Ada Development Environment on Linux

Sowebio SARL
15, rue du Temple
17310 – St Pierre d'Oléron – France

Capital 15 000 EUR – SIRET 844 060 046 00019 – RCS La Rochelle – APE 6201Z – TVA FR00844060046

Ed.	***Release***	***Comments***	
1	20230527	Initial release.	sr
6	20230530	Add GNAT Studio settings	sr
8	20230530	Add section Convert a project to Alire	sr
10	20230604	Add GNAT Studio documentation links	sr
17	20230605	Move AIDE documentation to this document	sr
32	20230609	Add section Convert a library to Alire, change Convert a project to Alire	alf
34	20230728	Unified logos	sr
36	20231128	Various updates in GNAT Toolchain chapter, relating to GCC 13.2	sr
38	20231224	Alire static build, Debian 11 diff. Python dependence for GNAT Studio 24w	sr
42	20240101	GNAT Remote utility	sr
46	20240202	Move Ada FAQ from v22 manual ans compiler issues to ADEL manual	sr
49	20240313	Add notes in GNAT Remote section	sr
52	20240317	Update GNAT Remote section with new features	sr
54	20240630	Add 25.0w release and update GNAT Remote section	sr
55	20241125	Update GNAT Remote section	sr
57	20241230	Vastly improve GNAT, GNAT Studio and Alire chapters	sr
64	20250210	Gpb deprecated utility removed	sr
65	20250210	GNAT Remote section redesign	sr
67	20250211	Improve Native Linux Compiler section in Alire chapter	sr
68	20250212	New introduction, new diagram in Alire chapter	sr
69	20250213	Update cover and footer	sr
70	20250218	Some updates after new install checking	ls
73	20250223	Complete overhaul to streamline station and remote installations	sr
78	20250311	Start of manual modified to emphasize quick installation	sr
83	20250326	General install update	xp
90	20250826	Add 26.0w release and update FAQ section	sr
92			

- Authors

  Stéphane Rivière - sriviere@soweb.io [Sowebio CTO]
  Arthur Le Floch - https://v22.soweb.io/people [Sowebio intern 2023]
  Lounès Souakri - https://v22.soweb.io/people [Sowebio intern 2025]
  Xavier Petit - dev@sowneo.com [Sowebio software architect 2025]

- Acknowledgments

  Adacore GNAT team, Alire author Alejandro Mosteo.

- Editor

  Stéphane Rivière [Number Six] - sriviere@soweb.io [CTO Sowebio]

  The “Excuse me I’m French” speech - The main author of this manual is a Frenchman with basic English skills. Frenchmen are essentially famous as frog eaters[1]. They have recently discovered that others ~~forms of communication~~ languages are widely used on earth. So, as a frog eater, I’ve tried to write some stuff in this foreign dialect loosely known here under the name of English. However, it’s a well known fact that frogs don’t really speak English. So your help is welcome to correct this bloody manual, for the sake of the wildebeests, and penguins too.

- Syntax notation

  Inside a command line:
  - A parameter between brackets **[ ]** is optional;
  - Two parameters separated by **|** are mutually exclusives.

  An important notice:

⇨ This is an important notice !

- Edition

  1 92 - 2025-08-26

[1] We could be famous as designers of the Concorde, Ariane rockets, Airbus planes or even Ada computer language but, definitely, Frenchmen have to wear beret with bread baguette under their arm to go eating frogs in a smokey tavern. That's *le cliché* :]

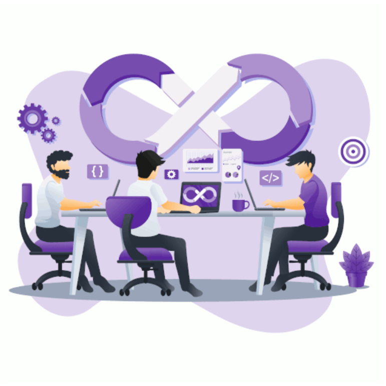
https://this-page-intentionally-left-blank.org

Contents

https://this-page-intentionally-left-blank.org

Introduction

Keep It Simple, Stupid.
Clarence Leonard "Kelly" Johnson



1 About this manual

For a development workstation under GNU/Linux, the recommended system is Debian or one of its derivatives like Mint. Ubuntu should be viewed with suspicion since snap packages have become absolutely mandatory.

This manual also applies to servers and virtual instances running Debian when using GNAT Remote tool for building and instant deploying services on development or production environments in datacentres.

Readers who find this manual useful might be interested in discovering the framework v22 https://github.com/sowebio/v22, which comes with a manual of almost 300 pages. This framework enables the development of text and web applications, and offers a wide range of modules covering all essential needs, as well as native UTF-8 support via simplified string management.

The author is a developer, system administrator and electronics engineer. He is not an Ada expert nor a fluent English writer: suggestions in order to improve this manual are (again) very welcome.
etup

2 Quick installation

Apply theses sections to setup a full Ada installation:

- GNAT Studio > Install
- GNAT Studio > Setup
- Alire > Install
- Station compiler > Install as a breeze

2.1 Philosophy

A station is installed in user mode. This gives the developer greater independence and eliminates permissions issues.

For a server instance, in conjunction with GNAT Remote, installation is performed as root, for building, service management and program deploying].

