

Installer ou construire GNATStudio 2026.2 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 comment installer l'application GNATStudio pour macOS prête à l'emploi et aussi pouvoir la construire à partir des sources de ses composants.

Sommaire

1.	Installer GnatStudio	3
2.	Configuration générale pour la construction	4
3.	Construire GTK	4
4.	Construire les extensions GTK en Python	8
5.	Construire GMP	10
6.	Construire Pip, Setuptools (Distutils) et VirtualEnv	10
7.	Configuration de l'environnement en Ada	11
8.	Construire GTKAda	12
9.	Construire XMLAda (Schema, DOM, SAX, Unicode)	12
10.	Construire Templates-Parser	13
11.	Construire GPR (inclut dans GPRBuild)	13
12.	Construire GNATColl (Core, Bindings et DB)	14
13.	Construire VSS Text et Extra	16
14.	Construire Spawn et Spawn_glib	16
15.	Construire AdaSAT	17
16.	Construire PrettierAda	17
17.	Construire Langkit	17
18.	Construire GPR2	18
19.	Construire Libadalang	19
20.	Construire Libadalang Tools	19
21.	Construire LAL-Refactor	20
22.	Construire GNATFormat	20
23.	Construire XDiff	20
24.	--Construire GNATHub	21
25.	--Construire Ada_libfswatch	21
26.	Construire Ada language server	22
27.	Construire GNATStudio	23
28.	Construire l'application macOS GNATStudio	25

1. Installer GnatStudio

Télécharger le fichier suivant sur le bureau du Mac :
20260625-Applications-GNATStudio.dmg,
depuis le site de Source Forge [/sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2026-ventura](https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2026-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 modifiant *GS_GNAT_PATH* et *GS_GPR_PATH* dans le fichier *Info.plist* de l'application *GNATStudio.app*. Attention, *Info.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_lancer  
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#gmat

2. Configuration générale pour la construction

Configuration : macOS 13.7, Xcode 14.3 et GNAT FSF 15.2 (depuis Alire).

Voir leur installation sur Blady :

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

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 3.24.52.

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 gitlab.gnome.org/GNOME/gtk-osx/-/wikis/home.

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-2026-diff.zip* sur blady.chez.com/telechargements/gtkada/xnadalib-2026-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=$PWD
% version=2026
% sudo mkdir src-$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

% bash $xnarchive/gtk-osx-setup.sh
...
PATH does not contain .../src-2026/.new_local/bin. You probably want to fix that.
...
```

Cette commande installe *jhbuild* dans le dossier *Source* et crée 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 contenus dans l'archive pour toutes les modifications des constructions suivantes :

```
% unzip $xnarchive/xnadalib-2026-diff.zip
```

```
% cd $xnadasrc/config
```

```
# Modifications jhbuildrc-custom.diff
```

```
% cd $xnadasrc
```

```
...
```

```
% jhbuild bootstrap-gtk-osx
```

```
...
```

```
% jhbuild build meta-gtk-osx-bootstrap
```

```
...
```

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

```
...
```

Des programmes 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 avec la commande :

```
% cd $xnadainst/bin; XDG_DATA_DIRS=$xnadainst/share ./gtk3-demo; cd -
```

Liste des modules installés :

- adwaita-icon-theme-50.0
- atk-2.38.0
- autoconf-2.73
- autoconf-archive-2024.10.16
- automake-1.18
- bison-3.8.2
- cairo-1.18.4
- cmake-4.3.1
- flex-2.6.4
- fontconfig-2.17.1
- freetype-2.14.3
- fridibi-1.0.16
- gdk-pixbuf-2.44.5
- gettext-1.0
- glib-2.88.0
- gobject-introspection-1.86.0
- gtk-3.24.52
- gtk-doc-1.36.0
- gtk-mac-integration-3.0.2
- gtk-osx-docbook-1.3
- harfbuzz-13.2.1
- hicolor-icon-theme-0.18
- icu
- intltool-0.51.0
- libepoxy-1.5.10
- libffi-3.5.2
- libiconv-1.19
- libjpeg-turbo-3.1.4.1
- libpng-1.6.56
- librsvg-2.62.1
- libtool-2.5.4
- libunistring-1.4
- libxml2-2.15.2
- m4-1.4.21
- make-4.4
- nasm-nasm-3.01
- pango-1.57.1
- pcre2-10.47
- pixman-0.46.4
- pkgconf-2.5.1
- tiff-4.7.1
- xz-5.8.2
- zlib-1.3.2

4. Construire les extensions GTK en Python

a) Construire Python3

Il s'agit d'un interpréteur de scripts en langage de programmation Python..

Site web : www.python.org.

La version installée est 3.13.2 (la version courante 3.14 ne fonctionne pas avec GNATStudio).

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% version=2026
% xnadasrc=$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
```

