Fundamentos de Software - Prácticas
Módulo II. Compilación y depuración de programas

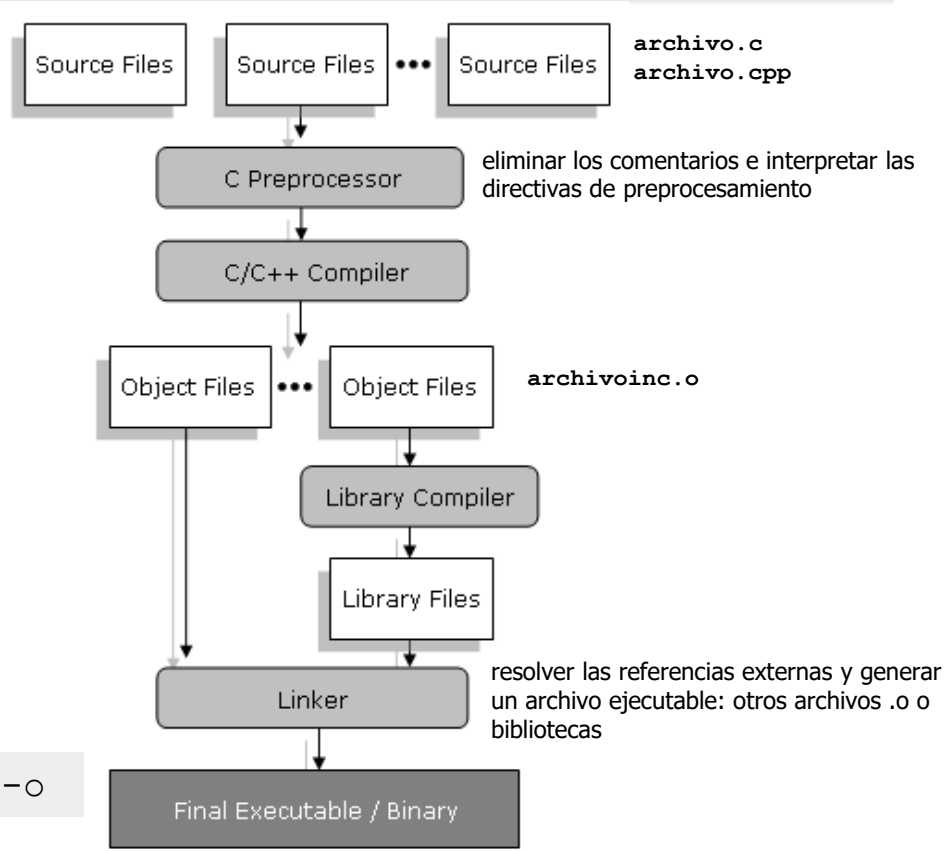
Práctica 7: Compilación de programas

Departamento de Lenguajes y Sistemas Informáticos – Universidad de Granada

UNIVERSIDAD DE GRANADA

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Conocer cómo la utilidad gcc/g++ realiza las distintas etapas del proceso de generación de un archivo ejecutable a partir de distintos archivos de código fuente..**
    - GCC: compilador integrado del proyecto GNU para los lenguajes C, C++, Objective C y Fortran. Archivo en código fuente → Ejecutable



Sección 7.2: Probar la opción `g++`, `g++ -c` y `g++ -o`

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Conocer las dependencias que se producen entre los distintos archivos implicados en el proceso de generación de un archivo ejecutable.**
  - Biblioteca: Colección de módulos objeto (constantes, variables y funciones)
  - Generar una biblioteca a partir de un conjunto de ficheros objeto:

```
ar -rvs libmates.a sin.o cos.o tan.o
```

  - Opciones de `g++`:
    - La opción `L` permite especificar directorios en donde g++ puede buscar las bibliotecas necesarias.
    - La opción `I` permite especificar directorios en donde g++ puede buscar los archivos de cabecera.

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Utilidad make:
    - Permite gestionar las dependencias y las comprobaciones de las modificaciones de archivos para construir el archivo ejecutable (objetivo: compilar solo los archivos que sean necesarios).
  - Archivo makefile:
    - Usar un fichero de texto para especificar las dependencias entre los archivos y las acciones que deben llevarse a cabo si se produce alguna modificación.

Sección 7.3

  - `make -f makefileA`
  - `make -p` (ver variables predefinidas)

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Estructura de un archivo makefile:
    - Una o varias reglas asociadas a la consecución de un objetivo concreto.
    - Las **reglas** están formadas por un **objetivo**, una **lista de dependencias** y las **acciones** u **órdenes** que son necesarias para alcanzar ese objetivo:
      - **Objetivo**: Normalmente un nombre identificativo (nombre de un programa)
      - **Dependencias**: especifican los archivos u otros objetivos posteriores de los que depende el objetivo asociado.
      - **Órdenes:** conjunto de una o más líneas de orden del shell y siempre deben tener un tabulador al principio de la línea de orden.

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Estructura de un archivo makefile:

```
programa1: main.o factorial.o hello.o
	g++ -o programa1 main.o factorial.o hello.o

main.o: main.cpp
	g++ -I./includes -c main.cpp

factorial.o: factorial.cpp
	g++ -I./includes -c factorial.cpp

hello.o: hello.cpp
	g++ -I./includes -c hello.cpp
```

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Estructura de un archivo makefile:
    - Diferentes resultados de la ejecución dependiendo de los ficheros que se hayan modificado.
    - Posibilidad de incluir funciones entre las dependencias.
    - Posibilidad de construir reglas que no tengan órdenes asociadas (objetivos simbólicos): comprobar lista de dependencias.
    - Posibilidad de reglas sin dependencias (reglas virtuales): normalmente al final del archivo si existen.
    - Posibilidad de incluir comentarios (#)

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Estructura de un archivo makefile – **Uso de variables**:
    - Definirlas al principio del fichero.
    - Incluirlas en las reglas entre paréntesis o llaves y anteponiéndoles el signo $

```
# Variable que indica el compilador que se va a utilizar
CC=g++

# Variable que indica el directorio en donde se encuentran los archivos de cabecera
INCLUDE_DIR= ./includes

# Variable que indica el directorio en donde se encuentran las bibliotecas
LIB_DIR= ./

programa2: main2.o factorial.o hello.o libmates.a
	$(CC) -L$(LIB_DIR) -o programa2 main2.o factorial.o hello.o -lmates
```

Práctica 7: Compilación de programas

- OBJETIVOS:
  - **Saber construir un archivo makefile sencillo que permita mantener las dependencias entre los distintos módulos de un pequeño proyecto software.**
  - Estructura de un archivo makefile – **Uso de variables**:
    - Variables especiales (Tabla 7.2)
      - **`$@:`** Nombre del objetivo de la regla en la que nos encontramos.
        - Representar el nombre que se le asociará al programa ejecutable.
      - **`$<:`** Primera dependencia de la regla en la que nos encontramos.
        - Representar el archivo aportado como primera dependencia en una regla.
      - **`$?:`** Dependencias de la presente regla que hayan sido actualizadas.
        - Referenciar directamente varias dependencias.
      - **`$^:`** Todas las dependencias separadas por un espacio en blanco.
        - Referenciar todos los archivos indicados en las dependencias de una regla.

Práctica 7: Compilación de programas

- COMANDOS Y UTILIDADES:
  - `gcc/g++`
  - `ar`
  - `make`