Programación en C

Compilación

El compilador

Compilador: programa que transforma código escrito en un lenguaje (source) a otro lenguaje (target). El término generalmente sólo es usado para referirse a transformaciones de código de alto nivel a código de bajo nivel.

En el caso de C, que es un lenguaje que compila a código nativo del procesador, el lenguaje source es C y el target es código binario particular a una arquitectura. Varios procesadores comparten una misma arquitectura y por lo tanto entienden el mismo código binario.

El Linux el compilador más usado es **gcc**.

Etapas

El proceso de pasar de C a binario consiste de 4 etapas, de las cuales sólo 1 es considerada “compilación”:

- Preprocessing
- Compilation
- Assembly
- Linking

Preprocessing

El preprocessor se encarga de resolver todas las directivas (instrucciones marcadas con #) y de remover comentarios.

- `#define x y`: reemplaza todas las instancias de `x` por `y`.
- `#include <file>|"file"`: inserta el contenido del archivo `file`.
  - `<file>`: busca en directorios definidos según la configuración del sistema.
  - `"file"`: busca primero en el directorio del archivo que contiene la directiva, sino lo encuentra, se comporta como `<file>`.
- `#if/#else/...`: incluyen/remueven código según una condición en tiempo de compilación

Ex 1.1, 1.2, 1.3, 1.4

Compilation & Assembly

Compilation

Traduce el código fuente en C en código assembly que es particular a una arquitectura.

Implica varios procesos para realizar distintas optimizaciones y convertir conceptos de alto nivel escritos en C en instrucciones simples de assembly.

Assembly

Traducción lineal de código assembly (legible) a binario.

Linking

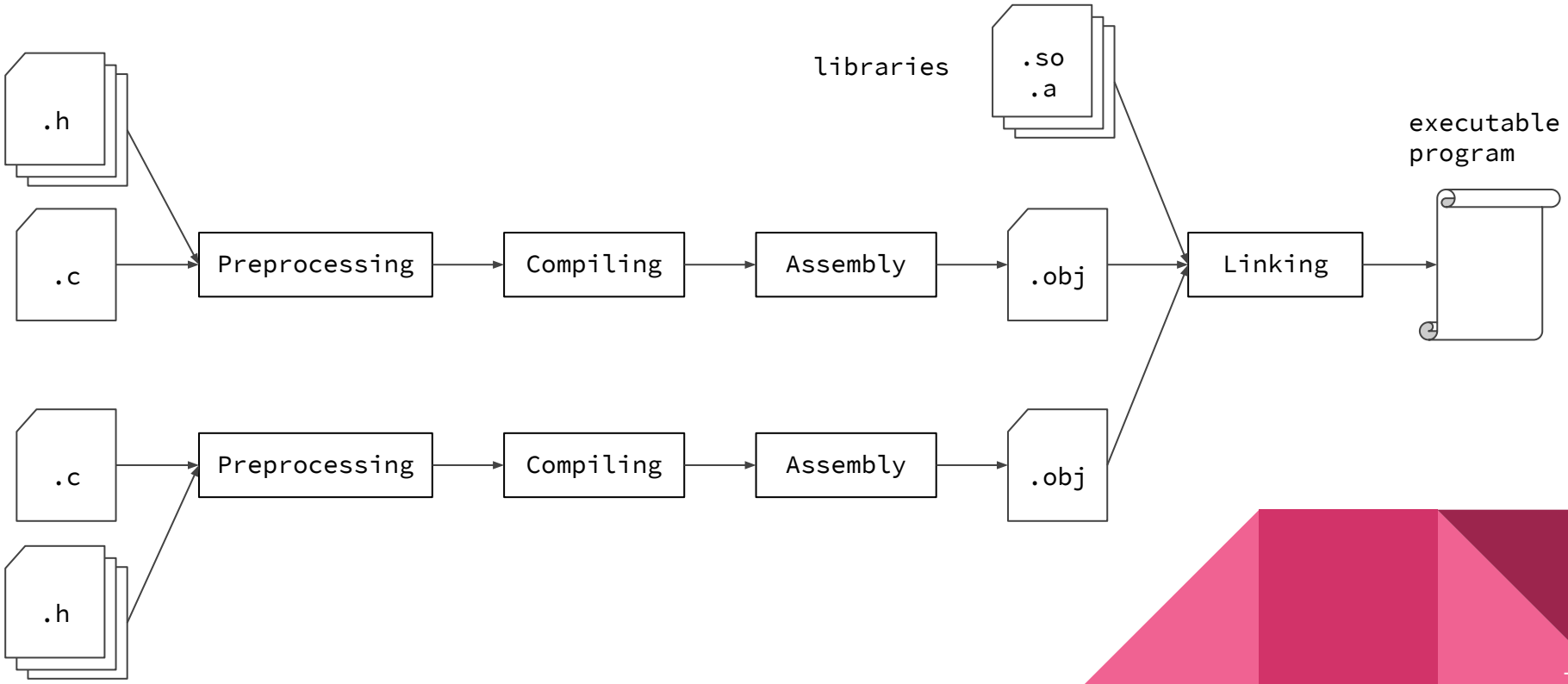
Las 3 fases anteriores trabajan sobre un único archivo de código fuente (.c).

Cada archivo puede declarar múltiples funciones que pretende usar sin definirlas. Al momento de linking, un programa llamado linker junta el binario de todos los archivos usados en el ejecutable y linkea el uso de una función en un archivo con su definición en otro.

Los archivos que usamos en los `include`, llamados headers (.h), permiten que una librería exponga las declaraciones de sus funciones para que el usuario las utilice sin tener acceso al código fuente.

Ex 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7

Compilación



Flags

Algunas flags útiles para gcc:

- `-g`: modo debug, permite trackear líneas de código y otros datos en un debugger como gdb.
- `-O0/-O1/-O2/-O3`: niveles de optimización (mayor número, más optimización).
- `-Wall`: activar todas las warnings

Para más info: **man gcc**.