Installer ou construire GTKAda Quartz et autres composants en Ada sur macOS X avec la bibliothèque XNAdaLib

GTKAda est la boite à outil graphique en Ada 95 basée sur GTK+ pour construire des applications portables sur la plupart des plateformes.

GTK+ est une bibliothèque graphique Linux conçue à l'origine pour fonctionner sur les systèmes Unix avec X-Windows... mais les développeurs GTK-OSX ont inclus une interface de macOS utilisant le moteur graphique Quartz. Un programme peut donc être conçu pour tourner sur macOS X sans activer le sous-système X11.

Site www.gtk.org/download/macos.php.

Nous pouvons soit construire GTKAda à partir des sources (voir §2 et suivants) soit le prendre prêt à l'emploi sur Source Forge (voir §1).

GTKAda est inclue dans la bibliothèque XNAdaLib qui comprend également le constructeur graphique d'application Glade ainsi que d'autres composants en Ada : Gate3, GNATColl, Florist, Simple Components, AICWL, AdaCurses, Zanyblue, PragmARC, Gnoga, SparForte et Alire.

Nous allons voir comme installer la bibliothèque XNAdaLib prêtre à l'emploi et aussi pouvoir la construire à partir des sources de ses composants.

Configuration : macOS 10.15.7, Xcode 12.0, GNAT Community 20200818.

Sommaire

1.	Installer GTKAda Quartz avec la bibliothèque XNAdaLib	2
2.	Construire GTK+ Quartz pour la bibliothèque XNAdaLib	3
3.	Construire GTKAda (sans OpenGL)	6
4.	Construire GLADE	7
5.	Construire Simple Components et AICWL	8
6.	Construire XMLAda (Schema, DOM, SAX, Unicode)	10
7.	Construire Template-Parser	11
8.	Construire GPRBuild / GPR	11
9.	Construire GNATColl (Core, Bindings et DB)	12
10.	Construire Florist	14
11.	Construire Gate3	14
12.	Construire AdaCurses	15
13.	Construire Zanyblue	16
14.	Construire PragmARC	17
15.	Construire Gnoga	17
16.	Construire SparForte	19
17.	Construire Alire	20

1. Installer GTKAda Quartz avec la bibliothèque XNAdaLib

Télécharger le fichier suivant sur le bureau du Mac :
xnadalib-gpl-2020-quartz-x86_64-apple-darwin17.7.0-bin.tgz,
depuis le site de Source Forge "sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2020-catalina/".

Lancer le Terminal dans un compte administrateur et taper les commandes suivantes :

```
% instbase=/usr/local # chemin d'installation obligatoire pour les bibliothèques dynamiques
% cd $instbase
% sudo tar xzf ~/Desktop/xnadalib-gpl-2020-quartz-x86_64-apple-darwin17.7.0-bin.tgz
```

GTK+, GTKAda, Glade, Gate3, GNATColl, Florist, Simple Components, AICWL, AdaCurses, Zanyblue, PragmARC, Gnoga, SparForte, Alire s'installent à partir du dossier : $instbase/xnadalib-2020.

Pour une utilisation courante, saisir aussi les commandes suivantes :

```
% instxada=$instbase/xnadalib-2020
% echo 'PATH=$instxada/bin:$PATH' >> ~/.profile
% echo 'PATH=$instxada/bin:$PATH' >> ~/.bashrc
% echo 'export MANPATH=$instxada/man:$instxada/share/man:$MANPATH' >> ~/.profile
% echo 'export MANPATH=$instxada/man:$instxada/share/man:$MANPATH' >> ~/.bashrc
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:
$GPR_PROJECT_PATH' >> ~/.profile
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:
$GPR_PROJECT_PATH' >> ~/.bashrc
% echo 'export XDG_DATA_DIRS=$instxada/share' >> ~/.profile
% echo 'export XDG_DATA_DIRS=$instxada/share' >> ~/.bashrc
```

Pour une utilisation temporaire, utiliser à chaque fois les commandes suivantes :

```
% instxada=$instbase/xnadalib-2020
% PATH=$instxada/bin:$PATH
% export MANPATH=$instxada/man:$instxada/share/man:$MANPATH
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:$GPR_PROJECT_PATH
% export XDG_DATA_DIRS=$instxada/share
```

