Quelles sont les bases fondamentales à connaitre pour utiliser GDB sur une machine LINUX ?

Pour utiliser GDB (GNU Debugger) sur Linux, il faut d'abord compiler votre programme avec l'option `-g` pour inclure les informations de débogage, puis lancer GDB avec l'exécutable et utiliser quelques commandes clés comme `break`, `run`, `print`, `step`/`next`, `continue` et `backtrace`. [1] [2] [3]

1. Préparation : compiler avec les symboles de débogage

GDB a besoin d'informations supplémentaires dans l'exécutable (numéros de ligne, noms de variables, etc.). Avec GCC, on les ajoute via l'option `-g` :

```
gcc -g -Wall programme.c -o programme
```

- `-g` : active les informations de débogage.
- `-Wall` : active les avertissements (utile, mais pas obligatoire pour GDB). [2] [3] [1]

Sans `-g`, GDB fonctionnera, mais vous n'aurez pas les numéros de ligne ni les noms de variables, ce qui rend le débogage beaucoup moins pratique. [3]

2. Lancer GDB

Depuis un terminal :

```
gdb ./programme
```

Variantes utiles :

- `gdb -tui ./programme` : mode avec interface texte (fenêtres pour le code, les registres, etc.). [1]
- `gdb ./programme core` : pour analyser un fichier core (plantage post-mortem). [4] [5]
- `gdb ./programme 2345` : attacher GDB à un processus déjà en cours (PID 2345). [5]

Pour quitter GDB : `quit` (ou `q`). [3] [1]

3. Commandes de base indispensables

Voici les commandes essentielles à connaître (avec leurs raccourcis courants) :

```
- `break <ligne>` / `b <ligne>` : poser un point d'arrêt à une ligne donnée\
```

Ex. : `b 15` ou `b main` ou `b fichier.c:42`.[3]

- `run` / `r` : lancer l'exécution du programme (s'arrête au premier point d'arrêt).[1] [3]
  - Avec arguments : `run arg1 arg2`.[1]
- `continue` / `c` : reprendre l'exécution jusqu'au prochain point d'arrêt.[3] [1]
- `step` / `s` : avancer d'une instruction, en entrant dans les fonctions appelées.[2] [3]
- `next` / `n` : avancer d'une instruction, sans entrer dans les fonctions (traite l'appel comme un bloc).[2] [3]

```
- `print <expr>` / `p <expr>` : afficher la valeur d'une variable ou d'une expressio
```

Ex. : `p x`, `p tableau[i]`.[2] [3]

- `backtrace` / `bt` : afficher la pile d'appels (où on est dans l'exécution).[2] [3]
- `list` / `l` : afficher le code source autour de la position actuelle.[3] [2]

```
- `watch <var>` / `w <var>` : arrêter l'exécution quand la variable change.[^2][^3]
```

- `clear <ligne>` : supprimer un point d'arrêt à une ligne donnée.[3]
- `delete` : supprimer tous les points d'arrêt (ou `delete <num>` pour un seul).[3]

Interruption manuelle : pendant l'exécution, `Ctrl+C` dans le terminal interrompt le programme et revient dans GDB, où vous pouvez faire `bt`, `p ...`, etc..[1]

4. Workflow typique pour débuter

Un scénario simple de débogage :

1. Compiler avec `-g` :

   ```
   gcc -g programme.c -o programme
   ```

2. Lancer GDB :

   ```
   gdb ./programme
   ```

3. Poser un ou plusieurs points d'arrêt :

   ```
   b main
   b 42
   ```

4. Lancer le programme :

```
run
```

5. À chaque arrêt :
   - inspecter les variables : `p variable`, `p *pointeur`
   - voir où on est : `bt`, `list`
   - avancer pas à pas : `n` ou `s`
   - continuer jusqu'au prochain point d'arrêt : `c`
6. Quitter :

```
quit
```

5. Points pratiques à retenir

- GDB s'utilise sur l'exécutable, pas directement sur le `.c`. [3]
- Les raccourcis (`b`, `r`, `c`, `n`, `s`, `p`, `bt`, `q`) sont très utilisés en pratique. [2] [3]
- Vous pouvez attacher GDB à un processus existant si vous connaissez son PID. [6] [5]
- Pour les programmes avec arguments, utilisez `run arg1 arg2` ou `gdb --args ./programme arg1 arg2`. [6] [1]

Si vous le souhaitez, je peux vous proposer un petit exercice guidé (programme C volontairement bogué + séquence de commandes GDB à taper pas à pas).

⁂

1. https://doc.ubuntu-fr.org/gdb
2. https://perso.ens-lyon.fr/daniel.hirschkoff/C_Caml/docs/doc_gdb.pdf
3. https://perso.ens-lyon.fr/francois.schwarzentruber/teaching/l3-prog/book/gdb.html
4. https://fr.linux-console.net/?p=7435
5. http://chl.be/glmf/www.linuxmag-france.org/lm23/memprof.html
6. https://c.developpez.com/cours/mode-emploi-gcc/
7. http://sdz.tdct.org/sdz/deboguer-son-programme-avec-gdb.html
8. https://devarea.developpez.com/dix-choses-faisables-avec-GDB/
9. https://linux-fra.com/?p=4204
10. https://fr.linux-console.net/?p=4551