Récupérer le module set contenant la version Python 3.13 (voir la configuration utilisée en [2025](#))

```
% jhbuild build python3
```

b) 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 1.27.0.

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% version=2026
% xnadasrc=$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
```

Récupérer le module set contenant la version Python 3.13 (voir la configuration utilisée en [2025](#))

```
% jhbuild build pycairo
```

c) 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 3.51.0.

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
```

```
% xnadabase=$PWD
% version=2026
% xnadasrc=$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

# Récupérer le module set contenant la version Python 3.13 (voir la configuration utilisée en 2025)
% jhbuild build pygobject3
```

d) Liste des modules installés

- openssl-3.6.2
- pycairo-1.27.0
- pygobject-3.5.0
- Python-3.13.2
- readline-8.3

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 6.3.0.

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% version=2026
% xnadasrc=$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, Setuptools (Distutils) et VirtualEnv

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 24.3.1 puis 25.1.1.

Saisir la commande suivante dans un nouveau Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% version=2026
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadolib-$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 setuptools
% pip3 install --upgrade pip

# Utile pour les scripts branchés uniquement sur la commande 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/](https://github.com/AdaCore/gnatstudio/releases/tag/gnatstudio-2026.2-20260409)

[gnatstudio-2026.2-20260409](https://github.com/AdaCore/gnatstudio/releases/tag/gnatstudio-2026.2-20260409)

Saisir les commandes suivantes dans le Terminal :

```
% cd /usr/local # obligatoire pour les bibliothèques dynamiques
% xnadabase=$PWD
% tar xzf gnatstudio-sources-x86_64-linux.tar.gz
```

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

```
% cd $xnadabase
% version=2026
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadolib-$version
% xnarchive=$xnadabase/gnatstudio-sources-x86_64-linux

# Activation du compilateur Ada GNAT FSF 15.2 (depuis Alire)
% PATH=$HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/bin:$HOME/.local/share/alire/toolchains/gprbuild_25.0.1_d8b2ccb7/bin:$PATH

# Contournement d'un manque lors de la construction de GNAT 15 pour macOS
% export CFLAGS="-isysroot /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk"
% export LDFLAGS="-isysroot /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk"

# Modification $HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/lib/gcc/x86_64-apple-darwin22.6.0/15.2.0/include-fixed/AvailabilityInternal.h

# Activation de la bibliothèque GTK-OSX
% PATH=$xnadainst/bin:$PATH

# Activation de la bibliothèque Ada
% export GPR_PROJECT_PATH=$xnadainst/share/gpr
```

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

```
% cd $xnadainst
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/lib/gcc/x86_64-apple-darwin22.6.0/15.2.0/adalib/libgnarl-15.dylib lib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/lib/gcc/x86_64-apple-darwin22.6.0/15.2.0/adalib/libgnat-15.dylib lib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/lib/libstdc++.6.dylib lib
% cp -p $HOME/.local/share/alire/toolchains/gnat_native_15.2.1_fffe07e8/lib/libgcc_s.1.1.dylib lib
```

8. Construire GTKAda

GTKAda est la boîte 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 27.0w.

