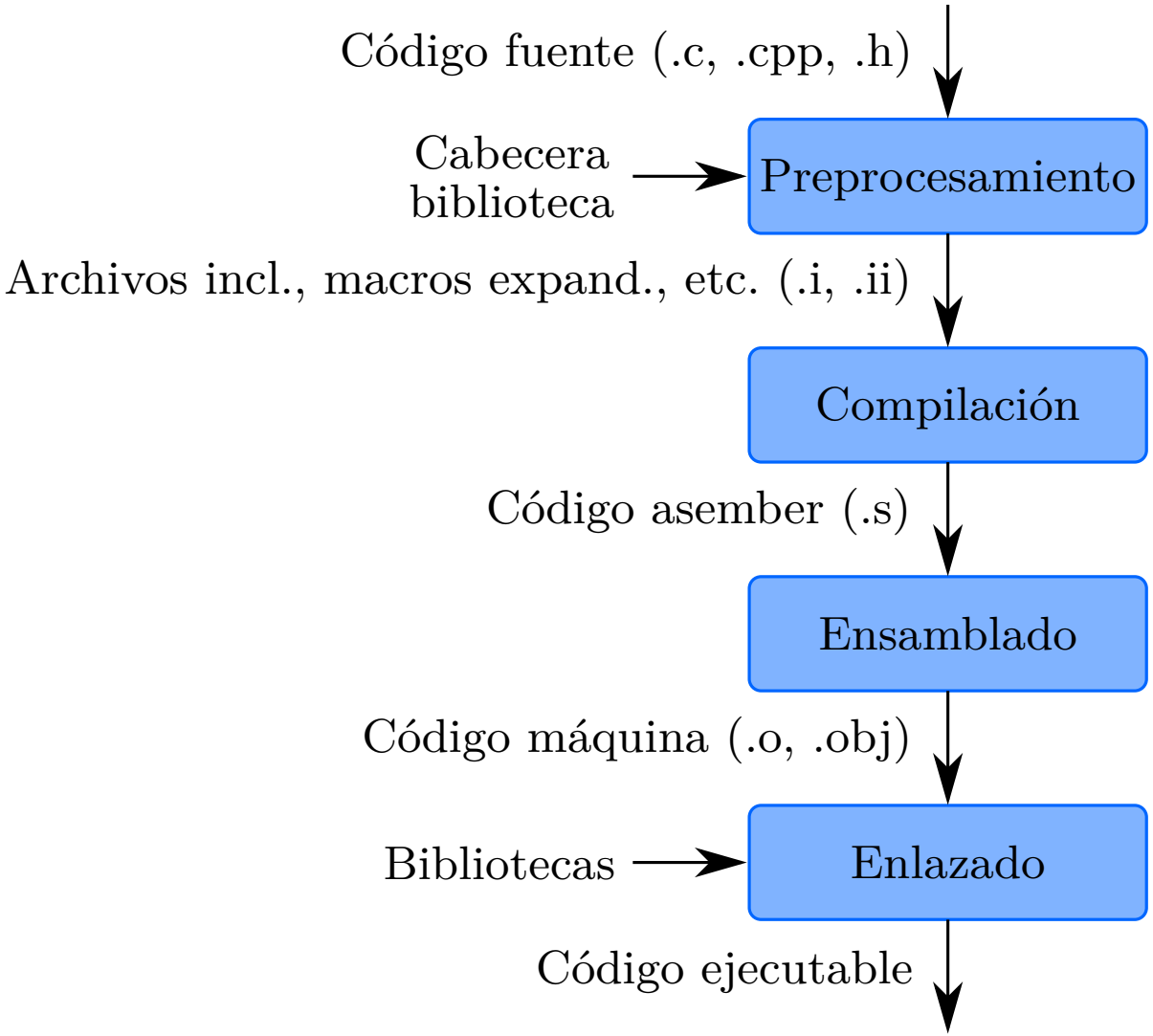
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



