# TD1

### Exercice 1

Regardez le contenu de argc et argv. Est-ce cohérent avec le contenu de launch.json ? Quel est le lien entre le caractère / et 47 ?

Oui c'est cohérent : **prog* contient le chemin du fichier exécuté, **arg* contient 456

/ est le caractère 47 de la table ASCCII

### Exercice 2

Avancez pas à pas jusqu'à la ligne 11. Observez les valeurs de prog et arg. Notez qu'une info bulle affiche leur contenu quand vous passez sur la variable dans l'éditeur.

J'ai vu

### Exercice 3

On remarque que *w = 2 ainsi que z, ce qui est normal puisque *w=z. On aimerait voir le contenu de &w. On peut pour cela ajouter un espion. Cliquer + dans la zone ESPION et tapez &w. Dépliez la valeur de &w. Comprenez-vous les relations entre : &w, *&w et **&w

&w = adresse mémoire de w
*&w = valeur de w
**&w = valeur pointée par l'adresse de w ➔ donc valeur de z

### Exercice 4

Avancez jusqu'au dernier point d'arrêt et regardez la sortie dans le terminal.

J'ai vu « bonjour »

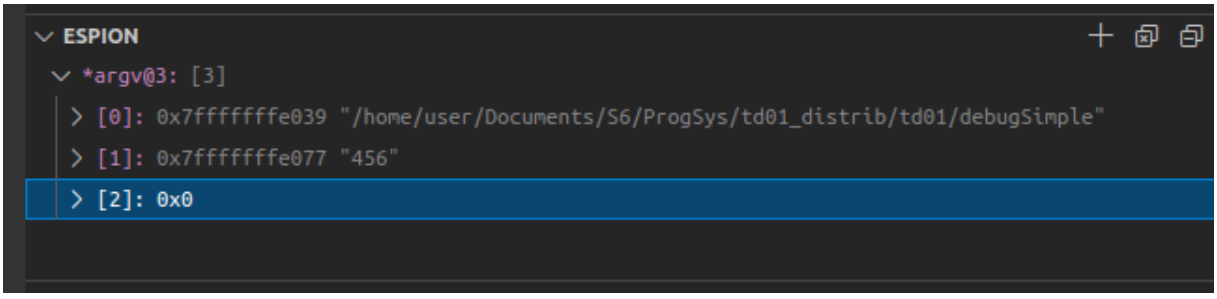
### Exercice 5

Enlevez tous les points d'arrêt. Placez un point d'arrêt conditionnel (clic droit) sur la ligne 10 à condition que z == 2. Lancez le debug, vous devez constater que vous vous arrêtez bien

Oui

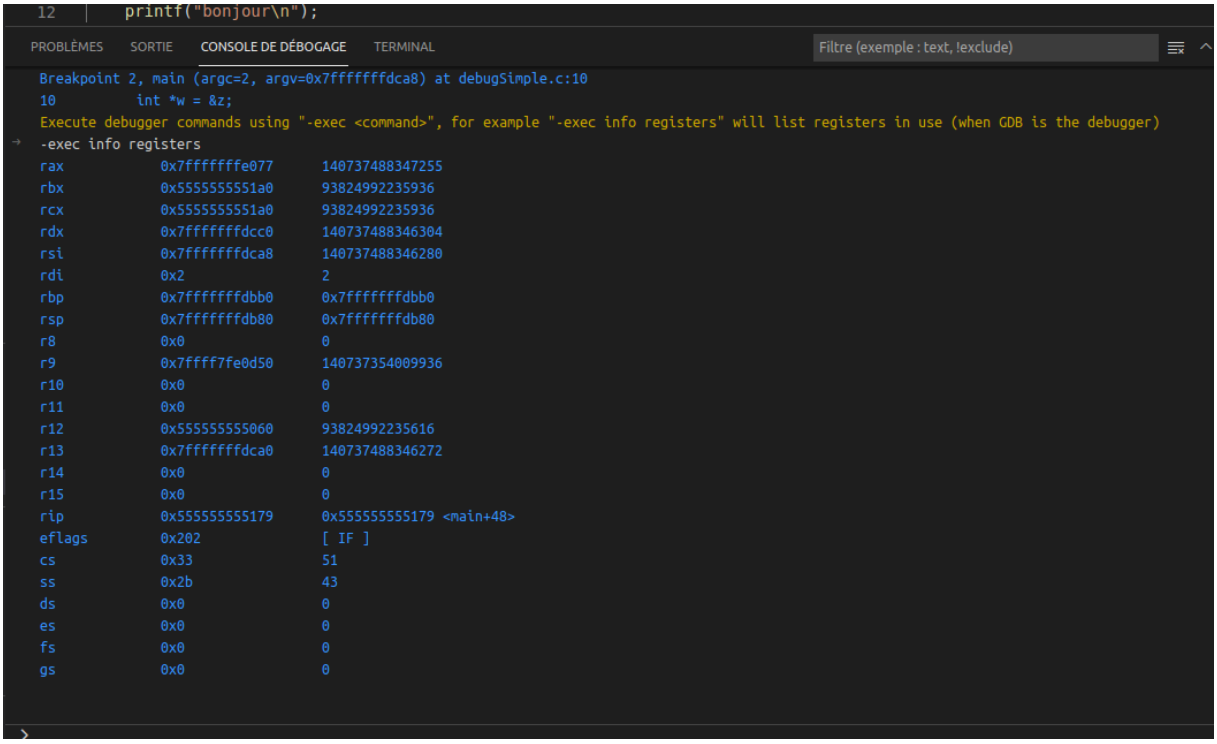
### Exercice 6

Placez un espion sur *argv@3.



