Manual breve de gdb

El presente documento no pretende ser de ningún modo un manual de gdb con todas sus opciones, sino más bien una pequeña guía práctica que ayude al programador a resolver de forma sencilla muchos de los problemas que se plantean al tratar de verificar la correcta ejecución de un programa.

Invocación de gdb

Para poder utilizar adecuadamente el depurador, es necesario compilar los programas con la opción `-g`, de este modo, podremos visualizar el código fuente del programa durante su depuración.

El depurador gdb puede ser invocado de varios modos, de los cuales los más comunes son los siguientes:

`$ gdb`	Sin argumentos
`$ gdb programa`	`programa` es el programa que deseamos depurar
`$ gdb programa core`	`programa` es el programa que generó el `core` y deseamos saber dónde y por qué se produjo el fallo que generó el `core`
`$ gdb programa pid`	Sirve para iniciar la depuración de un programa que se encuentra en ejecución. El identificador del proceso correspondiente es `pid`

Para finalizar la ejecución de gdb teclearemos la orden `quit` o `q`, en caso de que el programa que estemos depurando no haya finalizado su ejecución nos preguntará si realmente estamos seguros.

Órdenes más comunes

Cualquier orden que se indique a gdb es repetida automáticamente cada vez que se pulsa la tecla Enter. Seguidamente las órdenes más comúnmente empleadas.

list

Realiza el listado del programa o de una función
Ejemplos:

```
(gdb) l main
(gdb) l funcion
(gdb) l 1,40
```

La última línea se utiliza para realizar un listado desde la línea 1 a la 40

break

Coloca un breakpoint en una función o línea de programa
Ejemplos:

```
(gdb) b funcion
(gdb) b 30
```

run

Comienza la ejecución del programa con los argumentos que sean indicados
Ejemplos:

```
(gdb) run
(gdb) run par1 par2
```

print

Se utiliza para visualizar el valor de una expresión. Las expresiones que se desean visualizar deben pertenecer al contexto del punto actual de ejecución. Esto quiere decir que no es posible visualizar el valor de una expresión incluida en una función que no está en ejecución.
Ejemplos:

```
(gdb) p tamanno
(gdb) p matriz
```

backtrace

Sirve para visualizar las funciones que han sido invocadas y desde dónde han sido invocadas. Aparecen en forma de pila de modo que en la cima se muestra la función actual y el la base la función más vieja. Esta orden es muy útil para determinar cómo y dónde ha efectuado una operación ilegal un programa.
Ejemplo:

```
(gdb) bt
```

continue

Se emplea para continuar la ejecución de un programa detenido en un breakpoint.
Ejemplo:

```
(gdb) c
```

next

Sirve para ejecutar la próxima línea de programa sin entrar en funciones. Ejecuta toda la función en una única operación.
Ejemplo:

```
(gdb) n
```

step

Se utiliza para ejecutar la siguiente línea de programa, pero entrando en su caso en el código de la función.

Ejemplo:

```
(gdb) s
```

jump

Se utiliza para saltar al punto de programa indicado sin ejecutar el código intermedio.

Ejemplo:

```
(gdb) j 50
```

watch

Se emplea para detener la ejecución del programa cuando el valor de una expresión cambie.

Ejemplo:

```
(gdb) watch ret
```

info

Sirve para obtener información acerca de breakpoints, watchpoints, tipos, variables, etc.

Ejemplo:

```
(gdb) i b
```

Visualiza los actuales breakpoints

delete

Se utiliza para eliminar un breakpoint

Ejemplo:

```
(gdb) d 5
```

Elimina el breakpoint 5

set

Sirve para modificar el contenido de una variable

Ejemplo:

```
(gdb) set retorno=34
```

help

Se emplea para obtener ayuda

Ejemplo:

```
(gdb) help info
```