Déboguer confortablement en ligne de commande

GDB, Valgrind, LLDB, PDB, IPDB, JDB

2ème session LIFTech'

Auteur: Florent Jaillet

Date: 9 juin 2016

Plan

- Motivation
- Intérêt du débogage en ligne de commande
- GDB *(C, C++, Objective-C, Fortran, Go, Rust...)*
- Valgrind + GDB
- LLDB *(C, C++, Objective-C)*
- PDB et IPDB *(Python)*
- JDB *(Java)*

Motivation

- Présentation à CppCon 2015 de Greg Law : "Give me 15 minutes & I'll change your view of GDB"
- Outils en ligne de commande difficiles à apprendre et peu intuitifs, mais plus pratiques que laisse paraître leur première utilisation
- Objectif ici n'est pas de présenter en détail le fonctionnement des outils, mais plutôt d'identifier les fonctionnalités clefs qui rendent l'utilisation plus facile et pratique
- Utiliser ce support de présentation comme mémo pour creuser les points intéressants par la suite

Intérêt du débogage en ligne de commande

- Ligne de commande disponible partout (par défaut sous GNU/Linux, BSD, MAC OS X, Windows Subsystem for Linux sous Windows 10, installable sous android)
- Parfois la seule solution (machine distante, grappe de calcul, système embarqué...)
- Puissance des outils
- Éventuelle préférence pour la ligne de commande

GDB : Exemple d'utilisation

- Compilation :

```
g++ -g example.cpp -o example
```

- Lancement du débogage :

```
gdb ./example
```

GDB : Utilisation basique

Aide	**h**elp
Lancement	start, **r**un
Points d'arrêt	**b**reak, **dis**able, **en**able, **cl**ear, **d**elete
Flot d'exécution	**s**tep, **n**ext, **fin**ish, **c**ontinue
Affichage informations	**p**rint, **b**ack**t**race (*ou* where), **l**ist, **i**nfo
Arrêt et fin	**k**ill, **q**uit

GDB : Plus de confort avec tui

- Interface avec fenêtres textuelles multiples via le raccourci Ctrl+X+A ou la commande `tui enable`
- Informations affichables en plus de la ligne de commande de GDB :
  - code source
  - code assembleur
  - registres
- Choix de la disposition des fenêtres avec les raccourcis Ctrl+X+2 et Ctrl+X+1 ou la commande `layout`
- Raccourcis :
  - Ctrl+L : rafraichissement interface
  - Ctrl+P, Ctrl+N, Ctrl+B, Ctrl+F : remplacement des touches flêches

GDB : Plus de puissance

Points d'arrêt	**wa**tch, **cat**ch, **b**reak <where> if <condition>, **cond**ition, **comm**ands, *sauvegarde* (save, **so**urce)
Affichage	**dp**rintf
Appel à la volée	**cal**l
Rétro-débogage	**rec**ord, **r**everse-**s**tep, **r**everse-**n**ext, reverse-finish, **r**everse-**c**ontinue
Script et extension	**py**thon, **p**ython-**i**nteractive
Historique	set history save *(à mettre dans $HOME/.GDBinit)*
Commandes système	**!** (*ou* shell)

Valgrind

Suite d'outils pour l'analyse dynamique (en particulier mémoire), notamment :

- détection de l'utilisation de mémoire non allouée ou non initialisée

```
Invalid read of size 1
    at 0x4005C8: main (example_valgrind.c:18)
 Address 0x5202100 is 0 bytes after a block of size 16 alloc'd
    at 0x4C2BBCF: malloc (vg_replace_malloc.c:299)
    by 0x4005BF: main (example_valgrind.c:15)
```

- détection de fuites de mémoire

```
LEAK SUMMARY:
   definitely lost: 35 bytes in 2 blocks
   indirectly lost: 0 bytes in 0 blocks
     possibly lost: 0 bytes in 0 blocks
   still reachable: 0 bytes in 0 blocks
        suppressed: 0 bytes in 0 blocks
```

- profilage de l'utilisation mémoire (`--tool=massif`)

Valgrind + GDB

Utilisation interactive de Valgrind possible via GDB :

- Lancement de Valgrind : `valgrind --vgdb-error=0 ./example`
- Lancement de GDB dans une autre invite de commande : `gdb ./example`
- Puis dans GDB : `(gdb) target remote | vgdb`