2.2 Recommended system

Until 22.04 LTS, the recommended system was GNU/Linux Ubuntu (as we could get rid of snap] or one of its derivatives.

Since the 24.04 LTS release comes with a non removable snap package infection, the recommended system could be Mint or GNU/Linux Debian with XFCE, with the advantage of a more stable and much faster system.

2.3 Debian differences compared to Ubuntu

Compared to Ubuntu, the libcurl.so link could be missing in Debian:

```
user@system : cd /usr/lib/x86_64-linux-gnu
user@system : ln -s libcurl.so.4.7.0 libcurl.so
```

3 About Ada

Some general thoughts about Ada.

3.1 Introduction

This language is not known enough yet, at least not to the majority of us, much to the detriment of many potential users for that matter. Compared to the fashionable languages, Ada is more portable, more readable, allows for higher abstraction levels and has features and functionalities unseen in other languages. Ada also allows a more comfortable experience in system programming[2] and proves itself light enough to be usable on low class 8 bit processors[3].

Ada is the name of the first programmer to ever exist in humanity. And this first programmer was a woman: Augusta Ada Byron King, Countess of Lovelace, born in 1815, daughter of Byron, the great poet, Charles Babbage's assistant, she wrote programs destined to run on his famous machine.

Ada is an American military norm[4] as well as an international civil norm[5], it is the first object oriented language to be standardized at an international level. All Ada compilers must strictly adhere to the standard. There are hundreds of compilers destined to run on that many platforms but all of them will produce a code that runs identically.

Ada is used everywhere security is critical: Airbus (A320, A330, A340, A350, A380 civil airplanes and A400 military airplane], Alsthom (High speed train], Boeing (777

[2] Ada was designed too for embedded systems and excels in it.

[3] Components that have at their disposal a couple dozen bytes of RAM and a couple Kilobytes of programming memory.

[4] MIL-STD-1815

[5] ISO/IEC 8652

and 787 airplanes), EADS (Eurofighter, Ariane rockets, ATV spacecraft, all European spaces probes), Dassault (Rafale fighter), Lockeed Martin (F22 Raptor), STS (line 14 Meteor Paris unmanned metro system), NASA (Electric power supply of the International Space Station, European service module for Orion spacecraft Moon missions).

The list goes on and on. Everywhere reliability and security must come first, Ada is the language of choice.

3.2 Why use Ada

Ada was created because software engineering is a human activity. Humans make mistakes, the Ada compiler is friend to developers. Ada is also friend to project managers for large scale development. An Ada application is written, expanded and maintained very naturally. For these reasons, Ada is also friend to executives. Ada is the language of happy programmers, managers and users.

Because Ada is a comfortable language by it's expressiveness and a restful language by it's reliability, humans involved with Ada also reflect the image of their language. The Ada community is a very comfortable community to visit and most meetings are very enlighting. Free libraries are numerous and are usually of a very high quality. Finally, the Ada community is very highly active and by now growing again.

3.3 The ending word

When Boeing decided, two decades ago, that all software for the 777[6] would be exclusively written in Ada, the corporate associates of the constructor made the remark that they were using, for a long time, languages such as C, C++ and assembly language and that they were fully satisfied with them. Boeing simply answered that only firms that could provide Ada software would be considered in contracts offerings. Therefore, the firms converted themselves to Ada.

Today, the development of software for the Boeing 777 nicknamed « The Ada Plane », has been performed and it is essentially thanks to the very big commercial success of this plane that Boeing was able to maintain the revenues created by its civil activities[7].

And what do the Boeing partner firms do from now on ? They continue to develop their new software in none other than... Ada, and here's why:
- They noticed that the length of time to convert developers to Ada is usually rather short. In a week, the developer is comfortable enough to write software in Ada and in less than a month, he feels totally comfortable with the language;
- These firms did their accounting: written in Ada, software costs less, present less anomalies, are ready sooner and are easier to maintain.

4 About the Ada Community

At first, I would like to warmly thanks the Ada Community, definitely one of the best.

[6] The Boeing 777 is the world's biggest two engines plane and the first civil Boeing having electrical flight commands, ten years later the Airbus A320.

[7] This text was written well before the tragic management and engineering failure of the 737 Max.

AdaCore Ada compiler - https://www.adacore.com/community

Special thanks to these distinguished and friendly Ada gurus:

Rolf Ebert - https://github.com/RREE
Daniel Feneuille - http://d.feneuille.free.fr
Dmitry Kazakov - https://www.dmitry-kazakov.de
Gautier de Montmollin - https://github.com/zertovitch
Pascal Pignard - https://github.com/Blady-Com
Jean-Pierre Rosen - https://adalog.fr

The chapter heading quotes are extracted from Murphy's Law and other reasons why things go wrong - A. Bloch. They come from https://www.adalog.fr site created by Jean-Pierre Rosen.

5 Manual background

This manual has its roots from AIDE[8] 0.5 (2002) to AIDE v1.4 (2005), with an edition for Windows that was favored by the 5th edition of the LSM (Libre Software Meeting) on Bordeaux in 2004 the 8th of july. After introducing AIDE, Martin and Xavier (13 years both at this time) has explained how they learn programming in Ada.

Let's hear from Ludovic Brenta[9], a prominent and well-known member of the Ada community:

“I was most impressed by two 13-year-old youths who started learning programming in February this year, and are already Ada die-hard after playing with Python for a while, and also looking at Lisp, C and Java".

