Travaux Pratiques d'algorithmique n°6

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Compiler, déboguer et optimiser vos programmes sur DIVA

---

1 Compilation avec `cc`

Voici quelques options que vous connaissez dj et d'autres qui vous serviront par la suite :

- `-Aa` : option qui utilise la norme *C ANSI* ;
- `-o` nom : permet de donner un nom de fichier diffrent de `a.out` au fichier compil ;
- `-lm` : permet d'inclure la librairie mathmatique ;
- `-v` : demande au compilateur d'afficher ses actions ;
- `-g` : ncessaire pour utiliser le débogueur ;
- `-G` : ncessaire pour utiliser `gprof`.

2 Compilation et le pr-processeur

En fait, lorsque vous lancez la commande `cc`, le compilateur excute un autre programme appel le *pr-processeur C*. Pour vous en rendre compte, utilisez l'option `-v` (pour *verbose* ou *mode bavard*) afin de compiler votre programme : `cc -Aa -v -o prog prog.c`

Vous verrez alors que `cc` fait appel un programme situ dans `/lib` et s'appelant `cpp`. Ce programme s'appelle le *pr-processeur* et son but est de :

- substituer les macros `#define` par leur valeur relle ;
- substituer les `#include` par les prototypes des fonctions que les librairies contiennent ;
- permettre la compilation conditionnelle. Ainsi, certaines parties de votre code source peuvent tre ou ne pas tre compiles suivant la valeur des variables que vous dclarez.

Ainsi, le pr-processeur prpare votre programme pour que le compilateur puisse s'en servir pour gnrer du code binaire excutable. Pour vous rendre compte de ce que fait le pr-processeur, compilez un petit programme avec `cpp` : `/lib/cpp essai.c essai.cpp`

3 Dbogueur `xdb`

Un dbogueur est un outil trs utile qui permet de comprendre pourquoi votre programme se "plante" ou ne se comporte pas comme vous le dsirez. Un dbogueur vous permet de suivre l'excution pas pas de votre programme. Vous pouvez ainsi tout moment vrifier la valeur d'une variable, d'une structure et mme revenir en arrire dans l'excution de votre programme. Le dbogueur dont vous disposez sur diva est `xdb`. Pour l'utiliser vous devez compiler votre programme avec l'option `-g` : `cc -Aa -g prog.c`

Ensuite, tapez la commande suivante : `xdb a.out`

Vous verrez alors apparatre deux fentres. Celle du haut contient votre programme crit en *C* ou en code assembleur (si vous avez oubli de compiler avec l'option `-g`). La fentre du bas permet d'entrer les commandes de contrle du dbogueur. Pour passer de votre programme *C* au programme qui a t compil en assembleur, tapez dans la fentre de commande : `td`

3.1 Les commandes de base du debogueur

Voici les commandes de base du dbogueur :

- `w` taille permet de redfinir la taille de la fentre o se trouve le code *C* ;
- `U` nettoie et rafrachit l'cran.

Pour vous dplacer dans la fentre contenant votre programme `C`, vous devez utiliser les commandes suivantes :

- `v` numro de ligne vous dplace sur la ligne que vous avez indiqu ;
- `v` vous dplace du nombre de lignes contenu dans la fentre ;
- `+` (respectivement `-`) vous dplace d'une ligne vers le bas (respectivement vers le haut) partir de la ligne courante ;
- `/`chane recherche la ligne qui contient la chane dsire ;
- `n` (respectivement `N`) vous dplace vers le bas (respectivement vers le haut) sur la chane suivante trouve si vous faites une recherche.

Ainsi, si vous cherchez une variable toto, sans vous rappeler o elle se trouve dans votre programme, tapez ceci dans la fentre de commande de `xdb` :

`/toto`	recherche la ligne o se trouve le mot toto
`n`	recherche la seconde occurrence de toto dans le programme

3.2 Les points d'arrt

Vous avez la possibilit d'indiquer au dbogueur des lignes prcises, o les fonctions qui vous intressent plus particulirement. Pour cela, on utilise des points d'arrt (*Break Points*). Ainsi, le programme s'excutera normalement jusqu'au moment o un point d'arrt sera rencontr. Puis, le dbogueur vous rendra la main, pour vous permettre d'excuter votre programme pas pas, tout en contrlant les valeurs des variables. Les points d'arrt sont numrots au fur et mesure de leur cration.

Voici les commandes relatives aux points d'arrt :

- `b` *numro de ligne* insre un point d'arrt la ligne indique ;
- `b` *nom de la fonction* insre un point d'arrt la ligne contenant la fonction ;
- `lb` donne la liste de tous les points d'arrts crs ;
- `sb` *numro* suspend le point d'arrt du numro spcifi ;
- `db` *numro* supprime le point d'arrt du numro spcifi.

3.3 Excuter le programme et dboguer

Voici les commandes qui permettent de contrler l'excution du programme :

- `r` excute le programme jusqu'au premier point d'arrt ;
- `s` permet d'excuter une seule instruction la fois ;
- `g` permet d'aller la ligne dsire ;
- `k` permet d'arrter l'excution du programme dans le debogueur ;
- `p` permet de visualiser le contenu d'une variable ;
- `q` permet de quitter `xdb`.

Voici un exemple de session de dbogage avec `xdb` :

1. `cc -Aa -g essai.c`
2. `xdb a.out`
3. `b AlloueEnS` positionne un point d'arrt au dbut de la fonction
4. `b 105` positionne un point d'arrt la ligne 105
5. `r` (exécution du programme dans la fenêtre de commande)

On s'arrte sur la fonction `AlloueEns` et on veut savoir le contenu de la variable de type ensemble `E1` :

6. `p E1`
   `p (*E1).Tab(2)`

On excute deux commandes au pas pas :

7. s
   s

On suspend les points d'arrt, puis on continue l'exécution du programme :

8. sb *
   r

Pour quitter `xdb` :

9. quit

3.4 Visualiser la pile des appels de fonction

Lorsque votre programme se plante dans une fonction x, appele par une fonction y, elle-mme appele par une fonction z, on a souvent besoin de remonter dans la pile d'excution pour vrifier le contenu des variables lors de l'appel.

Pour cela, vous pourrez utiliser les commandes suivantes :

- `T` et `t` permettent de visualiser le contenu de la pile des appels de fonctions (avec ou sans les variables) ;
- `up` et `down` permettent respectivement de remonter dans le code de la fonction appelante, et de descendre dans le code de la fonction appele.

4 L'optimiseur de performance `gprof`

Pour optimiser au mieux un programme, il est ncessaire de savoir o le programme passe le plus de temps (dans une fonction, dans des entres-sorties). Les outils utiliss sont appels profiler et celui qui est disponible sur diva est `gprof`. Pour l'utiliser, vous devez compiler votre programme en utilisant l'option `-G`, soit : `cc -Aa -G essai.c`

Ensuite, vous excutez votre programme normalement, en vitant au maximum que votre programme soit interactif (vitez les `scanf`, `getchar`, etc.). Un fichier `gmon.out` sera alors cr et contiendra les statistiques sur votre programme que `gprof` saura dcoder.

Les statistiques donnes par `gprof` sont nombreuses, celles qui vous intressent sont celles qui concernent vos fonctions et le nombre de `malloc`, `calloc` et de `free`. Vous pouvez ainsi vrifier que vous faites autant de `malloc` que de `free`, ou que vous n'avez pas un programme qui passe 90% de son temps excuter la mme procdure. Voici la marche suivre :

```
cc -Aa -G -o essai essai.c
essai
gprof | more
```