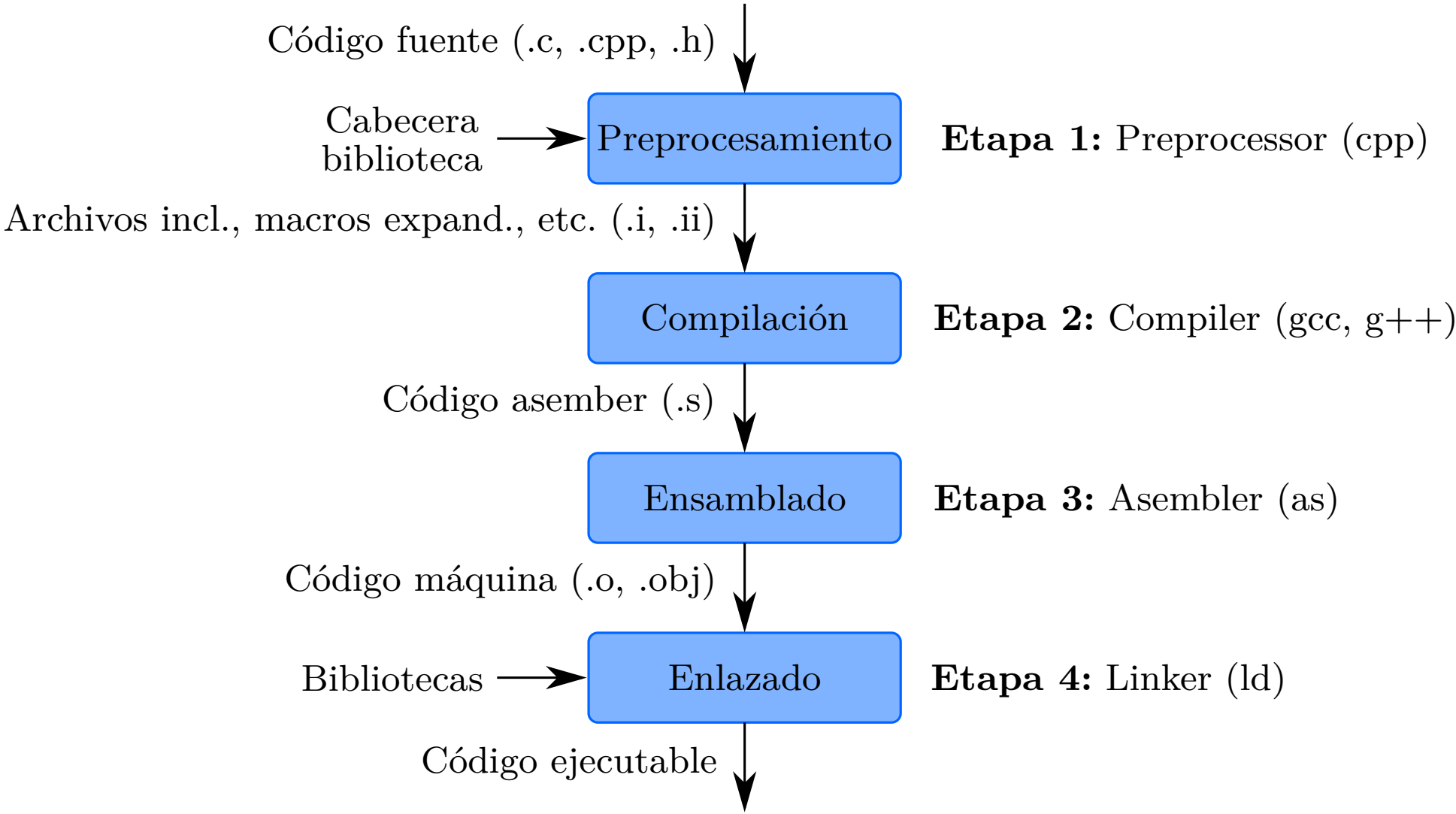
Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado).



Etapas de compilación/construcción

El proceso de compilación/construcción involucra 4 etapas (preprocesamiento, compilación, ensamblado, y enlazado). Conjunto de herramientas: *toolchain*.



Preprocesado

test.c

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

Preprocesado

test.c

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

```
> cpp test.c
```

Preprocesado

test.c

```c
#define TEST "Hola mundo!"
const char str[] = TEST;
```

```
> cpp 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!";
```

Preprocesado

test.c

```c
#define TEST "Hola mundo!"
const char str[] = TEST;
```

```
> cpp 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 preprocesador inserta líneas de registro de archivo fuente y el nro. de línea en la forma `#num-de-linea "archivo fuente"`

Preprocesado

hola1.c

```
1 #include <stdio.h>
2
3
4
5 int main(void)
6 {
7   printf("Hola mundo!\n");
8   return 0;
9 }
```

Preprocesado

hola1.c

```c
#include <stdio.h>



int main(void)
{
  printf("Hola mundo!\n");
  return 0;
}
```

Preprocesado:

```
> cpp hola1.c > hola1.i
```

Preprocesado

hola1.c

```c
#include <stdio.h>



int main(void)
{
  printf("Hola mundo!\n");
  return 0;
}
```

Preprocesado:

```
> cpp hola1.c > hola1.i
```

Preprocesado

hola2.c

```c
#include <stdio.h>

#define MENSAJE "Hola mundo!\n"

int main(void)
{
  printf(MENSAJE);
  return 0;
}
```

Preprocesado:

```
> cpp hola2.c > hola2.i
```

Preprocesado

hola2.c

```c
#include <stdio.h>

#define MENSAJE "Hola mundo!\n"

int main(void)
{
  printf(MENSAJE);
  return 0;
}
```

Preprocesado:

```
> cpp hola2.c > hola2.i
```

Observar: macros, comentarios, archivos cabecera (includes)

Compilado, ensamblado y enlazado

Compilar el archivo de salida del preprocesador

```
> gcc -Wall -S hola.i
```

Compilado, ensamblado y enlazado

Compilar el archivo de salida del preprocesador

```
> gcc -Wall -S hola.i
```

Ensamblado

```
> as hola.s -o hola.o
```

Compilado, ensamblado y enlazado

Compilar el archivo de salida del preprocesador

```
> gcc -Wall -S hola.i
```

Ensamblado

```
> as hola.s -o hola.o
```

Enlazado

```
> ld /usr/lib/x86_64-linux-gnu/crti.o \
/usr/lib/x86_64-linux-gnu/crtn.o \
/usr/lib/x86_64-linux-gnu/crt1.o \
-lc hola.o \
-dynamic-linker /lib64/ld-linux-x86-64.so.2 \
-o hola
```

(O bien `>gcc hola.o -o hola`)

Compilado, ensamblado y enlazado

Compilar el archivo de salida del preprocesador

```
> gcc -Wall -S hola.i
```

Ensamblado

```
> as hola.s -o hola.o
```

Enlazado

```
> ld /usr/lib/x86_64-linux-gnu/crti.o \
/usr/lib/x86_64-linux-gnu/crtn.o \
/usr/lib/x86_64-linux-gnu/crt1.o \
-lc hola.o \
-dynamic-linker /lib64/ld-linux-x86-64.so.2 \
-o hola
```

(O bien `>gcc hola.o -o hola`)

- `crti.o` y `crtn.o`: definen los prólogos y epílogos de las funciones
- `crt1.o`: contiene el símbolo `_start` donde comienza la ejecución
- Flag `-lc`: incluye la biblioteca estándar de C

Construcción paso a paso

Ejecutar diferentes etapas de construcción:

- `gcc -E`: Preprocesamiento sin compilación
- `gcc -S`: Compilación sin ensamblado
- `gcc -c`: Preprocesamiento, compilación y ensablado sin enlazado

Construcción paso a paso

Ejecutar diferentes etapas de construcción:

- `gcc -E`: Preprocesamiento sin compilación
- `gcc -S`: Compilación sin ensamblado
- `gcc -c`: Preprocesamiento, compilación y ensablado sin enlazado

Continuar la construcción desde una etapa anterior

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

La opción `-x` le indica que continúe desde la etapa indicada (`-c`, `-S`, `-c`). Ver página de manual `>man gcc`

Construcción paso a paso

Ejecutar diferentes etapas de construcción:

- `gcc -E`: Preprocesamiento sin compilación
- `gcc -S`: Compilación sin ensamblado
- `gcc -c`: Preprocesamiento, compilación y ensablado sin enlazado

Continuar la construcción desde una etapa anterior

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

La opción `-x` le indica que continúe desde la etapa indicada (`-c`, `-S`, `-c`). Ver página de manual `>man gcc`

Flag `-save-temps` genera archivos intermedios

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

(Ver flag `-v`)

Construcción paso a paso

Ejecutar diferentes etapas de construcción:

- `gcc -E`: Preprocesamiento sin compilación
- `gcc -S`: Compilación sin ensamblado
- `gcc -c`: Preprocesamiento, compilación y ensablado sin enlazado

Continuar la construcción desde una etapa anterior

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

La opción `-x` le indica que continúe desde la etapa indicada (`-c`, `-S`, `-c`). Ver página de manual `>man gcc`

Flag `-save-temps` genera archivos intermedios

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

(Ver flag `-v`)

Otros flags: `-std=c90`, `-Wall`, `-Werror`

Construcción paso a paso – archivos de salida

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

Construcción paso a paso – archivos de salida

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

```
> file hola
hola 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
```

Construcción paso a paso – archivos de salida

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

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

Construcción paso a paso – archivos de salida

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

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

(`objdump -x`, `hexdump -C`, `readelf -d`)

Construcción paso a paso – archivos de salida

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

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

(`objdump -x`, `hexdump -C`, `readelf -d`)

ELF: Executable and Linkable Format

El preprocesador – 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 preprocesador – Compilación condicional

dtest.c

