Installer ou construire GNATStudio 25.0w sur macOS

GNATStudio est l'environnement de programmation intégré (IDE) dédié au langage Ada. Il permet d'effectuer toutes les activités classique du développement d'un logiciel : le codage dans un éditeur dédié à Ada, l'intégration de bibliothèques en Ada, C ou C++, le test, le déverminage ou l'analyse de code.

Site www.adacore.com/gnatpro/toolsuite/gnatstudio.

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

Nous allons voir comme installer l'application GNATStudio pour macOS prêtre à l'emploi et aussi pouvoir la construire à partir des sources de ses composants.

Sommaire

1. Installer GnatStudio

Télécharger le fichier suivant sur le bureau du Mac :
20240623-Applications-GNATStudio.dmg,
depuis le site de Source Forge sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2024-ventura.

Ouvrir le fichier *DMG* et copier l'application dans un dossier local, par exemple : *$HOME/Applications*.

L'application n'est pas signée, il sera certainement nécessaire de supprimer les attributs de mise en quarantaine :

```
% xattr -r -d com.apple.quarantine $HOME/Applications/gnatstudio.app
```

Les emplacements du compilateur GNAT et des utilitaires GPR sont prédéfinis avec la version GNAT FSF 13.1 (*/opt/gcc-13.1.0/bin*). Si vous avez une autre version ou d'autres emplacements, vous pouvez les activer en modifiants *GS_GNAT_PATH* et *GS_GPR_PATH* dans le fichier *Info.plist* de l'application *GNATStudio.app*. Attention, I*nfo.plist* est mis en cache, vous devez alors le mettre dans un dossier temporaire puis le remettre dans l'application pour qu'il soit pris en compte. Une fois l'application lancée, vous pouvez vérifier votre configuration dans les menus *help->about* et *build->settings->toolchains*.

Ce successeur de GPS nommé GNATStudio va créer un nouveau dossier *.gnatstudio* de préférences dans le dossier *$HOME* en se basant sur le dossier existant *.gps*. Pour éviter tout problème lors de la conversion, je recommande de renommer temporairement *.gps* avant le premier lancement de GNATStudio.

Le premier lancement de GNATStudio peut prendre un certain temps dû au référencement de l'application par macOS. Si une fenêtre surgit demandant une autorisation pour accéder au dossier *Documents* alors cliquer sur le bouton *OK* pour donner l'accord.

En cas de problème, je conseil de lancer le programme depuis le Terminal pour observer les indications affichées :

```
% $HOME/Applications/gnatstudio.app/Contents/MacOS/gnatstudio_launcher
```
ou
```
% open -a $HOME/Applications/GNATStudio.app --stdout=`tty` --stderr=`tty`
```

Attention au contenu de votre variable *PATH*. Gardez la avec le minimum nécessaire. Évitez les chemins avec Python, MacPorts ou Brew.
Avec la première ligne, le compilateur sera recherché uniquement avec *PATH*. Avec la seconde ligne, le compilateur sera uniquement recherché avec *GS_GNAT_PATH* et *GS_GPR_PATH*.

Voir l'utilisation de GNATStudio avec des exemples sur Blady :
blady.chez.com/a_savoir.html#gnat

2. Configuration générale

Configuration : macOS 13.6, Xcode 14.3 et GNAT FSF 13.2 (depuis Alire).

Voir leur installation sur Blady :

- macOS et Xcode : blady.chez.com/liens.html#macosx
- GNAT : blady.chez.com/creations.html#gnatosxinstall

3. Construire GTK

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 3.24.38.

Récupérer dans le dossier *Téléchargements* 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.

Noter la date du téléchargement ou la date et le SHA du dernier commit sur :
gitlab.gnome.org/GNOME/gtk-osx/-/commits/master/gtk-osx-setup.sh.
Si vous avez un clone du dépôt vous pouvez apposer un TAG sur le dernier commit.

Et aussi *xnadalib-2024-diff.zip* sur blady.chez.com/telechargements/gtkada/xnadalib-2024-diff.zip.

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

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% version=2024
% sudo mkdir src-$version
% sudo chown $USER src-$version
% xnadasrc=$xnadabase/src-$version
% sudo mkdir xnadalib-$version
% sudo chown $USER xnadalib-$version
% xnadainst=$xnadabase/xnadalib-$version
% xnarchive=$HOME/Downloads
```

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

```
% export DEVROOT=$xnadasrc
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CONFIG_HOME=$xnadasrc/config
% export XDG_CACHE_HOME=$xnadasrc/cache

