

Installation de GNAT FSF 12 pour macOS 12

Deux sources du compilateur GNAT FSF 12 existent pour macOS 12 :

- le compilateur GNAT FSF 12.1 sur *Alire* (mai 2022) pour les langages Ada, C et C++, c'est le plus officiel,
- le compilateur GNAT FSF 12.2 sur *Github* (novembre 2022), pour les langages Ada, C et C++.

Sommaire

1.	Installation du compilateur GNAT FSF 12 depuis <i>Alire</i>	2
2.	Installation du compilateur GNAT FSF 12 depuis <i>Github</i>	5
3.	Utilisation avec le Terminal	7
4.	Les commandes utiles avec le Terminal	7

1. Installation du compilateur GNAT FSF 12 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 :

```
# L'installation peut se faire dans n'importe quel dossier,
# mais pour éviter les conflits je prends /opt
% mkdir /opt/gnat-alire
% cd /opt/gnat-alire
% alr get gnat_native
(i) Deploying gnat_native=12.1.2...
...
# L'installation ne comprend pas les outils de gestion de projet GPR
# Nous les ajoutons
% alr get gprbuild
(i) Deploying gprbuild=22.0.1...
##### 100,0%
gprbuild=22.0.1 successfully retrieved.
There are no dependencies.
% cd gnat_native_12.1*
% cd bin
% ln -s /opt/gnat-alire/gprbuild_22*/bin/gpr* .
# Pour simplifier les chemins j'ajoute un raccourci générique "gnat"
% cd /opt
% ln -s gnat-alire/gnat_native_12.1* gnat
```

L'éditeur intégré GPS 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
```

C'est presque terminé, comme il bien recommandé, utiliser à chaque fois la commande suivante pour une utilisation temporaire du compilateur :

```
$ PATH=/opt/gnat/bin:$PATH
$ export MANPATH=/opt/gnat/share/man:$MANPATH
```

Pour une utilisation courante, saisir aussi les commandes suivantes :

```
$ echo 'PATH=/opt/gnat/bin:$PATH' >> ~/.zshenv
$ echo 'export MANPATH=/usr/local/gnat/share/man:$MANPATH' >> ~/.zshenv
```

Une documentation aux formats "info" et "man" est disponible dans les répertoires /opt/gnat/share/info et /opt/gnat/share/man.

```
$ info -f /opt/gnat/share/info/dir
$ man -M /opt/gnat/share/man gcc
```

b) Le débogueur

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 déboguer :
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 apparaît :

- . 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 apparaît, 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 apparaît, 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 apparaît, 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 /opt/gnat/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.

c) Les bibliothèques dynamiques

Certaines bibliothèques dynamiques ne sont pas correctement référencées. Notamment en utilisant GNAT avec le langage C++, une erreur se produit :

```
dyld[7993]: Library not loaded: /Users/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-darwin/gcc/install/lib/libstdc++.6.dylib
  Referenced from: <4AF0F675-7ADD-308E-A987-18D508E8A10F> /Applications/gnatstudio.app/Contents/MacOS/ada_language_server
  Reason: tried: '/Applications/gnatstudio.app/Contents/Resources/lib/libstdc++.6.dylib' (no such file), '/Users/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-darwin/gcc/install/lib/libstdc++.6.dylib' (no such file),
  ...
```

Une première solution consiste à corriger à chaque fois le référencement dans l'application générée, comme par exemple ici :

```
% 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 /Applications/gnatstudio.app/Contents/MacOS/ada_language_server
```

Une deuxième solution consiste à changer définitivement le référencement dans la bibliothèque concernée, par exemple :

