GDB y Free Pascal Compiler

Gonzalo Soriano
gonzalo_soriano@yahoo.com.ar

8 de octubre de 2007

Resumen

GDB [1] o GNU Debugger es una herramienta de debuggin para la detección y corrección de errores en varios lenguajes de programación, entre los que se encuentran C, C++, Objective-C, Fortran, **Pascal**, Java, assembly, Modula-2 y Ada.
GDB fue escrito por Richard Stallman en 1988, y cuenta con la desventaja de no tener interfaz gráfica, por lo que debe usarse por línea de comandos. Sin embargo, existen otros depuradores (como el DDD [2]) que cuentan con una interfaz gráfica y usa el GDB.

1. Primeros pasos con el GDB

Lo primero que tenemos que hacer es compilar el programa, para eso usaremos el Free Pascal [3].

1.1. Compilar el fuente

Si suponemos que nuestro programa se llama `ejemplo.pas` entonces usualmente lo compilaríamos de la forma:

```
user$ fpc ejemplo.pas
```

Pero para informarle ahora tenemos que informarle al compilador que lo tiene que compilar de una forma particular para que después podamos usar el GDB; y eso lo hacemos con el *flag* `-g` antes del nombre del programa.

```
user$ fpc -g ejemplo.pas
```

1.2. Iniciar el GDB

Como dijimos en el resumen, el GDB se usa por línea de comandos, y una vez parados en ella con solo poner `user$ gdb ejemplo.pas` [1] iniciamos el GDB. Si queremos pasarle parámetros al programa que vamos a ejecutar, éste es el momento de pasarlos con el comando

```
set args param1 param2 ... paramN
```

Para empezar a ejecutar el código fuente del programa que queremos depurar, tenemos que usar el comando `start`. Una vez que empezamos a debuggear tenemos distintos comandos, que usaremos en función de nuestras necesidades; los más comunes son:

- **start:** Inicia la ejecución del programa, pero no ejecuta ninguna sentencia. Su llamada es `(gdb) start`.
- **step:** Ejecuta la próxima instrucción (equivalente al *Trace into* [F7] de Pascal). Su llamada es `(gdb) step n`. Donde `n` es la cantidad de pasos que tiene que seguir, si no se lo
- **next:** Ejecuta la próxima instrucción, pero si es un procedimiento o una función la ejecuta en un solo paso. Equivalente al *Step over* [F8] de Pascal. Su llamada es simplemente `(gdb) next`.
- **list:** Muestra la próxima instrucción a ejecutarse y su contexto. Su llamada es simplemente `(gdb) list`.
- **print:** Muestra el valor de una variable, una expresión lógica, o la llamada a una función. Su llamada es `(gdb) print param`, donde *param* es el nombre variable a observar, la llamada a la función o la expresión.
- **break:** Es el equivalente a los *breakpoints* del Turbo Pascal. Su función es de detener la corrida del programa en una línea en particular; pero sin terminar la ejecución. El *break* puede llamarse de las siguientes formas:

[1] Siendo *ejemplo.pas* el nombre del programa a debuggear.

- `(gdb) break n`. Inserta un *breakpoint* en la n-esima línea del fuente que estamos ejecutando.
- `(gdb) break fuente.pas:n`. Inserta un *breakpoint* en la n-esima línea del archvio `fuente.pas`.
- `(gdb) break funcion`. Inserta un *breakpoint* al comienzo de la función indicada.

- **delete *N*:** Borra el *breakpoint N*. Su llamada es simplemente `(gdb) delete N`.
- **continue:** Continúa la ejecución del programa hasta el próximo *breakpoint* o el fin del programa. Su llamada es simplemente `(gdb) continue`.
- **run:** Corre, desde el principio, el programa hasta encontrar un *breakpoint* o el fin del programa. Su llamada es simplemente `(gdb) run`.
- **help:** Muestra la ayuda del GDB. Su llamada es simplemente `(gdb) help`.
- **help running:** Muestra los comandos básicos de GDB para debuggear un programa. Su llamada es simplemente `(gdb) help running`.
- **help command:** Muestra una breve descripción del comando. Su llamada es simplemente `(gdb) help command`.
- **q/quit:** Sale del GDB. Su llamada es simplemente `(gdb) q`.

