# Bienvenue!

Conférence BRIEF-42
#00

# GDB, valgrind, memory and co.

*Salut! (Oui, vous êtes bien venu jusqu'à votre chaise, bravo! +0.1XP pour chacun d'entre vous.)*

Première conférence réalisée par et pour les étudiants de 42 Angoulême.

1 - Introduction à l'utilisation de GDB, avec *Marc*.

2 - Utilisation plus avancée de GDB, avec *Maël*.

3 - Démonstration avec GDB, avec *Sam*.

4 - Un peu de théorie sur la mémoire, avec *Thomas*.

5 - Introduction à l'utilisation de Valgrind, avec *Maxime*.

6 - Astuces d'utilisation de Bash, avec *Bruno*.

Si tout se passe bien* nous ferons plusieurs conférences tout au long de l'année. Pour nous organiser ensemble autour de ces projets pédagogiques nous créerons un club dédié au sein du BDE. ***Présentation en fin de conférence.***

** Ce qui inclut l'absence de décès de l'un ou plusieurs des intervenants, ni de malaise vagal dans le public, ni de faillite liée au budget sucreries de la conférence. (Quel sucreries dites-vous ? Euh y'en a plus, ne lisez pas les petits caractères!)*

**Durée estimée**: 1h30 *environ**

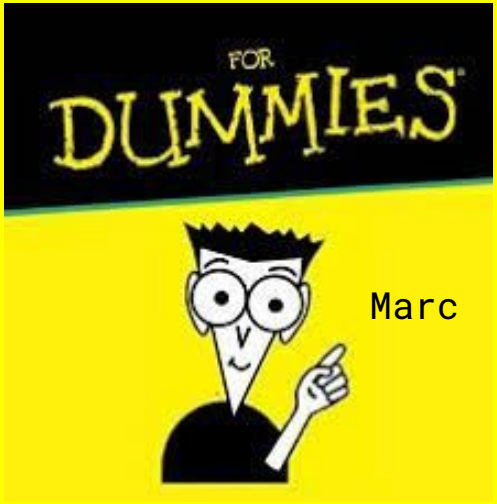
- 1 heure interventions
- 30 minutes FAQ sous forme de discussions et d'interventions libres.

# 00:/0:00

**C'est notre première, c'est une expérience d'apprentissage dans les deux sens, please be nice. Gardez les questions et discussions pour la FAQ, SVP.**

** Estimation réalisée par nos soins avec l'aide d'une horloge atomique afin de garantir une fin de conférence dans 1h30 et une marge d'erreur de très exactement 10 millièmes de secondes. (Ticket remboursé en cas de retard pour un rdv dentiste.)*

# 1 - Introduction à GDB



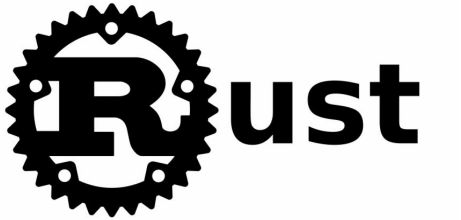
# G.D.B : Mais qu’est ce donc ????

- GNU Project Debugger
- G.D.B est un debugger
- Un debugger sert à regarder ce qu’il se passe dans le programme quand il s'exécute.
- GDB créé en 1986 par Richard Stallman est un logiciel libre sous GNU General Public License

*Hésitez pas à vous renseigner davantage sur le projet GNU et les débuts du logiciel libre. Saint Ignucius sera content.*

# G.D.B supporte :

- Ada
- Assembly
- C
- C++
- D
- Fortran
- Go
- Objective-C
- OpenCL
- Modula-2
- Pascal
- Rust



# G.D.B VS Printf

- Utiliser un debugger fait partie des bonnes pratiques professionnelles
- Gain de temps quand on sait s'en servir
- Évite de laisser des Printf partout dans ses programmes
- Possibilité de voir l'évolution d'une variable comme par exemple dans une while
- Permet de surveiller la mémoire (stack, heap, adressage)

# Déroulement:

- Compiler avec l’argument -g — gcc -g monProgBugger.c
- Lancer G.D.B avec l'exécutable — gdb a.out
- Créer un Break point
- Lancer le programme
- avancer pas à pas dans le programme
- observer le comportement des variables

