GDB - petit tutoriel

Hai Nam Tran

November 18, 2019

This presentation is created based on the document "GDB - petit tutoriel" written by Daniel Hirschkoff (https ://perso.ens-lyon.fr/daniel.hirschkoff/).

Compiler un programme pour le debugging

Pour pouvoir lancer l'exécution d'un programme sous gdb, il faut "préparer" celui-ci, en utilisant l'option spéciale -g lors de la compilation.

Compiler

```
gcc -Wall -g prog.c -o prog
```

Lancer gdb

```
gdb prog
« message d'accueil... »
(gdb)
```

prog.c

```c
#include <stdio.h>

int max(int a, int b) {
  int c=a;
  if (a<b){ c=b; };
  return c;
}

int f(int x) {
  return 2*x+max(x+4,2*x);
}

int main(){
  printf("%d et %d\n",f(3),f(5));
  return 0;
}
```

Exécuter le programme sous gdb

On peut exécuter le programme à l'aide de la commande run (r) de gdb

Exécuter

```
(gdb) run
```

L'exécution se déroule comme si le programme tournait "normalement". Lorsque l'utilisateur de gdb à la main, il peut choisir de terminer la session en tapant quit (q)

Déterminer des points d'arrêt

On peut exploiter gdb pour examiner le programme à différentes étapes de son exécution

- La commande break (br) permet d'indiquer des points d'arrêt
- On peut fournir un numéro de ligne dans le code source, ou bien le nom d'une fonction

Point d'arrêt

```
(gdb) break f
Breakpoint 1 at 0x679: file prog.c, line 10
(gdb) run
Starting program: /home/hntran/UBO/PCA/gdb/gdb_tutorial/prog
Breakpoint 1, f (x=5) at prog.c:10
```

Examiner la situation

Lorsque l'exécution du programme est interrompue, on peut examiner l'état de la mémoire à ce moment là, par exemple en affichant la valeur d'une variable à l'aide de la commande print(p)

print

```
(gdb) print x
$1 = 5
```

On peut également utiliser la commande list(l) pour se remémorer l'endroit dans le code o'u l'exécution a été interrompue.

Faire évoluer le programme

- La commande step (s) permet d'avancer pas à pas dans l'exécution, afin de bien contrôler l'évolution du programme. Elle permet de passer à la ligne successive dans le source.

print

```
(gdb) step
max (a=9, b=10) at prog.c:4
4 int c=a;
(gdb) print c
$2 = 0
(gdb)
```

À voir aussi : next (n), continue (c), backtrace (bt), clear (c), watch (w)

GDB script

Il est préférable d'automatiser une ou plusieurs étapes si nous devons exécuter gdb plusieurs fois

- Indiquer des points d'arrêt
- Lancer le programme
- Examiner la situation
- ...

fichier : gdb_command

```
break f
break max
break 14
run
```

Lancer gdb avec gdb_command

```
gdb prog --command=gdb_command
```