Déverminage

Luca Cirrottola (INRIA)

luca.cirrottola@inria.fr

Bordeaux INP ENSEIRB-MATMECA, Université de Bordeaux

Automne 2024

Introduction

1. « Bug »
2. gdb
3. valgrind

Déverminage

- Analyse d’un comportement incorrect
  1) Plantage du programme
  2) Résultat erroné
  3) Boucle infinie, blocage (inter-blocage …) (“deadlock”)
  4) Accès à des adresses mémoire erronés
  5) …

- gdb

- fonctionnement
  1) Identification rapide de la source du bug (ligne du code ..)
  2) Contrôle pas à pas
  3) Exploration des variables
  4) Ajout de points d’arrêt sur des conditions ou des lignes du programme

Outil interactif :

- segfaults
- résultats erronés
- boucles infinies...

- Comment ca marche ?
  1) Besoin d’informations additionnelles (symboles)
  2) Ajout de ces informations a la compilation en mettant l’option -g

```
> gcc -g  bug.c -o bug -lm
```

- Comment ca marche ?
  1) Gestion du prompt

[frue@mistral01 ~]$ gdb

(gdb)

- Comment ca marche ?
  1) Gestion du prompt

[frue@mistral01 ~]$ gdb

(gdb) run

- Comment ca marche ?
  1) Gestion du prompt

[frue@mistral01 ~]$ gdb

(gdb) run

Starting program:

No executable file specified.

Use the "file" or "exec-file" command.

(gdb) file bug

Reading symbols from /home/frue/bug...done.

- Lancement du programme
- Chargement des symboles

- Comment ca marche ?
  1) Gestion du prompt

[frue@mistral01 ~]$ gdb

(gdb) run

- Comment ca marche ?
  1) Gestion du prompt

- Erreur d’execution

[frue@mistral01 ~]$ gdb

(gdb) run

Starting program: /home/frue/bug

put some args

Program received signal SIGABRT, Aborted.

0x00007ffff77424d7 in kill () from /lib64/libc.so.6

Missing separate debuginfos, use: debuginfo-install glibc-2.17-260.el7.x86_64

- Quelques commandes
  1) Les points d’arrêt
     - breakpoint
     - break [filename]:[linenumber]
     - break main

- Quelques commandes
  1) Les points d’arrêt

```
[frue@mistral01 ~]$ gdb
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/frue/bug

Breakpoint 1, main (argc=1, argv=0x7fffffffde58) at bug.c:23
warning: Source file is more recent than executable.
23          if(argc < 3) usage();
```

- Quelques commandes
  1) Les points d’arrêt
     - breakpoint
     - break [filename]:[linenumber]
     - break main
  2) Le pas a pas
     - next – instruction suivante de la fonction courante
     - step – instruction suivante du programme

- Quelques commandes
  1) Les points d’arrêt
  2) Le pas a pas

[frue@mistral01 ~]$ gdb

(gdb) next

put some args

Program received signal SIGABRT, Aborted.

- Quelques commandes
  1) Les points d’arrêt
  2) Le pas a pas
  3) La pile

- Ou est on .. ?

[frue@mistral01 ~]$ gdb

(gdb) backtrace

#0 0x00007ffff77424d7 in kill () from /lib64/libc.so.6

#1 0x0000000004006ab in usage () at bug.c:11

#2 0x0000000000400705 in main (argc=1, argv=0x7fffffffde58) at bug.c:23

- Quelques commandes
  1) Les points d’arrêt
  2) Le pas a pas
  3) La pile

La fonction usage demande l’utilisation d’arguments

- Quelques commandes
  1) Les points d’arrêt
  2) Le pas a pas
  3) La pile

[frue@mistral01 ~]$ gdb

(gdb) run 1 3 7

Starting program: /home/frue/bug 1 3 7

- Quelques commandes
  1) Les points d’arrêt
  2) Le pas a pas
  3) La pile

[frue@mistral01 ~]$ gdb --args bug 1 3 7

(gdb) run

- Quelques commandes

```
Breakpoint 1, main (argc=4, argv=0x7fffffffde38) at bug.c:23
23          if(argc < 3) usage();
(gdb) next
26          a =(int*)malloc(argc*sizeof(int));
(gdb) next
29          for(i=0;i<=argc;i++) a[i] = 1/(atoi(argv[i])+1);
(gdb) next
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7746e67 in ___strtoll_l_internal () from /lib64/libc.so.6
```

- Quelques commandes

(gdb) backtrace

(gdb) frame 2

(gdb) list

(gdb) print i

(gdb) print argc

(gdb) print a@4

On corrige ...

- strtoll ?
- backtrace...
- Use frame

- Quelques commandes

On recompile

(gdb) file bug

(gdb) run 1 3 7

Le programme ne finit pas …

- Quelques commandes

On recompile

(gdb) file bug

(gdb) run 1 3 7

Le programme ne finit pas …

(gdb) run 1 3 7

(gdb) display i

- valgrind

- Comment ca marche ?
  1) Besoin d’informations additionnelles (symboles)
  2) Ajout de ces informations a la compilation en mettant l’option -g

> valgrind – –leak-check=full – –tool=memcheck <prog> <args>

- Verification des allocations..
- Et des liberations

- Comment ca marche ?
  1) Besoin d’informations additionnelles (symboles)
  2) Ajout de ces informations a la compilation en mettant l’option -g

> valgrind – –leak-check=yes – –tool=memcheck <prog> <args>

- Vérification des allocations..
- Et des libérations
- Vérification des droits en écriture

- Conclusion

  les debogueurs sont puissants et utiles pour découvrir les erreurs non triviales.

  - demandez les symboles de debug `-g`
  - pas d’optimisation (`-O0`) pour avoir la ligne précise de l’erreur

- Conclusion

  gdb : intéractif
  - blocages
  - calculs erronés
  - (simples) segfaults

  valgrind : fournit un rapport (détaillé)
  - segfaults
  - écriture/lecture hors limites des vecteurs
  - « dangling pointers »
  - « memory leaks »