Installer ou construire GNATStudio 23.0w 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 :
20221119-Applications-GNATStudio.dmg,
depuis le site de Source Forge sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2022-monterey.

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 12.1 (*/opt/gcc-12.1.0/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.pagesperso-orange.fr/a_savoir.html#gnat

2. Configuration générale

Configuration : macOS 12.6, Xcode 13.4.1, GNAT FSF 12.1 (depuis Alire).

Voir leur installation sur Blady :

- macOS et Xcode : blady.pagesperso-orange.fr/liens.html#macosx
- GNAT : blady.pagesperso-orange.fr/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 3.24.33.

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-2022-diff.tgz* sur blady.pagesperso-orange.fr/telechargements/gtkada/xnadalib-2022-diff.tgz.

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=2022
% cd $xnadabase
% sudo mkdir src-$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
...
PATH does not contain .../src-2022/.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 :
% tar xzf $xnarchive/xnadalib-2022-diff.tgz
% cd $xnadasrc/config
% patch -p0 < $xnarchive/xnadalib-2022-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-42.0
- atk-2.36.0
- autoconf-2.71
- autoconf-archive-2021.02.19
- automake-1.16.3
- bison-3.7.6
- cairo-1.17.6
- cmake-3.20.0
- flex-2.6.4
- fontconfig-2.13.1
- freetype-2.11.1
- fribidi-1.0.11
- gdk-pixbuf-2.42.8
- gettext-0.21
- glib-2.72.0
- gobject-introspection-1.72.0
- gtk-doc-1.33.2
- gtk-mac-integration-3.0.1
- gtk-osx-docbook-1.3
- gtk+-3.24.33
- harfbuzz-4.1.0
- hicolor-icon-theme-0.17
- icu4c-68_2-src
- intltool-0.51.0
- itstool-2.0.6
- jpegsrc.v9e
- libepoxy-1.5.4
- libffi-3.3
- libpng-1.6.37
- librsvg-2.54.0
- libtool-2.4.6
- libxml2-2.9.12
- libxslt-1.1.34
- m4-1.4.18
- make-4.3
- openssl-1.1.1n
- pango-1.50.6
- pcre-8.45
- pixman-0.40.0
- pkg-config-0.29.2
- Pygments-2.9.0
- Python-3.10.2
- readline-8.1
- tiff-4.2.0
- util-macros-1.19.3
- xz-5.2.5
- zlib-1.2.12

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 1.20.0.

Saisir les commandes suivantes dans le Terminal :

```
% xnadabase=/usr/local
% version=2022
% 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 3.40.0.

Saisir les commandes suivantes dans le Terminal :

```
% xnadabase=/usr/local
% version=2022
% 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=2022
% 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

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 3.10.2.

Saisir la commande suivante dans le Terminal :

```
% xnadabase=/usr/local
% version=2022
% 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
```

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-20220512

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=2022
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadalib-$version
% xnarchive=$HOME/Downloads

# Activation du compilateur Ada GNAT FSF 12.1 (depuis Alire)
% PATH=$HOME/.config/alire/cache/dependencies/gprbuild_22.0.1_b1220e2b/bin:$HOME/.config/alire/
cache/dependencies/gnat_native_12.1.2_587b912f/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 $xnadasrc
% tar xzf $xnarchive/gnatstudio-sources-x86_64-linux.tar.gz
```

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

```
% cd $xnadainst
% cp -p $HOME/.config/alire/cache/dependencies/gnat_native_12.1.2_587b912f/lib/gcc/x86_64-apple-
darwin19.6.0/12.1.0/adalib/libgnarl-12.dylib lib
% cp -p $HOME/.config/alire/cache/dependencies/gnat_native_12.1.2_587b912f/lib/gcc/x86_64-apple-
darwin19.6.0/12.1.0/adalib/libgnat-12.dylib lib
% cp -p $HOME/.config/alire/cache/dependencies/gnat_native_12.1.2_587b912f/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 23.0w.

