







0.1 Exemple de débogage : Gestion des pointeurs NULL selon le système d'exploitation

0.1.1 Contexte du problème

Lors de l'exécution du programme, une **erreur de segmentation** a été observée. Curieusement, ce comportement ne se manifestait que sur certaines machines de l'équipe, tandis que d'autres pouvaient exécuter le jeux sans problème. L'origine du bug etait dans la tentative de conversion d'un pointeur NULL, retourné par la fonction **getObject** pour un index invalide, en un pointeur d'un autre type.

0.1.2 Investigation initiale

A force de test nous sommes arrivé à la conclusion que :

- Les paramètres d'entrée étaient vérifier, et non égale à NULL.
- Deux éléments étaient suspectés d'être en cause :
 - La variable salleDep dans le contrôleur de scène
 - La structure de données scene

```
Placing emity at tile (6,10) => coords (96,160) with circle collision

Placing entity at tile (11,13) => coords (176,208) with circle collision

Placing entity at tile (17,13) => coords (272,208) with circle collision

Placing entity at tile (4,16) => coords (64,256) with circle collision

Placing entity at tile (10,17) => coords (160,272) with circle collision

Summary: Placed 1 spikes, 0 chests, and 12 barrels

Thread 1 "Ubissauphte" received signal SIGSEGV, Segmentation fault.

0x0000555555555252 in setScene (controller=0x555557c4f940, name=0x5555557c0bb "menuPrincipal") at src/utils/fscene.c:111

111

if (salleDep && strcmp(salleDep->name, "main")) {
(gdb) [

© 0 \( \Delta \) \( \Delta \) \( \Delta \)
```

FIGURE 1 – Trace d'exécution dans GDB montrant le point de crash

L'analyse à l'aide de GDB a permis d'observer :

- Une adresse mémoire non nulle (ex. 0x01) pour salleDep, même dans le cas où getObject devait renvoyer NULL.
- Une variation aléatoire de la valeur de salleDep entre différentes exécutions et machine (voir Fig. 3)

FIGURE 2 - Sortie de print salleDep montrant une adresse invalide

0.1.3 Origine du problème et correction du code

```
Listing 1 – Ligne fautive
```





```
PLSSING arguments.

(gdb) where
#0 0x000055555556de3d in setScene (controller=0x555556518930, name=0x5555557fd33 "menuPrincipal") at src/utils/fscene.c:111

#1 0x00005555555776c6 in main (argc=1, argv=0x7fffffffdd18) at src/main.c:139

(gdb) p salleDep

$1 = (t_scene *) 0x1

(gdb) [
```

FIGURE 3 – Variation de la valeur de salleDep entre deux exécutions

```
ajouter
[Thread 0x7fff954b8640 (LWP 83957) exited]
[Thread 0x7fffe62d9640 (LWP 83935) exited]
[Thread 0x7ffff62decc0 (LWP 83932) exited]
[Thread 0x7fffe6298640 (LWP 83936) exited]
[New process 83932]
[Inferior 1 (process 83932) exited normally]
(gdb)
```

Figure 4 – Comportement normal lorsque tout marche

Le problème venait de la tentative de conversion d'un pointeur NULL, retourné par la fonction getObject lorsqu'un indice invalide (Dans notre cas, -1) était utilisé. Nous avons théorisé que en raison des différences d'initialisation de la mémoire entre systèmes d'exploitation, certains environnements interprétaient correctement ce retour comme étant NULL, tandis que d'autres ne le faisaient pas, générant ainsi des valeurs indéterminées (par exemple : 0x01, 0xf0), pouvant inclure NULL, ne facilitant pas le debuggage.