% sh $xnarchive/gtk-osx-setup.sh
gtk-osx-setup.sh: Exception KeyError: 'name' Locking Failed! Voir https://github.com/jralls/gtk-osx-build/discussions/99 (modification .new_local/lib/python3.11/site-packages/pipenv/project.py, voir xnadalib-2024-diff)
% sh $xnarchive/gtk-osx-setup.sh
~/Documents/Programmation/XNAdaLib/2024a/src-2024/Source/pyenv ~/Documents/Programmation/XNAdaLib/2024a
Already up to date.
~/Documents/Programmation/XNAdaLib/2024a
pyenv: /Users/blady/Documents/Programmation/XNAdaLib/2024a/src-2024/.new_local/share/pyenv/versions/3.11.7 already exists
continue with installation? (y/N) N

...
PATH does not contain .../src-2024/.new_local/bin. You probably want to fix that.
...
```

Cette commande installe *jhbuild* dans le dossier *Source* et créé le dossier *.new_local/bin* avec les utilitaires requis pour la suite. Il installe aussi les scripts d'installation *config/jhbuildrc* et *config/jhbuildrc-custom* et copie les modules gtk-osx courants dans *Source/jhbuild/modulesets*. (Si ces derniers 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 :

```
% 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 :
% unzip $xnarchive/xnadalib-2024-diff.zip
% cd $xnadasrc/config
% patch -p0 < $xnarchive/xnadalib-2024-diff/jhbuildrc-custom.diff
% cd $xnadasrc
...
Lors de la première utilisation de Java, une fenêtre système surgit pour donner à Java l'accès au dossier Documents, cliquez sur OK
% jhbuild bootstrap-gtk-osx
...
% jhbuild build meta-gtk-osx-bootstrap
...
Python fourni avec macOS n'est pas installé avec le SDK complet nous installons le notre
Pygments est nécessaire pour gtk-doc

% jhbuild build pygments

% jhbuild build meta-gtk-osx-gtk3
...

Liste des modules installés :

- adwaita-icon-theme-44.0
- atk-2.38.0
- autoconf-2.71
- autoconf-archive-2023.02.20
- automake-1.16.5
- bison-3.8.2
- cairo-1.17.8
- cmake-3.26.4
- flex-2.6.4
- fontconfig-2.14.2
- freetype-2.13.1
- fribidi-1.0.13
- gdk-pixbuf-2.42.10
- gettext-0.22.3
- glib-2.76.3
- gobject-introspection-1.76.1
- gtk-doc-1.33.2
- gtk-mac-integration-3.0.1
- gtk-osx-docbook-1.3
- gtk+-3.24.38
- harfbuzz-7.3.0
- hicolor-icon-theme-0.17
- icu4c-73_2-src
- intltool-0.51.0
- jpegsrc.v9e
- libepoxy-1.5.10
- libffi-3.4.4
- libiconv-1.17
- libpng-1.6.40
- librsvg-2.56.1
- libtool-2.4.7
- libxml2-2.11.4
- m4-1.4.19
- make-4.4
- openssl-3.1.1
- pango-1.50.14
- pcre2-10.42
- pixman-0.42.2
- pkgconf-2.2.0
- Pygments-2.15.1
- Python-3.11.4
- readline-8.2
- tiff-4.5.1
- xz-5.4.3
- zlib-1.3.1

4. Construire les extensions GTK en Python

a) Construire PyCairo

Il s'agit d'une bibliothèque qui fournit les API Cairo pour le langage Python..
Site web : pypi.org/project/pycairo.
La version installée est 1.24.0.

Saisir les commandes suivantes dans le Terminal :

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% 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 pycairo
```

b) Construire PyGObject3

Il s'agit d'une bibliothèque qui fournit les API GObject (GTK, GStreamer, WebKitGTK, GLib, GIO...) pour le langage Python.
Site web : pypi.org/project/PyGObject.
La version installée est 3.44.1.

Saisir les commandes suivantes dans le Terminal :

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% 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 pygobject3
```

5. Construire GMP

Il s'agit d'une bibliothèque arithmétique multi-précisions pour des entiers, rationnels ou réels flottants.

Site web : gmplib.org.

La version installée est 6.2.1.

Saisir les commandes suivantes dans le Terminal :

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% 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 gmp
```

6. Construire Pip et VitualEnv

Pip est un programme d'installation de modules Python. VirtualEnv est un outil pour créer des environnements virtuels Python isolés.

Site web : pypi.org

Site web : virtualenv.pypa.io

La version installée est 3.11.4.

Saisir la commande suivante dans un nouveau Terminal :

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadalib-$version
# Activation de la bibliothèque GTK-OSX avec Python
% PATH=$xnadainst/bin:$PATH

% python3 -m ensurepip --upgrade
% python3 -m pip --version
% PATH=$PATH
% pip3 install virtualenv
% pip3 install --upgrade pip

# Utile pour les scripts branché uniquement sur python
% cd $xnadainst/bin
% ln -s python3 python
% cd -
```

7. Configuration de l'environnement en Ada

Nous allons utiliser les codes sources de la version pour Linux.
Télécharger les sources de GNATStudio dans le dossier *Téléchargements* :
gnatstudio-sources-x86_64-linux.tar.gz,
depuis le site Github : github.com/AdaCore/gnatstudio/releases/tag/gnatstudio-cr-20240506