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

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gtkada-23.0w-20220508-165B9-src.tar.gz
% cd gtkada-23.0w-20220508-165B9-src
% ./configure --prefix=$xnadainst --enable-build=Debug
% make -w
% make 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 23.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/xmlada-23.0w-20220414-1645D-src.tar.gz
% cd xmlada-23.0w-20220508-16164-src
% ./configure --prefix=$xnadainst --enable-build=Debug
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install
```

10. 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 23.0w.

Dépendance : XMLAda (optionnel).

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/templates_parser-23.0w-20220414-1661B-src.tar.gz
% cd templates_parser-23.0w-20220508-16328-src
% make DEBUG=true prefix=$xnadainst setup
% ADAFLAGS="-gnatwn -gnatU" make
% make 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 23.0w.

Dépendance : XMLAda.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gprbuild-23.0w-20220414-1643D-src.tar.gz
% cd gprbuild-23.0w-20220511-1637E-src
% make BUILD=debug prefix=$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 23.0w.

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 $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gnatcoll-core-23.0w-20220414-16553-src.tar.gz
% cd gnatcoll-core-23.0w-20220508-162EC-src
% make BUILD=DEBUG prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl Bindings
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gnatcoll-bindings-23.0w-20220414-165E6-src.tar.gz

# GNATColl Bindings gmp
% cd $xnadasrc
% cd gnatcoll-bindings-23.0w-20220508-1655B-src
% cd gmp
% CFLAGS=`pkg-config gmp --cflags` LDFLAGS=`pkg-config gmp --libs` python3 ./setup.py build --debug --gpr-opts=-gnatwn
```

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

# GNATColl Bindings iconv
% cd $xnadasrc
% cd gnatcoll-bindings-23.0w-20220508-1655B-src
% cd iconv
% python3 ./setup.py build --debug --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl Bindings python3
% cd $xnadasrc
% cd gnatcoll-bindings-23.0w-20220508-1655B-src
% cd python3
% python3 ./setup.py build --debug --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl DB
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gnatcoll-db-23.0w-20220414-16134-src.tar.gz

# GNATColl DB sql
% cd $xnadasrc
% cd gnatcoll-db-23.0w-20220508-1637E-src
% cd sql
% make BUILD=DEBUG prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB sqlite
% cd $xnadasrc
% cd gnatcoll-db-23.0w-20220508-1637E-src
% cd sqlite
% make BUILD=DEBUG prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB db2ada
% cd $xnadasrc
% cd gnatcoll-db-23.0w-20220508-1637E-src
% cd gnatcoll_db2ada
% make BUILD=DEBUG DB_BACKEND=db prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB xref
% cd $xnadasrc
% cd gnatcoll-db-23.0w-20220508-1637E-src
% cd xref
% make BUILD=DEBUG prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB gnatinspect
% cd $xnadasrc
% cd gnatcoll-db-23.0w-20220508-1637E-src
```

```
% cd gnatinspect
% make BUILD=DEBUG 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 23.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% tar xzf ./gnatstudio-sources-x86_64-linux/vss-23.0w-20220508-16294-src.tar.gz
% cd vss-23.0w-20220508-16294-src
% make -w
% make install 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 23.0w.