2. Anexo I: Compilando en Linux

Para compilar un programa en Linux, al igual que en cualquier plataforma, necesitamos un código fuente y un compilador.
El compilador que vamos a usar es el *Free Pascal*, el cual tiene la ventaja de ser multiplataforma y lo podemos obtener de `http://www.freepascal.org/download.var`.
Ahora lo que nos falta es el programa, de lo cual nos tenemos que encargar nosotros :). Para escribirlo podemos usar una IDE[2] como el Anjuta [4]; o cualquier editor de texto.
Usaremos como ejemplo el siguiente programa:

`Programa ejemplo.pas`

```
program ej1;
 uses crt;
 var
     int : integer;
 begin
       writeln('Funciona!!!!');
       readln(int);
       writeln('ingresastes el ',int);
       readkey;
 end.
```

[2]Integrated Development Environment, o Entorno de Desarrollo Integrado

Ahora que tenemos el código fuente del programa a compilar (ejemplo.pas) y el compilador (Free Pascal), simplemente tenemos que abrir una terminal, línea de comandos o consula y pararnos en el directorio donde esta guardado el archivo *ejemplo.pas*. Una vez ubicados en la posición correcta, escribimos:

```
user$ fpc ejemplo.pas
```
[3]

Si no tubimos problemas al momento de compilar, la salida será de la forma:

```
Free Pascal Compiler version 2.0.0 [2005/09/09] for i386
Copyright (c) 1993-2005 by Florian Klaempfl
Target OS: Linux for i386
Compiling ejemplo.pas
Linking ejemplo
10 Lines compiled, 0.6 sec
```

Si al mismo programa le agregamos una variable que no se usa, por ejemplo:

```
 int, aux : integer;
```

Free Pascal nos lo informará de la siguiente forma (ver línea 5):

```
Free Pascal Compiler version 2.0.0 [2005/09/09] for i386
Copyright (c) 1993-2005 by Florian Klaempfl
Target OS: Linux for i386
Compiling ejemplo.pas
ejemplo.pas(4,7) Note: Local variable "aux" not used
Linking ejemplo
10 Lines compiled, 0.2 sec
```

Si ahora cambiamos el programa usando una variable sin inicializar de la forma:

```
program ej1;
 uses crt;
 var
     int : integer;
 begin
       writeln('Funciona!!!!, de entrada int vale', int);
       readln(int);
       writeln('ingresastes el ',int);
       readkey;
 end.
```

Free Pascal nos lo informará de la siguiente forma (ver línea 5):

```
Free Pascal Compiler version 2.0.0 [2005/09/09] for i386
Copyright (c) 1993-2005 by Florian Klaempfl
Target OS: Linux for i386
Compiling ejemplo.pas
ejemplo.pas(6,2) Warning: Variable "int" does not seem to be initialized
Linking ejemplo
10 Lines compiled, 0.1 sec
```

[3] Recuerden que si después quieren usar el GDB para debuggearlo tienen que agregarle el flag -g.

Si nos hubieramos olvidado de declarar la variable int, Free Pascal de la siguiente forma:

```
Free Pascal Compiler version 2.0.0 [2005/09/09] for i386
Copyright (c) 1993-2005 by Florian Klaempfl
Target OS: Linux for i386
Compiling ejemplo.pas
ejemplo.pas(7,9) Error: Wrong number of parameters specified
ejemplo.pas(7,13) Error: Illegal expression
ejemplo.pas(8,28) Error: Wrong number of parameters specified
ejemplo.pas(8,32) Error: Illegal expression
ejemplo.pas(11) Fatal: There were 4 errors compiling module, stopping
ejemplo.pas(11) Error: Compilation aborted
Error: /usr/bin/ppc386 returned an error exitcode (normal if you did not
specifiy a source file to be compiled)
```

Notar que en la última línea nos indica que el programa no fue compilado.

Referencias

[1] http://sourceware.org/gdb/

[2] http://www.gnu.org/software/ddd/

[3] http://www.freepascal.org/

[4] http://www.anjuta.org/

Índice