

Comandos gcc y gdb

Diego Hildebrando Ramírez Aguilera

**Comandos gcc y gdb**

**Introducción**

Aquellos que desarrollan en C, conocen de las dificultades a las que se enfrenta cuando trata de depurar un programa, que, por ejemplo, por qué no se agrega un nodo a una lista o por qué no se copia determinado string. GDB (Gnu Project Debugger) es una herramienta que permite entre otras cosas, correr el programa con la posibilidad de detenerlo cuando se cumple cierta condición, avanzar paso a paso, analizar que ha pasado cuando un programa se detiene o cambiar algunas cosas del programa como el valor de las variables.

GDB es una herramienta muy poderosa que nos ayudará a encontrar esos errores difíciles, por ejemplo, cuando los punteros no apuntan a donde estamos pensando. Si bien este tutorial está pensado para el lenguaje C, probablemente también sirva para depurar programas en Fortran o C++ con los mismos comandos o similares.

**Comandos gcc**

GCC es un compilador integrado del proyecto GNU para C, C++, Objetive C y Fortran; 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.

La sigla GCC significa "GNU Compiler Collection". Originalmente significaba "GNU C Compiler"; todavía se usa GCC para designar una compilación en C. G++ refiere a una compilación en C++.

GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; 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.

La sigla GCC significa "GNU Compiler Collection". Originalmente significaba "GNU C Compiler"; todavía se usa GCC para designar una compilación en C. G++ refiere a una compilación en C++.

GDB es una herramienta muy poderosa que nos ayudará a encontrar esos errores difíciles, por ejemplo cuando los punteros no apuntan a donde estamos pensando. Si bien este tutorial está pensado para el lenguaje C, probablemente también sirva para depurar programas en Fortran o C++ con los mismos comandos o similares.

**Sintaxis.**

gcc [ opción | archivo ] ...

g++ [ opción | archivo ] ...

Las opciones van precedidas de un guión, como es habitual en UNIX, pero las opciones en sí pueden tener varias letras; no pueden agruparse varias opciones tras un mismo guión. Algunas opciones requieren después un nombre de archivo o directorio, otras no. Finalmente, pueden darse varios nombres de archivo a incluir en el proceso de compilación**.**

**Ejemplos.**

gcc hola.c

compila el programa en C hola.c, gener un archivo ejecutable a.out.

gcc -o hola hola.c

compila el programa en C hola.c, gener un archivo ejecutable hola.

g++ -o hola hola.cpp

compila el programa en C++ hola.c, gener un archivo ejecutable hola.

gcc -c hola.c

no genera el ejecutable, sino el código objeto, en el archivo hola.o. Si no s 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 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).

Opciones.

- c

realiza preprocesamiento y compilación, obteniento 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. No lleva espacio entre la I y la ruta, así: -I/usr/include

-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

-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).

-v

muestra los comandos ejecutados en cada etapa de compilación y la versión del compilador. Es un informe muy detallado.

**Etapas de compilación.**

El proceso de compilación involucra cuatro etapas sucesivas: preprocesamiento, compilación, ensamblado y enlazado. Para pasar de un programa fuente escrito por un humano a un archivo ejecutable es necesario realizar estas cuatro etapas en forma sucesiva. Los comandos gcc y g++ son capaces de realizar todo el proceso de una sola vez.

**1. Reprocesado.**

En esta etapa se interpretan las directivas al preprocesador. Entre otras cosas, las variables inicializadas con #define son sustituídas en el código por su valor en todos los lugares donde aparece su nombre.

Usaremos como ejemplo este sencillo programa de prueba, circulo.c:

/\* Circulo.c: calcula el área de un círculo.

Ejemplo para mostrar etapas de compilación.

\*/

#define PI 3.1416

main()

{

float area, radio;

radio = 10;

area = PI \* (radio \* radio);

printf("Circulo.\n");

printf("%s%f\n\n", "Area de circulo radio 10: ", area);

}

**Los comandos básicos GDB**

**Compilar programas en C con opcion -g**

% gcc -g prog.c ... -o <archivo binario>

Si no se especifica -o, se genera el archivo a.out

**Gdb se invoca con:**

gdb <archivo binario>

**Para obtener ayuda:**

help <comando>

Poner breakpoint en funciones:

(gdb) b main

Borrar breakpoints:

(gdb) del <nro. del break>

Mostrar nros. de breakpoints

(gdb) info break

Correr el programa:

(gdb) run

Ver el encadenamiento de funciones de la tarea actual

(la que le tiene la CPU en ese momento):

(gdb) where

subir y bajar en la pila para ver variables de funciones intermedias:

(gdb) up -> Si F llamo a G y estamos en G, pasa a F

(gdb) down -> vuelve a G

Especifico para programas en C:

**- Ejecutar paso a paso instrucciones en C (step y next):**

(gdb) s -> ejecuta una instruccion. Si hay una llamada a una funcion,

se detiene en la primera instruccion de esa funcion.

(gdb) n -> ejecuta una instruccion. Si hay llamadas a funciones, se

las ejecuta completamente sin detenerse.

**Imprimir valores de expresiones**

(gdb) p x->a.d + 1 ( p de print )

**Imprimir las variables locales de la funcion examinada:**

(gdb) info locals

**Especifico para programas en Assembler:**

**Ejecutar paso a paso instrucciones de maquina:**

(gdb) stepi -> ejecuta una instruccion de maquina.

(gdb) nexti -> si es un call, ejecuta el call hasta el retorno,

si no, ejecuta una instruccion de maquina.x

**Bibliografía**

[**https://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm**](https://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm)

[**https://lihuen.linti.unlp.edu.ar/index.php/C%C3%B3mo\_usar\_GDB**](https://lihuen.linti.unlp.edu.ar/index.php/C%C3%B3mo_usar_GDB)