Débugger un programme
Introduction à l'utilisation de GDB

Différents types de bugs dans un programme

1. à la compilation,
2. exécution qui s'arrête avant la fin (ex. erreur de segmentation) ou qui ne s'arrête jamais,
3. résultat du programme qui n'est pas celui attendu.

Différents types de bugs dans un programme

1. à la compilation,
2. exécution qui s'arrête avant la fin (ex. erreur de segmentation) ou qui ne s'arrête jamais,
3. résultat du programme qui n'est pas celui attendu.

Pour 1 :

- comprendre les messages d'erreur et d'alerte du compilateur ;

Différents types de bugs dans un programme

1. à la compilation,
2. exécution qui s'arrête avant la fin (ex. erreur de segmentation) ou qui ne s'arrête jamais,
3. résultat du programme qui n'est pas celui attendu.

Pour 1 :

- comprendre les messages d'erreur et d'alerte du compilateur ;

Pour 2 et 3 :

- faire des affichages avec `printf` dans le programme,
- utiliser le débuggeur GDB pour inspecter l'exécution du programme à certains endroits clef.

Déchiffrer un message d'erreur de compilation

```
essai.c: In function ‘main’:
essai.c:10:3: error: expected ‘;’ before ‘printf’

essai.c: In function ‘main’:
essai.c:10:3: warning: format ‘%d’ expects argument of type
```

Déchiffrer un message d'erreur de compilation

```
essai.c: In function ‘main’:
essai.c:10:3: error: expected ‘;’ before ‘printf’

essai.c: In function ‘main’:
essai.c:10:3: warning: format ‘%d’ expects argument of type
```

Règle numéro 1 :
quand il y a plusieurs messages d'erreur, seul **le premier** compte.

Erreurs communes (1)

```
'variable' undeclared (first use in this function)
    (Each undeclared identifier is reported only once...
```

Erreurs communes (1)

```
'variable' undeclared (first use in this function)
    (Each undeclared identifier is reported only once...
```

```
too few arguments to function ‘toto’
```

Erreurs communes (1)

```
'variable' undeclared (first use in this function)
    (Each undeclared identifier is reported only once...
```

```
too few arguments to function ‘toto’
```

```
warning: implicit declaration of function ‘toto’
```

Fonction `toto` non définie avant son utilisation

Erreurs communes (1)

```
'variable' undeclared (first use in this function)
    (Each undeclared identifier is reported only once...
```

```
too few arguments to function ‘toto’
```

```
warning: implicit declaration of function ‘toto’
```

Fonction `toto` non définie avant son utilisation

```
passing arg n of ‘toto’ makes pointer from integer without a c
```

erreur liée à l'usage des pointeur (vérifier le type).

Erreurs communes (1)

```
'variable' undeclared (first use in this function)
    (Each undeclared identifier is reported only once...
```

```
too few arguments to function 'toto'
```

```
warning: implicit declaration of function 'toto'
```

Fonction `toto` non définie avant son utilisation

```
passing arg n of 'toto' makes pointer from integer without a c
```

erreur liée à l'usage des pointeur (vérifier le type).

```
undefined reference to 'sqrt'
```

oubli de compiler avec l'option `-lm`

Erreurs communes (2)

```
warning: control reaches end of non-void function
```

il manque une valeur de retour à une fonction qui n'est pas de type `void`

Erreurs communes (2)

`warning: control reaches end of non-void function`

il manque une valeur de retour à une fonction qui n'est pas de type `void`

`warning: ISO C forbids nested functions`

probablement oubli de l'accolade fermante d'une fonction.

Erreurs communes (2)

`warning: control reaches end of non-void function`

il manque une valeur de retour à une fonction qui n'est pas de type `void`

`warning: ISO C forbids nested functions`

probablement oubli de l'accolade fermante d'une fonction.

`error: syntax error before '}' token`

point virgule manquant en fin de fonction.

GDB et compilation

Pour que GDB puisse relier l'exécution aux lignes du programme correspondante, il faut ajouter l'option `-g` à la compilation :

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

Démarrer GDB

GDB s'utilise dans le terminal de la façon suivante :

```
gdb ./exemple
```

Le logiciel DDD propose une version graphique de GDB.