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

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadalib-$version

# Activation du compilateur Ada GNAT FSF 12.1 (depuis Alire)
% PATH=$HOME/.local/share/alire/toolchains/gnat_native_13.2.2_9be96041/bin:$HOME/.local/share/alire/toolchains/gprbuild_22.0.1_b1220e2b/bin:$PATH

# Activation de la bibliothèque GTK-OSX
% PATH=$xnadainst/bin:$PATH
# Activation de la bibliothèque Ada
% export GPR_PROJECT_PATH=$xnadainst/share/gpr
```

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadabase
% tar xzf gnatstudio-sources-x86_64-linux.tar.gz
% xnarchive=$xnadabase/gnatstudio-sources-x86_64-linux
```

Pour l'exécution, les bibliothèques Ada GNAT et C++ GCC doivent être présentent dans le dossier d'installation :

```
% cd $xnadainst
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_13.2.2_9be96041/lib/gcc/x86_64-apple-darwin21.6.0/13.2.0/adalib/libgnarl-13.dylib lib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_13.2.2_9be96041/lib/gcc/x86_64-apple-darwin21.6.0/13.2.0/adalib/libgnat-13.dylib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_13.2.2_9be96041/lib/gcc/x86_64-apple-darwin21.6.0/13.2.0/adalib/libgnat-13.dylib lib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_13.2.2_9be96041/lib/libstdc++.6.dylib lib
```

8. Construire GTKAda

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 25.0w.

Dépendances : GTK, ATK, Cairo, GDK-Pixbuf, Glib, Harfbuzz, Pango, Pixman, PNG.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gtkada-25.0w-20240505-16451-src.tar.gz
% cd gtkada-25.0w-20240505-16451-src
% ./configure --prefix=$xnadainst
# Modification Makefile
% make -w
% make -w install prefix=$xnadainst PRJDIR=share/gpr
```

9. 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 25.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gtkada-25.0w-20240505-16451-src.tar.gz
% cd gtkada-25.0w-20240505-16451-src
% ./configure --prefix=$xnadainst
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install
```

10. Construire Templates-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 25.0w.

Dépendance : XMLAda.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/templates_parser-25.0w-20240408-16376-src.tar.gz
% cd templates_parser-25.0w-20240408-16376-src
% make -w prefix=$xnadainst setup
% ADAFLAGS="-gnatwn -gnatU" make -w
% make -w install
```

11.Construire GPR (inclut dans GPRBuild)

GPR est la bibliothèque de manipulation des fichiers GPR. Un programme complexe ayant plusieurs dépendances peut être décrit à travers un fichier projet GPR.

Site web : github.com/AdaCore/gprbuild.

La version installée est 25.0w.

Dépendance : XMLAda.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gprbuild-25.0w-20240408-162DA-src.tar.gz
% cd gprbuild-25.0w-20240408-162DA-src
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU" libgpr.build
% make -w libgpr.install
```

12.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 25.0w.

Dépendances :

- Core : GPR
- Bindings : GNATColl-Core, readline, iconv, lzma, openmp, z, gmp, python3, intl, dl, CoreFoundation
- DB : GNATColl-Core, GNATColl-Bindings, pthread, sqlite3

