Installation GNAT OS X 10.3

1) Installation du compilateur Gnat avec Carbon Ada et XCode

Pour pouvoir utiliser GNAT, il est indispensable d'avoir installé le compilateur version gcc 3.3 avec les derniers outils de développement XCode pour Mac OS X 10.3.
XCode est disponible avec les CDs Mac OS X 10.3 ou sur le site Apple : "http://connect.apple.com".
(voir en page *À Savoir sur Blady* pour l'installation Mac OS X).

Télécharger les fichiers suivants :
. Wavefront 10.3 : "gcc-3.3-gnat_installer-20040913.dmg",
. XCode/Interface Builder : "Xcode Bindings_wf.dmg",
. gdb for 10.3 : "gnatgdb-292.sitx",
sur le site "http://www.macada.org" aux pages "Compilers", "Bindings" et "Debuggers".

Le compilateur :
Si un compilateur gnat est déjà installé avec gcc 3.1 ou 3.3, lancer un script shell de préparation et de nettoyage :

```
$ cd /Volumes/GNAT\ for\ MacOS\ X\ 10.3\ Installer
$ sudo ./uninstall
```

Ouvrir "gcc-3.3-gnat_installer-20040913.dmg" sous le Finder, l'image disque est extraite et montée par Stuffit Expander. Lancer l'installation du compilateur contenu dans "gnat_for_macosx.pkg".

Le compilateur s'installe dans les répertoires :
/usr/bin
/usr/include/gcc/darwin/3.3
/usr/lib/gcc/darwin/3.3
/usr/libexec/gcc/darwin/ppc/3.3

Le dévermineur :
Ouvrir "gnatgdb-292.sit", placer le contenu de l'archive sur votre bureau.

```
$ sudo install -c -m 775 ~/Desktop/gnatgdb-292 /usr/bin/gdb
```

Le dévermineur s'installe dans le répertoire /usr/bin.

La bibliothèque Carbon Ada et le support de XCode :
Ouvrir "Xcode Bindings_wf.dmg", sous le Finder, l'image disque est extraite et montée par Stuffit Expander. Lancer l'installation des bibliothèques contenues dans "Carbon_Bindings_xcode.pkg".

Un script et les outils reformat et makebody s'installent dans le répertoire :
/usr/local/bin

La bibliothèque "Carbon Ada" s'installe dans le répertoire :
/usr/local/Bindings/Frameworks

La bibliothèque "ncurses" (fenêtrage avec le Terminal) s'installe en même temps dans le répertoire :
/usr/local/Bindings/ncurses

La bibliothèque "OpenGL" (Graphisme en 3D) s'installe en même temps dans le répertoire :
/usr/local/Bindings/AdaGL

Des exemples de programmes s'installent aussi en même temps dans les répertoires :
/Developer/Examples/Ada/Carbon
/Developer/Examples/Ada/Command_Line
/Developer/Examples/Ada/OpenGl

Les liens des bibliothèques (Framework) pour la recherche d'API s'installent également en même temps dans le répertoire :
/Developer/Extras

Les modèles de projets et les scripts pour XCode s'installent également en même temps dans le répertoire :
/Library/Application Support/Apple/Developer Tools

Les fichiers spécifications .ads n'ont pas toujours besoin d'être compilés. Ils le sont à travers les fichiers de code .adb avec Gnat. Aussi il y a lieu de prévenir XCode de ceci avec le patch suivant :

```
$ cd /Developer/Makefiles/pbx_jamfiles
$ sudo cp -p ProjectBuilderJambase ProjectBuilderJambase-apple
$ sudo patch -p0 ProjectBuilderJambase < /Volumes/Xcode\ Bindings/
jambase_patch
```

2) Utilisation avec le Terminal

La commande "gnatmake" seule, sans paramètre, donne justement la liste des paramètres possibles. Néanmoins, la simple commande suivante donnera de bons résultats :

$ gnatmake hello.adb

Le fichier hello.adb étant :

```
with Text_IO; use Text_IO;
procedure Hello is
begin
put_line("Hello again, avec Ada.");
end;
```

Et les résultats ne se font pas attendre :

```
[localhost:~/Programmation/essais] pascal$ gnatmake hello.adb
gcc -c hello.adb
gnatbind -x hello.ali
gnatlink hello.ali
[localhost:~/Programmation/essais] pascal$ ./hello
Hello again, avec Ada.
```

3) les commandes utiles avec le Terminal

La liste des commandes est obtenue de la façon suivante :

$ gnat
GNAT 3.3 20030304 (Apple Computer, Inc. build 1495) Copyright 1996-2002 Free Software Foundation, Inc.

