POSIX ADA

1) Introduction

Au vu de la prolifération des Unix non standard - dont Mac OS X / Darwin est un dernier avatar de plus - l'Institute of Electronic and Electical Engineers (plus connue sous IEEE) créé un groupe de travail à la des années 1980 pour développer un standard d'interface Unix : Portable Operating System Interface based on uniX (POSIX - IEEE Std 1003 - ISO 9945) dont le nom a changé par la suite en Portable Operating System Interface for Computer Environments puisque en fait Unix est une marque (crée par ATT dans les années 1970, Unix est maintenant la propriété de l'Open Group).

L'objectif du groupe POSIX est de créer un standard international basé sur les deux souches Unix principales ATT et BSD (l'autre version historique avec plus récemment Linux). Il s'agit de définir une interface de programmation standard (API) entre le code applicatif et le système d'exploitation. Pour cela, ce standard doit être indépendant des constructeurs, du système d'exploitation et de la plate-forme matérielle. De ce fait, la portabilité des applications au niveau du source peut être assurée.

Par contre, le type d'implémentation réalisée derrière n'est pas définie. La portabilité binaire n'est pas assurée par le standard.

Devant l'ampleur de la tâche plusieurs sous-groupes sont constitués. Les principaux sont le 1003.1 pour les interfaces applicatives avec 1003.1b pour les extensions temps-réel, 1003.1c pour les threads; le 1003.2 pour le langage de commande et les utilitaires. Il s'ajoute à ceux-ci le 1003.5 pour l'interface Ada.

Pourquoi donc une interface Ada, je vous le donne en mille : parce que les interfaces "standards universelles" de POSIX ont été en premier lieu écrites dans le langage naturel des unixiens : le langage C.

2) POSIX en Ada avec Florist

Pourquoi utiliser Posix en Ada ?

La question est pertinente puisque Ada nous apporte aussi une certaine portabilité au niveau source, contrairement au langage C où la même déclaration a autant de formes que d'Unix différents tout en fonctionnant ainsi plus ou moins bien d'où la standardisation apportée par POSIX.

Par contre, pour le programmeur Ada, POSIX permet d'utiliser des fonctions du système d'exploitation de plus bas niveau que celles proposées par les bibliothèques standards tout en restant "portable".

Une implémentation de POSIX Ada dans le domaine des logiciels libres est Florist. Florist est basé sur une implémentation de POSIX 1003.5: 1992, 1003.5b: 1996, et une partie de 1003.5c: 1998.

3) Construction de Florist GPL (juin 2012)

Récupérer l'archive suivante depuis le site Libre d'AdaCore "http://libre.adacore.com" à la page "Download GNAT GPL", sélectionner plateforme x86_64-linux 2012 puis Florist 2012 -> Sources :
- florist-gpl-2012-src.tgz 224 KB May 10, 2012

Avant de démarrer la compilation, GNAT et un répertoire d'installation doivent être activés, ajouter leur emplacement comme par exemple (si pas déjà fait) :

```
$ instxada=/usr/local/xadalib-2012 # ou tout autre emplacement existant
$ PATH=$instxada/bin:$PATH
$ PATH=/usr/local/gnat/bin:$PATH

$ cd
$ tar xzf ~/Desktop/florist-gpl-2012-src.tgz
$ cd florist-gpl-2012-src
# Appliquer les correctifs apportés sur Blady :
http://blady.pagesperso-orange.fr/telechargements/gnat/florist-gpl-2012.diff
$ patch -p0 < florist-gpl-2012.diff
$ ./configure CC='gcc -g -O0' --prefix=$instxada
$ Build=Debug make
$ make install
```

5) Utilisation de la bibliothèque Florist

La bibliothèque s'utilise de la manière suivante :

```
$ gnatmake -I$instxada/floristlib main -largs -lflorist
```

6) Test de la bibliothèque Florist avec pavt

Télécharger le fichier suivant sur le bureau du Mac : "pavt-1.3.tgz"
à partir du site http://www.cs.fsu.edu/~baker/florist.html.

```
$ cd ~/florist-gpl-2012-src
$ tar xzvf ~/Desktop/pavt-1.3.tgz
$ mv pavt-1.3 tests
# Appliquer les correctifs apportés sur Blady :
http://blady.pagesperso-orange.fr/telechargements/gnat/pavt-1.3.diff
$ patch -p0 < pavt-1.3.diff
$ make run_tests
```

Les résultats sont dans le fichier test.dir/run_tests_1.log. On y trouve une quantité d'erreurs du fait que l'adaptation de Florist à Darwin n'est sans doute pas parfaite.

7) Test de la bibliothèque Florist avec WPOSIX

Des programmes de test de Posix Ada sont disponible sur le site de Pascal Obry :
http://pobry.blogspot.fr/p/ada-contributions.html

Les sources de la bibliothèque wposix concernent l'API propre à Windows mais les tests sont utilisables avec Florist.

```
$ cd
$ git clone --recursive https://forge.open-do.org/anonscm/git/gnatpython/
gnatpython.git
Cloning into gnatpython...
$ cd gnatpython
$ python ./setup.py install --user
$ cd
$ git clone --recursive https://forge.open-do.org/anonscm/git/wposix/wposix.git
Cloning into wposix...
$ cd wposix/regtests
$ for i in demo*.adb ; do  gnatmake -I$instxada/floristlib -g $i -largs -lflorist; done
$ ./demo3
$ ./demo4
```

```
$ ./demo6
$ ./demo8
$ ./demo9
$ ./demo10
```

Tous devraient bien fonctionner.

Les autres tests nécessitent de modifier les .gpr dans les répertoires de "wposix/regtests/tests" en changeant auparavant <with "wposix"> par <with "florist">.

```
$ cd wposix
$ export GPR_PROJECT_PATH=$instxada/lib/gnat
$ make run_regtests
```

Des erreurs apparaissent car ces tests sont curieusement pour du Posix-Ada très orientés Windows.

Pascal Pignard, août, octobre 2002, septembre 2007, août 2012.