Travaux Dirigés Programmation C avancée

Informatique 1ère année.

—*Julien Allali* - `allali@enseirb-matmeca.fr` —

1 SVN

Dans cette partie, nous allons gérer les sources de la table de hachage dans un dépôt local svn.

▸**Exercice 1.** *Création d'un dépôt local :*
À la racine de votre compte, créer un répertoire .depots. *Dans ce répertoire créer un nouveau dépôt svn nommé* pg106 *avec la commande* `svnadmin --compatible-version 1.5 create pg106` *(l'option ajoutée est nécessaire du fait de la configuration locale).*

Ceci créer un dépôt svn. **Attention : on ne travaille jamais directement dans le dépôt mais dans un autre répertoire synchronisé avec le dépôt !**

Dans votre répertoire de travail, synchroniser vous avec le dépôt en utilisant la commande `svn checkout file://$HOME/.depots/pg106`. *Vous devriez obtenir la création d'un répertoire pg106 vide (à l'exception du dossier* `.svn`*).*

▸**Exercice 2.** *Structure de travail.*
Dans le dépôt créer trois répertoires : trunk, branches et tags. Transmettez ces modifications au dépôt.

▸**Exercice 3.** *Ajout des sources.*
Ajouter dans le trunk l'ensemble des sources de la table de hachage (`CMakeLists.txt`, `src/hash.c`, `src/hash.h` *etc...).*

Attention à ne pas ajouter des répertoires cachés type .git ou .svn.

Avant de transmettre les modifications au dépôt (commit), vérifier celles-ci avec la commande svn status.

A partir de maintenant, nous allons effectuer des développements dans des branches puis, une fois fini et testés, nous les rappatrieront dans le tronc du dépôt.

▸**Exercice 4.** *Création d'une branche.*
En utilisant la commande `svn copy`, *créer une copie du répertoire* `trunc` *dans un répertoire* `branches/test_hash`.

Nous allons maintenant travailler dans la branche nouvellement créée.

▸**Exercice 5.** *Environnement de test.*
Ajouter un répertoire appelé `tests`. *Dans ce répertoire, créer un premier fichier* `hash_unit_tests.c` *qui comportera le code :*

```
#include <assert.h>

int main(){
assert(false && "mon premier test!");
}
```

Ajouter un fichier `CMakeLists.txt` *qui permet la création de l'exécutable* `hash_unit_tests` *à partir de ce fichier source. Enfin, utiliser la commande* `add_test` *dans le fichier* `CMakeLists.txt` *pour automatiser le lancement de votre programme lors de l'appel à la commande* `make test` *depuis le répertoire de compilation.*

Vérifier bien que tout fonctionne avant de passer à la suite. Pour rappel, `make VERBOSE=1` *permet de voir les commandes de compilation utilisées.*

▸**Exercice 6.** *Tests unitaires*
Ecrire des tests unitaires pour chacunes des fonctions de la bibliothèque : init, ajout et recherche.

▸**Exercice 7.** *Nouveaux développements*
Nous allons maintenant ajouter une nouvelle fonction dans la bibilothèque : `hash_remove`*. Ajouter cette fonction dans le répertoire* `trunk` *du dépôt. Une fois développées, transmettre ces modifications au dépôt à l'aide de la commande* `svn commit`*.*

▸**Exercice 8.** *De retour dans la branche*
Lorsque l'on développe dans une branche, on essaye de toujours être synchronisé avec les développements en cours dans le tronc du dépôt. Pour cela, on utilise la commande `svn merge HEAD /path/to/trunk /path/to/branch`*.*

Utiliser cette commande pour récupérer les nouveaux développements dans votre branche liée aux tests. Penser à utiliser les commandes `status` *et* `commit`*.*

▸**Exercice 9.** *De la branche au tronc.*
Une fois la branche à jour et les développements de branche finis, on utilise la commande merge dans l'autre sens afin de rapatrier les modifications de la branche dans le tronc. Effectuer cette opération.

2 Couverture

La couverture est un élément important des tests. Cela correspond au nombre de lignes des fichiers sources qui ont été utilisées lors de l'exécution d'un test. Avant d'automatiser la génération de rapport de couverture, nous allons regarder comment cela fonctionne.

▸**Exercice 10.** *A la main.*
En utilisant `gcc`*, compiler à la main votre bibliothèque avec l'option* `--coverage`*. Vous verrez qu'en plus de la bibilothèque, de nouveaux fichiers ont été créés. Compiler votre programme de test avec cette bibilothèque. Lancer votre programme de test, cela crée des fichiers de trace d'exécution.*

Utiliser le programme `gcov` *sur le fichier* `hash.c`*. Si tout ce passe bien, un fichier* `hash.c.gcov` *a été créé. Ouvrez ce fichier et analysez son contenu.*

▸**Exercice 11.** *Compléter vos tests*
Utiliser la couverture pour compléter vos tests.

▸**Exercice 12.** *cmake*
En utilisant l'instruction `add_definition` *de cmake, ajouter un nouveau type de compilation (en plus des types Release et Debug) qui positionne les options de compilation nécessaires pour avoir les informations de couverture lors du lancement des tests.*