List of available commands

GNAT BIND gnatbind
réalise l'édition des liens Ada
GNAT CHOP gnatchop
découpe un fichier en unités utilisables par Gnat
GNAT COMPILE gnatmake -f -u -c
compile une entité Ada
GNAT ELIM gnatelim
détecte et élimine les sous-programmes inutilisés (non présent avec gnat-osx)
GNAT FIND gnatfind
liste toutes les utilisations d'une entité Ada
GNAT KRUNCH gnatkr
réduit les noms de fichiers au nombre de lettres spécifiés
GNAT LINK gnatlink
réalise l'édition des liens de l'exécutable
GNAT LIST gnatls
liste le contenu des objets générés
GNAT MAKE gnatmake
utilitaire optimisé de compilation multi-unités
GNAT NAME gnatname
???
GNAT PREPROCESS gnatprep
pré-processeur externe
GNAT STANDARD gnatpsta
affiche le package "Standard"
GNAT STUB gnatstub
non présent avec gnat-osx
GNAT XREF gnatxref
utilitaire d'édition des références croisées

De même chacune des commandes exécutée sans argument affichera justement la liste des arguments possibles.

$ gnatmake (extrait)

```
Usage: gnatmake  opts  name  {[-cargs opts] [-bargs opts] [-largs opts]}
  name  is a file name from which you can omit the .adb or .ads suffix
gnatmake switches:
  -a       Consider all files, even readonly ali files
  -b       Bind only
  -c       Compile only
  -f       Force recompilations of non predefined units
  -k       Keep going after compilation errors
  -l       Link only
  -M       List object file dependences for Makefile
  -s       Recompile if compiler switches have changed
  -u       Unique compilation. Only compile the given file.
  -v       Display reasons for all (re)compilations
  -z       No main subprogram (zero main)
To pass an arbitrary switch to the Compiler, Binder or Linker:
  -cargs opts   opts are passed to the compiler
  -bargs opts   opts are passed to the binder
  -largs opts   opts are passed to the linker
Compiler switches (passed to the compiler by gnatmake):
  -g            Generate debugging information
  -Idir         Specify source files search path
  -O[0123]      Control the optimization level
  -gnatf    Full errors. Verbose details, all undefined references
  -gnatv    Verbose mode. Full error output with source lines to stdout
  -gnat83   Enforce Ada 83 restrictions
```

Et aussi avec gcc :

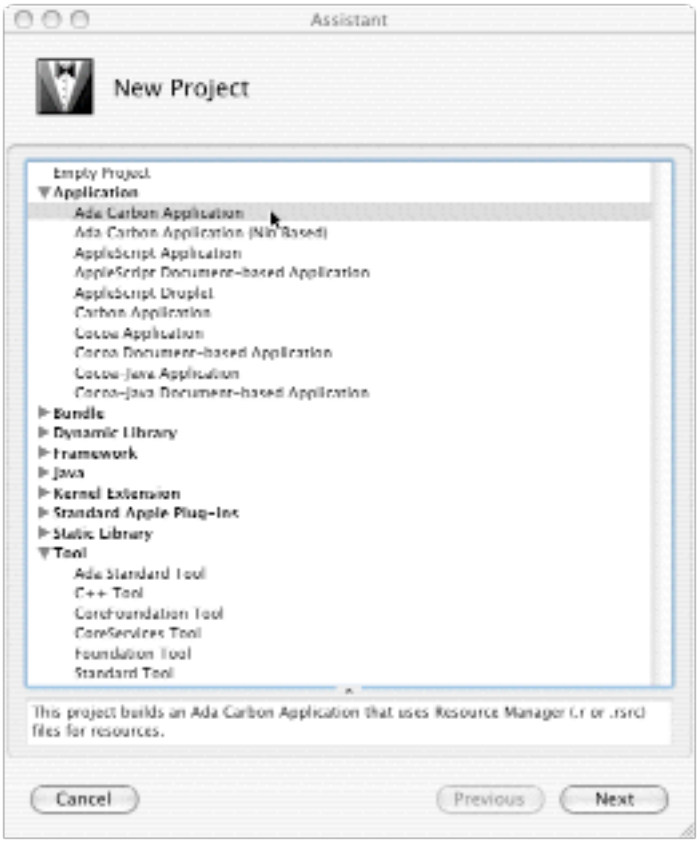
$ gcc --help

...

