GNU Debugger (GDB)

16 novembre 2020

GDB est un outil de type "débogeur symbolique". Il permet de contrôler l'exécution d'un programme, par exemple instruction après instruction, et de tracer l'évolution de l'état des variables.

1 Préparation du binaire

Avant d'utiliser le débogeur pour contrôler un programme, il est nécessaire de compiler ce programme avec l'option "-g". Cette option permet de conserver au sein du binaire une table, nommée "table des symboles", qui décrit les adresses d'implantation en mémoire des différentes variables du programme.

2 Commandes de base

Une fois démarré, un shell interactif permet d'analyser le fonctionnement du programme. Quelques commandes sont énumérées dans le tableau ci-dessous.

help	accès à la documentation des commandes
run	démarre l'exécution du programme
start	place un point d'arrêt sur main() et démarre l'exécution
break $< fonction >$	point d'arrêt au début d'une fonction
break $< fichier : numligne >$	point d'arrêt à une ligne d'un fichier
step	avance d'une instruction
next	avance d'une instruction ou exécute une fonction
continue	reprend l'exécution jusqu'au prochain point d'arrêt
finish	reprend l'exécution jusqu'à la fin de la fonction
print $< expr >$	affiche le résultat d'une expression
backtrace	pile des appels de fonctions
list $< fonction >$	affiche le code C d'une fonction
quit	sortie du débogueur

3 Exemple

Soit le programme suivant que l'on souhaite déboguer.

```
int main() {
        int *ptr;
        *ptr=17;
        return 0;
}
```

La trace qui suit montre une session simple de recherche d'erreurs dans le binaire *es*.

```
rubini@farfalle-linux:~$ gdb ./es
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2) 7.4-2012.04
Reading symbols from /home/rubini/es...done.
(gdb) run
Starting program: /home/rubini/es
Program received signal SIGSEGV, Segmentation fault.
0x080483bd in main () at es.c:3
3 *ptr=17;
(gdb) list
1 int main() {
2 int *ptr;
3 *ptr=17;
5 return 0;
6 }
(gdb) print ptr
$1 = (int *) 0x0
(gdb) print *ptr
Cannot access memory at address 0x0
(gdb) break main
Breakpoint 1 at 0x80483ba: file es.c, line 3.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/rubini/es
Breakpoint 1, main () at es.c:3
3 *ptr=17;
(gdb) step
Program received signal SIGSEGV, Segmentation fault.
0x080483bd in main () at es.c:3
3 *ptr=17;
(gdb) print ptr
$2 = (int *) 0x0
(gdb) step
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) quit
```