```c
#include <stdio.h>

int main(void)
{
#ifdef TEST
  printf("Modo test\n");
#endif
  printf("Ejecutando...\n");
  return 0;
}
```

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

El preprocesador – Compilación condicional

dtest.c

```c
#include <stdio.h>

int main(void)
{
#ifdef TEST
  printf("Modo test\n");
#endif
  printf("Ejecutando...\n");
  return 0;
}
```

```
> 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 preprocesador – Compilación condicional

dtest.c

```c
#include <stdio.h>

int main(void)
{
#ifdef TEST
  printf("Modo test\n");
#endif
  printf("Ejecutando...\n");
  return 0;
}
```

```
> 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 preprocesador – Macros con valor

dtestval.c

```c
#include <stdio.h>

int main(void)
{
  printf("El valor de NUM es %d\n", NUM);
  return 0;
}
```

El preprocesador – Macros con valor

dtestval.c

```c
#include <stdio.h>

int main(void)
{
  printf("El valor de NUM es %d\n", NUM);
  return 0;
}
```

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

(-DNAME=VALUE)

El preprocesador – Macros con valor

dtestval.c

```c
#include <stdio.h>

int main(void)
{
  printf("El valor de NUM es %d\n", NUM);
  return 0;
}
```

```
> 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 preprocesador – Macros con valor

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 preprocesador – Macros con valor

dtestval1.c

```c
#include <stdio.h>

int main(void)
{
  printf("Diez veces NUM es %d\n", 10 * (NUM));
  return 0;
}
```

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

El preprocesador – Macros con valor

dtestval1.c

```c
#include <stdio.h>

int main(void)
{
  printf("Diez veces NUM es %d\n", 10 * (NUM));
  return 0;
}
```

```
> 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 preprocesador – Macros con valor

dtestval1.c

```c
#include <stdio.h>

int main(void)
{
  printf("Diez veces NUM es %d\n", 10 * (NUM));
  return 0;
}
```

```
> 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 dtestval.c
> ./a.out
El valor de NUM es 1
```

El preprocesador – Macros con valor

dtestval1.c

```c
#include <stdio.h>

int main(void)
{
  printf("Diez veces NUM es %d\n", 10 * (NUM));
  return 0;
}
```

```
> 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 dtestval.c
> ./a.out
El valor de NUM es 1
```

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

Enlazando con bibliotecas externas

Biblioteca

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

Enlazando con bibliotecas externas

Biblioteca

Colección de archivos 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`)

Enlazando con bibliotecas externas

Biblioteca

Colección de archivos 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`

Enlazando con bibliotecas externas

Biblioteca

Colección de archivos 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 las funciones de biblioteca se encuentran en archivos de cabecera `.h`

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

```
> gcc -Wall -stdc=c90 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
```

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

```
> gcc -Wall -stdc=c90 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

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

```
> gcc -Wall -stdc=c90 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)

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

```
> gcc -Wall -stdc=c90 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

Enlazando con bibliotecas externas – Ejemplo

calc.c

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
  double x = 2.0;
  double y = sqrt(x);
  printf("La raíz cuadrada de %f es %f\n", x, y);
  return 0;
}
```

```
> gcc -Wall -stdc=c90 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`)

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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)

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

(`ldd calc`)

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

(`ldd calc`)

Enlazado con biblioteca dinámica

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

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

(`ldd calc`)

Enlazado con biblioteca dinámica

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

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

Enlazando con bibliotecas externas – Ejemplo

Enlazado con biblioteca estática (buscar `libm.a`)

```
> 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
```

(`ldd 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.

Archivos de cabecera de bibliotecas

badconv.c

```c
#include <stdio.h>

int main(void)
{
  double x = strtod("123", NULL);
  printf("El valor es %f\n", x);
  return 0;
}
```

Archivos de cabecera de bibliotecas

badconv.c

```c
#include <stdio.h>

int main(void)
{
  double x = strtod("123", NULL);
  printf("El valor es %f\n", x);
  return 0;
}
```

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);
              ^
```

Archivos de cabecera de bibliotecas

badconv.c

```c
#include <stdio.h>

int main(void)
{
  double x = strtod("123", NULL);
  printf("El valor es %f\n", x);
  return 0;
}
```

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
```

Archivos de cabecera de bibliotecas

badconv.c

```c
#include <stdio.h>

int main(void)
{
  double x = strtod("123", NULL);
  printf("El valor es %f\n", x);
  return 0;
}
```

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.

Otras opciones de compilación

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

Otras opciones de compilación

ERROR: `FILE.h: No such file or directory`
Qué significa?