## Exercice 7

En ligne de commande du debugger, taper « -exec info registers », cela affiche le contenu des registres de la machine




## Exercice 8

1. Placez un point d'arrêt sur chacune des lignes du programme.

Oui

2. Placez les espions

Oui

3. Tracez pas à pas jusqu'à avoir exécuté l'instruction char *ci=tab;. Que déduisez-vous de l'organisation mémoire ? Comment sont placés les octets à l'intérieur d'un int ?

Les int sont stockés en l'ordre inverse de ses bits

4. Y a-t-il une différence entre *ci@16 et *si@8 ?

Oui les adresses pointées ne sont pas les memes

Dans si on affiche les données en int contrairement à ci qui affiche les données en char

5. Tracez pas à pas jusqu'à la fin du programme et observez bien ii, si et ci. Pour quelle raison ii++ et si++ n'ont pas le même effet sur la valeur de ii et si ? De même, pourquoi ii++ et ci++ n'ont pas le même effet sur la valeur de ii et ci ?

Les incrémentations sont différentes pour les différents types de données

## Exercice 9

Tentez de découvrir quel est le bug de ce programme.

Le problème est que l'on va trop loin dans la boucle for, on override la valeur de la constante sig

### Exercice 10

Quel algorithme de tri implémente la fonction mysort ?

Tri Shell

### Exercice 11

Placer des points d'arrêts et des observateurs d'expression.

Oui

### Exercice 12

Corrigez maintenant le problème et recompilez avec ctrl-shift-b

Oui

### Exercice 13

Dans les programmes fournis dans l'archive, vous avec le programme list.c. Tentez de le déboguer le plus rapidement possible.

Vérification que *tail* ne soit pas null

### Exercice 14

Exécutez votre programme de tri corrigé avec ltrace. Quelles est (sont) la (les) bibliothèque(s) partagées que votre programme tri utilise. Quelles sont les fonctions de cette (ces) bibliothèques(s) qui sont utilisées ?

```
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$ gcc -o tri -z lazy tri.c
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$ ltrace tri 3 2 1
Can't execute `tri': No such file or directory
failed to initialize process 5017: No such file or directory
couldn't open program 'tri': No such file or directory
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$ ltrace ./tri 3 2 1
malloc(12)                                                = 0x55b66fe292a0
atoi(0x7fffca2050e5, 0, 0x55b66fe292a0, 0)                = 3
printf("%d ", 1)                                          = 2
putchar(10, 0x55b66f8fa007, 0, 01 2 3
)                                 = 10
free(0x55b66fe292a0)                                      = <void>
+++ exited (status 0) +++
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$
```

### Exercice 15

Utilisez le programme *strace* pour trouver où se trouve(nt) la (les) bibliothèques(s) qui sont chargées

```
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$ strace ./tri 2 3 1
execve("./tri", ["./tri", "2", "3", "1"], 0x7ffe92978558 /* 62 vars */) = 0
brk(NULL)                               = 0x565340e8f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffce558d0a0) = -1 EINVAL (Argument invalide)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Aucun fichier ou dossier de ce type)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=72651, ...}) = 0
mmap(NULL, 72651, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f64ce33e000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f64ce33c000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f64ce14a000
mprotect(0x7f64ce16f000, 1847296, PROT_NONE) = 0
mmap(0x7f64ce16f000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f64ce16f000
mmap(0x7f64ce2e7000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f64ce2e7000
mmap(0x7f64ce332000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f64ce332000
mmap(0x7f64ce338000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f64ce338000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f64ce33d540) = 0
mprotect(0x7f64ce332000, 12288, PROT_READ) = 0
mprotect(0x565340baf000, 4096, PROT_READ) = 0
mprotect(0x7f64ce37d000, 4096, PROT_READ) = 0
munmap(0x7f64ce33e000, 72651)           = 0
brk(NULL)                               = 0x565340e8f000
brk(0x565340eb0000)                     = 0x565340eb0000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "1 2 3 \n", 71 2 3
)                 = 7
exit_group(0)                           = ?
+++ exited with 0 +++
user@PNS-VirtualBox:~/Documents/GitHub/PNS-S6-ProgSys/td01_distrib/td01$
```

## Exercice 16

Enfin, voici un dernier programme rechercheBinaire.c pour vous entrainer au debug d'un programme C dans l'environnement Visual Code.

Ajout d'un +1 manquant dans le else de gauche.

milieu = (gau + droite) / 2;

if (tab[milieu] == x)

return milieu;

if (tab[milieu] > x)

droite = milieu - 1;

else

**gau = milieu + 1;**