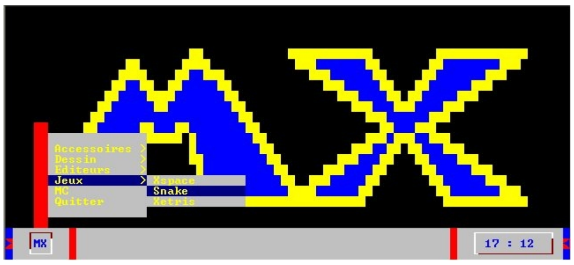
They understand that Ada is not a fashionable language but still prefer using a good language than a fashionable one. Even more stunning, they even prefer using Emacs instead of more graphical IDEs such as GPS[10]! They've written a 2000-line text-mode application in Ada that allows them to draw pictures using ASCII block characters, save them into text files, read back and display them. They designed the file format themselves, and it turns out it is quite similar to XPM.

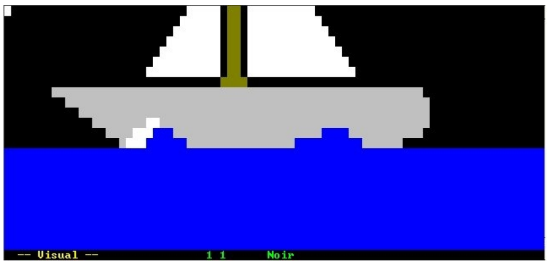
They have a second application that uses these files to display a "Start" menu with a number of applets, one of which is a fully working calculator. The father of one of these youths, Stéphane Rivière of AIDE fame, taught them the basics of Ada during 45-minute courses on Sundays, and they did all the rest by themselves with very little supervision. After only 4 months since their first exposure to programming, they understand and routinely use separate compilation and encapsulation, and were asking me questions about multitasking and game programming in Ada!”

[8] Ada Instant Environment Development, a ready to use Ada environment for Windows with Unix tools.

[9] https://comp.lang.ada.narkive.com/aKzBkWD5/ann-ada-on-the-2004-libre-software-meeting

[10] Previous name of GNAT Studio, GPS was renamed in 2020.





During these years, AIDE was a tool of choice for Ada trainers. They could set up an Ada training room in minutes on any PC!

Then time passed, Windows no longer exists for us, nor does it seem relevant for a free software developer concerned with his tools. Martin and Xavier had dreamed of a version of AIDE for Debian. It was time, in 2019, to re-create AIDE for our own needs - high availability servers cluster management and web applications - and to share it with the free software community.

Some time later, Alire, the Ada package manager, came along, making AIDE obsolete. Once again, we revised this manual to keep it up to date with the latest trends in Ada development. Alire is an important event for a more effective use of the Ada language

Documentation

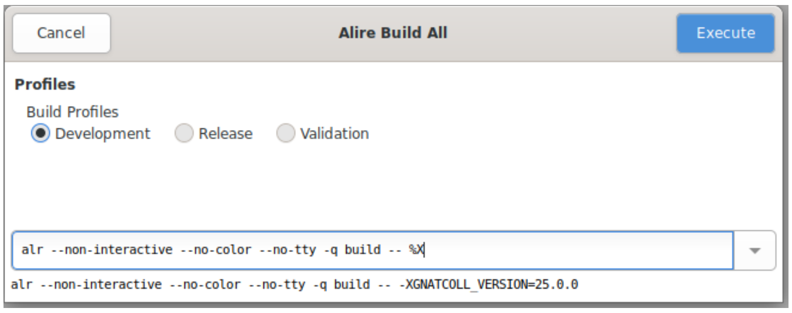
It's not a problem until it happens twice.
Jim Van Sickle

1 AdaCore resources

The documentation in various formats (PDF & HTML) is available here :https://www.adacore.com/documentation.

2 AdaCore GNAT Studio

Starting with the 25w version, GNAT Studio comes with a new build window named "Alire Build All". The "Alire Build All" window is triggered when finding an "alire.toml" file in the root project. One can find the Alire integration script in /opt/gnat-studio/share/gnatstudio/support/core/alire.py This script is the place to change the default alr build options.



Build Profiles: [✓ Development] Release Validation

```
alr --non-interactive --no-color --no-tty -q build -- %X
alr --non-interactive --no-color --no-tty -q build -- -XGNATCOLL_VERSION=25.0.0
```

From AdaCore documentation : When opening a .gpr file belonging to an Alire project (i.e: when the project's directory contains an `alire.toml` file), GNAT Studio is now able to correctly load it by running the `alr printenv` command to set up the needed environment, displaying a message in the *Locations* view once it's setup.

The default build targets to build and clean projects are also replaced by the respective Alire commands (`alr build` and `alr clean`).

2.1 Tutorial

https://docs.adacore.com/live/wave/gps/html/gps_tutorial/index.html

2.2 Manual

https://docs.adacore.com/live/wave/gps/html/gps_ug/index.html

2.3 Release notes

https://docs.adacore.com/live/wave/IDE-release-notes/html/IDE_release_notes/in-dex.html

This link above supercedes https://docs.adacore.com/live/wave/gnatstudio-release-notes/html/gnatstudio_release_notes/index.html

- Release notes 26w

  Not yet released.

