Comment migrer du C à l'Ada

1) Introduction

Au début, l'utilité de C2Ada s'est trouvé dans la traduction de fichiers d'en-têtes en C en unités Ada pour s'interfacer avec les bibliothèques correspondantes.
Pourtant C2Ada fait bien mieux en traduisant aussi du code complet en C en code Ada.
Une documentation est disponible sur :
http://c2ada.wiki.sourceforge.net

2) Installation

Nous devons aller chercher le site de développement de C2Ada sur Internet.
Le développement de cet utilitaire et de ses codes sources sont disponibles sur le site :
http://sourceforge.net/projects/c2ada
Il n'y a pas actuellement d'exécutable disponible pour Mac OS X.
Nous allons récupérer les sources avec SVN (voir sur Blady) et construire C2Ada.
Lancer une connection à Internet et le Terminal.

```
$ cd
$ svn co https://c2ada.svn.sourceforge.net/svnroot/c2ada c2ada
...
Checked out revision 14.
```

D'autre part, j'ai effectué quelques améliorations pour adapter l'utilitaire à GNAT sur MAC.
Les codes sources modifiés sont disponibles à l'adresse :
http://blady.pagesperso-orange.fr/alpha.html
À télécharger sur le bureau puis lancer le Terminal.

```
$ cd ~/Desktop
$ tar xzf c2ada-14.1.tgz
$ mv c2ada ~
```

Dans les deux cas, lancer la compilation de l'utilitaire avec :

```
$ cd ~/c2ada/branches/portability
$ make PYTHON_TOP=/usr PYTHON_VER=python2.3 HERE=~/c2ada/branches/
portability
```

3) Utilisation basique

C2Ada utilise le langage Python présent en standard sur Mac OS X.
Le script "setup" est écrit pour Mac OS X 10.4 / Darwin 8 en positionnant la version du langage Python à 2.3.

Pour une utilisation courante, saisir aussi les commandes suivantes :

```
$ echo 'PATH=~/c2ada/branches/portability:$PATH' >> ~/.profile
$ echo 'export PYTHONPATH=~/c2ada/branches/portability:/usr/lib/python2.3' >>
~/.profile
$ echo 'export PYTHONHOME=/usr/lib/python2.3' >> ~/.profile
$ echo 'PATH=~/c2ada/branches/portability:$PATH' >> ~/.bashrc
$ echo 'export PYTHONPATH=~/c2ada/branches/portability:/usr/lib/python2.3' >>
~/. bashrc
$ echo 'export PYTHONHOME=/usr/lib/python2.3' >> ~/. bashrc
```

Pour une utilisation temporaire, utiliser à chaque fois la commande suivante :

```
$ export PATH=~/c2ada/branches/portability:$PATH
$ export PYTHONPATH=~/c2ada/branches/portability:/usr/lib/python2.3
$ export PYTHONHOME=/usr/lib/python2.3
```

```
$ ./c2ada
Illegal invocation
Usage: ./c2ada [flags] input files
  flags:
      -Dname[=value]
            Define a macro with an optional value.  By default
                  macros will be defined with the value 1.
      -Uname
            Undefine a builtin macro.
      -Idir
            Add a search path for finding include files.
      -S
            Add a search path for system include files
      -builtin
            Display all predefined macros.
      -fun
            Flag all union declarations.
      -cs
            Add sizeof and alignof comments for all decls.
      -erc
            Always gen enum rep clauses.
      -exp
            Export functions, variables from .c file to Ada spec.
      -sih
            Suppress import declarations from included headers.(default)
      -rrc
            Always gen record rep clauses.
      -src
```

Suppress all record rep clauses.(default)

-ap
Automatic packaging. (default)

-C
Attempt to retain C comments in the translation.

-noref
No reference comments from Ada back to C.

-mwarn
Warnings about untranslated macros.(default)

-rational | -vads | **-gnat** | -icc
Rational | VADS | GNAT(default) | Irvine as target compiler.

-95
Output Ada 95 (default).

-pp
Predefined package name, default is C.

-mf
Map file cbind.map used to map unit names.

