COMANDOS GCC Y GDB

Felipe Alvarado Galicia

UNIVERSIDAD POLITÉCNICA
DE LA ZONA METROPOLITANA DE GUADALAJARA

11 DE FEBRERO DE 2020

UPZMG
Prof: Carlos Enrique Moran Garabito

Introducción:

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.

Marco teorico:

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.

Comandos gcc

PATH_HOLD=$PATH : Este comando guarda tu PATH actual antes de que sea modificado, para poder restaurarlo después de la instalación.

export PATH=/opt/gnat/bin:$PATH : Este comando permite encontrar el compilador Ada de GNAT para construir Ada.

touch treeprs.ads [es]info.h nmake.ad[bs] : Este comando crea los ficheros necesarios para construir Ada. Puedes omitir este paso si no quieres compilar el frontal (frontend) para Ada.

CC=/usr/bin/gcc : Este comando es para evitar el uso del nuevo *PATH* que pone al **gcc** de GNAT como compilador primario.

--enable-languages=c,c++,objc,f77,ada,java : Este comando construye todos los lenguajes disponibles en el paquete GCC. Puedes modificar este comando para eliminar los lenguajes que no desees.

--enable-shared --enable-threads=posix --enable-__cxa_atexit : Estos comandos son necesarios para construir las librerías C++ según los estándares publicados.

--enable-clocale=gnu : Este comando es un mecanismo de seguridad para datos de locale incompletos.

make gnatlib_and tools : Este comando completa el proceso de construcción de Ada. Omítelo si no incluiste Ada entre los lenguajes.

Contenido

El

paquete GCC contiene **c++**, **c++filt**, **cpp**, **g++**, **g77**, **gcc**, **gccbug**, **gcov**, **glob**, **gnat**, **gnatbind**, **gnatbl**, **gnatchop**, **gnatfind**, **gnatkr**, **gnatlink**, **gnatls**, **gnatmake**, **gnatprep**, **gnatpsta**, **gnatpsys**, **gnatxref** y las librerías GCC.

Descripciones

Los programas y librerías que no se describen aquí se encuentran documentados en la página sobre GCC-3.3.1 del libro LFS.

g77

g77 es el compilador de Fortran invocado por **gcc**.

add2line

add2line convierte los elementos orbitales de 2 líneas contenidos en un fichero del formato ASCII al binario y los añade a los ficheros orbdata.

gcov

gcov es un programa de chequeo de cobertura.

gdb

gdb es el depurador de GNAT.

gnatbind

gnatbind se usa para vincular los objetos compilados.

gnatbl

gnatbl es el enlazador de Ada.

gnatchop

gnatchop renombra ficheros para que cumplan con las convenciones de nombres de ficheros del Ada estándar.

gnatelim

gnatelim sirve para detectar y eliminar subprogramas sin usar en una partición Ada.

gnatfind

gnatfind es el buscador de definiciones/usos de GNAT.

gnatgcc

gnatgcc es el compilador.

gnathtml.pl

gnathtml.pl convierte ficheros de código Ada a HTML para visualizarlos con navegadores Web.

gnatkr

gnatkr sirve para determinar el nombre truncado de un fichero dado, cuando se trunca a un largo máximo especificado.

gnatlink

gnatlink se usa para enlazar programas y construir un ejecutable.

gnatls

gnatls es el navegador de unidades compiladas.

gnatemake

gnatmake es una utilidad automática para make.

gnatmem

gnatmem es la utilidad GNAT que supervisa la actividad de asignación y desasignación dinámica de un programa.

gnatprep

gnatprep es el preprocesador externo de GNAT.

gnatpsta

gnatpsta determina los valores de todos los parámetros relevantes en Standard y los muestra por la salida estándar.

gnatpsys

gnatpsys determina los valores de todos los parámetros relevantes en System y los muestra por la salida estándar.

gnatstub

gnatstub es un generador de cabos de cuerpo (body stubs).

gnatxref

gnatxref es referenciador cruzado de GNAT.

gvd

gvd es el Depurador Visual GNU.

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.

Ambas instrucciones muestran la direccion de la siguiente instruccion a ejecutar en hexadecimal. Ej.:

(gdb) 0x080483d4 in dump ()

- Para mostrar 10 instrucciones de maquina a partir de una direccion en hexadecimal:

(gdb) x/10i 0x080483d2

0x80483d2 <dump+6>: mov %ebp,%eax

0x80483d4 <dump+8>: mov %eax,0xfffffffc(%ebp)

0x80483d7 <dump+11>: mov 0xfffffffc(%ebp),%eax

- Para mostrar el contenido de los registros

(gdb) info register

- Para mostrar el contenido a partir de una dirección en hexadecimal:

(gdb) x/10x 0xbffff5c8

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.77777777777777777777777

Bibliografias:

https://users.dcc.uchile.cl/~lmateu/CC41C/gdb.txt

https://lihuen.linti.unlp.edu.ar/index.php/C%C3%B3mo_usar_GDB

http://www.nongnu.org/gccintro-es/gccintro.es.html

https://gcc.gnu.org/

https://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm

https://lihuen.linti.unlp.edu.ar/index.php/C%C3%B3mo_usar_GDB