TP 6 – Prise en main de `check`

Ce TP se concentre sur la prise en main du framework de test `check`. Par la suite, dans les prochains TP de C de ce module, votre code devra systèmatiquement être testé.

Exercice 1. Installation

Nous allons commencer par installer le framework de test sur les machines. Comme nous n'avons pas les droits administrateurs, nous allons devoir passer par une phase de compilation et d'installation manuelle.

1. Créer un répertoire `usr/check/src` dans votre répertoire personnel (`mkdir -p ~/usr/check`)

2. Récupérer les sources du projet à l'adresse suivante :
`https://github.com/libcheck/check/releases` et placez les dans le répertoire que vous avez précédemment créé.

3. Désarchivez l'archive récupérée (si `.tar.gz`, utilisez la commande suivante `tar xvzf yourarchive.tar.gz`), placez vous dans le répertoire créé et lancez les commances suivantes :

```
$ ./configure --prefix="$(echo ~/usr/check/)"
$ make
$ make install
```

4. Une fois que le framework est installé, téléchargez le fichier de macro `tsuite.h` à l'adresse suivante : `https://aranega.github.io/courses`. Ce fichier va permettre de réduire le code non pertinent pour les tests. Placez ce fichier dans `~/usr/check/include`

5. Pour tester l'installation du tout, créez un fichier `justtest.c` n'importe où et tappez le code suivant :

```
#include <tsuite.h>

START_TEST (dumb_check)
{
  ck_assert_int_eq(16, 4);
}
END_TEST

TSUITE(just_testing_install, "Basic installation testing", dumb_check)

RUN_ALL(just_testing_install)
```

6. Compilez le tout avec la commande suivante :

```
gcc -o justtest justtest.c -lcheck -L ~/usr/check/lib -I ~/usr/check/include
```

7. Exécutez le binaire produit de cette façon :

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/usr/check/lib/ ./justtest
```

8. Écrivez un rapidement `Makefile` avec une cible `test` pour ne pas avoir à tapper à nouveau tout votre code pour compiler ou lancer le binaire (vous pouvez vous inspirer de celui présent là où vous avez téléchargé `tsuite.h`).

Exercice 2. Premiers tests – API de gestion de comptes, création

Maintenant que le framework de test est installé et fonctionne, nous allons écrire quelques tests en mode TDD. Nous allons donc écrire des tests, les faires échouer et écrire le code nécessaire pour que les tests passent. Pour ce faire, écrivez les tests dans un fichier dédié, le code de l'API dans un fichier `.c` et son `.h` associé. Le `main` sera proposé dans un autre fichier.

Notre code exemple sera une API très simple pour la gestion de comptes (pas palpitant, mais ça ira pour une première prise en main). L'API devra être capable de créer des comptes, d'ajouter de l'argent à un compte et d'en retirer. Chaque compte est unique et possède un numéro unique d'identification et, est associé à un propriétaire (ici, une simple chaine de caractères).

1. Écrivez le test associé à la création d'un nouveau compte.

2. Proposez une structure capable de conserver les informations d'un compte.
3. Écrivez la fonction de création d'un nouveau compte.
4. Écrivez la fonction de libération de la mémoire d'un compte.

Exercice 3. Ajout et retrait d'argent
1. Écrivez les tests relatif à l'ajout d'argent dans un compte.
2. Quels sont les cas limites ? Écrivez des tests pour les illustrer.
3. Écrivez le code de la fonction d'ajout associé.
4. Écrivez les tests relatifs au retrait d'argent dans un compte.
5. Quels sont les cas limites ? Écrivez des tests pour les illustrer.
6. Écrivez le code de la fonction de retrait associé.

Exercice 4. Conservation de comptes (s'il vous reste du temps)
Pour l'instant, les fonctions proposées ne permettent que de travailler sur un seul compte et pas sur plusieurs. Nous allons rajouter la gestion transparente d'un tableau de comptes. Notre "mémoire" des comptes créés sera un simple tableau de pointeurs vers des comptes. Pour cet exercice, nous n'allons pas nous intéresser à la fermeture de comptes, mais seulement à leurs créations et à leur recherches.
1. Créez une variable globale à votre API de type tableau de pointeurs vers comptes d'une taille de 50 éléments.
2. Modifiez le code de la fonction de création d'un compte pour que soit automatiquement ajouté le compte créé à la liste des comptes.
3. Écrivez un test pour une fonction de recherche d'un compte en fonction de son numéro.
4. Écrivez le code associé.
5. Écrivez un test pour une fonction de recherche des comptes d'un utilisateur en particulier.
6. Écrivez le code associé.
7. Écrivez une fonction générale de destruction de tout les comptes enregistrés dans la liste des comptes.