# Les indispensables :

| | |
|---|---|
| Lancer GDB avec des arguments | gdb --args a.out “POLO” “&” “PAN” |
| Break point | break avec numéros de ligne ou fct |
| Enlever les Break point | delete |
| Afficher ou en est le code | list |
| Lancer le prg | run |
| Ligne suivante | next |
| répéter la commande | touche enter |
| rentrer dans une fct | step |
| Afficher une variable | print nomDelaVariable |
| Changer la valeur d’une variable | set var nomDelaVariable = X |

# T.U.I pour y voir clair

- Text User Interface
- Pour l’activer 2 méthodes
  - Soit dans le terminal — gdb -- tui a.out
  - Soit dans GDB — tui enable

# T.U.I cheat sheet

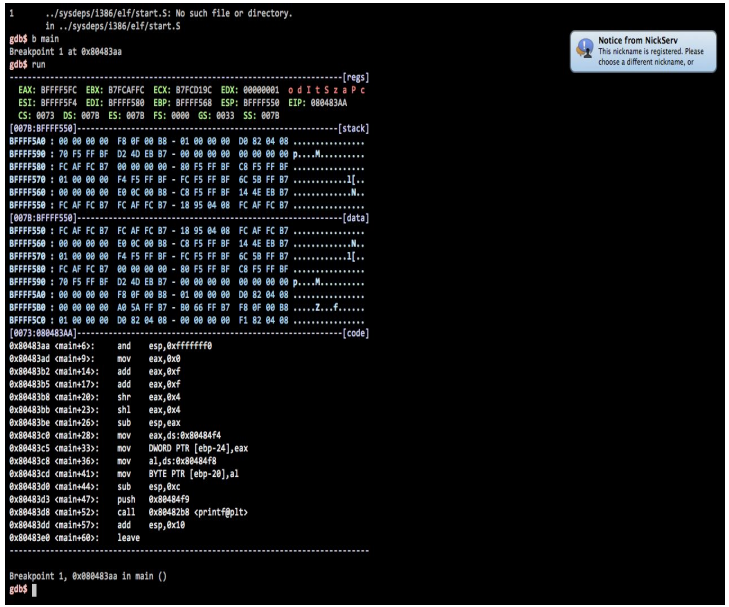
| | |
|---|---|
| Changer de fenêtre pour scroller | focus next / preview |
| Afficher les commandes ASM / registre | layout asm / reg |
| Afficher le code src et registre | layout split |
| Revenir au code source | layout src |

# Pimp my G.D.B

Possibilité de rajouter des extensions pour GDB

en modifiant ~/.gdbinit

ou avec des surcouches python




https://stackoverflow.com/questions/209534/how-to-highlight-and-color-gdb-output-during-interactive-debugging/17341335#17341335

# 2 - Utilisation de GDB

Maël


hackerman tips : bash -c "$(curl -fsSL http://gef.blah.cat/sh)" - to install GDB-GEF and became *pimped hackerman*

# Summary of regularly shorted used gdb command

- `breakpoint -> b`
- `watchpoint -> watch`
- `run -> r`
- `continue -> c`
- `next -> n`
- `step -> s`
- `list -> l`
- `print -> p`
- `backtrace -> bt`
- `info -> i`

# ‘print’ and ‘x’ like a rockstar

- print - *like printf u know*

  use it as : print optType1 targetName optType2

  - optType1 is an ‘in-case’ type to format target
  - targetName can be a pointer, an address or a variable name
  - optType2 another ‘in-case’ type formatter

  **example** : p *toto@100 - *print 100 first formatted element of toto*

- x - examine at a certain memory address *like printf++*

  use it as : x/nbrToShow optType1 targetAdress

  - optType1 can be : d (dec-10), a (ptr), c (char), s (char *), x(hex), o (oct-8)
  - nbrToShow is the number of optType1 that you want to show from targetAdress