Des exemples de programme Ada sont disponibles dans le dossier $instxada/share/examples/gtkada.
Une documentation au format HTML est disponible dans les dossiers
$instxada/share/doc/gtkada et $instxada/share/gtk-doc/html :

```
% open $instxada/share/doc/gtkada/gtkada_rm/index.html
% open $instxada/share/doc/gtkada/gtkada_ug/index.html
% open $instxada/share/gtk-doc/html/gtk3/index.html
```

Voir l'utilisation de GTKAda avec des exemples sur Blady :
blady.pagesperso-orange.fr/a_savoir.html#gtkada
ainsi que l'utilisation de Gnoga :
blady.pagesperso-orange.fr/a_savoir.html#gnoga

2. Construire GTK+ Quartz pour la bibliothèque XNAdaLib

Gtk+ est une bibliothèque graphique en C pour X-Window et Win32. Elle fut développée initialement pour Gimp. Nous allons construire la version comportant le rendu natif macOS directement avec Quartz.

Site web : www.gtk.org.

La version installée est 3.24.20.

Récupérer sur le bureau le script d'installation gtk-osx-setup.sh à l'adresse :
gitlab.gnome.org/GNOME/gtk-osx/raw/master/gtk-osx-setup.sh
Source wiki.gnome.org/Projects/GTK+/OSX/Building.

Et aussi xnadalib-2020-diff.tgz sur blady.pagesperso-orange.fr/telechargements/gtkada/xnadalib-2020-diff.tgz.

Saisir les commandes suivantes dans le Terminal d'une session administrateur tout en étant connecté à Internet :

```
% instbase=/usr/local # obligatoire pour les bibliothèques dynamiques
% cd $instbase
% sudo mkdir src-2020
% sudo chown $USER src-2020
% cd src-2020
% xnadasrc=$PWD
% export DEVROOT=$xnadasrc
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CONFIG_HOME=$xnadasrc/config
% export XDG_CACHE_HOME=$xnadasrc/cache
% sh ~/Desktop/gtk-osx-setup.sh
...
PATH does not contain .../src-2020/.new_local/bin. You probably want to fix that.
Warning: Python 3.8 was not found on your system… # (le cas échéant)
Would you like us to install CPython 3.8.5 with pyenv? [Y/n]: Y
...
```

Cette commande installe jhbuild dans "Source" et créé ".new_local/bin/jhbuild". Il installe aussi "config/jhbuildrc" et "config/jhbuildrc-custom" et copie les modules gtk-osx courants dans "Source/jhbuild/modulesets". (Si ces fichiers sont déjà présents, je recommande de les supprimer avant de lancer la commande ci-dessus.)

Comme suggérer nous allons ajouter l'accès demandé dans notre environnement :

```
% echo 'PATH=$xnadasrc/.new_local/bin:$PATH' >> ~/.profile
% echo 'PATH=$xnadasrc/.new_local/bin:$PATH' >> ~/.bashrc
```

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

```
% PATH=$xnadasrc/.new_local/bin:$PATH
```

Pour toute la construction de GTK+ nous utiliserons le compilateur natif du Mac, adapter la variable PATH en conséquence :

```
% which gcc
/usr/bin/gcc
```

Saisir les commandes suivantes tout en étant connecté à Internet :

```
# Appliquer les correctifs apportés sur Blady :
% cd ~/Desktop
% tar xzf ~/Desktop/xnadalib-2020-diff.tgz
% cd $xnadasrc/config
% patch -p0 < ~/Desktop/xnadalib-2020-diff/jhbuildrc-custom.diff
% cd $instbase
% sudo mkdir xnadalib-2020
% sudo chown $USER xnadalib-2020
% instxada=$PWD/xnadalib-2020
% export XML_CATALOG_FILES=$instxada/share/xml/catalog
% jhbuild bootstrap-gtk-osx
...
# Lors de la première utilisation de Java, une fenêtre système surgit pour donner à Java l'accès au dossier Documents, cliquez sur OK

# Python fourni avec macOS n'est pas installé avec le SDK nous installons le notre
% jhbuild build python
...
% jhbuild build meta-gtk-osx-bootstrap
...
% jhbuild build meta-gtk-osx-gtk3
...
```

Liste des modules installés :

- adwaita-icon-theme-3.36.1
- atk-2.36.0
- autoconf-2.69
- autoconf-archive-2020.01.06
- automake-1.16.2
- bison-3.6
- cairo-1.16.0
- cmake-3.17.2
- flex-2.6.0
- fontconfig-2.13.1
- freetype-2.10.2
- fribidi-1.0.9
- gdk-pixbuf-2.40.0
- gettext-0.20.2
- glib-2.64.2
- gobject-introspection-1.64.1
- gtk-doc-1.32
- gtk-mac-integration-2.1.3
- gtk-osx-docbook-1.2
- gtk+-3.24.20
- harfbuzz-2.7.1
- hicolor-icon-theme-0.15
- intltool-0.51.0
- itstool-2.0.6
- jpeg-9d

