El compilador GCC

Juan Bertoni, Santiago Calligari, Leandro Spagnolo, Julián Cabrera

3 de agosto de 2020

Índice

Resumen

GCC es un conjunto de compiladores creados por el proyecto GNU. GCC es software libre y lo distribuye la FSF. Originalmente GCC significaba GNU C Compiler, por que solo compilaba el lengua C y posteriormente se extendió a otros como C++, Fortran, Ada y otros. Su principal función es la 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. El compilador g++ se comporta de manera muy similar, solo que tiene algunas configuraciones especiales para compilar C++.

1. Sintaxis

gcc [opción | archivo] ...
g++ [opción | archivo] ...

2. Sufijos en nombres de archivo

Esta tabla te servirá para entender con qué tipo de archivo estás trabajando en cada momento:

.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++ preprocesado
.s	Fuente en lenguaje ensamblador
.o	Código objeto
.h	Archivo para preprocesar(encabezados); no suelen figurar en la línea de comando de gcc

Cuadro 1: Extensiones más comunes para make

Opciones del comando:

- - c realiza preprocesamiento y compilación, obteniendo el archivo en código objeto; no realiza el enlazado.
- - E realiza solamente el preprocesamiento, enviando el resultado a la salida estándar.
- -o archivo indica el nombre del archivo de salida, cualesquiera sean las etapas cumplidas.
- -Iruta especifica la ruta hacia el directorio donde se encuentran los archivos marcados para incluir en el programa fuente (estos son los archivos .h que se declaran al principio con #include). No lleva espacio entre la I y la ruta, así: -I/usr/include. Si se ingresa muchas veces, busca en cada uno de los directorios especificados en el orden especificado; por ejemplo -I/usr/include -I/home/usuario. También podemos escribir, por ejemplo, -I../headers (usamos dos puntos en vez de uno), que indica buscar tanto en directorio actual como en su subdirectorio ./headers.
- -L especifica la ruta hacia el directorio donde se encuentran los archivos de biblioteca con el código objeto de las funciones referenciadas en el programa fuente. No lleva espacio entre la L y la ruta, así: -L/usr/lib. Las mismas consideraciones que aplican a -I también lo hacen a -L.
- -Wall muestra todos los mensajes de error y advertencia del compilador, incluso algunos cuestionables pero en definitiva fáciles de evitar escribiendo el código con cuidado.
- -g incluye en el ejecutable generado la información necesaria para poder rastrear los errores usando un depurador, tal como GDB (GNU Debugger). En otras palabras, realiza un enlazado con el código fuente que permite acceder al mismo y ejecutarlo línea a línea para poder depurar bugs.
- . -v muestra los comandos ejecutados en cada etapa de compilación y la versión del compilador. Es un informe muy detallado.

3. Mi primer programa

Imaginemos que quieres crear un primer programa con el conocido "Hola mundo" en lenguaje C. El primer paso sería crear un archivo con tu editor de texto favorito, (por ejemplo nano). Mi primer programa se llamará Hola.c (Acordarse de especificar qué tipo de archivo estamos trabajando por eso pongo .c como sufijo). Acto seguido nos queda escribir el código típico de C como el que venimos trabajando en Codeblocks desde años anteriores; usaremos el código fuente Hola.c que se encuentra en el apéndice. Una vez hecho esto, lo guardan y les debería quedar en su directorio el código de su programa.

A partir de acá se pueden hacer varias cosas: El primer caso sería el que solo quieras compilar este código y no tengas que enlazarlo con ningún otro. Para hacer esto utilizaremos los siguientes comandos:

$ gcc -o Hola Hola.c

Esto creará un ejecutable con fuente de un solo archivo el cual es remarcado en color verde, y para ejecutarlo hay que usar ./Hola

Ahora el segundo caso que puede suceder es que quieras enlazar dos módulos objeto para crear un archivo ejecutable hecho a partir de varios códigos, para esto simplemente tenés que primero conseguir el código objeto con el comando:

$ gcc -c Hola.c

$ gcc -c buen_dia.c

Y luego de esto solo hace falta enlazarlos con el comando:

$ gcc -o Bienvenida Hola.o buen_dia.o

4. Ventajas del uso de GCC

Lo primero de todo, GCC es un compilador portable —se ejecuta en la mayoría de las plataformas disponibles hoy, y puede producir salidas para muchos tipos de procesadores. Además de procesadores usados en ordenadores personales, también soporta microcontroladores, DSPs y CPUs de 64 bits. GCC no es solo un compilador nativo —también puede compilar cruzado cualquier programa, produciendo ficheros ejecutables para un sistema diferente desde el que GCC está siendo usado. Esto permite compilar software para sistemas embebidos que no son capaces de ejecutar un compilador. GCC está escrito en C con un fuerte enfoque hacia la portabilidad, y puede compilarse a sí mismo, así puede ser adaptado a nuevos sistemas fácilmente. GCC tiene múltiples frontends, para parsear diferentes lenguajes. Los programas en cada lenguaje pueden ser compilados, o compilados de manera cruzada, para cualquier arquitectura. Por ejemplo, un programa en ADA puede ser compilado para un microcontrolador, o un programa en C para un supercomputador. GCC tiene un diseño modular, permitiendo que el soporte para nuevos lenguajes y arquitecturas sea añadido. Añadir un nuevo front-end a GCC habilita el uso de este lenguaje en cualquier arquitectura y proporciona que estén disponibles facilidades (tales como librerías) en tiempo de ejecución. De manera similar, si se añade soporte para una nueva arquitectura éste se vuelve disponible para todos los lenguajes. Finalmente, y de manera más importante, GCC

es software libre, distribuido bajo la GNU General Public License (GNU GPL). Esto significa que se tiene la libertad para usar y modificar GCC, como con todo el software de GNU. Si se necesita soporte para un nuevo tipo de CPU, un nuevo lenguaje, o una nueva funcionalidad es posible añadirla uno mismo o contratar a alguien para mejorar GCC de manera personalizada. Se puede contratar a alguien para arreglar un error si esto es importante en el trabajo cotidiano. Más allá, hay libertad para compartir cualquier mejora hecha a GCC. Como resultado de esta libertad, se pueden usar las mejoras hechas a GCC por otras personas. Las muchas funcionalidades ofrecidas por GCC hoy muestran cómo esta libertad de cooperar funciona en tu beneficio, y en el de cualquiera que use GCC.

5. Apéndice

5.1. Hola.c

```
#include <stdio.h>
int main(){ printf("Hola Mundo"); return 0; }
```