Saisir les commandes suivantes dans le Terminal :

```
# GNATColl Core
% cd $xnadasrc
% tar xzf $xnarchive/gnatcoll-core-25.0w-20240408-16118-src.tar.gz
% cd gnatcoll-core-25.0w-20240408-16118-src
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl Bindings
% cd $xnadasrc
% tar xzf $xnarchive/gnatcoll-bindings-25.0w-20240408-162B5-src.tar.gz

# GNATColl Bindings gmp
% cd $xnadasrc
% cd gnatcoll-bindings-25.0w-20240408-162B5-src
% cd gmp
% CFLAGS=`pkg-config gmp --cflags` LDFLAGS=`pkg-config gmp --libs` python3 ./setup.py build --gpr-opts=-gnatwn
```

```
% CFLAGS=`pkg-config gmp --cflags` LDFLAGS=`pkg-config gmp --libs` python3 ./setup.py install --
prefix=$xnadainst

# GNATColl Bindings iconv
% cd $xnadasrc
% cd gnatcoll-bindings-25.0w-20240408-162B5-src
% cd iconv
## % python3 ./setup.py build --gpr-opts=-gnatwn

# Ne pas prendre Libiconv de macOS mais celui de GTK-OSX
% gprbuild -j0 -p -gnatwn -Pgnatcoll_iconv.gpr --target=x86_64-darwin -XGNATCOLL_VERSION=25.0w
-XBUILD=PROD -XGNATCOLL_OS=osx -XGNATCOLL_ICONV_OPT=-liconv
-XLIBRARY_TYPE=relocatable -XXMLADA_BUILD=relocatable -XGPR_BUILD=relocatable
-XCFLAGS=-I$xnadainst/include -XLDFLAGS=-L$xnadainst/lib
% gprbuild -j0 -p -gnatwn -Pgnatcoll_iconv.gpr --target=x86_64-darwin -XGNATCOLL_VERSION=25.0w
-XBUILD=PROD -XGNATCOLL_OS=osx -XGNATCOLL_ICONV_OPT=-liconv -XLIBRARY_TYPE=static
-XXMLADA_BUILD=static -XGPR_BUILD=static -XCFLAGS=-I$xnadainst/include -XLDFLAGS=-
L$xnadainst/lib

% python3 ./setup.py install --prefix=$xnadainst

# GNATColl Bindings python3
% cd $xnadasrc
% cd gnatcoll-bindings-25.0w-20240408-162B5-src
% cd python3
% python3 ./setup.py build --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl DB
% cd $xnadasrc
% tar xzf $xnarchive/gnatcoll-db-25.0w-20240408-1639A-src.tar.gz

# GNATColl DB sql
% cd $xnadasrc
% cd gnatcoll-db-25.0w-20240408-1639A-src
% cd sql
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB sqlite
% cd $xnadasrc
% cd gnatcoll-db-25.0w-20240408-1639A-src
% cd sqlite
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB db2ada
% cd $xnadasrc
% cd gnatcoll-db-25.0w-20240408-1639A-src
% cd gnatcoll_db2ada
% make DB_BACKEND=db prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install
```

```
# GNATColl DB xref
% cd $xnadasrc
% cd gnatcoll-db-25.0w-20240408-1639A-src
% cd xref
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB gnatinspect
% cd $xnadasrc
% cd gnatcoll-db-25.0w-20240408-1639A-src
% cd gnatinspect
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install
```

13.Construire VSS

VSS est une bibliothèque de manipulation de chaines de caractères Unicode et de textes de divers formats.

Site web : github.com/AdaCore/VSS.

La version installée est 25.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/vss-25.0w-20240408-1642F-src.tar.gz
% cd vss-25.0w-20240408-1642F-src
% make -w build-all-libs
% make -w install-all-libs PREFIX=$xnadainst
```

14.Construire Spawn et Spawn_glib

Spawn est une bibliothèque très simple pour lancer des sous-processus et communiquer avec eux. Elle vient sous deux formules : intégrée à Glib et indépendante.

Site web : github.com/AdaCore/spawn.

La version installée est 25.0w.

Dépendance : GTKAda pour la version Glib, sans sinon.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/spawn-25.0w-20240505-164A1-src.tar.gz
% cd spawn-25.0w-20240505-164A1-src
% gprbuild -p -P gnat/spawn_glib.gpr -XOS=osx -XSPAWN_WARN_ERRORS=false
% gprinstall --prefix=$xnadainst -p -P gnat/spawn_glib.gpr -XOS=osx -XSPAWN_WARN_ERRORS=false
% gprbuild -p -P gnat/spawn.gpr -XOS=osx -XSPAWN_WARN_ERRORS=false
% gprinstall --prefix=$xnadainst -p -P gnat/spawn.gpr -XOS=osx -XSPAWN_WARN_ERRORS=false
```

15.Construire AdaSAT

AdaSAT est une implémentation d'un solveur SAT basé sur DPLL.
Site web : github.com/AdaCore/AdaSAT.
La version installée est 25.0w.
Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/adasat-25.0w-20240408-16385-src.tar.gz
% cd adasat-25.0w-20240408-16385-src
% make -w all-libs BUILD_MODE=prod
% make -w install BUILD_MODE=prod INSTALL_DIR=$xnadainst
```

16.Construire PrettierAda

PrettierAda le portage du formateur Prettier pour le langage Ada.
Site web : github.com/AdaCore/prettier-ada.
La version installée est 25.0w.
Dépendance : GNATColl-Core, VSS.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/prettier-ada-25.0w-20240408-16284-src.tar.gz
% cd prettier-ada-25.0w-20240408-16284-src
% make -w all BUILD_MODE=prod
% make -w install BUILD_MODE=prod PREFIX=$xnadainst
```

17.Construire Langkit

Langkit est une boite à outil qui rend plus facile la création d'analyseurs syntaxiques et sémantiques.
Site web : github.com/AdaCore/langkit.
La version installée est 25.0w.
Dépendance : Python3 avec plusieurs utilitaires dont Mako, GNATColl core, GnatColl gmp, GNATColl iconv, XMLAda, GPR.