Pour l'aide :

```
man gdb
```

Executer le programme avec GDB

En ligne de commande :

```
(gdb)  run [arguments du programme]
```

- si pas de “gros” problème le programme s’exécute,
- si le programme s’arrête avant la fin, alors GDB renvoie des informations tel que
  - la ligne du programme correspondant au crash,
  - les valeurs des paramètres dans la fonction où a eu lieu le crash.

Executer le programme avec GDB

En ligne de commande :

```
(gdb)  run [arguments du programme]
```

- si pas de “gros” problème le programme s’exécute,
- si le programme s’arrête avant la fin, alors GDB renvoie des informations tel que
  - la ligne du programme correspondant au crash,
  - les valeurs des paramètres dans la fonction où a eu lieu le crash.

Pour terminer GDB

```
(gdb) quit
```

Afficher des informations

Afficher la pile des appels (backtrace)

```
(gdb) bt
```

Afficher des informations

Afficher la pile des appels (backtrace)

```
(gdb) bt
```

Afficher la valeur des variables (print)

```
(gdb) p variable
(gdb) p tableau[0]
```

Inspecter l'exécution du programme

Mettre sur pause l'exécution du programme en ajoutant des **points d'arrêt**.

```
(gdb) break exemple.c:6
(gdb) break fonction
```

Inspecter l'exécution du programme

Mettre sur pause l'exécution du programme en ajoutant des **points d'arrêt**.

```
(gdb) break exemple.c:6
(gdb) break fonction
```

A chaque fois que le programme atteint cette partie du programme (si il l'atteint), il se met en pause et attend une nouvelle instruction.

Plusieurs points d'arrêt peuvent être définis avant ou pendant l'exécution.

Inspecter l'exécution du programme

Mettre sur pause l'exécution du programme en ajoutant des **points d'arrêt**.

```
(gdb) break exemple.c:6
(gdb) break fonction
```

A chaque fois que le programme atteint cette partie du programme (si il l'atteint), il se met en pause et attend une nouvelle instruction.

Plusieurs points d'arrêt peuvent être définis avant ou pendant l'exécution.

Les points d'arrêts peuvent être supprimés.

```
(gdb) delete exemple.c:6
```

Inspecter l'exécution du programme

Mettre sur pause l'exécution du programme en ajoutant des **points d'arrêt**.

```
(gdb) break exemple.c:6
(gdb) break fonction
```

A chaque fois que le programme atteint cette partie du programme (si il l'atteint), il se met en pause et attend une nouvelle instruction.

Plusieurs points d'arrêt peuvent être définis avant ou pendant l'exécution.

Les points d'arrêts peuvent être supprimés.

```
(gdb) delete exemple.c:6
```

Infos sur les points d'arrêt :

```
(gdb) info breakpoints
```

Se déplacer dans l'exécution du programme

- `run` (re-)démarre l'exécution,

Se déplacer dans l'exécution du programme

- `run` (re-)démarre l'exécution,
- `continue` continue l'exécution jusqu'au prochain arrêt,

Se déplacer dans l'exécution du programme

- `run` (re-)démarre l'exécution,
- `continue` continue l'exécution jusqu'au prochain arrêt,
- `step` exécute une instruction de base (instruction-expression),

Se déplacer dans l'exécution du programme

- `run` (re-)démarre l'exécution,
- `continue` continue l'exécution jusqu'au prochain arrêt,
- `step` exécute une instruction de base (instruction-expression),
- `next` exécute une instruction sans entrer dans les blocs.

Autres fonctionnalités

```
(gdb) watch variable
```

Le programme s'interrompt à chaque fois que la variable `variable` est modifiée.

Autres fonctionnalités

```
(gdb) watch variable
```

Le programme s'interrompt à chaque fois que la variable `variable` est modifiée.

Point d'arrêt conditionnel

```
(gdb) break exemple.c:6 if i < 0
```

Autres fonctionnalités

```
(gdb) watch variable
```

Le programme s'interrompt à chaque fois que la variable `variable` est modifiée.

Point d'arrêt conditionnel

```
(gdb) break exemple.c:6 if i < 0
```

Affecter une valeur à une variable

```
(gdb) set variable nom_variable = expression
```