Dépendance : GTKAda pour la version Glib, sans sinon.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/spawn-23.0w-20220508-162E0-src.tar.gz
% cd spawn-23.0w-20220508-162E0-src
% gprbuild -p -P gnat/spawn_glib.gpr -XOS=osx -XBUILD_MODE=dev
-XSPAWN_WARN_ERRORS=false
% gprinstall --prefix=$xnadainst -p -P gnat/spawn_glib.gpr  -XOS=osx -XBUILD_MODE=dev
-XSPAWN_WARN_ERRORS=false
% gprbuild -p -P gnat/spawn.gpr -XOS=osx -XBUILD_MODE=dev -XSPAWN_WARN_ERRORS=false
% gprinstall --prefix=$xnadainst -p -P gnat/spawn.gpr  -XOS=osx -XBUILD_MODE=dev
-XSPAWN_WARN_ERRORS=false
```

15.Construire Libadalang avec 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 23.0w.

Dépendance : Python3 avec plusieurs utilitaires dont Mako, GNATColl core, GnatColl gmp, GNATColl iconv, XMLAda, GPR.

Libadalang est bibliothèque pour analyser la sémantique du langage Ada.

Site web : github.com/AdaCore/libadalang.

La version installée est 23.0w.

Dépendance : Python3 avec plusieurs utilitaires dont Mako, Langkit, GNATColl core, GnatColl gmp, GNATColl iconv, XMLAda, GPR.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/langkit-23.0w-20220414-16178-src.tar.gz
% cd langkit-23.0w-20220511-1638A-src
% pip3 install .

% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/libadalang-23.0w-20220414-16466-src.tar.gz
% cd libadalang-23.0w-20220511-16354-src
% python3 -mvenv env
% source env/bin/activate
% pip3 install -r REQUIREMENTS.dev
% ln -s $xnadasrc/langkit-23.0w-20220511-1638A-src langkit
% cd langkit
% pip3 install .
% LIBRARY_PATH=$xnadainst/lib python3 manage.py build-langkit-support --library-types=static,static-pic,relocatable
% LIBRARY_PATH=$xnadainst/lib python3 manage.py install-langkit-support --library-types=static,static-pic,relocatable $xnadainst
% cd ..
% python3 manage.py generate
% LIBRARY_PATH=$xnadainst/lib python3 manage.py build --library-types=static,static-pic,relocatable
% LIBRARY_PATH=$xnadainst/lib python3 manage.py install $xnadainst --library-types=static,static-pic,relocatable
% 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
```

16.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 23.0w.

Dépendance : Libadalang.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/libadalang-tools-23.0w-20220414-16646-src.tar.gz
% cd libadalang-tools-23.0w-20220512-16327-src
% LIBRARY_PATH=$xnadainst/lib make lib
% LIBRARY_PATH=$xnadainst/lib make bin
% LIBRARY_PATH=$xnadainst/lib make install-lib DESTDIR=$xnadainst
% LIBRARY_PATH=$xnadainst/lib make install-bin-strip DESTDIR=$xnadainst
```

17.Construire GPR2

GPR2 est la nouvelle bibliothèque d'analyse des fichiers projets GPR.

Site web : github.com/AdaCore/gpr.

La version installée est 23.0w.

Dépendance : GNATColl, Langkit, GPRConfig KB, Python3.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gpr2-edge-23.0w-20220414-1619D-src.tar.gz

% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gprconfig-kb-23.0w-20220414-16437-src.tar.gz
% cd gpr2-edge-23.0w-20220512-1618A-src
% cd src/kb
% ln -s $xnadasrc/gprconfig-kb-23.0w-20220512-161E7-src/db gprconfig_kb

% cd $xnadasrc
% cd langkit-23.0w-20220511-1638A-src
% pip3 install .

% cd $xnadasrc
% cd gpr2-edge-23.0w-20220512-1618A-src
% LIBRARY_PATH=$xnadainst/lib make -w
# Modification makefile : ajout install-libs
% patch -p0 < $xnarchive/xnadalib-2022-diff/gpr2-makefile.diff
% LIBRARY_PATH=$xnadainst/lib make -w install-libs
```

18.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/xmlada.

La version installée est 23.0w.