Saisir les commandes suivantes dans le Terminal :
(Une connexion Internet est requise)

```
% cd $xnadasrc
% tar xzf $xnarchive/langkit-25.0w-20240411-1627B-src.tar.gz
% cd langkit-25.0w-20240411-1627B-src
% pip3 install .
```

18.Construire GPR2

GPR2 est la nouvelle bibliothèque d'analyse des fichiers projets GPR.
Site web : github.com/AdaCore/gpr.
La version installée est 25.0w.
Dépendance : GNATColl, Langkit, GPRConfig KB, Python3.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gpr2-25.0w-20240409-162B5-src.tar.gz

% tar xzf $xnarchive/gprconfig-kb-25.0w-20240408-16484-src.tar.gz

% cd gpr2-25.0w-20240409-162B5-src
# Modification makefile
% make -w prefix=$xnadainst setup GPR2KBDIR=$xnadasrc/gprconfig-kb-25.0w-20240505-16517-src/
db MFILE="./Makefile)" PYTHON=python3 GPR2_BUILD=release
% LIBRARY_PATH=$xnadainst/lib make -w build-libs MFILE="./Makefile)"
% LIBRARY_PATH=$xnadainst/lib make -w install-libs MFILE="./Makefile)"t
```

19.Construire Libadalang

Libadalang est bibliothèque pour analyser la sémantique du langage Ada.
Site web : github.com/AdaCore/libadalang.
La version installée est 25.0w.
Dépendance : Python3 avec plusieurs utilitaires dont Mako, Langkit, GNATColl core, GnatColl gmp, GNATColl iconv, XMLAda, GPR, GPR2, AdaSAT, PrettierAda.

Saisir les commandes suivantes dans le Terminal :
(Une connexion Internet est requise)

```
% cd $xnadasrc
% tar xzf $xnarchive/libadalang-25.0w-20240411-16289-src.tar.gz
% cd libadalang-25.0w-20240411-16289-src

% python3 -mvenv env
% source env/bin/activate
% pip3 install -r REQUIREMENTS.dev

% ln -s $xnadasrc/langkit-24.0w-20230428-16136-src langkit
% cd langkit
% pip3 install .

% LIBRARY_PATH=$xnadainst/lib python3 manage.py build-langkit-support --library-types=static,static-
pic,relocatable --build-mode=prod
% LIBRARY_PATH=$xnadainst/lib python3 manage.py install-langkit-support --library-types=static,static-
pic,relocatable $xnadainst --build-mode=prod

% cd ..
% python3 manage.py generate
```

```
% LIBRARY_PATH=$xnadainst/lib python3 manage.py build --library-types=static,static-pic,relocatable --build-mode=prod
% LIBRARY_PATH=$xnadainst/lib python3 manage.py install $xnadainst --library-types=static,static-pic,relocatable --build-mode=prod

% deactivate
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/nameres
% install_name_tool -add_rpath @executable_path/../lib bin/lal_parse
% install_name_tool -add_rpath @executable_path/../lib bin/lal_prep
% install_name_tool -add_rpath @executable_path/../lib bin/lal_dda
% install_name_tool -add_rpath @executable_path/../lib bin/gnat_compare
% install_name_tool -add_rpath @executable_path/../lib bin/navigate
% install_name_tool -add_rpath @executable_path/../lib bin/lal_unparse
```

20. Construire Libadalang Tools

Les Libadalang tools sont le regroupement des utilitaires gnatpp, gnatmetric, gnatstub, gnattest.

Site web : github.com/AdaCore/libadalang-tools.

La version installée est 25.0w.

Dépendance : Libadalang.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/libadalang-tools-25.0w-20240408-1625A-src.tar.gz
% cd libadalang-tools-25.0w-20240505-16471-src
# Modification Makefile
% LIBRARY_PATH=$xnadainst/lib make lib BUILD_MODE=prod LIBRARY_TYPE=relocatable
% LIBRARY_PATH=$xnadainst/lib make bin BUILD_MODE=prod LIBRARY_TYPE=relocatable
% LIBRARY_PATH=$xnadainst/lib make install-lib DESTDIR=$xnadainst BUILD_MODE=prod LIBRARY_TYPE=relocatable
% LIBRARY_PATH=$xnadainst/lib make install-bin-strip DESTDIR=$xnadainst BUILD_MODE=prod LIBRARY_TYPE=relocatable
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/gnatmetric
% install_name_tool -add_rpath @executable_path/../lib bin/gnatpp
% install_name_tool -add_rpath @executable_path/../lib bin/gnatstub
% install_name_tool -add_rpath @executable_path/../lib bin/gnattest
% install_name_tool -add_rpath @executable_path/../lib bin/utils-var_length_ints-test
```

21.Construire LAL-Refactor

Refactor est un ensemble d'outils de remaniement du code source pour le langage Ada utilisant LibAdaLang.

Site web : github.com/AdaCore/lal-refactor.

La version installée est 25.0w.