- Release notes 25w

  https://docs.adacore.com/live/wave/IDE-release-notes/html/IDE_release_notes/rel-notes_25.html

- Release notes 24w

  https://docs.adacore.com/live/wave/IDE-release-notes/html/IDE_release_notes/rel-notes_24.html

2.4 Commit changes

⇨ You should press [Load more commits] until the last commit is loaded.

- Commits from 25w to 26w

  https://github.com/AdaCore/gnatstudio/compare/gnatstudio-cr-20240506...gnatstu-dio-cr-20250417

- Commits from 24w to 25w

  https://github.com/AdaCore/gnatstudio/compare/gnatstudio-cr-20230501...gnatstu-dio-cr-20240506

3 Free Software Foundation

The documentation in various formats (PDF, PS, HTML or Texinfo) is available here : https://gcc.gnu.org/onlinedocs

Choose your corresponding GCC version. In our context, it could be the last one. These manuals below total some 2,000 pages. GNAT is well documented:

- GCC Manual.pdf
- GCC GNAT User's Guide.pdf
- GCC GNAT Reference Manual.pdf
- GNAT Coding Style Manual.pdf (at the end of the web page)

4 Alire

Alire is the **A**da **LI**brary **RE**pository manager for the Libre and Open Source Ada ecosystem. It's also a build manager and more.

A comprehensive presentation paper, from the Alire author, can be found in AUJ Vol 39, Number 3, Sept 2018, P 189. http://www.ada-europe.org/archive/auj/auj-39-3.pdf

The main documentation source is https://alire.ada.dev/docs

➯ This documentation is not a substitute for the Alire documentation, but rather a complement on architecture, implementation and other points.

4.1 Caveat

Like all successful new software, Alire is constantly evolving. You may encounter messages like:

```
warn: Index 'community' version (1.2.1) is older than the newest supported by alr (1.3.0)
warn: You can disable this warning with settings key 'warning.old_index'
ⓘ If you experience any problems loading this index, you may need to reset the community index with 'alr
index --reset-community'. Note that this operation will delete any local changes to the community index.
```

Don't hesitate to update Alire, then delete the local community index, to get back to an up-to-date repository. Without this, you could miss out on new crates or new versions of crates:

```
user@system : alr index --reset-community
user@system : alr search --list
user@system : alr search --full gnat_native
```

4.2 Contributions

From a functional point of view :

- A package management standard
- A package repository
- Dependency management
- Toolchain management

From a community point of view:

- Alire provides a **centralized repository** for Ada packages and development environments, something that was clearly lacking in the past.
- You could easily do without it, but Alire is a **powerful unifying force** for the Ada community.
- This **massive adoption** shows that Alire has filled a gap.

4.3 Architecture

Here's some information I would have liked to have had when I started using Alire.

Alejandro R Mosteo, Alire's author had done an incredible work. Designing Alire must have been all the more difficult as it arrived well after GNAT and GPRBuild: Alire had to work with what already existed.

Alire is carefully designed. However:
- Knowing how Alire is implemented could avoid wasting time.
- Alire interface is still evolving, and anomalies and/or misunderstandings may appear if certain checks are not carried out and/or the Alire way is not strictly followed.

Alire's architecture is based on:
- The respect for the operating system - it is not modified by Alire
- A sandbox approach where each Ada project is independent
- Independence from any revision control system
- Staying close to the free FSF/AdaCore Ada tooling like GNAT (Ada Compiler) and GPRBuild (GNAT Project file)

4.4 Implementation

Using Alire, **the only program in the system PATH** could be alr.

The Alire environment itself is located in several directories:

- Alire global paths (per user basis)

  - $HOME/.alire - active toolchain
  - $HOME/.config/alire - settings, repository index
  - $HOME/.local/share/alire - builds, releases, toolchains

  These provisions ensure that the system is not altered (no PATH modification, for example).

  The GNAT Ada runtime system is located in $HOME/.alire/lib/gcc/x86_64-pc-linux-gnu/14.2.0/adainclude

