Installation de GNAT FSF 16 pour macOS 13

Une seule distribution du compilateur GNAT FSF existe pour macOS :

- le compilateur GNAT FSF 16.1 sur Alire (juillet 2026) pour les languages Ada, C, C++ et Objective C.

Sommaire

1. Installation du compilateur GNAT FSF 16 depuis Alire

a) Le compilateur

Installez auparavant l'utilitaire *alr* pour avoir accès à la bibliothèque Alire, voir sur Blady.

Lancer le Terminal dans une session administrateur et taper les commandes suivantes :

% **alr toolchain --select**
...
If you choose "None", Alire will use whatever version is found in the environment.

ⓘ Currently configured: gnat_native=xx.xx

Please select the gnat version for use with this configuration
1. gnat_native=16.1.0
2. None
3. gnat_aarch64_elf=15.3.1
4. gnat_arm_elf=15.3.1
5. gnat_avr_elf=15.3.1
6. gnat_native=15.3.1
7. gnat_riscv64_elf=15.3.1
8. gnat_x86_64_elf=15.3.1
9. gnat_xtensa_esp32_elf=15.3.1
0. gnat_arm_elf=15.2.1
a. (See more choices...)

Enter your choice index (first is default):
> 1
ⓘ Selected tool version gnat_native=16.1.0

ⓘ Choices for the following tool are narrowed down to releases compatible with just selected gnat_native=16.1.0

ⓘ Currently configured: gprbuild=26.0.1

Please select the gprbuild version for use with this configuration
1. gprbuild=26.0.1
2. None
3. gprbuild=25.0.1
4. gprbuild=22.0.1
5. gprbuild=21.0.2
6. gprbuild=21.0.1

Enter your choice index (first is default):
> 1
ⓘ Selected tool version gprbuild=25.0.1

Le compilateur GNAT et le constructeur de projets GPRbuild s'installent dans le dossier : *$HOME/.local/share/alire/toolchains*.

b) L'environnement de développement intégré

L'éditeur intégré GNAT Studio ne fait pas partie de cette installation, cependant celui de la livraison de 2019 est toujours utilisable, à installer dans un autre dossier (voir sur Blady) puis déclarer l'alias suivant avec la commande suivante, par exemple :

```
% alias gps=/usr/local/adacore/2019/bin/gps
```

Une version de GNAT Studio est néanmoins disponible également sur Blady.

c) Le débuggeur

D'autre part, le debugger gdb n'est pas encore fonctionnel, il est bloqué par le système de surveillance de macOS et provoque cette erreur à l'exécution du programme à débugger :

```
Unable to find Mach task port for process-id 6633: (os/kern) failure (0x5).
 (please check gdb is codesigned - see taskgated(8))
```

Nous allons suivre la procédure décrite sur le blog de Simon Wright pour qu'il fonctionne avec macOS (forward-in-code.blogspot.com/2018/11/mojave-vs-gdb.html).Lancer le Terminal dans une session administrateur et taper les commandes suivantes :

i) Ouvrir l'application *Trousseau d'accès* dans le dossier *Applications -> Utilitaires*. Sélectionner le menu *Trousseau d'accès -> Assistant de certification -> Créer un certificat…*.
Dans la fenêtre qui apparait :
. donner le nom *gdb-cert*,
. le type d'identité à *Racine auto-signée*,
. le type de certificat à *Signature de code*,
. cocher le case *Me laisser ignorer les réglages pas défaut*.
Puis cliquer sur le bouton Continuer plusieurs fois jusqu'à ce qu'apparaisse le panneau *Indiquez l'emplacement du certificat*.

ii) Sélectionner alors *Système*, puis sur le bouton *Créer*. Une fenêtre d'autorisation de modification du trousseau apparait, entrer le mot de passe puis cliquer sur le bouton *Modifier le trousseau*. Le certificat est créé, cliquer alors sur le bouton *Terminer*.

iii) Dans la fenêtre des trousseaux, sélectionner le trousseau *Système* et double-cliquer sur le certificat *gdb-cert*. Dans la fenêtre qui apparait, déployer le triangle *Se fier* et en face de *Lors de l'utilisation de ce certificat* sélectionner *Toujours approuver* puis fermer la fenêtre en cliquant sur sa bulle rouge de fermeture. Une fenêtre d'autorisation de modification des réglages apparait, entrer le mot de passe puis cliquer sur le bouton *Mettre à jour les réglages*.