Dépendance : LibadalangTools, VSS.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/lal-refactor-25.0w-20240505-16130-src.tar.gz
% cd lal-refactor-25.0w-20240505-16130-src
# Modifiifcation testsuite/ada_drivers/gnat/lal_refactor_test_drivers.gpr
% LIBRARY_PATH=$xnadainst/lib make -w all BUILD_MODE=prod LIBRARY_TYPE=relocatable
% LIBRARY_PATH=$xnadainst/lib PREFIX=$xnadainst make -w install BUILD_MODE=prod
LIBRARY_TYPE=relocatable
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/lalrefactor
```

22.Construire GNATHub

GNATHub est un utilitaire en ligne de commande qui collecte et agrège les résultats de plusieurs outils d'analyse et .les enregistre dans un fichier SQLite.

Site web : github.com/AdaCore/gnatdashboard/tree/master/gnathub.

La version installée est 25.0w.

Dépendance :GPR2, GNATColl core, GnatColl Python, GNATColl SQLite. Et webui en option.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gnathub-edge-25.0w-20240408-162AB-src.tar.gz
% cd gnathub-edge-25.0w-20240505-163AA-src
# Modifiction du Makefile
% make -w PYTHON=python3
% make -w PYTHON=python3 prefix=$xnadainst install
# Copie de $xnadainst/lib/python3.11 dans $xnadainst/share/gnathub/python
```

23.--Construire Ada_libfswatch

(N'est pas utilisée) Ada LibFSWatch est une sur-couche Ada de la bibliothèque FSWatch qui surveille les notifications quand le contenu des fichiers et dossiers spécifiés est modifié.

Site web : github.com/AdaCore/ada_libfswatch.

La version installée est 24.0w.

Dépendance : FSWatch.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/ada_libfswatch-24.0w-20230428-16626-src.tar.gz
% cd ada_libfswatch-24.0w-20230428-16626-src
% tar xzf $xnarchive/fswatch-1.17.1.tar.gz
% cd fswatch-1.17.1
% ./configure --prefix=$xnadasrc/ada_libfswatch-24.0w-20230428-16626-src/libfswatch
% make -w
% make -w install
% cd ..
% make -w
% make -w install DESTDIR=$xnadainst
```

24.Construire Ada language server

Il s'agit d'un server conforme au standard Language Protocol de Microsoft pour les langages Ada et SPARK.

Site web : github.com/AdaCore/ada_language_server.

La version installée est 25.0w.

Dépendance : GNATColl, Libadalang, Libadalang-tools, VSS, Gnatdoc (source), GPR2, Spawn / Spawn_glib, Ada_libfswatch (interne ALS avec un projet bouchon).

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/als-25.0w-20240506-162AE-src.tar.gz
% tar xzf $xnarchive/gnatdoc4-25.0w-20240505-164DF-src.tar.gz
% cd als-25.0w-20240506-162AE-src

# Modification makefile
# Modification source/ada/lsp-ada_execute_command.adb et source/gpr/lsp-gpr_files.adb

% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-25.0w-20240506-162AE-src/
subprojects/stubs:$xnadasrc/gnatdoc4-25.0w-20240505-164DF-src/gnat:$GPR_PROJECT_PATH make
-w BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-25.0w-20240506-162AE-src/
subprojects/stubs:$xnadasrc/gnatdoc4-25.0w-20240505-164DF-src/gnat:$GPR_PROJECT_PATH make
-w install prefix=$xnadainst BUILD_MODE=prod

% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-25.0w-20240506-162AE-src/
subprojects/stubs:$xnadasrc/gnatdoc4-25.0w-20240505-164DF-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-build BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-25.0w-20240506-162AE-src/
subprojects/stubs:$xnadasrc/gnatdoc4-25.0w-20240505-164DF-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-install prefix=$xnadainst BUILD_MODE=prod
```

ou avec Ada LibFSWatch externe (non retenue):

```
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/
gnatdoc4-24.0w-20230428-16616-src/gnat:$GPR_PROJECT_PATH make -w BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/
gnatdoc4-24.0w-20230428-16616-src/gnat:$GPR_PROJECT_PATH make -w install prefix=$xnadainst
BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/
gnatdoc4-24.0w-20230428-16616-src/gnat:$GPR_PROJECT_PATH make -w lsp_client_glib-build
BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/
gnatdoc4-24.0w-20230428-16616-src/gnat:$GPR_PROJECT_PATH make -w lsp_client_glib-install
prefix=$xnadainst BUILD_MODE=prod
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/ada_language_server
% install_name_tool -add_rpath @executable_path/../lib bin/tester-run
```

25.Construire GNATStudio

GNATStudio est l'IDE dédié aux langages Ada et SPARK qui prend aussi en charge les langages C et C++.

Site web : github.com/AdaCore/xmlada.

La version installée est 25.0w.