- Alire project paths

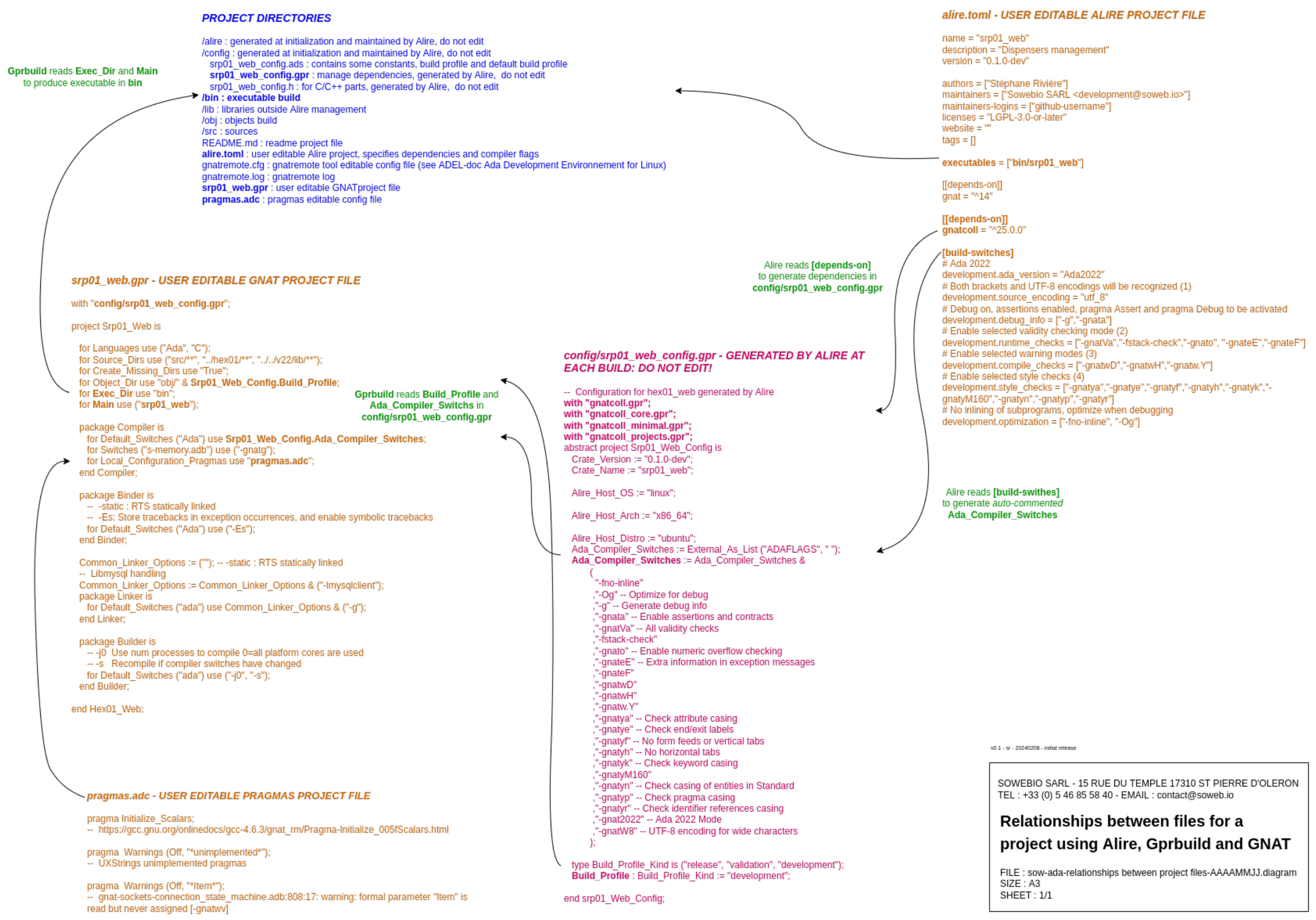
  When a new <project> is created, Alire creates one file and three new directories at the root level:

  - <project>.toml, auto-generated "root manifest"
  - alire, storing its auto-generated confs files
  - config, storing auto-generated Ada code <project>_config.ad? and a auto-generated GPR file <projet>_config.gpr
  - share, storing nothing at the beginning

  The ~/<project>.toml contains basic information about the project.

  The ~/config/GPR <project>_config.gpr file stores information managed by Alire, like dependencies, compilation options and development profiles.

As Alire had to work with what already existed, the following diagram could help to understand how Alire, GprBuild an GNATthings are tied together.



Gprbuild reads Exec_Dir and Main to produce executable in bin
Alire reads [depends-on] to generate dependencies in config/srp01_web_config.gpr
Gprbuild reads Build_Profile and Ada_Compiler_Switchs in config/srp01_web_config.gpr
Alire reads [build-swithes] to generate *auto-commented* Ada_Compiler_Switches

PROJECT DIRECTORIES

```
/alire : generated at initialization and maintained by Alire, do not edit
/config : generated at initialization and maintained by Alire, do not edit
   srp01_web_config.ads : contains some constants, build profile and default build profile
   srp01_web_config.gpr : manage dependencies, generated by Alire,  do not edit
   srp01_web_config.h : for C/C++ parts, generated by Alire,  do not edit
/bin : executable build
/lib : libraries outside Alire management
/obj : objects build
/src : sources
README.md : readme project file
alire.toml : user editable Alire project, specifies dependencies and compiler flags
gnatremote.cfg : gnatremote tool editable config file (see ADEL-doc Ada Development Environnement for Linux)
gnatremote.log : gnatremote log
srp01_web.gpr : user editable GNATproject file
pragmas.adc : pragmas editable config file
```

alire.toml - USER EDITABLE ALIRE PROJECT FILE

```
name = "srp01_web"
description = "Dispensers management"
version = "0.1.0-dev"

authors = ["Stéphane Rivière"]
maintainers = ["Sowebio SARL <development@soweb.io>"]
maintainers-logins = ["github-username"]
licenses = "LGPL-3.0-or-later"
website = ""
tags = []

executables = ["bin/srp01_web"]

[[depends-on]]
gnat = "^14"

[[depends-on]]
gnatcoll = "^25.0.0"

[build-switches]
# Ada 2022
development.ada_version = "Ada2022"
# Both brackets and UTF-8 encodings will be recognized (1)
development.source_encoding = "utf_8"
# Debug on, assertions enabled, pragma Assert and pragma Debug to be activated
development.debug_info = ["-g","-gnata"]
# Enable selected validity checking mode (2)
development.runtime_checks = ["-gnatVa","-fstack-check","-gnato", "-gnateE","-gnateF"]
# Enable selected warning modes (3)
development.compile_checks = ["-gnatwD","-gnatwH","-gnatw.Y"]
# Enable selected style checks (4)
development.style_checks = ["-gnatya","-gnatye","-gnatyf","-gnatyh","-gnatyk","-
gnatyM160","-gnatyn","-gnatyp","-gnatyr"]
# No inlining of subprograms, optimize when debugging
development.optimization = ["-fno-inline", "-Og"]
```