Aide	**mo**nitor **h**elp
Fuite mémoire	**mo**nitor **l**eak-check
Validité mémoire	**mo**nitor **g**et-vbits, **mo**nitor **c**heck_memory
Occupation mémoire	**mo**nitor **s**napshot *(nécessite l'option --tool=massif de Valgrind)*

Plus de détails dans la documentation de Valgrind et ce tutoriel

LLDB : Utilisation basique

	GDB	LLDB
Aide	**h**elp	**h**elp
Lancement	start, **r**un	**r**un **-s**, **r**un
Points d'arrêt	**b**reak, **dis**able, **en**able, **cl**ear, **d**elete	**b**reakpoint set, **br**eakpoint **di**sable, **br**eakpoint **e**nable, **br**eakpoint **cl**ear, **br**eakpoint **de**lete
Flot d'exécution	**s**tep, **n**ext, **fin**ish, **c**ontinue	**s**tep, **n**ext, **fin**ish, **c**ontinue
Affichage informations	**p**rint, **b**ack**t**race (*ou* where), **l**ist, **i**nfo	**p**rint, bt, **l**ist, (**fr**ame **v**ariable, **br**eakpoint **l**ist, **wa**tchpoint **l**ist,...)
Arrêt et fin	**k**ill, **q**uit	**k**ill, **q**uit

LLDB : Plus de puissance 1/2

	GDB	LLDB
Mutlifenêtre	tui	gui
Points d'arrêt	**wa**tch, **cat**ch, **b**reak <where> if <condition>, **cond**ition, **comm**ands, *sauvegarde* (save, **so**urce)	**wa**tchpoint **s**et **v**ariable, **br**eakpoint **s**et **-F**, **br**eakpoint **s**et **-c**, **br**eakpoint **m**odify **-c**, **br**eakpoint **co**mmand **a**dd, ø
Affichage	**dp**rintf	ø
Appel à la volée	**cal**l	**expr**ession
Rétro-débogage	**rec**ord,...	ø

LLDB : Plus de puissance 2/2

	GDB	LLDB
Script et extension	**py**thon, **p**ython-**i**nteractive	**sc**ript, **sc**ript
Historique	set history save *(à mettre dans $HOME/.GDBinit)*	*disponible par défaut*
Commandes système	**!** (*ou* shell)	**pla**tform **sh**ell

Page avec plus d'infos sur les correspondances GDB / LLDB

PDB et IPDB

	GDB	PDB et IPDB
Aide	**h**elp	**h**elp
Lancement	start, **r**un	ø, **r**un
Points d'arrêt	**b**reak, **dis**able, **en**able, **cl**ear, **d**elete, **cond**ition, **comm**ands	**b**reak, disable, enable, **cl**ear, **cl**ear, condition, commands
Flot	**s**tep, **n**ext, **fin**ish, **c**ontinue	**s**tep, **n**ext, **r**eturn, **c**ontinue
Affichage informations	**p**rint, **b**ack**t**race (*ou* where), **l**ist, **i**nfo	(p, pp, print), **w**here, **l**ist, (dir(), locals(), globals(), **b**reak,...)
Arrêt et fin	**k**ill, **q**uit	ø, **q**uit
Appel, script	**cal**l, **py**thon, **p**ython-**i**nteractive	[**!**]
Système	**!** (*ou* shell)	os.system("<cmd>")

Voir pdb++ ou pudb pour des fonctionnalités plus avancées

JDB

Fonctionnalités limitées, pas de fonctionnalités avancées :

	GDB	JDB
Aide	**h**elp	help
Lancement	start, **r**un	ø, run
Points d'arrêt	**b**reak, **dis**able, **en**able, **cl**ear, **d**elete, **wa**tch, **cat**ch	stop, ø, ø, clear, clear, watch, catch
Flot d'exécution	**s**tep, **n**ext, **fin**ish, **c**ontinue	step, next, step up, cont
Affichage informations	**p**rint, **b**ack**t**race (*ou* where), **l**ist, **i**nfo	(print, dump), where, list, (locals, clear,...)
Appel à la volée	**cal**l	eval (*ou* print)
Arrêt et fin	**k**ill, **q**uit	kill, quit