  **example** : x/16c &toto - *show 16 formatted char from toto address*

# ‘set’ like a rockstar

- set - *like ‘=’ u know*

  use it as : set optTarget1 targetName = myCoolValue

  - optTarget1 can be lot  of things, first thing first : var
  - targetName is the name of the object
  - myCoolValue is what you want to write in target

  **example** : set var toto = “tutu”

# ‘break’ and ‘watch’ to stop the damn process

- break - *insert breakpoint into an instruction*

  use it as : b * target

  ➢ target can be an address, a line or a call to a function,

  **example** : b * 15 (break line 15), b * 0xdeadbeef (break at 0xdeadbeef address), b * main (break when main is called to the stack)

- watch - *insert a watchpoint into an instruction*

  this command work like breakpoints in any point except that you need to use watch command and this will break if targeted value has changed.

# BONUS tips

- whatis - *what's the type of ?*

  useful command to see a type of variable

- set $pc = 0xdeadbeef - *jump EVERYWHERE*

  super useful command to jump to 0xdeadbeef address

- *logical condition*

  you can choose to add logical conditions in your gdb statement like
  **b * 0xdeadbeef if (i == 2)**

- define

  define is a command that allow you to create macro-like in gdb

# 3 - Démo GDB

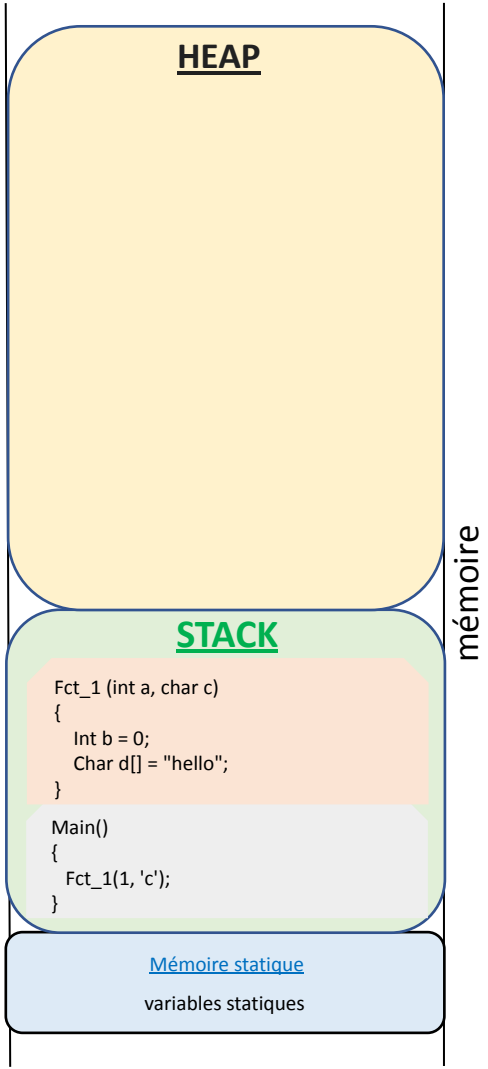
Sam

# 4 - Un peu de théorie sur la mémoire

Thomas

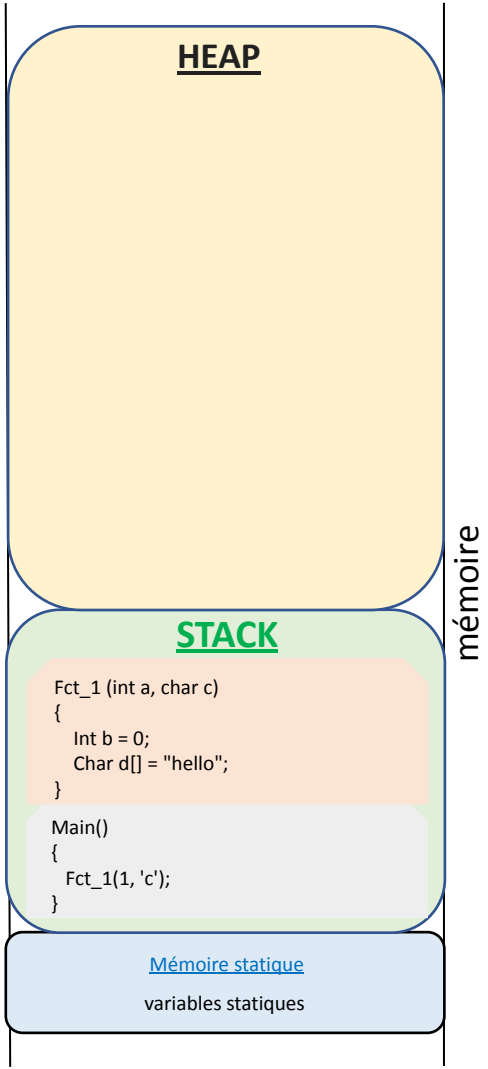
```
Fct_1(int a, char c)
{
   Int b = 0;
   Char d[] =
"hello";
}
```

```
Main()
{
    Fct_1(1, 'c');
}
```