Dépendance : GTKAda, PyGobject, PyCairo, GNATColl, VSS, Spawn, Libadalang, Libadalang-tools, Ada Language Server, XMLAda, Templates Parser.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gnatstudio-25.0w-20240506-161C6-src.tar.gz
% cd gnatstudio-25.0w-20240506-161C6-src
# Modification code source
% patch -p0 < $xnarchive/xnadalib-2024-diff/gnatstudio.diff
% PYTHON=python3 ./configure --prefix=$xnadainst
% LIBRARY_TYPE=relocatable make -w PYTHON=python3 BUILD=Production
% LIBRARY_TYPE=relocatable make -w PYTHON=python3 install BUILD=Production
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/gnatstudio
% install_name_tool -add_rpath @executable_path/../lib bin/gnatstudio_cli
% install_name_tool -add_rpath @executable_path/../lib bin/gnatdoc3
```

Installer les modules Python de Libadalang, Jedi et PyCodeStyle :

```
% cd $xnadainst
% cp -rp python share/gnatstudio
% pip3 install jedi
% pip3 install pycodestyle
```

Voilà notre exécutable GNATStudio est créé, c'était bien long ... très long et ce n'est pas tout à fait fini. Avant de le lancer, il nous faut positionner les variables d'environnement pour GTK, Python et GNATStudio.

Variables d'environnement pour GTK :

- *XDG_DATA_DIRS* : référence les données d'application habituellement misent dans le dossier *share*

```
% export XDG_DATA_DIRS="$xnadainst/share"
```

- *GTK_EXE_PREFIX* : référence le dossier *lib* à la place de celui pris lors de la compilation

```
% export GTK_EXE_PREFIX="$xnadainst"
```

- *GDK_PIXBUF_MODULE_FILE* : référence le fichier de description des modules *GDKPixbuf* à charger à la place de celui donné lors de la compilation par *libdir*

```
% export GDK_PIXBUF_MODULE_FILE="$xnadainst/lib/gdk-pixbuf-2.0/2.10.0/loaders.cache"
```

- *GI_TYPELIB_PATH* : référence le dossier des descriptions pour *GObject Introspection*

```
% export GI_TYPELIB_PATH="$xnadainst/lib/girepository-1.0"
```

Variables d'environnement pour Python :

- *GNATSTUDIO_PYTHONHOME* : indique au moteur d'exécution Python où trouver son environnement d'exécution

```
% export GNATSTUDIO_PYTHONHOME=$xnadainst
```

- *DYLD_LIBRARY_PATH* : indique au moteur d'exécution Python où trouver les bibliothèques GTK et GDK

```
% export DYLD_LIBRARY_PATH="$xnadainst/lib:$xnadainst/lib/gdk-pixbuf-2.0/2.10.0/loaders"
```

Variables d'environnement pour GNATStudio :

- *GPS_ROOT* : indique au programme où trouver l'environnement d'exécution

```
% export GPS_ROOT=$xnadainst
```

- *GNATSTUDIO_HOME* (optionnel) : indique au programme où trouver le dossier des préférences, par défaut *$HOME*

```
% export GNATSTUDIO_HOME=$xnadasrc
```

Nous utiliserons le script shell *gnat_launcher.sh* qui fera tout ça pour nous en lui indiquant le dossier d'installation avec la variable *prefix* :

```
% prefix=$xnadainst $xnadasrc/gnatstudio-25.0w-20240506-161C6-src/gnat_launcher.sh $xnadainst/bin/gnatstudio
```

Et voilà le résultat :

26.Construire l'application macOS GNATStudio

Nous allons prendre notre exécutable GNATStudio pour le transformer en une application macOS autonome. Elle pourra être utilisée sans besoin d'installation juste par un simple copier / coller.

Pour cela nous utiliserons l'utilitaire *GTK-Mac-Bundler* avec plusieurs fichiers de configuration.

a) Configuration de l'environnement JHBuild

```
% xnadabase=/usr/local
% version=2024
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadalib-$version
% PATH=$xnadasrc/.new_local/bin:$PATH
% export DEVROOT=$xnadasrc
% export PIP_CONFIG_DIR=$xnadasrc/config/pip
% export XDG_CONFIG_HOME=$xnadasrc/config
% export XDG_CACHE_HOME=$xnadasrc/cache
```

b) Construire GTK-Mac-Bundler

Il s'agit d'un script Python qui créé des applications macOS (bundle app) à partir de programmes GTK. Il est conçu pour fonctionner avec GTK construit avec *jhbuild* (voir §3).

Site web : github.com/jralls/gtk-mac-bundler.

La version installée est mi-2024 (0.7.4+).

Dépendances : Python, JHBuild, GTK.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% git clone https://github.com/Blady-Com/gtk-mac-bundler.git
% cd gtk-mac-bundler
% make -w install
```

c) Construire les fichiers de configuration

Les fichiers de configuration sont :

- *gnatstudio.bundle* : configuration principale du bundle
  - Indique le dossier de création du bundle
  - Indique la version de GTK
  - Indique l'emplacement du fichier Info.plist
  - Indique l'emplacement de l'exécutable principal
  - Indique les modules et données GTK à transformer
  - Indique les modules et données Python à transformer
  - Indique les exécutables secondaires
  - Indique l'emplacement de l'icône de l'application
- *gnatstudio.icns* : icône de l'application au format ICNS

