Exemple d’utilisation du débogueur GDB :

Le débogueur GDB s’est révélé être d’une grande aide tout au long du projet pendant lequel nous avons découvert son potentiel, notamment lors d'une modification importante de la structure des fonctions du programme dans l’objectif de se débarrasser de certaines variables globales superflues et de pouvoir appliquer les fonctions à n’importe quel élément passé en paramètre (dans cet exemple n’importe quel joueur) afin de respecter les concepts de la programmation modulaire.

Il s’agit ici du cas d’une erreur de segmentation dont la cause à pu être découverte rapidement et sans avoir à relire méticuleusement l’intégralité du code concerné.
L’exécution du programme sous GDB informe d’une erreur de segmentation dans la fonction « equiper_sac_slot » et suggère un déréférencement illégal vers la variable « sac ».

```
Thread 1 "jeux.bin" received signal SIGSEGV, Segmentation fault.
equiper_sac_slot (joueur=0x555555de69e0, slot=1) at src/inventaire.c:141
141         lobjet_t * sac = joueur->inventaire->sac;
(gdb) up
#1  0x000055555556048b in afficher_inventaire (joueur=0x555555de69e0) at src/menus.c:236
236                             equiper_sac_slot(joueur, slot_selectionne);
(gdb)
#2  0x00005555555592e3 in keyDown (ev=0x7fffffffe410, joueur=0x555555de31a0) at src/event.c:46
46                       afficher_inventaire(joueur->inventaire);
(gdb)
#3  0x00005555555596d1 in jeu_event (joueur=0x555555de31a0) at src/event.c:174
174                 case SDL_KEYDOWN : keyDown((SDL_KeyboardEvent*)&lastEvent.key, joueur); break;
```

Aucun paramètre nul n’étant observé, la pile des fonctions appelées est remontée en utilisant la commande « up ».
On observe alors les valeurs à la recherche d’une incohérence.
Dans cet exemple, on remarque que l’argument « joueur » est modifié entre l’appel des fonctions « keyDown » et « afficher_inventaire », il est alors judicieux d’investiguer dans le code de la première fonction.
Les valeurs proches des deux arguments mettent sur la piste du passage d’un membre de la structure joueur en paramètre plutôt que du pointeur sur la structure, cependant aucune erreur n’es déclarée à la compilation.
Dans le doute je supprime les fichiers objets et exécutables du projet à l’aide de la fonction mrproper du makefile avant de le recompiler, ce qui révèle enfin le problème :

```
src/event.c: In function ‘keyDown’:
src/event.c:46:43: warning: passing argument 1 of ‘afficher_inventaire’ from incompatible pointer type [-Wincompatible-pointer-types]
   46 |                 afficher_inventaire(joueur->inventaire);
      |                                     ~~~~~~^~~~~~~~~~~~
      |                                           |
      |                                           inventaire_t * {aka struct inventaire_s *}
In file included from include/commun.h:19,
                 from src/event.c:1:
include/menus.h:35:44: note: expected ‘joueur_t *’ {aka ‘struct joueur_s *’} but argument is of type ‘inventaire_t *’ {aka ‘struct inventaire_s *’}
   35 | extern void afficher_inventaire(joueur_t * joueur);
      |                                  ~~~~~~~~~~~~^~~~~~
```

Comme envisagé, le compilateur informe que le type d’argument ne correspond pas au paramètre attendu.

GDB a beaucoup servi à suivre la valeur des variable lors des phases de débogage, en particulier pour suivre la valeur des variables avec la commande « display » et le pistage de leur modifications avec la commande « watch ».