- libcroco-0.6.13
- libepoxy-1.5.4
- libffi-3.3
- libpng-1.6.37
- librsvg-2.40.17
- libtool-2.4.6
- libxml2-2.9.9
- libxslt-1.1.34
- make-4.2.1
- pango-1.44.7
- pixman-0.40.0
- pkg-config-0.29.2
- Python-2.7.17
- readline-8.0
- tiff-4.1.0
- util-macros-1.19.2
- xz-5.2.5
- zlib-1.2.11

Et surtout, des programme de test et démo donnent plein de situations d'emploi des objets GTK avec leur description, le code source C correspondant et une démonstration du résultat :

```
% jhbuild shell
% gtk3-demo
```

3. Construire GTKAda (sans OpenGL)

GTKAda est la boite à outil graphique en Ada basée sur GTK+ pour construire des applications portables sur la plupart des plateformes.

Site Web : www.adacore.com/gtkada

La version installée est 20.2.

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

```
# Activation du compilateur Ada GNAT
% PATH=/usr/local/gnat/bin:$PATH
% instxada=/usr/local/xnadalib-2020
# Activation de la bibliothèque GTK-OSX
% PATH=$instxada/bin:$PATH
```

Récupérer ensuite sur le bureau le fichier Makefile à partir du site :
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/Makefile
et
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/patches/gtkada-patch.txt

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% cp -p ~/Desktop/Makefile .
% mkdir -p patches
% cp -p ~/Desktop/gtkada-patch.txt patches
% make adacore-version=20.2 rm-git-repo= os=darwin prefix=$instxada gtkada-options=--enable-build=Debug gtkada
# Utilisation de Sphinx et LaTeX avec MacPorts
% PATH=/opt/local/bin:$PATH make -C gtkada-build docs
% make adacore-version=20.2 os=darwin sudo= gtkada-install
% cp -p gtkada-build/po/build_skeleton.pl $instxada/bin
```

La bibliothèque GTKAda s'est installée dans le dossier $instxada.

Pour une utilisation courante, nous positionnons les variables d'environnement PATH pour une utilisation en ligne de commande et GPR_PROJECT_PATH pour une utilisation avec un projet GPS :

```
% echo 'PATH=$instxada/bin:$PATH' >> ~/.profile
% echo 'PATH=$instxada/bin:$PATH' >> ~/.bashrc
% echo 'export MANPATH=$instxada/man:$MANPATH' >> ~/.profile
% echo 'export MANPATH=$instxada/man:$MANPATH' >> ~/.bashrc
% echo 'export MANPATH=$instxada/share/man:$MANPATH' >> ~/.profile
% echo 'export MANPATH=$instxada/share/man:$MANPATH' >> ~/.bashrc
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:$GPR_PROJECT_PATH' >> ~/.profile
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:$GPR_PROJECT_PATH' >> ~/.bashrc
```

Pour une utilisation temporaire, utiliser à chaque fois les commandes suivantes :

```
% PATH=$instxada/bin:$PATH
% export MANPATH=$instxada/man:$MANPATH
% export MANPATH=$instxada/share/man:$MANPATH
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:$GPR_PROJECT_PATH
```

Une démo tout en Ada est présente dans le dossiers "$instxada/share/examples/gtkada" :

```
% export XDG_DATA_DIRS=$instxada/share
% cd $instxada/share/examples/gtkada/testgtk
% ./testgtk
```

Une documentation au format PDF et HTML sous forme d'un manuel utilisateur et d'un manuel de référence est disponible dans le dossier "$instxada/share/doc/gtkada".

