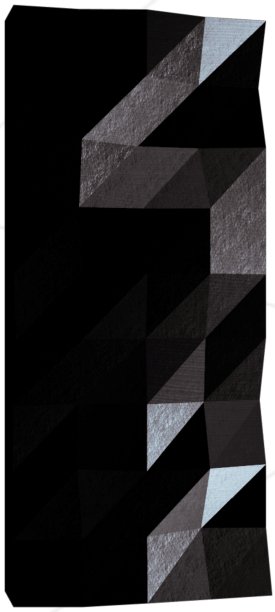

RUSH

Low Level Debugger

42 Staff pedago@staff.42.fr

Summary: Simple rush to discover debugging techniques that solve all your problems.

Contents

Chapter I

Foreword

- Preparation time: 15 minutes
- Cooking time : 10 minutes
- Ingredients (for 4 persons) :
  - 85 g butter
  - 1 egg
  - 85 g sugar
  - Vanilla essence or 1 packer of vanilla sugar
  - 150 g of flour
  - 100 g of dark chocolate
  - 1 teaspoon of salt
  - 1 teaspoon of baking powder

Preparation of the recipe :

- Preheat the oven to 180°C (thermostat 6).
- Soften the butter at room temperature. In a salad bowl, put 75 g of butter, add the sugar, the whole egg, the vanilla and mix everything.
- Gradually add the flour mixed with the yeast, salt and chocolate cut into small pieces.

- Butter a baking sheet and form cookies on the plate.

- To form cookies, use 2 tablespoons and make small piles spaced from each other; they will grow up while cooking.

Chapter II

Introduction

A debugger is a software that helps a developer to analyze the program for inconsistencies or bugs. For this, it allows you to run the program step-by-step, to display the state of the variables at any time, or to put breakpoints on conditions or any place of the program ...

This is the application of the computer programming to the process of troubleshooting.

Chapter III

Objectives

This rush has a simple purpose of introducing you to the wide variety of useful debugging methods for the less-painful bug detection in your programs.

Upon the completion of this rush, the LLDB debugger will no longer be a mystery to you, hopefully.

Chapter IV

Mandatory part

For this project you will have to create three different directories, each containing the requested files for each exercise.

IV.1 Part I - Configuration

At first you will have to make a config file named .lldbinit. This config file will specify preferences at the launch of lldb . Put this file in the folder named "00-init" on your repo. The requested configuration is :

- Intel style disassembly syntax.
- Change the prompt to the team leader login.
- Automatic launch of a script displaying a banner of your team.

Here is an example of a session with the valid configuration script :

```
# lldb example
(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 Part II - Usage

You will have to use lldb for real! To do so you will have to compile the C++ code snippet below like this ‘clang++ -Wall -g source.cpp -o example‘:

```
#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;
}
```

Once the compilation is finished you have to make this program work without bugs. For this you have to use lldb and write the following commands:

```
# 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) _
```

Once you get to this point you should make sure that the program works properly. You have to fix the program in the least amount of instructions possible. You have to list all the commands you used in a file named "commands" in your "01-usage" directory present on a repo.

The number of instructions will affect your final grade..

Here is an example of a valid file:

```
# 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 Part III - Script

Now that you know how to use lldb properly we will be able to personalize this program by adding our own scripts ! You have to create a small Python script whose purpose will be to display the name of the program attached to lldb but in reverse! You will finally add a small FT_ at the beginning of programs name. The script will be named "reverse.py" and must be located in the folder named "02-script". Here is an example of possible use:

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

Chapter V

Bonus part

As a bonus, only this three items available.

- The first will be to add color to your banner containing the login of a team leader of your rush.
- The second one will automate the commands launching in a LLDB configuration file.
- The last one will be to create other "USEFUL" functions in lldb.

Chapter VI

Turn-in and peer-evaluation

- This project will be corrected only by peers.
- You must handle errors in a reasonable way. In no case your program will exit unexpectedly (crash, etc)
- Send your work on your `GiT` repo as usual. Only the work present on your repo will be evaluated during defense.
- Three directories (+ bonus) present on your repo must be named like this:
  - 00-init
  - 01-usage
  - 02-script
- Your repo will look like this :

```
# 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
```