srp01_web.gpr - USER EDITABLE GNAT PROJECT FILE

```
with "config/srp01_web_config.gpr";

project Srp01_Web is

   for Languages use ("Ada", "C");
   for Source_Dirs use ("src/**", "../hex01/**", "../../v22/lib/**");
   for Create_Missing_Dirs use "True";
   for Object_Dir use "obj/" & Srp01_Web_Config.Build_Profile;
   for Exec_Dir use "bin";
   for Main use ("srp01_web");

   package Compiler is
      for Default_Switches ("Ada") use Srp01_Web_Config.Ada_Compiler_Switches;
      for Switches ("s-memory.adb") use ("-gnatg");
      for Local_Configuration_Pragmas use "pragmas.adc";
   end Compiler;

   package Binder is
      --  -static : RTS statically linked
      --  -Es: Store tracebacks in exception occurrences, and enable symbolic tracebacks
      for Default_Switches ("Ada") use ("-Es");
   end Binder;

   Common_Linker_Options := (""); -- -static : RTS statically linked
   --  Libmysql handling
   Common_Linker_Options := Common_Linker_Options & ("-lmysqlclient");
   package Linker is
      for Default_Switches ("ada") use Common_Linker_Options & ("-g");
   end Linker;

   package Builder is
      -- -j0  Use num processes to compile 0=all platform cores are used
      -- -s   Recompile if compiler switches have changed
      for Default_Switches ("ada") use ("-j0", "-s");
   end Builder;

end Hex01_Web;
```

pragmas.adc - USER EDITABLE PRAGMAS PROJECT FILE

```
pragma Initialize_Scalars;
--  https://gcc.gnu.org/onlinedocs/gcc-4.6.3/gnat_rm/Pragma-Initialize_005fScalars.html

pragma  Warnings (Off, "*unimplemented*");
--  UXStrings unimplemented pragmas

pragma  Warnings (Off, "*Item*");
--  gnat-sockets-connection_state_machine.adb:808:17: warning: formal parameter "Item" is
read but never assigned [-gnatwv]
```

config/srp01_web_config.gpr - GENERATED BY ALIRE AT EACH BUILD: DO NOT EDIT!

```
--  Configuration for hex01_web generated by Alire
with "gnatcoll.gpr";
with "gnatcoll_core.gpr";
with "gnatcoll_minimal.gpr";
with "gnatcoll_projects.gpr";
abstract project Srp01_Web_Config is
   Crate_Version := "0.1.0-dev";
   Crate_Name := "srp01_web";

   Alire_Host_OS := "linux";

   Alire_Host_Arch := "x86_64";

   Alire_Host_Distro := "ubuntu";
   Ada_Compiler_Switches := External_As_List ("ADAFLAGS", " ");
   Ada_Compiler_Switches := Ada_Compiler_Switches &
          (
            "-fno-inline"
           ,"-Og" -- Optimize for debug
           ,"-g" -- Generate debug info
           ,"-gnata" -- Enable assertions and contracts
           ,"-gnatVa" -- All validity checks
           ,"-fstack-check"
           ,"-gnato" -- Enable numeric overflow checking
           ,"-gnateE" -- Extra information in exception messages
           ,"-gnateF"
           ,"-gnatwD"
           ,"-gnatwH"
           ,"-gnatw.Y"
           ,"-gnatya" -- Check attribute casing
           ,"-gnatye" -- Check end/exit labels
           ,"-gnatyf" -- No form feeds or vertical tabs
           ,"-gnatyh" -- No horizontal tabs
           ,"-gnatyk" -- Check keyword casing
           ,"-gnatyM160"
           ,"-gnatyn" -- Check casing of entities in Standard
           ,"-gnatyp" -- Check pragma casing
           ,"-gnatyr" -- Check identifier references casing
           ,"-gnat2022" -- Ada 2022 Mode
           ,"-gnatW8" -- UTF-8 encoding for wide characters
          );

   type Build_Profile_Kind is ("release", "validation", "development");
   Build_Profile : Build_Profile_Kind := "development";

end srp01_Web_Config;
```

v0.1 - sr - 20240208 - initial release

SOWEBIO SARL - 15 RUE DU TEMPLE 17310 ST PIERRE D'OLERON
TEL : +33 (0) 5 46 85 58 40 - EMAIL : contact@soweb.io

Relationships between files for a project using Alire, Gprbuild and GNAT

FILE : sow-ada-relationships between project files-AAAAMMJJ.diagram
SIZE : A3
SHEET : 1/1

4.5 Compilation options

In the GPR file located at the root of the project, it is not advisable to specify compilation options (except, for example, options linked to a particular source).

Some compilation options are already defined by Alire. They are stored in ~/alire/build_hash_inputs, which is also an auto-generated file.

With the help of https://alire.ada.dev/docs/#build-profiles-and-switches, we can read there are three build profiles available in Alire: Development, Release & Validation.