```
% open $instxada/share/doc/gtkada/gtkada_ug/GtkAda.pdf
% open $instxada/share/doc/gtkada/gtkada_ug/index.html
% open $instxada/share/doc/gtkada/gtkada_rm/index.html
```

Voir l'utilisation de GTKAda avec des exemples sur Blady :
blady.pagesperso-orange.fr/a_savoir.html#gtkada

4. Construire GLADE

Glade est un outil graphique de développement d'interfaces utilisateurs pour la bibliothèque GTK. Les fichiers XML produits par Glade peuvent être utilisés par de nombreux langage de programmation comme C, C++, C#, Vala, Java, Perl, Python et ... Ada ;-).
Site web : glade.gnome.org.
La version installée est 3.22.1.

Saisir les commandes suivantes dans le Terminal :

```
% xnadasrc=/usr/local/src-2020
% PATH=$xnadasrc/.new_local/bin:$PATH
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CACHE_HOME=$xnadasrc/cache
% export XDG_CONFIG_HOME=$xnadasrc/config
% instxada=/usr/local/xnadalib-2020
% export XML_CATALOG_FILES=$instxada/share/xml/catalog
% jhbuild build glade
```

Ne pas oublier de configurer LANG (pour l'affichage français) et XDG_DATA_DIRS avant de lancer Glade :

```
% export XDG_DATA_DIRS=$instxada/share
% export LANG=fr_FR.UTF-8
% $instxada/bin/glade
```

Voir sur Blady pour les premiers pas avec un exemple :
blady.pagesperso-orange.fr/a_savoir.html#gtkada

5. Construire Simple Components et AICWL

Simple Components et AICWL proposés par Dmitry Kazakov contiennent les composants :

- Simple components : graphes, ensembles, piles, vecteurs, analyseurs d'expressions, primitives de synchronisation, nombre pseudo-aléatoires...
- Strings edit : mise à l'échelle des axes, nombre romains, entiers et réels, codage UTF-8 et Unicode, recherche avec jokers...
- Tables : container de données avec recherche par chaines de caractères,
- AICWL (Ada industrial control widget library) : collection de widgets de visualisation type compteur de vitesse et vue mètre, d'horloges, d'oscillogramme, éditeur de widget...
- GtkAda contributions : multi-tâche, vue arborescente, navigation de fichiers, image en code source Ada, fichier de ressources graphiques, modèle de couleur HSL, des boutons, exécution de processus asynchrones...

Site web : www.dmitry-kazakov.de.

a) Construire GTKSourceView

Il s'agit d'un widget qui étend GtkTextView avec les fonctionnalités d'un éditeur de code source comme par exemple la coloration syntaxique.

Site web : wiki.gnome.org/Projects/GtkSourceView.

La version installée est 3.24.4.

Saisir les commandes suivantes dans le Terminal :

```
% xnadasrc=/usr/local/src-2020
% PATH=$xnadasrc/.new_local/bin:$PATH
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CONFIG_HOME=$xnadasrc/config
% export XDG_CACHE_HOME=$xnadasrc/cache
% jhbuild build gtksourceview3
```

b) Construire GNUTLS

Il s'agit d'une bibliothèque qui contient des API pour les protocoles SSL, TLS et DTLS.

Site web : www.gnutls.org.

La version installée est 3.6.14.

Saisir la commande suivante dans le Terminal :

```
% xnadasrc=/usr/local/src-2020
% PATH=$xnadasrc/.new_local/bin:$PATH
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CONFIG_HOME=$xnadasrc/config
% export XDG_CACHE_HOME=$xnadasrc/cache
% jhbuild build gnutls
```

c) Construire Simple Components

Site web : www.dmitry-kazakov.de/ada/components.htm.
Version installée : 4.50.
Récupérer l'archive suivante sur le bureau :
sourceforge.net/projects/simplecomponentsforada/files/releases/components_4_50.tgz/download

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=$instxada/bin:$PATH
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% mkdir components_4_50
% cd components_4_50
% tar xzf ~/Desktop/components_4_50.tgz
# Récupérer les projets GPR apportés sur Blady
% unzip ~/Desktop/xnadalib-2020-diff/components_gpr.zip
% gprbuild -p -P lib_components.gpr -XSC_OS=OSX


# Certains sources provoquant une erreur avec gnatdoc sont ignorés
% gnatdoc --enable-build -P lib_components.gpr -XSC_OS=OSX
% gprinstall -f -p --prefix=$instxada -P lib_components.gpr -XSC_OS=OSX
```

Une documentation au format HTML est disponible dans le dossier $instxada/share/doc/components :

```
% open $instxada/share/doc/components/strings_edit.htm
% open $instxada/share/doc/components/tables.htm
% open $instxada/share/doc/components/components_rm/index.html
```

Des exemples de programme sont disponibles dans le dossier $instxada/share/examples/components.

d) Construire AICWL

Site web : www.dmitry-kazakov.de/ada/aicwl.htm.
Version installée : 3.24.
Récupérer l'archive suivante sur le bureau :
sourceforge.net/projects/aicwl/files/releases/aicwl_3_24.tgz/download