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

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gtkada-src.tar.gz
% cd gtkada-27.0w-20260409-163D3-src
# Modification configure, Makefile.in
% ./configure --prefix=$xnadainst
% make -w
% make -w install prefix=$xnadainst PRJDIR=share/gpr
```

Des programmes de test et démo donnent plein de situations d'emploi des objets GTKAda avec leur description, le code source Ada correspondant et une démonstration du résultat avec la commande :

```
% cd $xnadainst/share/examples/gtkada/testgtk; XDG_DATA_DIRS=$xnadainst/share ./testgtk; cd -
```

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

XMLAda est une boîte à outil pour analyser les fichiers XML.

Site web : github.com/AdaCore/xmlada.

La version installée est 27.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/xmlada-src.tar.gz
% cd xmlada-27.0w-20260324-16261-src
# Modification xmlada_shared.gpr.in et *.gpr (add LD_FLAGS support)
% ./configure --prefix=$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 27.0w.

Dépendance : XMLAda.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/templates_parser-src.tar.gz
% cd templates_parser-27.0w-20260324-1652A-src
% make -w prefix=$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 27.0w.

Dépendance : XMLAda.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gprbuild-src.tar.gz
% cd gprbuild-27.0w-20260324-1674D-src
# Modification gpr/gpr.gpr '-(add CFLAGS and LDFLAGS support)
% make -w prefix=$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 27.0w.

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 $xnarchive/gnatcoll-core-src.tar.gz
% cd gnatcoll-core-27.0w-20260324-16311-src
% ADAFLAGS="-gnatwn -gnatU" make -w prefix=$xnadainst
% make -w install prefix=$xnadainst

# GNATColl Bindings
% cd $xnadasrc
% tar xzf $xnarchive/gnatcoll-bindings-26.0w-20250407-1614B-src.tar.gz

# GNATColl Bindings gmp
% cd $xnadasrc
% cd gnatcoll-bindings-27.0w-20260324-16166-src
% cd gmp
% CFLAGS="-pkg-config gmp --cflags` $CFLAGS" LDFLAGS="-pkg-config gmp --libs` $LDFLAGS"
python3 ./setup.py build --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl Bindings iconv
% cd $xnadasrc
% cd gnatcoll-bindings-27.0w-20260324-16166-src
% cd iconv
# Ne pas prendre Liblconv de macOS mais celui de GTK-OSX (option --force-libiconv n'est valable que
pour Linux sinon -liconv pour les autres)
% CFLAGS="-I$xnadainst/include $CFLAGS" LDFLAGS="-L$xnadainst/lib $LDFLAGS" python3 ./
setup.py build --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl Bindings python3
% cd $xnadasrc
% cd gnatcoll-bindings-27.0w-20260324-16166-src
% cd python3
% python3 ./setup.py build --gpr-opts=-gnatwn
% python3 ./setup.py install --prefix=$xnadainst

# GNATColl DB
% cd $xnadasrc
% tar xzf $xnarchive/gnatcoll-db-src.tar.gz
```

```
# GNATColl DB sql
% cd $xnadasrc
% cd gnatcoll-db-27.0w-20260324-165F4-src
% cd sql
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB sqlite
% cd $xnadasrc
% cd gnatcoll-db-27.0w-20260324-165F4-src
% cd sqlite
% make -w prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB db2ada
% cd $xnadasrc
% cd gnatcoll-db-27.0w-20260324-165F4-src
% cd gnatcoll_db2ada
% make DB_BACKEND=db prefix=$xnadainst setup
% make -w GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB xref
% cd $xnadasrc
% cd gnatcoll-db-27.0w-20260324-165F4-src
% cd xref
% make -w prefix=$xnadainst setup
# Ne pas prendre Liblconv de macOS mais celui de GTK-OSX
% CFLAGS="-I$xnadainst/include $CFLAGS" LDFLAGS="-L$xnadainst/lib $LDFLAGS" make -w
GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install

# GNATColl DB gnatinsect
% cd $xnadasrc
% cd gnatcoll-db-27.0w-20260324-165F4-src
% cd gnatinsect
% make -w prefix=$xnadainst setup
% CFLAGS="-I$xnadainst/include $CFLAGS" LDFLAGS="-L$xnadainst/lib $LDFLAGS" make -w
GPRBUILD_OPTIONS="-gnatwn -gnatU"
% make -w install
```

13. Construire VSS Text et Extra

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

Site web : github.com/AdaCore/vss-text et github.com/AdaCore/vss-extra.