4) Utilisation des modèles de projets avec XCode

Lancer XCode et choisir "New Project..." dans le menu "File". La fenêtre "New Project" s'affiche avec le choix entre trois projets Ada :

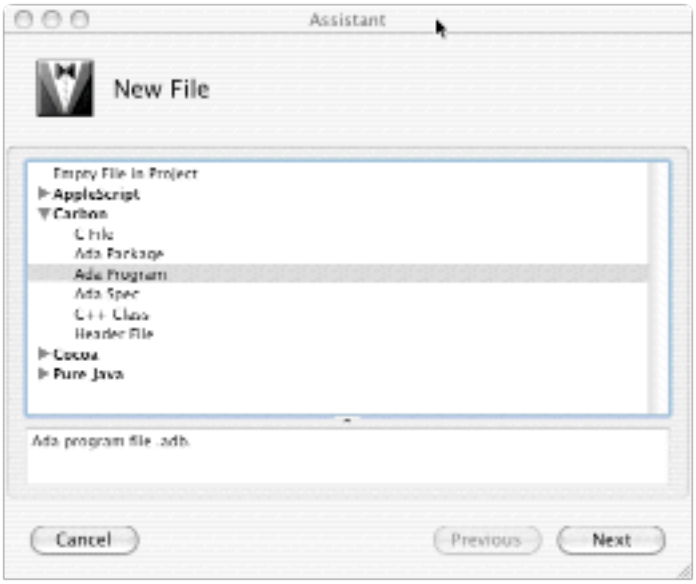
- Ada Carbon Application (Nib Based) : permet de construire des applications Carbon avec une interface utilisateur construite avec InterfaceBuilder.
- Ada Carbon Application : permet de construire des applications Carbon avec une interface utilisateur construite à base de ressources "rcrs",
- Ada Standard Tool : permet de construire des programmes utilisables sous Terminal.



Les applications se construisent alors de manière classique avec XCode.

L'ajout de fichiers s'opère en choisissant "New File..." dans le menu "File". La fenêtre "New File" s'affiche avec le choix entre trois fichiers Ada :

- Ada Package : permet de construire le corps d'une unité Ada,
- Ada Program : permet de construire la procédure principale d'un programme Ada.
- Ada Spec : permet de construire la spécification d'une unité Ada,



5) Utilisation des scripts avec XCode

Les scripts suivants sont disponibles dans le menu script de XCode :

1. Reformat Ada - remplace le contenu d'un fichier par le code source formarté. Pour modifier les options de formatage, éditez le fichier /Library/Application Support/Apple/Developer Tools/Scripts/10-User Scripts/70-Ada/20-ada_reformat.sh.
Les options sont les listées dans le fichier.
Pour les simples besoins de clarification du code source j'utilise la syntaxe suivante : -id Mixed_No_Force_Lower -rw Lower_Case -i 2.

2 Comment - Commente le texte sélectionné,

3. Uncomment - Enlève les commentaires du texte sélectionné,

4. Sweep CVS - nettoie les traces CVS dans les répertoires du projet courant,

5. Makebody - génère le corps d'une spécification dans le presse papier.

6) Convention des noms de fichiers avec GNAT

'main.ads'
Main (spec), contient la spécification d'une procédure ou d'une fonction, utile pour l'appeler d'une autre unité.

'main.adb'
Main (body), contient le code de la procédure ou de la fonction principale du programme.

'arith_functions.ads'
Arith_Functions (package ou proc ou func spec), contient les spécifications d'une unité Ada.

'arith_functions.adb'
Arith_Functions (package ou proc ou func body), contient le code d'une unité Ada.

'func-spec.ads'
Func.Spec (child package spec), contient les spécifications d'une unité fille Ada.

'func-spec.adb'
Func.Spec (child package body), contient le code d'une unité fille Ada.

'main-sub.adb'
Sub (subunit of Main separate), contient le code d'une unité Ada définie comme séparée.