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=$instxada/bin:$PATH
```

```
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% mkdir aicwl_3_24
% cd aicwl_3_24
% tar xzf ~/Desktop/aicwl_3_24.tgz
# Récupérer les projets GPR apportés sur Blady
% unzip ~/Desktop/xnadalib-2020-diff/aicwl_gpr.zip
% gprbuild -p -P lib_aicwl.gpr -XSC_OS=OSX
% gnatdoc --enable-build --no-subprojects -P lib_aicwl.gpr -XSC_OS=OSX
% gprinstall -f -p --prefix=$instxada -P lib_aicwl.gpr -XSC_OS=OSX
% gprbuild -p -P xpm2gtkada/build_xpm2gtkada.gpr -XSC_OS=OSX
% gprinstall -f -p --prefix=$instxada -P xpm2gtkada/build_xpm2gtkada.gpr -XSC_OS=OSX
```

Une documentation au format HTML est disponible dans le dossier $instxada/share/doc/aicwl :

```
% open $instxada/share/doc/aicwl/aicwl.htm
% open $instxada/share/doc/aicwl/gtkada_contributions.htm
% open $instxada/share/doc/aicwl/aicwl_rm/index.html
```

Des exemples de programme sont disponibles dans le dossier $instxada/share/examples/aicwl.

Ne pas oublier de configurer XDG_DATA_DIRS avant de lancer les exemples :

```
% cd $instxada/share/examples/aicwl
% gprbuild -p -P build_examples.gpr -XSC_OS=OSX
% export XDG_DATA_DIRS=$instxada/share
% ./bin/oscilloscope_plotter
...
```

6. Construire XMLAda (Schema, DOM, SAX, Unicode)

XMLAda est une boite à outil pour analyser les fichiers XML.
Site web : github.com/AdaCore/xmlada.
La version installée est 20.2.
Dépendance : sans.

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
```

Récupérer ensuite sur le bureau le fichier Makefile à partir du site :
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/Makefile

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% cp -p ~/Desktop/Makefile .
% make adacore-version=20.2 rm-git-repo= os=darwin prefix=$instxada xmlada-options=--enable-build=Debug GPRBUILD_OPTIONS="-gnatwn -gnatU" xmlada
# Utilisation de Sphinx et LaTeX avec MacPorts
% PATH=/opt/local/bin:$PATH make -C xmlada-build docs
% make adacore-version=20.2 os=darwin sudo= xmlada-install
```

7. Construire Template-Parser

Il s'agit d'un composant qui remplace des zones de textes balisées dans des modèles.
Site Web : github.com/AdaCore/templates-parser
La version installée est 20.2.
Dépendance : XMLAda (optionnel).

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
```

Récupérer ensuite sur le bureau le fichier Makefile à partir du site :
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/Makefile

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% cp -p ~/Desktop/Makefile .
% make adacore-version=20.2 rm-git-repo= os=darwin prefix=$instxada templates-parser-options=DEBUG=true GPROPTS="-gnatwn -gnatU" templates-parser
# Utilisation de Sphinx et LaTeX avec MacPorts
% PATH=/opt/local/bin:$PATH make -C templates-parser-build build-doc
% make adacore-version=20.2 os=darwin sudo= templates-parser-install
```

8. Construire GPRBuild / GPR

GPRBuild est le moteur du construction d'un programme à travers ses sources en enchainant les étapes de compilation, l'édition des liens Ada, l'édition des liens du binaires. Un programme complexe peut être décrit à travers un fichier projet GPR. La bibliothèque de manipulation des fichiers GPR est aussi installée avec GPRBuild.
Site web : github.com/AdaCore/gprbuild.
La version installée est 20.2.
Dépendance : XMLAda.

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
% export GPR_PROJECT_PATH=$instxada/share/gpr
```

Récupérer ensuite sur le bureau le fichier Makefile à partir du site :
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/Makefile

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% cp -p ~/Desktop/Makefile .
% make adacore-version=20.2 rm-git-repo= os=darwin prefix=$instxada gprbuild-options=BUILD=debug GPRBUILD_OPTIONS="-gnatwn -gnatU" gprbuild
# Utilisation de Sphinx et LaTeX avec MacPorts
% PATH=/opt/local/bin:$PATH make -C gprbuild-build doc
# La doc PDF n'est pas générée suite à des erreurs
% make adacore-version=20.2 os=darwin sudo= gprbuild-install
```

9. Construire GNATColl (Core, Bindings et DB)

GNAT Component Collection (GNATColl) est une bibliothèque d'usage générale utilisée pour les outils d'AdaCore comme GPS. Elle inclue une vingtaine de composants dont les traces, la mémoire, les chaînes de caractères, les e-mail, la logique trois états, JSON, SQL, ReadLine...
Site web : github.com/AdaCore/gnatcoll.
La version installée est 20.2.

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% export GPR_PROJECT_PATH=$instxada/share/gpr
% PATH=/usr/local/gnat/bin:$PATH
```