Dépendance : Libadalang, Libadalang-tools, VSS, Gnatdoc, GPR2, Spawn.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gnatdoc4-23.0w-20220512-161F4-src.tar.gz
% tar xzf ./gnatstudio-sources-x86_64-linux/als-23.0w-20220512-161DF-src.tar.gz
% cd als-23.0w-20220512-161DF-src
# modifications code source et makefile
% patch -p0 < $xnarchive/xnadalib-2022-diff/als.diff
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-23.0w-20220512-161DF-src/
subprojects/stubs:$xnadasrc/gnatdoc4-23.0w-20220512-161F4-src/gnat:$GPR_PROJECT_PATH make
-w
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-23.0w-20220512-161DF-src/
subprojects/stubs:$xnadasrc/gnatdoc4-23.0w-20220512-161F4-src/gnat:$GPR_PROJECT_PATH make
-w install prefix=$xnadainst
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-23.0w-20220512-161DF-src/
subprojects/stubs:$xnadasrc/gnatdoc4-23.0w-20220512-161F4-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-build
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-23.0w-20220512-161DF-src/
subprojects/stubs:$xnadasrc/gnatdoc4-23.0w-20220512-161F4-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-install prefix=$xnadainst
```

Ajuster la localisation des bibliothèques dynamiques :

```
% cd $xnadainst
% install_name_tool -add_rpath @executable_path/../lib bin/ada_language_server
% install_name_tool -change /Users/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-
darwin/gcc/install/lib/libc++.6.dylib @rpath/libstdc++.6.dylib bin/ada_language_server
% install_name_tool -add_rpath @executable_path/../lib bin/codec_test
% install_name_tool -add_rpath @executable_path/../lib bin/tester-run
```

19.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 23.0w.

Dépendance : GTKAda, PyGobject, PyCairo, GNATColl, VSS, Spawn, GPR2, Libadalang.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf ./gnatstudio-sources-x86_64-linux/gnatstudio-23.0w-20220512-162BF-src.tar.gz
% cd gnatstudio-23.0w-20220512-162BF-src
# modifications code source
% patch -p0 < $xnarchive/xnadalib-2022-diff/gnatstudio.diff
% PYTHON=python3 ./configure --prefix=$xnadainst
% make -w PYTHON=python3
% make -w PYTHON=python3 install
```

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/gnat_compare
```

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

