API — TP1 : Test de programmes simples

Exercice 1 : compter les 'LE' et tests «boîte noire»

Vous disposez de 9 versions d'un programme destiné à compter le nombre d'occurences de "LE" dans une séquence de caractères. Vous avez les sources de ces programmes (répertoire sources) et les exécutables (répertoire binaires). Certaines versions fonctionnent, d'autres non. Il va donc vous falloir les *tester* pour réussir à identifier les problèmes éventuels.

> *En informatique, un test désigne une procédure de vérification partielle d'un système. Le but est de trouver un nombre maximum de comportements problématiques du logiciel, car il est impossible de prouver qu'un logiciel fonctionne bien dans tous les cas. Plus le nombre d'erreurs trouvées est important, plus il y a de chances qu'il y ait davantage d'erreurs dans le composant logiciel visé. Les tests de vérification ou de validation visent à s'assurer que ce système réagit de la façon prévue par ses concepteurs (spécifications) ou est conforme aux attentes du client l'ayant commandé (besoins), respectivement. (cf. wikipedia)*

Il va donc vous falloir les tester pour réussir à identifier les erreurs éventuelles. Vous devez, dans un premier temps uniquement avec les exécutables, tester ces programmes [1]. Faites un catalogue détaillé des séquences que vous utilisez pour les tests. Testez systématiquement ces séquences pour tous les programmes. Dans un second temps, vous pouvez tenter de vous reposer sur les sources [2], même si elles utilisent des notions que vous ne connaissez pas encore.

Exercice 2 : automatisation des tests

En entreprise, et dans tout projet dont la conduite est professionnelle, les procédures de tests sont définies dans des *plans de tests*, et leur exécution est *automatisée*. Sur des projets de taille assez grande, des outils dit d'intégration continue tels que *CruiseControl* ou *Hudson/Jenkins* permettent de lancer automatiquement les tests et de vérifier le bon fonctionnement, généralement, la nuit.

Nous allons donc tenter à un très modeste niveau d'automatiser ces tests en passant par un script shell (interpréteur de commande) qui vous est fourni. Le script a besoin de savoir quel est le répertoire contenant les executables (BIN_PATH), quel est le fichier dans lequel seront imprimés les résultats (LOG_FILE), et enfin d'un fichier contenant les entrées (INPUT_FILE).

Lisez le fichier de script `test.sh`, essayez de comprendre son fonctionnement. Complétez le fichier d'entrée avec les tests identifiés lors du premier exercice, puis exécutez ce script (depuis le répertoire TP1) :

```
$ sources/test.sh
```

Exercice 3 : compilation et test «boîte blanche»

Le programme `atester.adb` vous est fourni. On compile un programme Ada en lançant gnatmake :

```
$ gnatmake atester
```

- Lisez le fichier `atester.adb` et tentez de comprendre quelles seront les entrées qui le mettront en défaut. Ces entrées étaient-elles comprises dans les tests établis aux exercices précédents ?
- Écrivez les entrées qui peuvent mettre en défaut ce programme dans votre fichier de test, et incluez le programme `atester` dans la liste des logiciels que vous testez.
- Une fois l'erreur identifiée, essayez de corriger ce programme.

1. il s'agit d'un test dit «boîte noire» : le jeu de tests est établi uniquement à partir de la spécification
2. test dit «boîte blanche», établi à partir de la spécification *et* de l'implémentation