La version installée est 27.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/vss-text-src.tar.gz
% cd vss-text-27.0w-20260324-1616F-src
# Modification *.gpr '-(add LDFLAGS support)
% make -w build-all-libs
% make -w install-all-libs PREFIX=$xnadainst
```

```
% cd $xnadasrc
% tar xzf $xnarchive/vss-extra-src.tar.gz
% cd vss-extra-27.0w-20260409-167B2-src
# Modification *.gpr '-(add LDFLAGS support)
% make -w build-all-libs
% make -w install-all-libs PREFIX=$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 27.0w.

Dépendance : GTKAda pour la version Glib, sans sinon.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/spawn-src.tar.gz
% cd spawn-27.0w-20260409-16169-src
# Modification *.gpr '-(add CFLAGS et LDFLAGS support)
% gprbuild -p -P gnat/spawn_glib.gpr -XOS=osx -XSPAWN_WARN_ERRORS=false
% gprinstall --prefix=$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 27.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/adaset-src.tar.gz
% cd adaset-27.0w-20260324-1629D-src
# Modification *.gpr '(add LDFLAGS support)
% make -w all-libs BUILD_MODE=prod
% make -w install BUILD_MODE=prod INSTALL_DIR=$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 27.0w.

Dépendance : GNATColl-Core, VSS.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/prettier-ada-src.tar.gz
% cd prettier-ada-27.0w-20260324-166DC-src
# Modification *.gpr '(add LDFLAGS support)
% make -w all BUILD_MODE=prod
% make -w install-all BUILD_MODE=prod PREFIX=$xnadainst
```

17. Construire Langkit

Langkit est une boîte à outil qui rend plus facile la création d'analyseurs syntaxiques et sémantiques.

Site web : github.com/AdaCore/langkit.

La version installée est 27.0w.

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

Saisir les commandes suivantes dans le Terminal :

(Une connexion Internet est requise)

```
% cd $xnadasrc
% tar xzf $xnarchive/langkit-src.tar.gz
% cd langkit-27.0w-20260324-164EA-src
# Modification *.gpr '(add LDFLAGS support)
```

1. Blady

2 mai 2026 à 18:53:11

gpr2.gpr:29:1182:

"gpr2.build.view_tables.update_sources_list" is not a unit of this project

gpr2.gpr:29:2933:

"gpr2.tree_internal.load_autoconf"

2. Blady

8 mai 2026 à 15:14:45

cannot generate code for file metrics-line_counting.adb (missing subunits)

gpr2-build-

view_tables.adb:1553:04: warning: subunit

"GPR2.BUILD.VIEW_TABLES.UPDATE_SOURCES_LIST" in file

"gpr2-build-view_tables-update_sources_list.adb" not found [enabled by default]

```
% (cd langkit; git clone https://github.com/AdaCore/adasat)
% pip3 install .
```

```
% LIBRARY_PATH=$xnadainst/lib python3 manage.py make --no-mypy --library-types=static,static-pic, relocatable
```

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

```
% LIBRARY_PATH=$xnadainst/lib python3 -m langkit.scripts.lkm install -c lkt/langkit.yaml $xnadainst --library-types=static,static-pic, relocatable --disable-all-mains
% pip3 install lkt/build/python
```

18. Construire GPR2

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

Site web : github.com/AdaCore/gpr.

La version installée est 27.0w.

Dépendance : GNATColl, Langkit, GPRConfig KB, Python3.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
```

```
% tar xzf $xnarchive/gprconfig-kb-src.tar.gz
```

```
% tar xzf $xnarchive/gpr2-src.tar.gz
```

```
% cd gpr2-27.0w-20260324-164D3-src
```

```
# Modification makefile
```

```
# Modification *.gpr '-(add CFLAGS and LDFLAGS support) et gpr2.gpr ajout interfaces manquantes
```

```
% make -w prefix=$xnadainst setup GPR2KBDIR=$xnadasrc/gprconfig-kb-27.0w-20260324-1617F-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)"
```