Récupérer ensuite sur le bureau les fichiers Makefile et gnatcoll-db-patch.txt à partir du site :
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/Makefile
et
raw.githubusercontent.com/Blady-Com/gnat-builder/xnadalib-2020/patches/gnatcoll-db-patch.txt

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% cp -p ~/Desktop/Makefile .
% mkdir -p patches
% cp -p ~/Desktop/gnatcoll-db-patch.txt patches

# GNATColl Core
% make adacore-version=20.2 rm-git-repo= os=darwin prefix=$instxada BUILD=DEBUG GPRBUILD_OPTIONS="-gnatwn -gnatU" GNATCOLL_VERSION=20.2 gnatcoll-core
# Utilisation de Sphinx et LaTeX avec MacPorts
```

```
% PATH=/opt/local/bin:$PATH make -C gnatcoll-core-build/docs html
% PATH=/opt/local/bin:$PATH make -C gnatcoll-core-build/docs latexpdf
% make adacore-version=20.2 os=darwin sudo= gnatcoll-core-install

# GNATColl Bindings
% cd /usr/local/src-2020
# Utilise GMP installé auparavant avec GNUTLS
% C_INCLUDE_PATH=$instxada/include LIBRARY_PATH=$instxada/lib make adacore-version=20.2 rm-git-repo= os=darwin gnatcoll-bindings-options="--debug --gpr-opts=-gnatwn" gnatcoll-bindings
% make adacore-version=20.2 os=darwin prefix=$instxada sudo= gnatcoll-bindings-install

# GNATColl DB
% cd /usr/local/src-2020
% make adacore-version=20.2 os=darwin prefix=$instxada BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-db-build

% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-sql
% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 prefix=$instxada sudo= gnatcoll-sql-install

% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-sqlite
% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 prefix=$instxada sudo= gnatcoll-sqlite-install

% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-gnatcoll_db2ada
% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 prefix=$instxada sudo= gnatcoll-gnatcoll_db2ada-install

% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-xref
% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 prefix=$instxada sudo= gnatcoll-xref-install

% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 gnatcoll-gnatinspect
% make adacore-version=20.2 os=darwin BUILD=DEBUG GNATCOLL_SQLITE=external DB_BACKEND=sqlite GNATCOLL_VERSION=20.2 prefix=$instxada sudo= gnatcoll-gnatinspect-install
```

Des exemples de programme sont disponibles dans le dossier $instxada/share/examples/gnatcoll.

Une documentation au format PDF et HTML est disponible dans le dossier $instxada/share/doc/gnatcoll.

10.Construire Florist

Florist contient les composants conforment aux standards Posix Ada : IEEE Standards 1003.5: 1992, IEEE STD 1003.5b: 1996 et en partie IEEE STD 1003.5c: 1998.
Site Web : www.cs.fsu.edu/~baker/florist.html
La version installée est mid-2020.

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=$instxada/bin:$PATH
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone https://github.com/Blady-Com/florist.git
% cd florist
% git checkout xnadalib-2020
% ./configure --prefix=$instxada
% Build=Debug make
% make rm-doc
% make install
```

Une documentation au format HTML est disponible dans le dossier $instxada/share/doc/florist/florist_rm.

11.Construire Gate3

Gate3 est un utilitaire qui produit du code Ada à partir d'un fichier Glade.
Gate3 a été développé par Francois Fabien sous licence MIT.
Site web : sourceforge.net/projects/lorenz.
La version installée est 0.5c.