By default, the root crate is in “Development” and the dependencies are in “Release". Theses informations can be found in ~/alire/settings.toml:

```
last_build_profile                                                                    =
“gnat_native=RELEASE,gnatcoll=RELEASE,libgpr=RELEASE,srp01_joe=DEVELOPMENT,xmlada=RELEASE”
```

The “root manifest” [i.e. the <project>.toml at the root of the project] may contain [build-profiles] and [build-switches] lists.

```
[build-profiles]
lib_under_test = “validation
lib_to_debug = “development

[build-switches]
release.optimization = [“-O1”, “-gnatn”]
```

The “release” part of the parameter is used to differentiate between Development, Release and Validation build options. A compilation option for all profiles will be noted as follows:

```
“*”.optimization = ...
```

The “optimization” part of the parameter can be a predefined set from Performance, Size or Debug [see catalog-format-spec below] or a sequence of options:

```
release.optimization = “Debug
release.optimization = [“-O1”, “-gnatn”]
```

The “root manifest” ~/<project>.toml format is described in https://alire.ada.dev/docs/catalog-format-spec.

Practical example, with a srp01_joe.toml containing:

```
name = "srp01_joe"
description = ""
version = "0.1.0-dev"

authors = ["Your Name"]
maintainers = ["Your Name <example@example.com>"]
maintainers-logins = ["github-username"]
licenses = "MIT OR Apache-2.0 WITH LLVM-exception"
website = ""
tags = []

executables = ["srp01_joe"]

[[depends-on]]
gnat = "^14"

[[depends-on]]
gnatcoll = "^25.0.0"

[build-switches]
# Ada 2022
development.ada_version = "Ada2022"
# Both brackets and UTF-8 encodings will be recognized (1)
development.source_encoding = "utf_8"
# Debug on, assertions enabled, pragma Assert and pragma Debug to be activated
development.debug_info = ["-g","-gnata"]
# Enable selected validity checking mode (2)
development.runtime_checks = ["-gnatVa","-fstack-check","-gnato", "-gnateE","-gnateF"]
# Enable selected warning modes (3)
development.compile_checks = ["-gnatwD","-gnatwH","-gnatw.Y"]
# Enable selected style checks (4)
development.style_checks  =  ["-gnatya","-gnatye","-gnatyf","-gnatyh","-gnatyk","-gnatyM160","-gnatyn","-
gnatyp","-gnatyr"]
# No inlining of subprograms, optimize when debugging
development.optimization = ["-fno-inline", "-Og"]
```

```
#  https://gcc.gnu.org/onlinedocs/gcc-14.1.0/gnat_ugn/Alphabetical-List-of-All-Switches.html
#
#  (1)
#  https://gcc.gnu.org/onlinedocs/gcc-4.8.5/gnat_ugn_unw/Character-Set-Control.html
#  https://gcc.gnu.org/onlinedocs/gcc-4.8.5/gnat_ugn_unw/Wide-Character-Encodings.html#Wide-Character-En-
codings
#  (2)
#  https://gcc.gnu.org/onlinedocs/gcc-14.1.0/gnat_ugn/Validity-Checking.html
#  a turn on all validity checking options
#  -fstack-check wise when using pragma Initialize_Scalars
#  -gnato  enable overflow checking in STRICT mode
#  -gnateE generate extra information in exception messages
#  -gnateF check overflow on predefined Float types
#  (3)
#  https://gcc.gnu.org/onlinedocs/gcc-14.1.0/gnat_ugn/Warning-Message-Control.html#ed
#  .e turn on every optional info/warning (no exceptions)
#  D  turn off warnings for implicit dereference (default)
#  H  turn off warnings for hiding declarations (default)
#  .Y turn off info messages for why pkg body needed (default)
#  (4)
#  https://gcc.gnu.org/onlinedocs/gcc-14.1.0/gnat_ugn/Style-Checking.html#ee
#  a  check attribute casing
#  e  check end/exit labels present
#  f  check no form feeds/vertical tabs in source
#  h  no horizontal tabs in source
#  k  check casing rules for keywords
#  Mn check line length <= n characters
#  n  check casing of package Standard identifiers
#  p  check pragma casing
#  r  check casing for identifier references
#  (5)
#  Options starting with -g, -f, -m, -O, -W, or --param are automatically passed on to the various
#  sub-processes invoked by gcc.  In order to pass  other options on to these processes the
#  -W<letter> options must be used.
#  (6) All warnings and style messages are treated as errors. -gnatg implies -gnatw.ge and -gnatyg
#  so that all standard warnings and all standard style options are turned on. All warnings and
#  style messages are treated as errors.
```

It might be wise to impose a validated compiler version, when you know that a particular version could cause problems. For example, version 13 of the GNAT compiler creates memory leaks with UXStrings, whereas version 14 does not.

