Département IMA / 3A (S6) / 2013 **Programmation Avancée**
Jonathan Aceituno, Vincent Aranéga, Nathalie Devésa, Rachid Elkouri et Laure Gonnord

TP5 - Debug avec GDB

Objectifs Découvrir et appliquer les principes du debogage avec gdb, pour trouver les erreurs d'algorithmique, traquer les boucles infinies et résoudre les erreurs de segmentation. Utilisation sous emacs.

Avant de commencer, récupérez *à la ligne de commande* sur le compte polytech de Laure Gonnord l'archive `tp5_gdb.tgz` (adresse /home/imaEns/lgonnord/PA2013/TP5/) et désarchivez-la dans votre répertoire de PA *toujours à la ligne de commande!*.

1 Tutoriel GDB

(Source : Matthieu Moy pour Ensimag)

Ouvrir le fichier `gdb-tutorial2.c` avec Emacs et faire ce tutoriel en lançant gdb depuis la ligne de commande.

On utilisera deux terminaux, un pour compiler et un pour lancer gdb. à chaque modification du fichier source, on compilera, mais il ne sera pas nécessaire de quitter GDB.

2 Programmes à déboguer

Debug assisté Le programme contenu dans le source `buginsa.c` prend en argument un entier n à la ligne de commande, crée un tableau de taille n, l'initialise en mettant i^2 à la case i, puis désalloue le tableau. Mais "il segfaulte". Pourquoi ?

1. Lancez gdb et chargez le binaire.
2. Lancez l'exécution avec 3 comme argument (`run 3`), repérer la ligne où a lieu l'erreur.
3. Afficher la pile d'appel de fonction menant à l'erreur (commande `backtrace`).
4. Tapez `help breakpoints` pour avoir la liste des commandes permettant d'utiliser les point d'arrêt. Mettez un point d'arrêt au début de la fonction main, un autre au début de la fonction traitement. Enlevez celui de la fonction main.
5. Relancez le programme.
6. Lorsque vous êtes arrêté dans la fonction traitement, mettez une surveillance de la variable i (commande watch). Entrez la commande `watch i` pour afficher i à chaque arrêt.
7. Tapez `cont` pour continuer, surveillez l'exécution jusqu'à ce que vous trouviez le problème.

(Source : Tanguy Risset pour INSA Lyon)

Autres bugs à trouver

1. Utilisez l'interface graphique `ddd` pour trouver et corriger les bugs de `listsegf.c` et `list.c`. On utilisera les fonctions de dessin des listes pour regarder leur évolution au cours de l'exécution du programme (un document sur `ddd` vous a été distribué au S5 (TP8) et est encore dispo à l'adresse http://laure.gonnord.org/pro/teaching/progStructuree.html).

2. Ouvrir le fichier `exo-bug.c`. Le programme recherche un élément `element` dans un tableau `tab` *trié* de taille `nb` par dichotomie. Corriger ses deux bogues. Pour arrêter une exécution avec Emacs/gdb, il faut taper CTRL-c/CTRL-c (deux fois). (Source : Antoine Miné pour ÉNS Ulm)

3 Questions s'il vous reste du temps

Petits exos

1. Trouvez et corrigez le bug de `boucle.c`. (Source : Amélie Lambert pour ENSIIE).
2. **Pointeurs/Tableaux** Trouvez et corrigez le bug de `perror.c`. (Source : idem)
3. **Dépassement de capacité** Trouvez et corrigez le bug de `scan.c`. (source : A. Miné)
4. Faire "marcher" `prime.c`. (Source http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Debug.html)

Stack Overflow Le dépassement de tampon est un bug très classique pouvant avoir des conséquences graves pour la sécurité du système en permettant d'exécuter du code avec les droits de l'utilisateur. La pile est une partie de la mémoire disponible pour stocker les variables locales pour un programme. C'est une file LIFO où s'empilent les variables. Si une variable déborde sur la pile, il est alors possible de modifier d'autres variables (ou l'adresse de retour d'une fonction)

Dans DDD (fichier `stack.c`) :

1. Placez un point d'arrêt sur le printf de la fonction `foo`
2. Visualisez les variables myc.a, myc.c et bar pour 11 puis 12 puis 15, puis 20 fois la lettre 'A' passée en argument
3. Il est possible de visualiser l'état de la mémoire avec data/memory en choisissant par exemple 25 octets à partir de myc.c (decimal/bytes/Display).

(Source : Sylvain Ferrand, École Polytechnique)