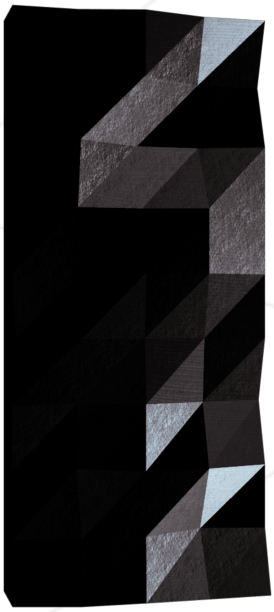

RUSH

Low Level Debugger

42 Staff pedago@staff.42.fr

Résumé: Simple rush pour découvrir des méthodes pour débug vos programmes.

Table des matières

Chapitre I

Préambule

- Temps de préparation : 15 minutes
- Temps de cuisson : 10 minutes
- Ingrédients (pour 4 personnes) :
  - 85 g beurre
  - 1 oeuf
  - 85 g sucre
  - Essence de vanille ou 1 sachet de sucre vanillé
  - 150 g de farine
  - 100 g de chocolat noir
  - 1 cuillère à café de sel
  - 1 cuillère à café de levure chimique

Préparation de la recette :

- Préchauffer le four à 180°C (thermostat 6).
- Faire ramollir le beurre à température ambiante. Dans un saladier, mettre 75 g de beurre, incorporer le sucre, l'oeuf entier, la vanille et mélanger le tout.
- Ajouter petit à petit la farine mélangée à la levure, le sel et le chocolat coupé en petits morceaux.
- Beurrer une plaque allant au four et former les cookies sur la plaque.
- Pour former les cookies, utiliser 2 cuillères à soupe et faire des petits tas espacés les uns des autres ; ils grandiront à la cuisson.

Chapitre II

Introduction

Un débogueur (ou débugueur, de l'anglais debuger) est un logiciel qui aide un développeur à analyser les bugs d'un programme. Pour cela, il permet d'exécuter le programme pas-à-pas, d'afficher la valeur des variables à tout moment, de mettre en place des points d'arrêt sur des conditions ou sur des lignes du programme ...

Il s'agit de l'application à la programmation informatique du processus de troubleshooting.

Chapitre III

Objectifs

Ce rush a pour simple but de vous faire découvrir des méthodes utilisables par un débogueur dans le but est de détecter les bugs sur vos programmes en général.

Une fois ce rush terminé l'utilisation de lldb spécifiquement n'aura plus aucun secret pour vous.

Chapitre IV

Partie obligatoire

Pour ce projet vous allez devoir créer trois dossiers différents contenant dans chacun les fichiers demandés pour chaque exercice.

IV.1 Partie I - Configuration

Dans un premier temps vous allez devoir rendre un fichier de config du nom de .lldbinit. Ce fichier de config va devoir faire en sorte d'avoir une configuration précise au lancement de lldb. Ce fichier sera alors à mettre dans un dossier du nom de "00-init" présent à la racine de votre dépôt. La configuration demandée :

- Syntax intel.
- Changement du prompt par le login du chef du groupe.
- Lancement automatique d'un script afficant un banner.

Voici un exemple de session avec le script de configuration valide :

```
# lldb exemple
(wandre) target create "example"
Current executable set to 'example' (x86_64).


                                      iiii  lllllll
                                     i::::i l:::::l
                                      iiii  l:::::l
                                            l:::::l
wwwwwww           wwwww           wwwwwwwiiiiiii  l::::l
 w:::::w         w:::::w         w:::::w i:::::i  l::::l
  w:::::w       w:::::::w       w:::::w   i::::i  l::::l
   w:::::w     w:::::::::w     w:::::w    i::::i  l::::l
    w:::::w   w:::::w:::::w   w:::::w     i::::i  l::::l
     w:::::w w:::::w w:::::w w:::::w      i::::i  l::::l
      w:::::w:::::w   w:::::w:::::w       i::::i  l::::l
       w:::::::::w     w:::::::::w        i::::i  l::::l
        w:::::::w       w:::::::w       i::::::il::::::l
         w:::::w         w:::::w        i::::::il::::::l
          w:::w           w:::w         i::::::il::::::l
           www             www          iiiiiiiillllllll


(wandre) disassemble --name main
example[0x4009c0] <main>: push   rbp
example[0x4009c1] <main+1>: mov    rbp, rsp
example[0x4009c4] <main+4>: sub    rsp, 0x60
example[0x4009c8] <main+8>: movabs rdi, 0x6013c0
example[0x4009d2] <main+18>: movabs rsi, 0x400cd4
example[0x4009dc] <main+28>: mov    dword ptr [rbp - 0x4], 0x0
example[0x4009e3] <main+35>: call   0x400810                 ;  + 112
example[0x4009e8] <main+40>: movabs rsi, 0x400840
example[0x4009f2] <main+50>: mov    rdi, rax
example[0x4009f5] <main+53>: call   0x400830                 ;  + 144
example[0x4009fa] <main+58>: mov    dword ptr [rbp - 0x14], 0x0
example[0x400a01] <main+65>: mov    qword ptr [rbp - 0x20], rax
[...]
(wandre) _
```