Attention, par défaut, le nom du fichier doit être identique à celui de l'unité définie dans le source. Ne pas mettre d'espaces ou de tirets.

7) Les options utiles

La commande de compilation issue d'une commande en ligne ou de l'utilitaire de compilation gnatmake est en standard : **gcc -c essai.adb** (respectivement le compilateur, l'option de compilation et un source Ada).

Gnat comporte quelques options qui nous viennent en aide suivant les différentes phases de développement de notre programme :

- mise au point source (maximum d'information à la compilation),
- mise au point (maximum d'information à l'exécution),
- inspection du code et des données, pour la livraison de l'exécutable (maximum de performance à l'exécution).

Pour avoir l'ensemble des informations sur le code source lors des premières compilations, j'utilise les options : **-gnatc -gnatf -gnatwa**
(respectivement pas de génération de code, l'ensemble des erreurs et l'ensemble des avertissements) ou plus simplement -gnatcfwa.
A ce stade la génération du code exécutable n'est pas nécessaire.

Pour obtenir le listing du code source avec les erreurs, on peut utiliser **-gnatv** ou **-gnatl** pour le listing complet, mais aujourd'hui avec les environnements de développement intégré comme Xcode ou jGrasp se n'est pas nécessaire.
Pour afficher la liste de l'ensemble des unités requises pour la compilation, on peut utiliser l'option **-gnatu**.

Lorsque tout le code source compile, pour obtenir l'ensemble des informations de déverminage, j'utilise les options : **-fstack-check -g -gnato -gnatVa -gnatE**
(respectivement le contrôle de la pile, la génération des information pour le déverminateur, le contrôle des dépassements numériques, le contrôle des validités des valeurs avant l'utilisation, le contrôle dynamique des élaborations de code).
A ce stade la génération d'un exécutable optimisé n'est pas nécessaire.

Pour activer les pragmas Assert et Debug, on peut utiliser l'option **-gnata**.
Pour obtenir les instructions assembleurs générées, on peut utiliser les options : **-fverbose-asm -S**.
Pour obtenir des informations détaillées sur la structure des données on peut utiliser l'option **-gnatR2s**.

Lorsque tout fonctionne, pour obtenir un exécutable optimisé, j'utilise les options : **-gnatp -gnatN -O2**
(respectivement la suppression de tous les contrôles, la mise du code en ligne pour les procédures désignées par le pragma inline, optimisation étendue de gcc).

Pour aller plus loin dans l'optimisation, on peut utiliser l'option **-O3** qui produit en plus la mise de toutes les procédures en ligne. Ce qui n'est pas forcément intéressant sur des procédures avec beaucoup de code.

Bien sûr, il est possible d'obtenir tout d'un coup avec les options : -gnatEfNowa -gnatVa -fstack-check -g -O2, au risque de mélanger la nature des problèmes qui vont surgir et donc à rendre plus complexe leur résolution.
D'une part, il est aussi étrange de vouloir faire des vérifications tout en optimisant le code…
On peut prendre alors les options -gnatfNpwa -O2 au dépend de la sécurité.

D'autre part, les aussi, les environnements de développement intégré comme XCode permette de définir facilement plusieurs cibles avec des options différentes.

8) Utilisation de bibliothèques par défaut

Par défaut GNAT ne connaît l'emplacement que de ses unités de compilation situées dans les répertoires 'adainclude' et 'adalib'. Lors de l'utilisation répétée d'unités de compilation Ada, il est pratique d'indiquer au compilateur le chemin de ces unités une fois pour toute et non pas à chaque compilation.
Cette indication est réalisée avec GNAT au moyen de deux fichiers de configuration : 'ada_source_path' pour les codes sources (.ads essentiellement) et 'ada_object_path' pour les codes objets (.ali et .o).
Ces fichiers doivent contenir une ligne de texte pour chaque emplacement des unités. Les répertoires du code source et du code objet peuvent être identiques, il faut cependant le préciser de façon identique dans les deux fichiers.
Les répertoires par défaut de GNAT doivent être aussi obligatoirement mentionnés : 'adainclude' dans 'ada_source_path' et 'adalib' dans 'ada_object_path'.
Ces deux fichiers de configurations sont placés au même endroit que le fichier de spécification du compilateur. Cette information est obtenue avec la commande 'gcc –v'.

Pascal Pignard, janvier - mars, septembre 2004, août 2005, janvier 2006.