Otras opciones de compilación

ERROR: `FILE.h: No such file or directory`
Qué significa? No se encuentra un archivo de cabecera de una biblioteca

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`

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

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`

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/bib`
- `/usr/local/lib`

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/bib`
- `/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
```

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/bib`
- `/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
```

- `-IPATH`: `-I/usr/include/libusb`, `-I$HOME/milib/include/`, etc.
- `-LPATH`: `-L/usr/lib/graphivz`, `-L$HOME/milib/lib`, etc.

Estándares del lenguaje C

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

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[1] útiles para sistemas GNU, por ejemplo:

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] útiles para sistemas GNU, por ejemplo:

- funciones definidas dentro de funciones

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] útiles para sistemas GNU, por ejemplo:

- funciones definidas dentro de funciones
- vectores de tamaño variable

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] útiles para sistemas GNU, por ejemplo:

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

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] 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`

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] 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`
- `-Wall`: habilita todos las advertencias (warnings)

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] 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`
- `-Wall`: habilita todos las advertencias (warnings)
- `-Werror`: convierte las advertencias en errores

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] 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`
- `-Wall`: habilita todos las advertencias (warnings)
- `-Werror`: convierte las advertencias en errores
- `-pedantic`: genera advertencias si se utiliza alguna extensión de GNU

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

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[1] 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`
- `-Wall`: habilita todos las advertencias (warnings)
- `-Werror`: convierte las advertencias en errores
- `-pedantic`: genera advertencias si se utiliza alguna extensión de GNU
- `-std`: puede ser `-std=c90`, `-std=c99`, etc.

[1] https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html

Estándares del lenguaje C

ansi.c

```c
#include <stdio.h>

int main(void)
{
  const char asm[] = "6502";
  printf("La cadena asm es: '%s'\n", asm);
  return 0;
}
```

Ejemplo de programa incompatible con la extensión de GNU C.

Estándares del lenguaje C

ansi.c

```c
#include <stdio.h>

int main(void)
{
  const char asm[] = "6502";
  printf("La cadena asm es: '%s'\n", asm);
  return 0;
}
```

Ejemplo de programa incompatible con la extensión de GNU C.

```
> 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);
                                      ^
```

Estándares del lenguaje C

ansi.c

```c
#include <stdio.h>

int main(void)
{
  const char asm[] = "6502";
  printf("La cadena asm es: '%s'\n", asm);
  return 0;
}
```

Ejemplo de programa incompatible con la extensión de GNU C.

```
> 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'
```

Estándares del lenguaje C – ANSI/ISO estricto

gnuarray.c

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i, n = argc;
  double x[n];

  for(i = 0; i < n; i++)
    x[i] = i;

  return 0;
}
```

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

Estándares del lenguaje C – ANSI/ISO estricto

gnuarray.c

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i, n = argc;
  double x[n];

  for(i = 0; i < n; i++)
    x[i] = i;

  return 0;
}
```

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

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

Estándares del lenguaje C – ANSI/ISO estricto

gnuarray.c

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i, n = argc;
  double x[n];

  for(i = 0; i < n; i++)
    x[i] = i;

  return 0;
}
```

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]
```

Construcción de bibliotecas

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

Construcción de bibliotecas

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

hola_fn.c

```c
#include <stdio.h>
#include "hola.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

Construcción de bibliotecas

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

hola_fn.c

```c
#include <stdio.h>
#include "hola.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

adios_fn.c

```c
#include <stdio.h>
#include "hola.h"

void adios(void)
{
  printf("Adios!\n");
}
```

Construcción de bibliotecas

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

hola_fn.c

```c
#include <stdio.h>
#include "hola.h"

void hola(const char * nombre)
{
  printf("Hola, %s!\n", nombre);
}
```

adios_fn.c

```c
#include <stdio.h>
#include "hola.h"

void adios(void)
{
  printf("Adios!\n");
}
```

hola.h

```c
#ifndef HOLA_ADIOS_H
#define HOLA_ADIOS_H

void hola(const char * );
void adios(void);

#endif
```

Construcción de bibliotecas – Estática

Compilar

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

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
```

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 (reemplazándolo si existe)

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 (reemplazándolo si existe)

Listar los archivos objetos de la biblioteca

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

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 (reemplazándolo si existe)

Listar los archivos objetos de la biblioteca

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

(Ver comando `nm` – `man`)

Construcción de bibliotecas – Dinámica

Compilar

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

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)

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
```

Actividad práctica

- 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.

Actividad práctica

- 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.

Actividad práctica

- 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`

Actividad práctica

- 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.

Actividad práctica

- 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`).

Actividad práctica

- 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`).