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=$instxada/bin:$PATH
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone https://github.com/Blady-Com/gate3.git
% cd gate3
% make
% make PREFIX=$instxada install
```

Des exemples sont construits avec :

```
% make editor
% make calculator
% make lady
% make lorenz
```

Leur exécution :

```
% ./bin/editor
% ./bin/calculator
% ./bin/lady
% ./bin/lorenz
```

Les sources Ada sont construits par Gate3 à partir d'un fichier Glade avec gate3.sh. Voir le tutoriel Factoriel sur Blady :
blady.pagesperso-orange.fr/a_savoir.html#gtkada

Une documentation au format texte est disponible dans le dossier $instxada/share/gate3/doc. Un tutoriel mettant en oeuvre l'application factorielle est présent dans le dossier $instxada/share/gate3/tutorial.

Les fichiers modèles pour la génération des sources Ada dans le dossier $instxada/share/gate3/tmplt peuvent être modifiés en incluant notamment la licence dans les modèles des paquetages spécifications et corps :

- gate3_license.txt : licence MIT par défaut, à changer à votre convenance
- gate3_header.tmplt : en-tête de la procédure principale, inclut gate3_license.txt
- gate3_main.tmplt : modèle de la procédure principale
- gate3_spec.tmplt : modèle du paquetage spécification des callbacks
- gate3_body.tmplt : modèle du paquetage corps des callbacks

12.Construire AdaCurses

AdaCurses est une bibliothèque Ada 95 basée sur NCurses. La correspondance avec les fonctions de NCurses n'est pas directe mais a été construite dans l'esprit Ada de privilégier la lisibilité.

Site web : www.gnu.org/software/ncurses.

La version installée est 6.2.

Avant de démarrer la compilation, GNAT et le dossier d'installation doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=$instxada/bin:$PATH
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone https://github.com/Blady-Com/ncurses.git
% cd ncurses
% git checkout xnadalib-2020
% ./configure CC='gcc -g -O0' --enable-pc-files --with-ada-sharedlib --with-shared --enable-symlinks --prefix=$instxada --with-ada-libname=adacurses
% make
% make -C Ada95/src rm-docs
% make install
% make -C Ada95/src install-new
```

Des exemples de programme sont disponibles dans le dossier $instxada/share/examples/adacurses.

Une documentation au format HTML est disponible dans le dossier $instxada/share/doc/adacurses.

13.Construire Zanyblue

Zanyblue est une bibliothèque native en Ada pour l'internalisation d'un logiciel avec l'affichage de textes traduits dans différentes langues.

Site web : zanyblue.sourceforge.net.

Version installée : 1.4.0.

Récupérer l'archive suivante sur le bureau :

sourceforge.net/projects/zanyblue/files/zanyblue-1.4.0.tar.gz

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% tar xzf ~/Desktop/zanyblue-1.4.0.tar.gz
% cd zanyblue-1.4.0
% cd src
% make
% make INSTALL_DIR=$instxada install
```

Une documentation au format PDF et HTML est disponible dans le dossier $instxada/share/doc/zanyblue :

```
% open $instxada/share/doc/zanyblue/ZanyBlue.pdf
% open $instxada/share/doc/zanyblue/index.html
% open $instxada/share/doc/zanyblue/ref/index.html
```

Des exemples de programme sont disponibles dans le dossier $instxada/share/examples/zanyblue.

14.Construire PragmARC

PragmAda Reusable Components (PragmARCs) est une collection de composants mathématiques, de dates, de listes proposée par Jeff Carter.

Site web : pragmada.x10hosting.com/pragmarc.htm.

La version installée est mid-2020.

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

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone https://github.com/Blady-Com/PragmARC.git
% cd PragmARC
% git checkout xnadalib-2020
% gprbuild -p -P lib_pragmarc.gpr
% gnatdoc --enable-build -P lib_pragmarc.gpr
% gprinstall -f -p --prefix=$instxada lib_pragmarc.gpr
```

Une documentation au format HTML est disponible dans le dossier $instxada/share/doc/pragmarc :

```
% open $instxada/share/doc/pragmarc/pragmarc_rm/index.html
```

15.Construire Gnoga

Gnoga est une bibliothèque graphique créée nativement en Ada. Ce n'est pas une sur-couche Ada à une bibliothèque existante en C ou C++. Sa particularité est de permettre de construire des applications graphiques orientées Web indépendantes de la plateforme. Double indépendance garantie d'une part de fait du langage Ada lui même, Ada assure qu'un code source aura un comportement identique quelque soit la plate-forme d'exécution de part son compilateur (s'il accepte la compilation), d'autre part avec l'utilisation du HTML, CSS et Javascript pour le rendu graphique dans un navigateur Web.

Source : www.gnoga.com.

La version installée est 1.6-beta.