```
% install_name_tool -id @rpath/libstdc++.6.dylib /opt/gnat/lib/libstdc++.6.dylib
```

2. Installation du compilateur GNAT FSF 12 depuis Github

a) Le compilateur

Télécharger le fichier suivant dans votre dossier Téléchargements :
Compilateur FSF 12.2 : *gcc-12.2.0-x86_64-apple-darwin15.pkg*,
sur le site GitHub de Simon Wright github.com/simonjwright/distributing-gcc/releases/tag/gcc-12.2.0-x86_64.

(Les instructions de construction du compilateur avec les langages supplémentaires C et C++ sont sur :

github.com/simonjwright/building-gcc-macos-native

Et un ensemble de scripts pour aider la construction sont sur :

github.com/simonjwright/distributing-gcc)

Le fichier n'est pas signé, une fois téléchargé, ne pas double-cliquer dessus mais faire un clic droit puis sélectionner *Ouvrir* dans le menu contextuel et accepter l'avertissement en cliquant sur le bouton *Ouvrir*. Une fenêtre surgit, indiquer un nom et un mot de passe administrateur puis cliquer sur le bouton *Modifier mes réglages*. L'installation se lance, cliquer sur le bouton *Continuer* 3 fois puis sur le bouton *Agree* et finalement sur le bouton *Installer*. Une fenêtre surgit, indiquer un nom et un mot de passe administrateur puis cliquer sur le bouton *Installer le logiciel*. Enfin cliquer sur le bouton *Fermer*. Une fenêtre demande si vous voulez placer le programme d'installation dans la corbeille, pour ma part, je clique sur le bouton *Conserver*.

Le compilateur s'installe à partir du dossier : */opt/gcc-12.2.0*.

Il contient les outils de gestion de projet *GPRBuild*, les bibliothèques *AUnit*, *GNATColl*, *Langkit*, *LibAdaLang*, *LibAdaLangTools* et *XMLAda*.

Pour une utilisation courante, saisir aussi les commandes suivantes :

```
$ echo 'PATH=/opt/gcc-12.2.0/bin:$PATH' >> ~/.zshenv
```

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

```
$ export PATH=/opt/gcc-12.2.0/bin:$PATH
```

Une documentation aux formats "info" et "man" est disponible dans les répertoires */opt/gcc-12.2.0/share/info* et */opt/gcc-12.2.0/share/man* :

```
$ info -f /opt/gcc-12.2.0/share/info/dir  
$ man -M /opt/gcc-12.2.0/share/man gcc
```

b) Le débogueur

Suivre alors la procédure du §1 b) pour signer *GDB*.

c) Les bibliothèques dynamiques

L'environnement Python requis a été référencé dans le dossier `/Library/Frameworks` alors qu'il peut être fourni par Xcode dans le dossier `/Applications/Xcode.app/Contents/Developer/Library/Frameworks` d'où une erreur lors de l'utilisation de GNATColl Python et de gdb => n'est plus présent pour gdb dans la distribution 12.2 et fait référence à Python 3.9 non présent dans Xcode 13.

```
dyld[70240]: Library not loaded: '/Library/Frameworks/Python.framework/Versions/3.8/Python'  
-Referenced from: '/opt/gcc-12.1.0/bin/gdb'  
-Reason: tried: '/Library/Frameworks/Python.framework/Versions/3.8/Python' (no such file), '/  
System/Library/Frameworks/Python.framework/Versions/3.8/Python' (no such file)
```

La solution est de référencer l'emplacement de Xcode :

```
sudo install_name_tool -change /Library/Frameworks/Python.framework/Versions/3.8/Python  
/Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/Versions/  
3.8/Python3 /opt/gcc-12.1.0/lib/libgnatcoll_python3.dylib
```

```
% sudo install_name_tool -change /Library/Frameworks/Python.framework/Versions/3.8/  
Python /Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/  
Versions/3.8/Python3 gdb
```

3. 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.
```

4. Les commandes utiles avec le Terminal

La liste des commandes est obtenue de la façon suivante :

```
$ gnat
GNAT 12.2.0
Copyright 1996-2022, Free Software Foundation, Inc.
List of available commands
GNAT BIND          gnatbind      réalise l'édition des liens des unités Ada compilées
GNAT CHOP          gnat chop     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        gnat check    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         gnat find    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	gnatstest	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] [-larges 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
- larges opts opts are passed to the linker
- margs opts opts are passed to gnatmake

Compiler switches (passed to the compiler by gnatmake):

- ldir 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.