1 Modification de \$xnadainst/share/gpr/gpr2.gpr pour retirer 2 éléments (**separate**) :

- gpr2.build.view_tables.update_sources_list
- gpr2.tree_internal.load_autoconf

2 Ajout dans \$xnadainst/include/gpr2.relocatable de 2 éléments (**separate**) depuis le dossier gpr2.static:

- gpr2-build-view_tables-update_sources_list.adb
- gpr2-tree_internal-load_autoconf.adb

19. Construire Libadalang

Libadalang est bibliothèque pour analyser la sémantique du langage Ada.

Site web : github.com/AdaCore/libadalang.

La version installée est 27.0w.

Dépendance : Python3 avec plusieurs utilitaires dont Mako, Langkit, GNATColl core, GnatColl gmp, GNATColl iconv, GPR, GPR2, AdaSAT, PrettierAda.

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

```
% cd $xnadasrc
% tar xzf $xnarchive/libadalang-src.tar.gz
% cd libadalang-27.0w-20260324-1687D-src

% ln -s $xnadasrc/langkit-27.0w-20260324-164EA-src langkit

# Change la version de autopep8 dans requirements-pypi.txt
% python3 -mvenv env
% source env/bin/activate
% pip install -r requirements-github.txt
% pip install -r requirements-pypi.txt

% PYTHONPATH=$xnadainst/lib/python3.14/site-packages DYLD_LIBRARY_PATH=$xnadainst/lib
python3 -m langkit.scripts.lkm generate
% PYTHONPATH=$xnadainst/lib/python3.14/site-packages LIBRARY_PATH=$xnadainst/lib python3 -m
langkit.scripts.lkm build --library-types=static,static-pic,rellocatable
%PYTHONPATH=$xnadainst/lib/python3.14/site-packages python3 -m langkit.scripts.lkm install
$xnadainst --library-types=static,static-pic,rellocatable

% deactivate
```

20. Construire Libadalang Tools

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

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

La version installée est 27.0w.

Dépendance : Libadalang.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/libadalang-tools-src.tar.gz
% cd libadalang-tools-27.0w-20260324-16542-src
# Modification Makefile
# Modification *.gpr '-(add LDFLAGS support)
% LIBRARY_PATH=$xnadainst/lib OS=osx make lib BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib OS=osx make bin BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib OS=osx make install-lib DESTDIR=$xnadainst BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib OS=osx make install-bin-strip DESTDIR=$xnadainst
BUILD_MODE=prod
```

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 27.0w.

Dépendance : LibadalangTools, VSS.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/lal-refactor-src.tar.gz
% cd lal-refactor-27.0w-20260409-16484-src
# Modification testsuite/ada_drivers/gnat/lal_refactor_test_drivers.gpr
# Modification *.gpr '-(add LDFLAGS support)
% LIBRARY_PATH=$xnadainst/lib make -w all BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib PREFIX=$xnadainst make -w install BUILD_MODE=prod
```

22.Construire GNATFormat

GNATformat est un outil pour formater un code source Ada valide selon le style de codage décrit dans le guide GNAT Coding Style.

Site web : github.com/AdaCore/gnatformat.

La version installée est 27.0w.