- *Info-gnatstudio.plist* : informations de l'application
  - Indique l'exécutable à lancer
  - La version de notre programme
  - Le code APPL de notre application
  - Les variables d'environnement spécifiques pour le fonctionnement de notre programme

Pour récupérer les trois fichiers, saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% mkdir gs_bundle
% cd gs_bundle
% unzip $xnarchive/xnadalib-2024-diff/gs_bundle_conf.zip
```

d) Construire le bundle

Nous allons exécuter *gtk-mac-bundler* dans l'environnement *jhbuild*.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% cd gs_bundle
% jhbuild shell
[JH] % gtk-mac-bundler gnatstudio.bundle
[JH] % exit
```

e) Construire le lanceur

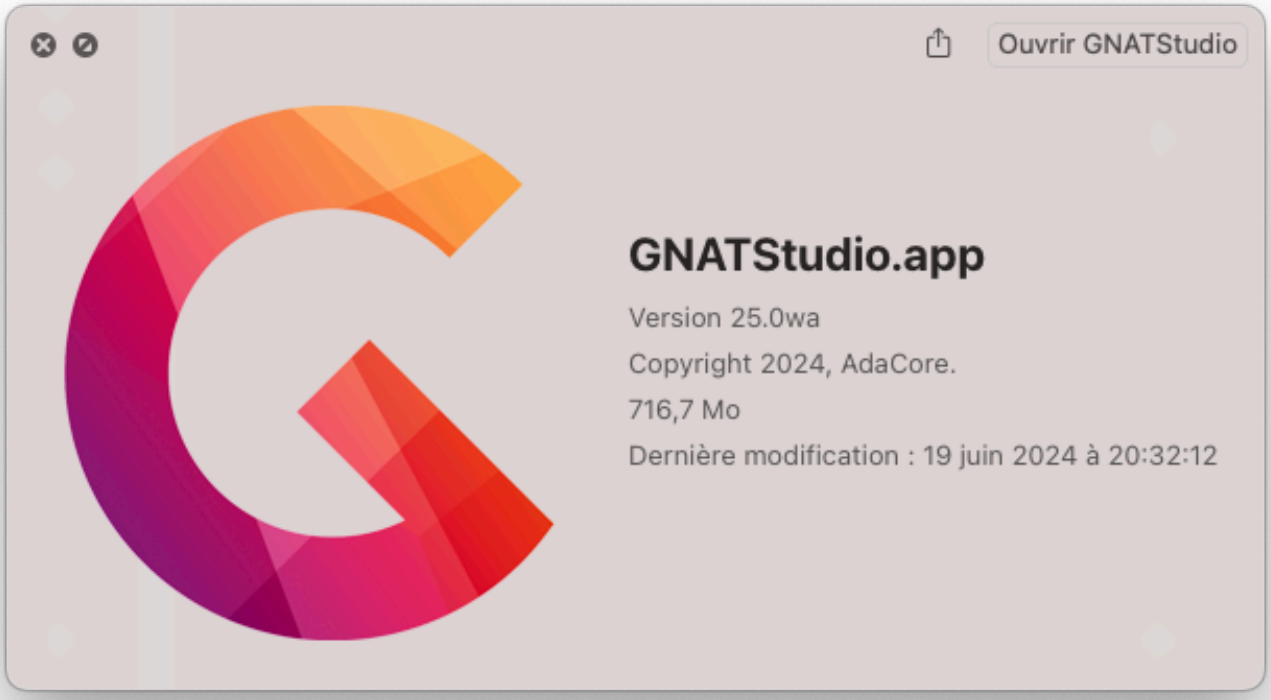
Pour positionner les variables d'environnement GTK et GNATStudio, nous ne pouvons pas utiliser le script shell précédent car le système de sécurité GateKeeper de macOS l'interdit. Nous allons donc construire un lanceur en Ada qui va positionner les variables nécessaires avant d'exécuter le programme gnatstudio.

```
% cd $xnadasrc
% unzip $xnarchive/xnadalib-2024-diff/gnatstudio_launcher.zip
% cd gnatstudio_launcher
% gprbuild -p -P gnatstudio_launcher.gpr

# Ajoutons le launcher dans le bundle
% cd $xnadasrc
% cd gs_bundle/_build/Applications/GNATStudio.app/Contents/MacOS
% mv gnatstudio_launcher gnatstudio
% cp -p $xnadasrc/gnatstudio_launcher/bin/gnatstudio_launcher .
```

L'application est créée dans le dossier *gs_bundle/_build/Applications*. L'application se lance de façon classique avec un double clic ou avec la commande suivante :

```
% open -a $xnadasrc/gs_bundle/_build/Applications/GNATStudio.app --stdout=`tty` --stderr=`tty`
```



Pascal Pignard, juin 2020, décembre 2022, juillet 2023, septembre 2024.
http://blady.chez.com