Quitter l'application Trousseau, ça été long et ce n'est pas encore tout à fait fini. Il nous faut alors redémarrer le Mac (c'est malheureusement nécessaire) puis télécharger le fichier de description gdb.xml à partir de Blady :
blady.pagesperso-orange.fr/telechargements/gnat/gdb.xml

Nous pouvons alors enfin signer *GDB* :

```
$ cd <gnat path>/bin
$ codesign --force --sign gdb-cert --entitlements ~/Downloads/gdb.xml gdb
Password:
```

Une fenêtre s'ouvre *macOS souhaite effectuer des modifications*, saisir nom administrateur et mot de passe puis cliquer sur le bouton *Autoriser* pour autoriser la signature.

Si l'erreur *errSecInternalComponent* s'affiche alors vérifiez que vous êtes bien dans une session administrateur.

d) Adaptation à macOS 13

Le compilateur a été construit pour macOS 15 et compatible avec macOS 26 (nouvelle numérotation, la suivante en faite) ce qui provoque quelques incompatibilités de compilation de programmes en C++ avec notre macOS 13 :

- Error with stdio.h

```
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include/stdio.h:67:8:
**error:** '**FILE**' does not name a type
67 | extern **FILE** *__stdinp;
| **^~~~**
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include/stdio.h:65:1:
**note:** '**FILE**' is defined in header '**<cstdio>**'; this is probably fixable by adding '**#include <cstdio>**'
64 | #include <_stdio.h>
+++ |+#include <cstdio>
```

Correction : nous allons supprimer la surcharge de _stdio.h fait par GNAT sur celle de Xcode.

```
% cd $HOME/.local/share/alire/toolchains/gnat_native_16.1.0_bb6d1434/lib/gcc/x86_64-apple-darwin24.6.0/16.1.0/include-fixed
% mv _stdio.h _stdio.h@0
```

- Erreur avec cstdlib

```
/Users/me/GNAT-16/gnat-x86_64-darwin-16.1.0-rc2/include/c++/16.1.0/cstdlib:147:11:
**error:** '**at_quick_exit**' has not been declared in '**::**'
147 | using ::**at_quick_exit**;
| **^~~~~~~~~~~~~~**
**/Users/me/GNAT-16/gnat-x86_64-darwin-16.1.0-rc2/include/c++/16.1.0/cstdlib:170:11:
**error:** '**quick_exit**' has not been declared in '**::**'
170 | using ::**quick_exit**;
| **^~~~~~~~~~**
```

Correction : mise en commentaire des éléments non déclarés.

```
% cd $HOME/.local/share/alire/toolchains/gnat_native_16.1.0_bb6d1434/include/c++/16.1.0
% diff cstdlib cstdlib@0
--- ./cstdlib@0 2026-06-29 15:26:58
+++ ./cstdlib    2026-07-11 09:53:37
@@ -144,7 +144,7 @@
   using ::atexit;
 #if __cplusplus >= 201103L
 # ifdef _GLIBCXX_HAVE_AT_QUICK_EXIT
-  using ::at_quick_exit;
+//  using ::at_quick_exit;
 # endif
 #endif
   using ::atof;
@@ -167,7 +167,7 @@
   using ::qsort;
 #if __cplusplus >= 201103L
 # ifdef _GLIBCXX_HAVE_QUICK_EXIT
-  using ::quick_exit;
+//  using ::quick_exit;
 # endif
 #endif
   using ::rand;
```

- Avertissement sur la différence de version lors de l'édition des liens

ld: warning: dylib (/Users/blady/Documents/Programmation/AdaCore/GNAT/GNAT-16/gnat-x86_64-darwin-16.0.1-20260419/lib/libstdc++.dylib) was built for newer macOS version (15.0) than being linked (13.0)

