INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Enseignant : *Vincent Migliore*
vincent.migliore@insa-toulouse.fr

ARCHITECTURE MATERIELLE
Introduction à GDB

GDB est un outil de debug de la chaîne de compilation de GCC. Prenant en paramètre un binaire exécutable, il permet au programmeur de faire des opérations avancées notamment :

→ l'arrêt du déroulement du programme à des instants précis (points d'arrêts) ;

→ d'observer l'état des registres et de la mémoire ;

→ de faire le lien entre le code C et le code désassemblé (sous réserve d'avoir utilisé l'option -g pendant la compilation).

Remarque : Étant un outil travaillant au niveau du binaire, une bonne connaissance du langage d'assemblage est un prérequis indispensable à son utilisation.

Toute machine linux où GCC est installé a accès à GDB (il fait partie de la chaîne de compilation), qui se trouve généralement dans le PATH, simplifiant son utilisation. Pour tester un binaire *mon-binaire.bin* par exemple, le plus simple est de se trouver dans le même dossier que ce dernier et taper **gdb mon-binaire.bin**. En cas de succès, le préfixe (gdb) devrait apparaître au niveau de la ligne de commande.

quit	Permet de quiter gdb.

1 Gestion de l'interface

GDB fonctionne entièrement dans un seul terminal. En fonction des informations à afficher (invite de commande, du code, etc...), le terminal sera visuellement découpé en 1, 2 ou 3 parties. Il existe 3 fenêtres différentes : La fenêtre de l'invite de commande, nommée **cmd**; la fenêtre pour le code désassemblé, nommée **asm**; et la fenêtre pour le code source en c (uniquement si l'option -g a été utilisée), nommée **src**.

Pour choisir les fenêtres à afficher, on utilise la commande **layout** :

layout asm	Invite de commande + code désassemblé.
layout src	Invite de commande + code source.
layout split	Invite de commande + code désassemblé + code source.

Afin de pouvoir naviguer dans les différentes fenêtres (permettant notamment de se déplacer dans le code source, dans le code désassemblé, ou de pouvoir récupérer vos précédentes commandes, il faut utiliser la commande **focus** :

focus asm	Permet de se placer dans la fenêtre du code désassemblé.
focus src	Permet de se placer dans la fenêtre du code source.
focus cmd	Permet de se placer dans la fenêtre de l'invite de commande.

2 Navigation dans le code

Pour se déplacer dans le code, deux stratégies sont possibles (et peuvent se combiner) : placer des points d'arrêts à des moments précis, puis exécuter le code qui se stoppera au premier point d'arrêt atteint; et/ou faire du pas à pas.

Pour le placement de points d'arrêts, on utilise la fonction **breakpoint** (simplifiable par **b**) :

b * *addresse*	Point d'arrêt à l'adresse *addresse* du binaire.
b *fichier:ligne*	Point d'arrêt à une ligne du code c. Note : **b *ligne*** pour le faire sur le fichier source courant.
b *fichier:fonction*	Point d'arrêt à une fonction du code c. Note : **b *fonction*** pour le faire sur le fichier source courant.
delete *i*	Suppression du point d'arrêt d'indice *i*.

Pour faire avancer l'exécution, différentes commandes sont disponibles :

run	Permet de charger le programme et de l'exécuter.
continue / c	Relance de l'exécution (qui s'arrêtera au premier point d'arrêt trouvé, ou à la fin du code).
step / s	Avance d'une ligne du code source (i.e. le code c).
next / n	Avance d'une ligne du code source (i.e. le code c) sans descendre dans les fonctions.
stepi / si	Avance d'une ligne du code désassemblé.
nexti / ni	Avance d'une ligne du code désassemblé sans descendre dans les fonctions.

3 Observation des variables, des registres et de la mémoire

Lecture des variables/registres, on utilise la commande **print** (abréviation : **p**). Pour faire la distinction entre variable et registre, les registres ont le préfixe $ (exemple $eax pour afficher le registre eax).

p *variable*	Affichage du contenu d'une variable.
p/x *$registre*	Affichage en hexadécimal du contenu d'un registre.
p {*variable,$registre*}	Affichage du contenu de plusieurs variables / registres.

Pour la mémoire, on utilise la commande **x** :

x/x ***adresse*	Affichage en hexadécimal du contenu d'une adresse.
x/x *$registre*	Affichage en hexadécimal du contenu d'une adresse à partir d'un registre.
x/16x *$registre*	Affichage en hexadécimal du contenu de 16 adresses successives.