Outils de développement et compilation

GDB, Valgrind, ASan, perf...

Timothée Ravier, LIFO, INSA-CVL, LIPN

1re année cycle ingénieur STI, 2013 – 2014

Objectifs

— Découvrir GDB, Valgrind (memcheck, callgrind), ASan, perf ;
— Tester des EDIs (vim, emacs, KDevelop, Eclipse, Netbeans...).

Pour ce TD, nous allons utiliser les code sources d'exemple disponibles à l'adresse : https://tim.siosm.fr/downloads/cours/tools-td-example.tar.gz.

```
$ curl -o td-tools.tar.gz https://tim.siosm.fr/downloads/cours/tools-td-example.tar.gz
$ tar xf td-tools.tar.gz
$ cd devtools/tools/td/
```

1 GDB

Lisez, compilez et exécutez le programme `test-gdb` :

```
$ make test-gdb
$ ./test-gdb
```

Nous constatons que le programme se termine de façon abrupte. Nous allons donc chercher à comprendre pourquoi en utilisant GDB :

```
$ gdb test-gdb
$ (gdb) run
```

Lorsque le programme atteint une instrcution levant une erreur, GDB stoppe l'exécution et nous redonne le contrôle. Nous allons alors afficher la suite de fonction en cours d'exécution pour déterminer pour quelle raison notre programme a planté :

```
$ (gdb) backtrace
$ (gdb) print a
```

- **Question 1 :** Quelle est l'erreur présente dans ce programme ?

Essayer les commandes suivantes dans GDB :

```
$ (gdb) info locals
$ (gdb) print <variable>
$ (gdb) list
```

2 Valgrind

2.1 Memcheck

Lisez, compilez et exécutez le programme `test-memcheck` :

```
$ make test-memcheck
$ ./test-memcheck
```

Le programme semble s'exécuter normalement. Utilisez maintenant Valgrind avec l'outil memcheck pour essayer de trouver les erreurs dans ce programme :

```
$ valgrind --tool=memcheck ./test-memcheck
```

- **Question 2 :** Quelles sont les erreurs présentes dans ce programme ?

2.2 Callgrind

Utilisez maintenant Valgrind avec l'outil callgrind pour obtenir des informations sur les fonctions appelées dans un programme :

```
$ valgrind --tool=callgrind <n'importe quel programme>
```

Utilisez l'interface KCachegrind pour visualiser les données obtenues.

3 ASan

Lisez le code du programme `test-asan`, compilez et exécutez le programme `test-asan` :

```
$ make test-asan
$ ./test-asan
```

- **Question 3 :** Que se passe-t-il ?

Compilons la version avec la fonction Address Sanitizer d'activée :

```
$ make test-asan-on
$ ./test-asan-on
```

- **Question 4 :** Quelle est l'erreur dans ce programme ?

4 perf

Lisez, compilez et exécutez le programme `test-perf` :

```
$ make test-perf
$ ./test-perf
```

Essayer les commandes perf principales sur ce programme (top, record, report) la consommation en mémoire de ce programme avec `htop`.

5 Références

— Address Sanitizer : http://code.google.com/p/address-sanitizer/wiki/AddressSanitizer
— Beej's Quick Guide to GDB : http://beej.us/guide/bggdb/
— Valgrind tutorial : http://valgrind.org/docs/manual/quick-start.html