POLYTECH LILLE

Université Lille1 Sciences et Technologies

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

TP1 : à la découverte de GCC et des librairies

Objectifs/Instructions

Découvrir certaines caractéristiques de gcc. Compilation et utilisation de librairies statiques et dynamiques. Avant de commencer, récupérez sur la page web de Laure Gonnord l'archive `tp1_gcc.tgz`.

1 Allocation statique/dynamique

Compilez et exécutez les programmes `alloc_statique.c` et `alloc_dyn.c`.

- Comparez les temps d'exécution avec l'aide de la commande Unix `time` : `time(./alloc_dyn)`
- Augmentez la taille de la matrice dans `alloc_statique.c`. Que se passe-t-il ? Pourquoi ?
- Mêmes questions avec `alloc_dyn.c`.
- Modifiez le code de `alloc_dyn.c` en ajoutant une vérification de la valeur de retour de `malloc`.

▶On vérifie toujours la valeur de retour de malloc !

2 Parcours en ligne/en colonne

Copiez le fichier `alloc_dyn.c` en `parcours_mat.c`. Comparez (sur de grandes tailles de matrices) les temps d'exécution des parcours ligne par ligne et colonne par colonne.

▶On favorise le parcours par ligne.

3 Compilation séparée et édition de liens

Cas simple : deux fichiers c d'un même projet Récupérer les fichiers `pile.h`, `pile.c` et `main.c` de l'archive.

1. Essayer de compiler "normalement" `main.c` :

   ```
   gcc -o main main.c
   ```

   Que se passe-t-il ? Pourquoi ?

2. Essayer de compiler `main.c` en code objet :

   ```
   gcc -c main.c
   ```

   Que se passe-t-il ? Pourquoi ?

3. Ajouter `#include "pile.h"` et compiler de nouveau avec `gcc -c`. Appeler `nm` sur le fichier binaire obtenu. Il manque donc le code donné dans `pile.c`.

4. Générer le code binaire pour `pile.c`. Vérifier que le binaire contient le code pour `initPile` (avec nm).

5. Compiler les deux fichiers objet ensemble :

   ```
   gcc -o testpile main.o pile.o
   ```

6. Exécuter le binaire pour vérifier que tout est bon. Faire nm sur le binaire.
7. Si l'on modifie uniquement `main.c`, comment faire pour recompiler ?

Utilisation d'une bibliothèque : le cas statique Cf transparents de cours :

1. Transformer le fichier objet `pile.c` en bibliothèque statique :

   ```
   gcc -c pile.c
   ar -cvq libpile.a pile.o
   ```

2. Regarder le contenu de `libpile.a` avec `nm`.
3. Ensuite, compiler de façon classique :

   ```
   gcc -o test main.o libpile.a
   ```

   ou ?

   ```
   gcc -o test libpile.a main.o
   ```

4. Mettre la librairie (.a) dans un autre répertoire et noter le chemin. Compiler avec :

   ```
   gcc -o testpile main.c -L<chemin> -lpile
   ```

5. Vérifier avec nm que votre binaire a bien embarqué le code de la librairie.
6. Changer libpile.a d'endroit, puis réexécuter le binaire.

►Lier une librairie statiquement revient à embarquer le code des fonctions (utilisées) dans le binaire.

Utilisation d'une bibliothèque : le cas dynamique Cf transparents de cours :

1. Transformer le fichier objet `pile.c` en bibliothèque dynamique :

   ```
   gcc -c pile.c
   gcc -o libpile.so -shared pile.o
   ```

2. Regarder le contenu de la lib, compiler dans le même répertoire, exécuter et vérifier que cela ne marche pas :

   ```
   error while loading shared libraries: libpile.so: cannot open shared
   object file: No such file or directory
   ```

3. Comparer les tailles des deux binaires : avec liaison statique et avec liaison dynamique.

►Lier une librairie dynamiquement impose de savoir à l'exécution à quel endroit se trouve le .so

1. Créer à la racine de votre compte un répertoire `lib`
2. Y placer `libpile.so`
3. Ouvrir (ou créer) le fichier `.bashrc` (à la racine de votre compte) et y placer la ligne suivante :

   ```
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/lib
   ```

4. Relancer un autre terminal pour prendre en compte la modification.
5. Vérifier que votre binaire s'exécute bien maintenant.

►Lors de l'exécution d'un binaire lié avec une lib dynamique, les libs sont cherchées dans le `LD_LIBRARY_PATH`

4 Les options d'optimisation de gcc

Reprenez vos codes du projet du premier semestre. Comparez les temps d'exécution en compilant sans option d'optimisation ou avec. Comparez aussi les tailles des binaires.

	(sans option)	`-O0`	`-O1`	`-O2`	`-O3`
temps d'exécution					
taille du binaire					

►L'option `-O2` permet souvent de gagner en temps d'exécution.