POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S6) **Programmation Avancée 2010/2011**

TP2 : Débuggons !

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 et avec ddd.

Avant de commencer, récupérez sur la page web du cours l'archive `tp2_gdb.tgz` et désarchivez dans votre répertoire.

1 Tutoriel GDB

Ouvrir le fichier `gdb-tutorial.c` avec xemacs et refaire ce tutoriel (vous l'avez fait au S5, si, si !)

(Source : Matthieu Moy pour Ensimag)

2 GDB à la ligne de commande

Exo 1 Regarder puis compiler le programme `buginsa.c`, exécutez-le avec un entier comme argument sur la ligne de commande. Essayez de trouver l'erreur à vue (record sans débuggeur : 2'30). Même si vous avez trouvé, lancez gdb. Sous gdb, tapez help pour avoir la liste de commande, taper `help cmd` pour l'aide sur la commande cmd.

1. Chargez le programme bug (commande `file`).
2. Lancez l'exécution avec 3 comme argument (`run 3`), repérer la ligne où a lieu l'erreur. Afficher les lignes autour (commande `list`).
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 bug.
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)

3 Programmes à déboguer - Sous emacs obligatoirement

Exo 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 en utilisant gdb. Pour arrêter une exécution avec emacs/gdb, il faut taper CTRL-c/CTRL-c (deux fois).

(Source : Antoine Miné pour ÉNS Ulm)

Exo 3 : Boucle infinie Trouvez et corrigez le bug de `boucle.c`. (Source : Amélie Lambert pour ENSIIE)

Exo 4 : Dépassement de capacité Trouvez et corrigez le bug de `scan.c`.
(Source : idem)

Exo 5 : Pointeurs/tableaux Trouvez et corrigez le bug de `perror.c`.
(Source : idem)

Exo 6 : Un exemple plus conséquent Faire "marcher" `prime.c`.
(Source http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Debug.html)

4 Avec l'interface graphique ddd

Exo 7 : Listes Utilisez l'interface graphique `ddd` pour trouver et corriger les bugs de `list.c`. On utilisera les fonctions de dessin des listes pour regarder leur évolution au cours de l'exécution du programme.
(Source : code Laure Gonnord pour TP de SD IMA)

Exo 8 : 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)

5 Un peu de doc

GDB sous emacs : http://sourceware.org/gdb/current/onlinedocs/gdb/Emacs.html