-Pfilename
project configuration file name

-Opathname
output directory (default=bindings)

```
$ ./c2ada -builtin
__FILE__
__LINE__
__ANSI_CPP__    1
_LANGUAGE_C     1
LANGUAGE_C      1
__STDC__        1
__SVR4  1
```

4) Utilisation avec Mac OS X / Darwin PPC

Traduction d'un exemple simple de programme "main.c" :

```
$ cat > main.c
#include <stdio.h>

int main (int argc, const char * argv[]) {
    // insert code here...
    printf("Hello, World with C2Ada!\n");
    return 0;
}
^D
```

Les bibliothèques standards du C sur Mac s'attendent à certaines définitions par défaut venant du compilateur. Nous les ajoutons :

```
$ ./c2ada -D__ppc__ -D__restrict= main.c
...
```

Nous créons le programme principal Ada :

```
$ cd bindings
$ python ../AdaMain.py main C run run.adb
```

Le résultat de la traduction est présent par défaut dans le répertoire bindings.
Une grande partie du travail de traduction est ainsi fait.
IL reste quelques corrections manuelles à apporter.
Certains fichiers de Darwin commencent par un souligné, ce qui n'est pas valide en Ada. Nous créons un script qui remplace les soulignés par le caractère "u" :

```
$ cat > c2ada.sh
#!/bin/tcsh
foreach i (*.ad?)
  foreach f (*-*.ads)
    set fr = $f:r
    echo Translating $i with $fr
    set fa = `echo $fr | sed s/-/./`
    sed s/$fa/`echo $fa | sed s/_/u/`/g $i >$i.tmp
    mv $i.tmp $i
  end
end
foreach f (*-*.ad?)
mv $f `echo $f | sed s/_/u/g`
end
$ chmod +x c2ada.sh
$ ./c2ada.sh
```

Nous créons les unités manquantes :

```
$ cat > sys.ads
package sys is
end sys;
$ cat > ppc.ads
package ppc is
end ppc;
```

Nous lançons la compilation Ada :

```
$ gnatmake -gnatf -I.. run
...
```

5) Correction des erreurs

Il reste quelques erreurs que nous allons corriger.
Dans le fichier sys-utypes.ads :
Remplacer pour obtenir le type correct :

```
-    DARWIN_NULL           : constant System.Address := null;
+    DARWIN_NULL           : constant System.Address := System.Null_Address;
```

Ajouter en fin de fichier une structure bidon qui n'a pas été traduite :

```
+    type struct_mcontext is
+      record
+      es : Integer;
+      end record;
+    type struct_mcontext64 is
+      record
+      es : Integer;
+      end record;
```

Dans le fichier stdio.ads :
Ajouter en fin de fichier une structure bidon qui n'a pas été traduite :

```
+    type struct_sFILEX is
+      record
+      es : Integer;
+      end record;
```

Ajouter en fin de fichier les constantes qui n'ont pas été traduites :

```
+    stdin  : constant access FILE := sF(0)'access;
+    stdout : constant access FILE := sF(1)'access;
+    stderr : constant access FILE := sF(2)'access;
```

Dans le fichier stdio.adb :
Remplacer pour obtenir le type correct :

```
+    function To_p2_func_access is new Unchecked_Conversion (System.Address,
p2_func_access);
     begin
-       return funopen(cookie, fn, 0, 0, 0);        -- /usr/include/stdio.h:364
+        return funopen(cookie, To_p2_func_access(fn), null, null, null);
```

et

```
+    function To_p3_func_access is new Unchecked_Conversion (System.Address,
p3_func_access);
     begin
-       return funopen(cookie, 0, fn, 0, 0);        -- /usr/include/stdio.h:365
+        return funopen(cookie, null, To_p3_func_access(fn), null, null);
```

Remplacer pour éviter d'appeler la fonction zopen :

```
-       return zopen(
+        return fopen(
               Tmp_p1(Tmp_p1'First)'unchecked_access,
-              Tmp_p2(Tmp_p2'First)'unchecked_access,
-             p3);
+               Tmp_p2(Tmp_p2'First)'unchecked_access);--,
+              --p3);
```

Dans le fichier main.ads :
Ajouter la fonction qui n'a pas été traduite :

```
+function main(
+              argc: C.int;
+               argv: access C.const_charp)
+             return C.int;
```

Cette fois tout devrait bien se compiler :

```
$ gnatmake -gnatf -I.. run
$ ./run
Hello, World with C2Ada!
```

Nous verrons le mois prochain comment traduire de fichiers d'en-têtes Carbon en C en unités Ada pour s'interfacer avec les bibliothèques correspondantes.

Pascal Pignard, novembre-décembre 2007.