Maintenance de Logiciels

Travaux Dirigés N° 03

Débogueur - GDB

Préliminaires :

GDB est un débogueur symbolique permettant de contrôler le déroulement de programmes compilés avec le compilateur **GCC** (C, C++, Fortran...). **GDB** permet (entre autres) de mettre des points d'arrêt dans un programme, de visualiser l'état de sa pile d'appels ou de ses variables, de calculer des expressions, d'appeler interactivement des fonctions...

L'objectif de ce TD est d'explorer l'ensemble des commandes de GDB. Pour cela, on reprendra la liste des commandes présentes dans cours. Comme exemple de test, on utilisera un exemple de programme de tri qui sera fourni pendant la séance. Par ailleurs, un fichier **Makefile** sera fourni. Le programme qui sera fourni est un programme de tri incomplet (réalisant seulement un tri sur les 4 demandés). À titre d'exercice, on complètera le programme en ajoutant 4 fonctions de comparaison et en utilisant des pointeurs sur fonctions C dans la fonction `quicksort`.

Exercice 01 : Prise en main de GDB

Décompresser l'archive `td_gdb.tar.gz` obtenue puis entrer dans le dossier `td_gdb` :

```
~% tar xzvf td_gdb.tar.gz
~% cd td_gdb
```

Question 1 :

Modifier le fichier **Makefile** en ajoutant les éléments nécessaires pour incorporer les informations de débogage dans l'exécutable `tri`.

Question 2 :

Visionner les différents fichiers sources du programme. Dans un terminal, lancer l'exécutable `tri` et fournir quelques exemples d'informations à trier.

Question 3 :

Lancer le programme avec le débogueur **GDB** et tester les différentes commandes :

```
~% gdb ./tri
```

- Exécuter le programme `tri` à travers **GDB** (commande **run**)
- Positionner des points d'arrêt en utilisant les différentes possibilités la commande **break**
- Afficher les points d'arrêt créés (**info b**)
- Effacer certains points d'arrêt et vérifier que la liste des points d'arrêt a été remise à jour
- Exécuter le programme et vérifier qu'il s'arrête bien dans les points d'arrêts spécifiés

- Tester le fonctionnement des commandes **next** et **step**
- Afficher le contenu de quelques variables (**print**)
- Utiliser la commande **display** pour faire afficher automatiquement le contenu de variable
- Faire afficher l'état de la pile dans le cas d'appels de procédures imbriqués (**backtrace** ou **bt**)
- Faire afficher le code en cours d'exécution (**list**)
- Se déplacer le niveau d'appel étudié avec les commandes **up** et **down** et vérifier avec la commande **list** que le déplacement a bien eu lieu.

Exercice 02 : Ajout des tris manquants

La fonction `quicksort` présente dans le programme ne permet de trier que des entiers dans un ordre croissant. Dans cet exercice, il s'agira de rendre cette fonction plus générique en lui ajoutant un paramètre supplémentaire qui représente la procédure de comparaison implantée par l'utilisateur.

Ainsi, on pourra ordonner les informations par ordre alphabétique par exemple. Pour ce faire, on utilisera les pointeurs sur des fonctions C. Bien évidemment, pour chaque problème rencontré lors de l'enrichissement du programme, on utilisera le débogueur **GDB** pour aider à le résoudre.

Question 1 :

Dans le fichier `types.h`, définir un type `compare_t` qui est un pointeur sur une fonction ayant la signature suivante :

- type de retour entier (0 ou 1)
- prend deux paramètres de types `ITEM`

Définir dans le fichier `tri.c` les 4 fonctions (pouvant être pointées par `compare_t`) suivantes :

- `inf_age_croissant` : retourne 1 si l'age de son premier paramètre est inférieur à celui de son second paramètre et 0 sinon
- `inf_age_decroissant` : retourne 1 si l'age de son premier paramètre est supérieur à celui de son second paramètre et 0 sinon
- `inf_alpah_nom` : retourne 1 si le nom de son premier paramètre est inférieur (alphabétiquement) à celui de son second paramètre et 0 sinon
- `inf_alpha_prenom` : retourne 1 si le prénom de son premier paramètre est inférieur (alphabétiquement) à celui de son second paramètre et 0 sinon

Question 2 :

Modifier les fonctions `Trier` et `quicksort` pour qu'elles prennent un paramètre supplémentaire de type `compare_t` et qu'elles utilisent la fonction pointée par ce paramètre pour effectuer le tri rapide.

Question 3 :

Compléter la fonction `Choix` pour supporter les 3 tris restants