```
% export DYLD_LIBRARY_PATH="$xnadainst/lib"
```

Variables d'environnement pour GNATStudio :

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

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

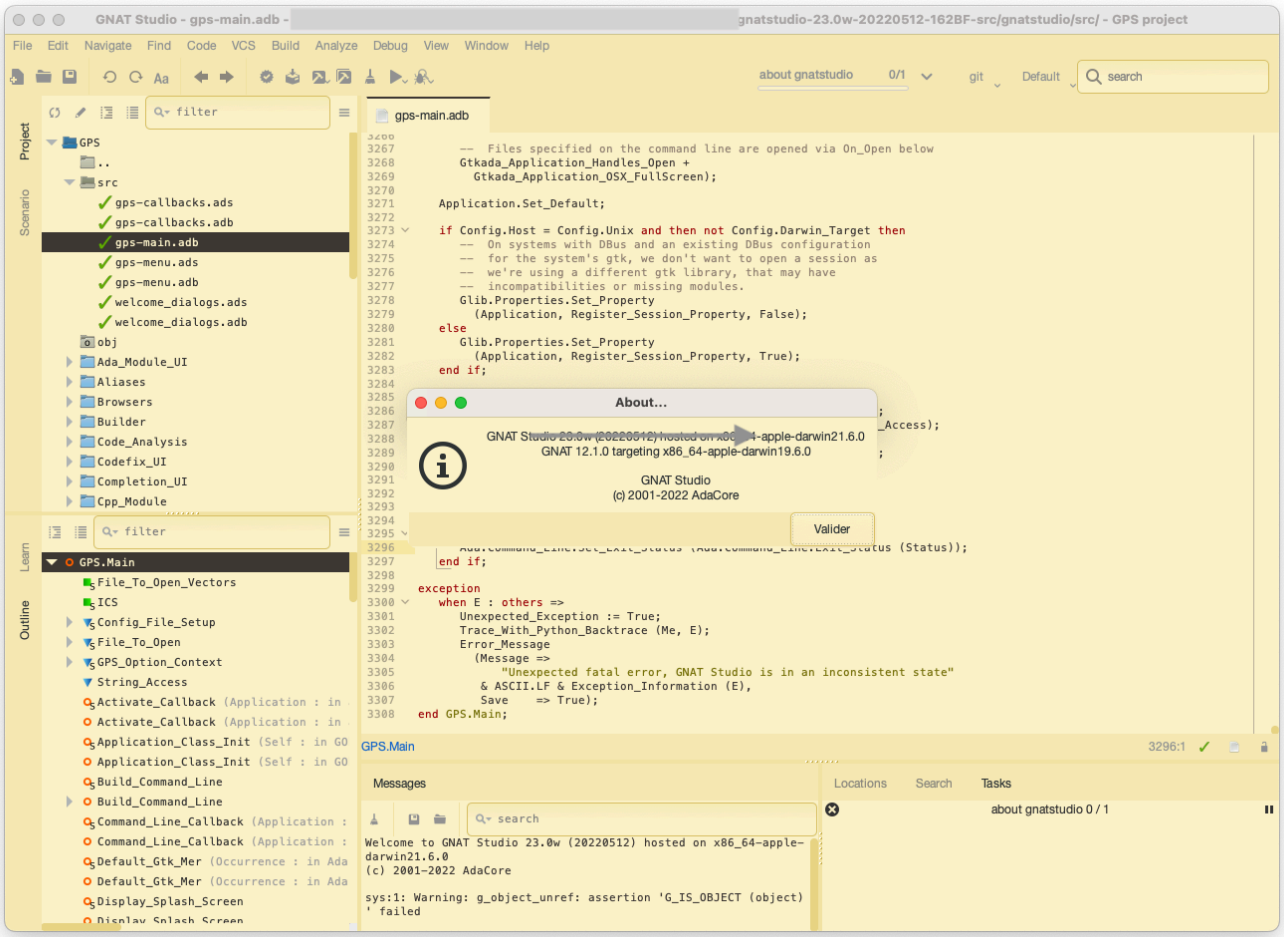
- *GNATSTUDIO_HOME* (optionnel) : indique au programme où trouver le dossier des préférences, par défaut *$HOME*

```
% export GNATSTUDIO_HOME=$xnadasrc/gnatstudio-23.0w-20220512-162BF-src
```

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-23.0w-20220512-162BF-src/gnat_launcher.sh $xnadainst/bin/gnatstudio
```

Et voilà le résultat :



20.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=2022
% 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-2022 (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 -b proposal02
% cd gtk-mac-bundler
% make 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
% unzip $xnarchive/xnadalib-2022-diff/gs_bundle_conf.zip
```

d) Construire le lanceur

Pour positionner les variables d'environnement GTK et GNATStudio, nous ne pouvons pas utiliser le script shell du §19 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-2022-diff/gnatstudio_launcher.zip
% cd gnatstudio_launcher
% gprbuild -p -P gnatstudio_launcher.gpr
```

e) 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

# L'application est crée, quelques références de bibliothèques dynamiques doivent être ajustées
% install_name_tool -change $xnadasrc/.new_local/share/pyenv/versions/3.10.2/lib/libpython3.10.dylib
@executable_path/../Resources/lib/libpython3.10.dylib $xnadasrc/gs_bundle/_build/Applications/
gnatstudio.app/Contents/Resources/lib/python3.10/site-packages/libxml2mod.so
% install_name_tool -change /Users/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-
darwin/gcc/install/lib/libstdc++.6.dylib @executable_path/../Resources/lib/libstdc++.6.dylib $xnadasrc/
_build/20221104-Applications/gnatstudio.app/Contents/MacOS/ada_language_server

# Ajoutons le laucher
% cp -p $xnadasrc/gnatstudio_launcher/bin/gnatstudio_launcher $xnadasrc/gs_bundle/_build/
Applications/GNATStudio.app/Contents/MacOS
```

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.
http://blady.pagesperso-orange.fr