Correction : faire croire que l'on est sur macOS 15.

```
% export MACOSX_DEPLOYMENT_TARGET=15.0
```

2. Utilisation avec le Terminal

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

```
$ gnatmake hello.adb
```

Le fichier hello.adb étant :

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

Et les résultats ne se font pas attendre :

```
$ gnatmake hello.adb
gcc -c hello.adb
gnatbind -x hello.ali
gnatlink hello.ali
$ ./hello
Hello again, avec Ada.
```

3. Les commandes utiles avec le Terminal

La liste des commandes est obtenue de la façon suivante :

```
$ gnat
GNAT 16.1.0
Copyright 1996-2025, Free Software Foundation, Inc.
List of available commands
GNAT BIND          gnatbind       réalise l'édition des liens des unités Ada compilées
GNAT CHOP          gnatchop       découpe un fichier en unités pour satisfaire les
conventions Gnat
GNAT CLEAN         gnatclean      nettoie les fichiers générés par gnat
GNAT COMPILE       gnatmake -f -u -c      compile une entité Ada
GNAT CHECK         gnatcheck      vérifie le code source suivant des règles définies (non
présent avec gnat-osx)
GNAT ELIM          gnatelim       détecte et élimine les sous-programmes inutilisés
GNAT FIND          gnatfind       liste toutes les utilisations d'une entité Ada
GNAT KRUNCH        gnatkr         réduit les noms de fichiers au nombre maximal de lettres
spécifié
GNAT LINK          gnatlink       réalise l'édition des liens de l'exécutable
GNAT LIST          gnatls         liste le contenu des objets générés
GNAT MAKE          gnatmake       utilitaire optimisé de compilation multi-unités
GNAT METRIC        gnatmetric     statistiques sur le code Ada
GNAT NAME          gnatname       réalise la correspondance entre les unités Ada et les
noms des fichiers lorsque ceux-ci ne sont pas au standard GNAT
GNAT PREPROCESS        gnatprep       pré-processeur externe
GNAT PRETTY        gnatpp         reformate le source Ada
GNAT STACK         gnatstack      calcul la taille de pile mémoire maximale théorique (non
présent avec gnat-gpl)
GNAT STUB          gnatstub       créé le squelette d'un corps d'une spécification
GNAT TEST          gnattest       créé ou exécute la suite de test unitaire
GNAT XREF          gnatxref       utilitaire d'édition des références croisées
```

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

```
$ gnatmake (extrait)
Usage: gnatmake  opts  name  {[-cargs opts] [-bargs opts] [-largs opts]}
  name  is a file name from which you can omit the .adb or .ads suffix
gnatmake switches:
  --version   Display version and exit
  --help      Display usage and exit
  -c       Compile only
  -f       Force recompilations of non predefined units
  -k       Keep going after compilation errors
  -m       Minimal recompilation
  -n       Check objects up to date, output next file to compile if not
  -o name  Choose an alternate executable name
  -p       Create missing obj, lib and exec dirs
  -Pproj   Use GNAT Project File proj
  -s       Recompile if compiler switches have changed
  -v       Display reasons for all (re)compilations
```

To pass an arbitrary switch to the Compiler, Binder or Linker:
  -cargs opts   opts are passed to the compiler
  -bargs opts   opts are passed to the binder
  -largs opts   opts are passed to the linker
  -margs opts   opts are passed to gnatmake
Compiler switches (passed to the compiler by gnatmake):
  -Idir   Specify source files search path
  -gnat2012 Ada 2012 mode (default)
  -gnat2022 Ada 2022 mode

Et aussi avec gcc :

$ gcc --help
...

Pour en savoir plus : voir l'utilisation avancée des outils GNAT, l'environnement de développement GPS ainsi que du dévermineur GDB et des exceptions Ada sur la page à savoir de Blady.

Pascal Pignard, juin 2018, mai 2019, janvier 2020, février 2020, octobre 2020, juillet 2021, juillet-décembre 2022, mai 2023, septembre 2024, février-juillet 2025, juillet 2026.