Avant de démarrer la compilation, GNAT doit être activé, ajouter son emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone git://git.code.sf.net/p/gnoga/code gnoga-code
% cd gnoga-code
% git checkout dev_1.6
% make gnoga-config
% make html-docs
% make rm-docs
% make PREFIX=$instxada install_gnoga_sa
% cp -p bin/gnoga-config $instxada/bin
% tar -cf - demo tutorial | tar -C $instxada/share/gnoga -xf -
```

Pour une utilisation courante, nous positionnons les variables d'environnement PATH pour une utilisation en ligne de commande et GPR_PROJECT_PATH pour une utilisation avec un projet GPS :

```
% echo 'PATH=$instxada/bin:$PATH' >> ~/.profile
% echo 'PATH=$instxada/bin:$PATH' >> ~/.bashrc
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:
$GPR_PROJECT_PATH' >> ~/.profile
% echo 'export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:
$GPR_PROJECT_PATH' >> ~/.bashrc
```

Pour une utilisation temporaire, utiliser à chaque fois les commandes suivantes :

```
% PATH=$instxada/bin:$PATH
% export GPR_PROJECT_PATH=$instxada/lib/gnat:$instxada/share/gpr:$GPR_PROJECT_PATH
```

Des démos et tutoriels sont présents respectivement dans les dossiers "$instxada/share/gnoga/demo" et "$instxada/share/gnoga/tutorial".

Une documentation sous forme de manuels utilisateur et d'un manuel de référence est disponible dans le dossier "$instxada/share/gnoga/html".

```
% open $instxada/share/gnoga/html/user_guide.html
% open $instxada/share/gnoga/html/api_summary.html
% open $instxada/share/gnoga/html/gnoga_rm/index.html
```

Voir l'utilisation de Gnoga avec des exemples sur Blady :
blady.pagesperso-orange.fr/a_savoir.html#gnoga

16.Construire SparForte

SparForte est un interpréteur du langage AdaScript, il peut servir de shell Ada ou de moteur de création de page HTML en AdaScript qui est un petit sous-ensemble du langage de programmation Ada.

Site web : sparforte.com.

Version installée : 2.3.1.

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
# Utilisation de SDL et SDL_Image avec MacPorts
% PATH=/opt/local/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone https://github.com/Blady-Com/SparForte.git
% cd SparForte/
% git checkout xnadalib-2020
% ./configure --prefix=$instxada --manprefix=$instxada --without-bdb --without-mysql --without-opengl --without-postgres --without-readline --without-sound --arch=native --without-pcre
% make
% make install
```

Le manuel est dans le dossier $instxada/man/man1:

```
% MANPATH=$instxada/man man spar
```

L'interpréteur se lance ainsi:

```
% cd $instxada/bin
% ./spar
Type "help" for help
=> put_line ("Hello again with Ada!")
Hello again with Ada!
=> return
```

17.Construire Alire

Alire est un catalogue de bibliothèques Ada prêtes à l'emploi et un utilitaire en ligne de commande pour construire vos programmes avec ces bibliothèques.

Site web : alire.ada.dev.

Version installée : 0.6.1.

Avant de démarrer la compilation, GNAT et le dossier d'installation de XNAdaLib doivent être activés, ajouter leur emplacement comme par exemple :

```
% instxada=/usr/local/xnadalib-2020
% PATH=/usr/local/gnat/bin:$PATH
```

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local/src-2020
% git clone --recursive https://github.com/alire-project/alire.git
% cd alire
% OS=macOS gprbuild -j0 -p -P alr_env
% cp -p bin/alr $instxada/bin
```

L'utilitaire alr se lance ainsi:

```
% cd $instxada/bin
% alr --help
Usage: alr [global options] <command> [command options] [arguments]
 -c, --config=ARG      Override configuration folder location
 -f, --force           Keep going after a recoverable troublesome situation
 -h, --help            Display general or command-specific help
 -n, --non-interactive Assume default answers for all user prompts
 --no-color            Disables colors in output
 --no-tty              Disables control characters in output
 --prefer-oldest       Prefer oldest versions instead of newest when resolving dependencies
 -q                    Limit output to errors
 -v                    Be more verbose (use twice for extra detail)
 -d, --debug[ARG]      Enable debug-specific log messages
% ./alr list gnat
gnat            GNAT is a compiler for the Ada programming language
gnatcoll        GNAT Components Collection - Core packages
gnatcoll_gmp    GNAT Components Collection - Bindings to GMP
gnatcoll_iconv  GNAT Components Collection - Bindings to libiconv
```

Pascal Pignard, avril-septembre 2011, août 2014, juillet 2015, août-septembre 2016, octobre 2017, septembre 2018, octobre 2020.

http://blady.pagesperso-orange.fr