II.2. EL COMPILADOR GCC

GCC es un compilador originario del proyecto GNU que se utiliza para compilar programas escritos en C, C++, Objective C y Fortran. Este compilador es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr (actualmente tanto dentro de entornos UNIX y LINUX como dentro de entornos-Windows).

GCC son las siglas de "***G***NU ***C***ompiler ***C***ollection". Una variante de éste compilador es el ***G++*** que se utiliza para compilar programas escritos en C++.

II.2.1.- Sintaxis.

```
gcc [ -opción [argumento(s)_opción]] nombre_fichero
g++ [ -opción [argumento(s)_opción]] nombre_fichero
```

Las opciones en la sintaxis van precedidas de un guión, como es habitual en UNIX y se conforman de una o más letras, sin embargo, no pueden agruparse varias opciones tras un mismo guión. Algunas opciones requieren de argumentos (que en su mayoría son nombres de archivos o directorios) y otras no. Al final de la sintaxis, se pueden dar varios nombres de archivo a incluir en el proceso de compilación.

II.2.2.- Sufijos más comunes en nombres de archivo.

Son habituales las siguientes extensiones o sufijos de los nombres de archivo:

`.c`	fuente en C
`.C .cc .cpp .c++ .cp .cxx`	fuente en C++; se recomienda .cpp
`.m`	fuente en Objective-C
`.i`	C preprocesado
`.ii`	C++ preprocesdo
`.s`	fuente en lenguaje ensamblador
`.o`	código objeto
`.h`	archivo para preprocesador (encabezados), no suele figurar en la linea de comando de gcc

II.2.3.- Descripción de las opciones más importantes

`-c`

Realiza solamente el preprocesamiento y la compilación de los ficheros fuentes. No se lleva a cabo la etapa de enlazado.

`-o`

fichero_salida especifica el nombre del fichero de salida, resultado de la tarea solicitada al compilador.

Si no se especifica la opción -o, el compilador generará un fichero ejecutable (su tendencia es la de realizar el trabajo completo: compilar y enlazar) y le asignará un nombre por defecto: a.exe (MS-DOS) o a.out (Linux/Unix). Si se especifica la opción -c generará el fichero objeto *nombre_fichero*.o

`-I`

path especifica el directorio donde se encuentran los ficheros a incluir por la directiva #include. Se puede utilizar esta opción varias veces para especificar distintos directorios.

`-L`

path especifica al enlazador el directorio donde se encuentran los ficheros de biblioteca. Como ocurre con la opción -I, se puede utilizar la opción -L varias veces para especificar distintos directorios de biblioteca. Los ficheros de biblioteca que deben usarse se especifican con la opción -l*fichero*. Esta opción hace que el enlazador busque en los directorios de bibliotecas (entre los que están los especificados con -L) un fichero de biblioteca llamado lib*fichero*.a y lo usa para enlazarlo.

`-D`

nombre[=*cadena*] define una constante simbólica llamada *nombre* con el valor *cadena*. Si no se especifica el valor, *nombre* simplemente queda definida. *cadena* no puede contener blancos ni tabuladores. Equivale a una línea #define al principio del fichero fuente, salvo que si se usa -D, el ámbito de la macrodefinición incluye todos los ficheros especificados en la llamada al compilador.

`-W`

all Muestra todos los mensajes de advertencia del compilador.

`-g`

Incluye en el ejecutable la información necesaria para poder trazarlo empleando un depurador.

`-v`

Muestra con detalle en stderr las órdenes ejecutadas por gcc.

II.2.4.- Etapas de compilación.

El proceso de compilación involucra cuatro etapas sucesivas:

- Preprocesado
- Compilación
- Ensamblado
- Enlazado

Para pasar de un programa fuente a un archivo ejecutable sería necesario realizar estas cuatro etapas en forma sucesiva, sin embargo gcc y g++ son capaces de llevar a cabo todo el proceso de una sola vez.

1. **Preprocesado.**
   En esta etapa directivas son interpretadas por el preprocesador. Las variables inicializadas con #define se sustituyen en el código por su valor en todos los lugares donde aparece su nombre.

2. **Compilación.**
   La compilación transforma el código C o C++ en el lenguaje ensamblador propio del procesador de nuestra máquina.

3. **Ensamblado.**
   El ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto, que es un archivo binario en lenguaje de máquina ejecutable por el procesador.

4. **Enlazado**
   Las funciones de C/C++ incluidas en el código fuente, tal como printf(), se encuentran ya compiladas y ensambladas en bibliotecas existentes en el sistema. Es preciso incorporar de algún modo el código binario de estas funciones a nuestro ejecutable. En esto consiste la etapa de enlace o ligado, donde se reúnen uno o más módulos en código objeto con el código existente en las bibliotecas, en este caso del lenguaje C/C++.

II.2.4.1.- Todo en un solo paso.

Los cuatro pasos definidos de la etapa de compilación se puede llevar a cabo en un solo paso con el uso del gcc o g++ según corresponda. Veamos los siguientes ejemplos:

gcc hola.c
compila el programa en C hola.c, generando un archivo ejecutable a.out (a.exe)

gcc -o hola hola.c
compila el programa en C hola.c, generando un archivo ejecutable hola.

g++ -o hola hola.cpp
compila el programa en C++ hola.cpp, generando un archivo ejecutable hola.

gcc -c hola.c
no genera el ejecutable, sino el código objeto, en el archivo hola.o. Si no se indica un nombre para el archivo objeto, usa el nombre del archivo en C y le cambia la extensión por .o.

gcc -c -o objeto.o hola.c
genera el código objeto indicando el nombre de archivo.

g++ -c hola.cpp
igual para un programa en C++.

g++ -o ~/bin/hola hola.cpp
genera el ejecutable hola en el subdirectorio bin del directorio propio del usuario.

g++ -L/lib -L/usr/lib hola.cpp
indica dos directorios donde han de buscarse las bibliotecas. La opción -L debe repetirse para cada directorio de búsqueda de bibliotecas.

g++ -I/usr/include hola.cpp
indica un directorio para buscar archivos de encabezado (de extensión .h).