Dépendance : Libadalang, PrettierAda, VSS.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/gnatformat-26.0w-src.tar.gz
% cd gnatformat-27.0w-20260409-162F3-src
# Modification *.gpr '-(add LDFLAGS support)
% LIBRARY_PATH=$xnadainst/lib make -w all BUILD_MODE=prod
% LIBRARY_PATH=$xnadainst/lib PREFIX=$xnadainst make -w install BUILD_MODE=prod
```

23.Construire XDiff

XDiff est un binding Ada de la bibliothèque xdiff incluse dans le projet Git.

Site web : github.com/AdaCore/xdiff.

La version installée est 27.0w.

Dépendance : sans.

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/xdiff-src.tar.gz
% cd xdiff-27.0w-20260409-16729-src
# Modification Makefile, *.gpr '-(add LDFLAGS support)
% make -w all BUILD_MODE=release
% make -w install BUILD_MODE=release INSTALL_DIR=$xnadainst
```

24.--Construire GNATHub

(N'est pas utilisé) 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.

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

Saisir les commandes suivantes dans le Terminal :

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

25.--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
```

26.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 27.0w.

Dépendance : GNATColl, Libadalang, Libadalang-tools, VSS, Gnatdoc (source, pas d'installation fournie), LAL_Refactor, GPR2, GNATFormat, Spawn / Spawn_glib, Ada_libfswatch (interne ALS avec un projet bouchon).

Saisir les commandes suivantes dans le Terminal :

```
% cd $xnadasrc
% tar xzf $xnarchive/als-src.tar.gz
% tar xzf $xnarchive/gnatdoc4-src.tar.gz
% cd als-27.0w-20260409-16629-src

# Modification makefile
# Modification *.gpr -(add LDFLAGS support)
# Modification source/ada/lsp-uyils.adb

% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-27.0w-20260409-16629-src/
subprojects/stubs:$xnadasrc/gnatdoc4-27.0w-20260409-1625F-src/gnat:$GPR_PROJECT_PATH make
-w BUILD_MODE=prod VERSION=27.0 BUILD_DATE=20260510
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-27.0w-20260409-16629-src/
subprojects/stubs:$xnadasrc/gnatdoc4-27.0w-20260409-1625F-src/gnat:$GPR_PROJECT_PATH make
-w install prefix=$xnadainst BUILD_MODE=prod VERSION=27.0 BUILD_DATE=20260510

% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-27.0w-20260409-16629-src/
subprojects/stubs:$xnadasrc/gnatdoc4-27.0w-20260409-1625F-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-build BUILD_MODE=prod VERSION=27.0 BUILD_DATE=20260510
% LIBRARY_PATH=$xnadainst/lib GPR_PROJECT_PATH=$xnadasrc/als-27.0w-20260409-16629-src/
subprojects/stubs:$xnadasrc/gnatdoc4-27.0w-20260409-1625F-src/gnat:$GPR_PROJECT_PATH make
-w lsp_client_glib-install prefix=$xnadainst BUILD_MODE=prod VERSION=27.0
BUILD_DATE=202605010
```

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
```

27. 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 2026.2.

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 $xnarchive/gnatstudio-src.tar.gz
% cd gnatstudio-2026.2-20260409-1617D-src
# Modifications code source (Pull Request Github)
% PYTHON=python3 ./configure --prefix=$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
4 % pip3 install pyyaml
```

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"

- GL_TYPELIB_PATH : référence le dossier des descriptions pour GObject Introspection
% export GL_TYPELIB_PATH="$xnadainst/lib/girepository-1.0"
```

5. Blady

21 juin 2026 à 11:20:35

Modification déjà dans la version
26

Variables d'environnement pour Python :

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

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

5

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

```
% export DYLD_FALLBACK_LIBRARY_PATH=/usr/lib:"$bundle_lib:$bundle_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
```