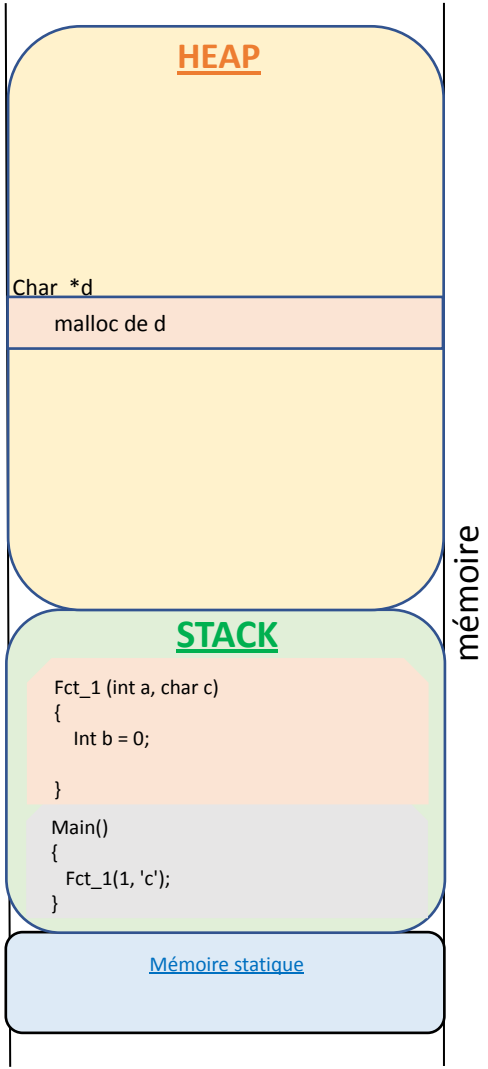


```
Fct_1(int a, char c)
{
   Int b = 0;
   Char d[] =
"hello";
}
```

```
Main()
{
    Fct_1(1, 'c');
}
```



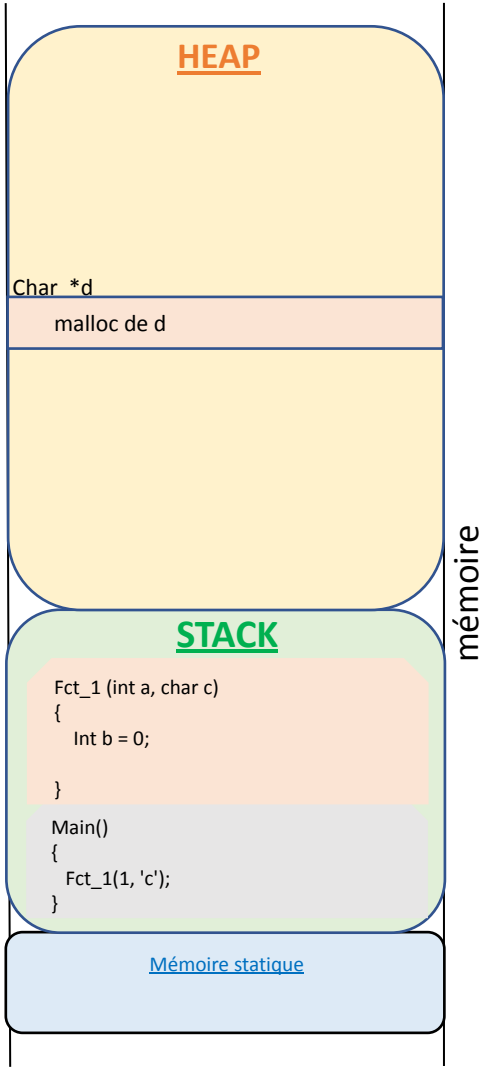
# BRIEF-42 : Conf#00




```
Fct_1(int a, char c)
{
   Int b = 0;
   Char  *d;
   d = malloc....;
}
```

```
Main()
{
    Fct_1(1, 'c');
}
```