IV.2 Partie II - Utilisation

Vous allez devoir commencer à utiliser lldb vraiment ! Pour se faire vous allez devoir compiler par la simple commande ‘clang++ -Wall -g source.cpp -o example‘ ce petit programme codé en c++ :

```
#include <iostream>
#define MAX 3

double      average(int min[], int max);
int         max(int min[], int max);

int         main(void)
{
    int     tab[MAX];
    int     count;

    std::cout << "3 numbers: " << std::endl;
    for ( count = 0; count < MAX ; count++ )
    {
        std::cout << "Next number : ";
        std::cin >> tab[1];
    }
    for ( count = 1; count < MAX ; count++ )
        std::cout << "Value [" << count << "] = " << tab[count] << std::endl;
    std::cout << "Average = " << average(tab, MAX) << std::endl;
    std::cout << "MAX = " << max(tab, MAX) << std::endl;
    return 0;
}

double      average(int min[], int max)
{
    double  tmp;

    tmp = 0.0;
    for (int i = 0; i > max; i++)
        tmp += min[0];
    tmp /= max;
    return tmp;
}

int         max(int min[], int max)
{
    int     biggest;

    biggest = 0;
    for (int i = 1; i < max; i++)
        if (biggest <= min[i])
            biggest = min[0];
    return biggest;
}
```

Une fois la compilation finis vous allez devoir tenter de faire fonctionner ce programme sans qu’il ne bug. Pour cela vous devez utiliser lldb et écrire la suite de commande suivante :

```
# lldb example
(wandre) breakpoint set --name main
Breakpoint 1: where = example`main + 35 at example.cpp:12, address = 0x00000000004009e3
(wandre) proces launch
Process 14127 launching
Process 14127 launched: '/PATH/example' (x86_64)
(wandre) Process 14127 stopped
* thread #1: tid = 14127, 0x0000000 [...]
(wandre) _
```

Une fois arrivé à ce point vous devez maintenant simplement faire en sorte que le programme fonctionne correctement avec le moins d'action possible. Vous allez devoir lister alors la liste de commandes (pas de raccourci) tapées dans un fichier nommé "commandes" qui sera alors présenté dans votre second dossier créer "01-usage" à la racine de votre dépôt de rendu.

Le nombre de commandes va influer sur votre notation finale..

Voici un exemple de fichier de rendu valide :

```
# cat -e commandes
thread backtrace all$
frame select 12$
frame info$
frame select -relative=4$
register read$
register write rax 521$
register read --all$
memory read --size 4 --format x --count 4 0xdeadbeef$
memory read `argv[4]`$
thread step-in$
thread step-out$
# _
```

IV.3 Partie III - Script

Vous savez maintenant utiliser lldb de manière correcte on va pouvoir personaliseer ce programme en ajoutant nos propres scripts ! Vous devez donc maintenant créer un petit script (en python) dont le but sera d'afficher simplement le nom du programme attaché à lldb mais en reverse ! On va enfin ajouter un petit FT_ au début de ce nom. Le fichier script sera nommé "reverse.py" et sera à l'intérieur du dernier dossier situé à la racine de votre dépôt de rendu nommé "02-script". Voici un exemple d'utilisation possible :

```
# lldb example
(wandre) target create "example"
(wandre) command script import reverse.py
(wandre) reverse
FT_elpmaxe
(wandre) _
```

Chapitre V

Partie bonus

En bonus, il n'y en aura que trois disponibles.

- Le premier sera de rajouter de la couleur durant l'affichage du banner contenant le login du chef du groupe de votre rush.
- Le second sera d'automatiser le lancement des commandes dans un fichier de configuration lisible par lldb.
- Le dernier sera de créer d'autres fonctions "UTILES" dans lldb.

Chapitre VI

Rendu et peer-évaluation

- Ce projet ne sera corrigé que par des humains.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas vos scripts ne doivent quitter de façon inattendue (crash, etc)
- Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.
- Les trois dossiers présents auront pour nom :
  - 00-init
  - 01-usage
  - 02-script
- Votre rendu sera alors de cette forme :

```
# ls -alR
.:
total 20
drwxr-xr-x 5 root root 4096 Mar  9 12:08 .
drwxr-xr-x 4 root root 4096 Mar  9 12:06 ..
drwxr-xr-x 2 root root 4096 Mar  9 12:08 00-init
drwxr-xr-x 2 root root 4096 Mar  9 12:08 01-usage
drwxr-xr-x 2 root root 4096 Mar  9 12:09 02-script

./00-init:
total 12
drwxr-xr-x 2 root root 4096 Mar  9 12:08 .
drwxr-xr-x 5 root root 4096 Mar  9 12:08 ..
-rw-r--r-- 1 root root  231 Mar  9 12:08 .lldbinit

./01-usage:
total 12
drwxr-xr-x 2 root root 4096 Mar  9 12:08 .
drwxr-xr-x 5 root root 4096 Mar  9 12:08 ..
-rw-r--r-- 1 root root  236 Mar  9 12:08 commandes

./02-script:
total 12
drwxr-xr-x 2 root root 4096 Mar  9 12:09 .
drwxr-xr-x 5 root root 4096 Mar  9 12:08 ..
-rw-r--r-- 1 root root  300 Mar  9 12:09 reverse.py
```