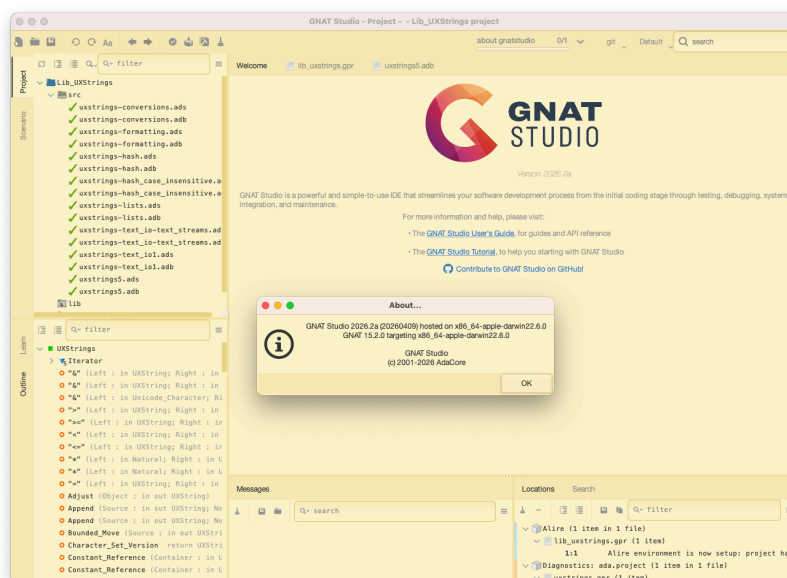
- `GNATSTUDIO_HOME` (optionnel) : indique au programme où trouver le dossier des préférences, par défaut `$HOME`

```
% 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-2026.2-20260409-1617D-src/gnat_launcher.sh $xnadainst/bin/gnatstudio
```

Et voilà le résultat :



28. 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=2026
% xnadasrc=$xnadabase/src-$version
% xnadainst=$xnadabase/xnadolib-$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 crée 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 $xnadasrc
% git clone https://github.com/Blady-Com/gtk-mac-bundler.git
% cd gtk-mac-bundler
% git checkout XNADALIB-2026A-MACOS-13.7-XCODE-14.3
% make -w install
% PATH=$PATH
```

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
% git clone https://github.com/Blady-Com/gs_macos_bundle.git
% cd gs_macos_bundle
% git checkout GNATStudio-2026.2a
```

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_macos_bundle
% jhbuild shell
[JH] % gtk-mac-bundler gnatstudio.bundle
[JH] % exit

# Nécessaire pour setuptools (remplace distutils)
% cp -p "$xnadainst/lib/python3.13/site-packages/setuptools/_vendor/jaraco/text/Lorem ipsum.txt"
$xnadasrc/_build/Applications/GNATStudio.app/Contents/Resources/lib/python3.13/site-packages/
setuptools/_vendor/jaraco/text
```

e) Construire le lanceur

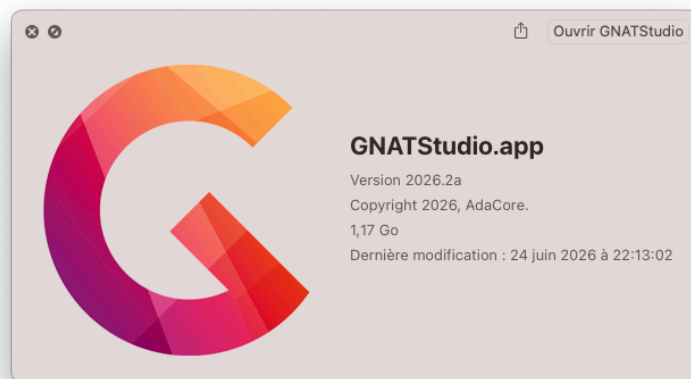
Pour positionner les variables d'environnement GTK et GNATStudio, nous ne pouvons pas utiliser le script shell précédent car le système de sécurité GateKeeper de macOS l'interdit. Nous allons donc construire un lanceur en Ada qui va positionner les variables nécessaires avant d'exécuter le programme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

```
% cd $xnadasrc
% git clone https://github.com/Blady-Com/gnatstudio_launcher.git
% cd gnatstudio_launcher
% git checkout GNATStudio-2026.2a
% gprbuild -P gnatstudio_launcher.gpr

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

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

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



Pascal Pignard, juin 2020, décembre 2022, juillet 2023, septembre 2024, juillet 2025, juin 2026.
<http://blady.chez.com>