# BRIEF-42 : Conf#00




```
Fct_1(int a, char c)
{
   Int b = 0;
   Char  *d;
   d = malloc....;
}
```

```
Main()
{
    Fct_1(1, 'c');
}
```

# 5 - Introduction à Valgrind

Maxime

# SOMMAIRE

# 6 - Time for some Bash

Bruno

# Command interpreter history

a. Shell
b. Bash
c. Zsh

# How to “chain” command

a. Flux and piping
b. ‘&&’ / AND operator
c. ‘||’ / OR operator

# Brace Expansion + BONUS

a. basic of list exp
b. Search and replace
c. dir stack :)

# Bonus Track: Présentation TED-42

*Bon et bien au revoir, merci d'être venu! (-666XP si vous vous levez maintenant traîtres!)*

**TED-42** copyright (ou **BRIEF** copyleft)

**TE**chnology + **ED**ucation

*(malheureusement ça marche qu'avec TED sinon **B**ière **R**istournes **I**dentité **E**mbrouilles et **F**…)*

Projet de club du BDE pour accompagner à la création et à la diffusion de conférences et de rencontres pour les étudiants de **42** Angoulême.

*Oula c'est serieux là, aucune images, documents noir et blanc. (et rien à manger en attendant, désolé la prod est parti avec la cais…)*

**Mais avant tout, une petite mise au point entre nous.**

**Chacun avance à son rythme.**

Si on perd un wagon, on perd le principe du pair à pair.

Et on entre sur un autre principe, celui du chacun pour soi.

Ce n'est pas un jugement, c'est nous qui décidons ce que nous faisons de cette école.
Alors la question qu'on se pose ensemble c'est, on fait quoi ?

*Bon ok deux images en couleur, pas plus, on va nous tomber dessus pour les droits intellectuels. :(*

# Objectifs :

1. **Apprendre à travailler ensemble** autour de la création et le partage d'outils et de contenu pédagogiques qui s'inscrivent sur le long du cursus 42 Angoulême: tronc commun, spécialités et autres selon les besoins de chacun.

2. Accompagner dans une forme ouverte tout les étudiants d'aujourd'hui et de demain sans discrimination de niveau technique, de promo ou même de coalition pour permettre un flux de communication à travers toute l'école et afin de **diffuser et structurer un savoir commun**.

3. **Tisser des liens** entre étudiants mais aussi avec des professionnels extérieurs pour aller plus loin dans l'exploration de ses connaissances et pratiques afin d'aider à l'élaboration de son projet personnel.

**Rejoignez nous pour:**

- Vous entraîner à consolider davantage vos connaissances pour vous même tout en les partageant avec nous autres, et tout en acquérant des “soft skills” de communication.

- Travailler avec le club pour apprendre avec nous comment construire un réseau qui connectera des ponts entre les étudiants et des professionnels du milieu tech.

- Participer à un projet pédagogique pair à pair innovant et ambitieux afin de suivre son évolution au fur et à mesure de ses itérations.

- Ou simplement venir discuter et partager vos critiques constructives, vos besoins ou vos idées pour de futurs conférences et projets.

Contactez nous sur le discord du futur club: lien ici?

*Prochaine conférence:*
dans 3 semaines “environ”

BRIEF-42: Conf#01
**Le thème sera décidé ensemble!**

**Venez en discuter, la préparer et la construire avec nous.**

*Non, il n’est pas sur l’écran, il est sur le Discord de la promo dans 3, 2, 1...*

# Au revoir!

# FAQ

Pour ceux qui le souhaitent.