Présentation de l'outil ALCOTEST « Another Light C prOgram TESTer »

P. Ayrault & F. Pessaux*

Le 1er mars 2005

1 Introduction

Ce fichier présente le mode d'emploi de l'outil ALCOTEST, « Another Light C prOgram TESTer ».

Il contient :

- une brève présentation des fichiers d'un exemple;
- le mode de réalisation des tests unitaires;
- enfin, la grammaire en EBNF du fichier de tests unitaires.

2 Fichiers d'un exemple

Dans le répertoire `Exemple`, vous trouverez les fichiers suivants:

- `sort.o` : le fichier objet contenant le code sous test
- `sort_tu.tu` : la description des tests unitaires
- `sort_tu.c` : le fichier résultat de l'outil Alcotest
- `sort_tu.txt` : le fichier résultat de l'exécution des tests unitaires
- `Makefile` : un petit *Makefile* pour automatiser la procédure; la variable `ALCOTEST` contient le chemin d'accès à l'exécutable de l'outil; la variable `FICHIER_A_TESTER` le nom du fichier objet à traiter

3 Procédure de réalisation des tests unitaires

1. Écriture du fichier de tests unitaires avec votre éditeur de texte favori, par exemple `emacs sort_tu.tu`
2. Génération du script C de tests unitaires : `./alcotest.x -o sort_tu.c sort_tu.tu`
3. Compilation des fichiers de tests unitaires : `gcc -Wall -o sort_tu.x sort_tu.c sort.o`
4. Exécution des tests unitaires : `./sort_tu.x >sort_tu.txt`

*Valérie Ménissier-Morain pour la documentation, l'installation et le `Makefile`

5. Analyse des résultats de tests

Les phases 2 à 4 sont automatisées par le *Makefile*.

4 Organisation du fichier de tests unitaires

Le fichier de description des tests unitaires est constitué obligatoirement de 4 parties dans cet ordre

`function`	définit le nom du composant sous test
`parameters`	définit les paramètres effectifs du composant
`glovars`	définit les variables globales utilisées par le composant
`test`	définit les valeurs des entrées et les résultats attendus pour le jeu de tests.

Il y a une section `test` par jeu de test unitaire.
Les commentaires commence par le signe `#` et se finissent à la fin de la ligne.

5 Grammaire en EBNF

⟨*FIC_TEST*⟩ ::= ⟨*FUNCTION*⟩ ⟨*PARAMETERS*⟩ ⟨*GLOBVARS*⟩ ⟨*JEU_TEST*⟩+

⟨*FUNCTION*⟩ ::= '`%function`' '`{`' ⟨*COMPOSANT*⟩ '`}`'

⟨*PARAMETERS*⟩ ::= '`%parameters`' '`{`' ⟨*DECL*⟩* '`}`'

⟨*GLOBVARS*⟩ ::= '`%globvars`' '`{`' ⟨*DECL*⟩* '`}`'

⟨*JEU_TEST*⟩ ::= '`%test`' ⟨*NUM*⟩ ⟨*CHAINE*⟩ '`{`'
'`%inputs`' '`{`' ⟨*LIAISON*⟩ '`}`'
'`%results`' '`{`' ⟨*LIAISON*⟩ '`}`'
'`}`'

⟨*DECL*⟩ ::= ⟨*FLUX*⟩ ⟨*NOM_VAR*⟩ '`:`' ⟨*TYPE*⟩ ⟨*INIT*⟩? '`;`'

⟨*FLUX*⟩ ::= '`%in`' | '`%out`' | '`%inout`'

⟨*TYPE*⟩ ::= '`INT`' | '`INT[`' ⟨*NUM*⟩ '`]`' | STRING | TUPLE

⟨*INIT*⟩ ::= '`=`' ⟨*VALEUR*⟩ // Obligatoire pour toutes les variables en `out` ou `inout`

⟨*LIAISON*⟩ ::= '`(`' ⟨*VARIABLE*⟩ '`,`' ⟨*VALEUR*⟩ '`)`'

⟨*VALEUR*⟩ ::= ⟨*NUM*⟩ | ⟨*CHAINE*⟩ | ⟨*VAL_TUPLE*⟩

⟨*VARIABLE*⟩ ::= ⟨*NOM_VAR*⟩ | ⟨*NOM_VAR*⟩ '`[`' ⟨*NUM*⟩ '`]`'

⟨*COMPOSANT*⟩ ::= nom de composant légal en C

⟨*NOM_VAR*⟩ ::= nom de variable légal en C

⟨*CHAINE*⟩ ::= '`"`' chaîne de caractères '`"`'

⟨*NUM*⟩ ::= nombre entier

⟨*VAL_TUPLE*⟩ ::= '(' ⟨*NUM*⟩ ',' ⟨*NUM*⟩ ')'