Alire will generate a ~/config/srp01_joe_config.gpr :

```
--  Configuration for srp01_joe generated by Alire
with "gnatcoll.gpr";
with "gnatcoll_core.gpr";
with "gnatcoll_minimal.gpr";
with "gnatcoll_projects.gpr";
abstract project Srp01_Joe_Config is
   Crate_Version := "0.1.0-dev";
   Crate_Name := "srp01_joe";

   Alire_Host_OS := "linux";

   Alire_Host_Arch := "x86_64";

   Alire_Host_Distro := "ubuntu";
   Ada_Compiler_Switches := External_As_List ("ADAFLAGS", " ");
   Ada_Compiler_Switches := Ada_Compiler_Switches &
          (
            "-fno-inline"
           ,"-Og" -- Optimize for debug
           ,"-g" -- Generate debug info
           ,"-gnata" -- Enable assertions and contracts
           ,"-gnatVa" -- All validity checks
           ,"-fstack-check"
           ,"-gnato" -- Enable numeric overflow checking
           ,"-gnateE" -- Extra information in exception messages
           ,"-gnateF"
           ,"-gnatwD"
           ,"-gnatwH"
           ,"-gnatw.Y"
           ,"-gnatya" -- Check attribute casing
           ,"-gnatye" -- Check end/exit labels
           ,"-gnatyf" -- No form feeds or vertical tabs
           ,"-gnatyh" -- No horizontal tabs
           ,"-gnatyk" -- Check keyword casing
           ,"-gnatyM160"
           ,"-gnatyn" -- Check casing of entities in Standard
           ,"-gnatyp" -- Check pragma casing
           ,"-gnatyr" -- Check identifier references casing
           ,"-gnat2022" -- Ada 2022 Mode
           ,"-gnatW8" -- UTF-8 encoding for wide characters
          );

   type Build_Profile_Kind is ("release", "validation", "development");
   Build_Profile : Build_Profile_Kind := "development";

end Srp01_Joe_Config;
```

The main GPR file:

```
-------------------------------------------------------------------------------
--  @file      srp01_joe.gpr
--  @copyright See authors list below and v22.copyrights file
--  @licence   LGPL v3
--  @encoding  UTF-8
-------------------------------------------------------------------------------
--  @summary
--  srp01_joe program
--
--  @description
--  Job executor
--
--  @authors
--  Stéphane Rivière - sr - sriviere@soweb.io
--
--  @versions
--  See git log
-------------------------------------------------------------------------------

with "config/srp01_joe_config.gpr";

project Srp01_Joe is

   for Languages use ("Ada", "C");
   for Source_Dirs use ("src/**",  "../srp01/**", "../../v22/lib/**");

   for Create_Missing_Dirs use "True";
   for Object_Dir use "obj/" & Srp01_Joe_Config.Build_Profile;
   for Exec_Dir use "bin";

   for Main use ("srp01_joe.adb"); --  Mains specified with directory separators are now rejected.

   ----------------------------------------------------------------------------
   --  Compiler options (Gnat)
   ----------------------------------------------------------------------------

   package Compiler is
      --for Default_Switches ("Ada") use Common_Compiler_Options;
      for Default_Switches ("Ada") use srp01_joe_config.Ada_Compiler_Switches;
      for Switches ("s-memory.adb") use ("-gnatg");
      for Local_Configuration_Pragmas use "pragmas.adc";
   end Compiler;

   ----------------------------------------------------------------------------
   --  Binder options (gnatbind)
   ----------------------------------------------------------------------------

   package Binder is
      --  -Es: Store tracebacks in exception occurrences, and enable symbolic tracebacks
      for Default_Switches ("Ada") use ("-Es");
   end Binder;

   ----------------------------------------------------------------------------
   --  Linker options (ld)
   ----------------------------------------------------------------------------

   Common_Linker_Options := (""); -- RTS statically linked ("-static");
   --  Libmysql handling
   Common_Linker_Options := Common_Linker_Options & ("-lmysqlclient");
   --  SQLite C source handling
   Common_Linker_Options := Common_Linker_Options & ("-L../v22/lib/sqlite", "-lsqlite3");
   --  Libcurl handling
   --Common_Linker_Options := Common_Linker_Options & ("-lcurl");

   package Linker is
      for Default_Switches ("Ada") use Common_Linker_Options & ("-g");
   end Linker;

   -----------------------------------------------------------------------------
   --  Builder options (gprbuild)
   -----------------------------------------------------------------------------

   package Builder is
      -- -d   Display compilation process
      -- -j0  Use num processes to compile 0=all platform cores are used
      -- -s   Recompile if compiler switches have changed
      for Default_Switches ("ada") use ("-j0", "-s"); --  , "-vh"
   end Builder;

   ----------------------------------------------------------------------------
   --  Document options (gnatdoc)
   ----------------------------------------------------------------------------

   package Documentation is
      for Documentation_Dir use "doc-generated";
   end Documentation;

   ----------------------------------------------------------------------------
   --  Printer options (gnatpp)
   ----------------------------------------------------------------------------

   package Pretty_Printer is
      for Default_Switches ("ada") use ("-M120", "-W8", "--comments-unchanged");
   end Pretty_Printer;

   ----------------------------------------------------------------------------
   --  Install ()
   ----------------------------------------------------------------------------

   package Install is
      for Artifacts (".") use ("share");
   end Install;

-------------------------------------------------------------------------------
end Srp01_Joe;
-------------------------------------------------------------------------------
```

4.6 Links

https://github.com/alire-project
https://alire.ada.dev

https://gitter.im/ada-lang/Alire
https://www.reddit.com/r/ada
https://twitter.com/mosteobotic

GNAT toolchain

Doubling the number of programmers on a late project does not make anything else than double the delay.
Second Brook's Law

1 GNAT Studio

1.1 Install

Pick the latest in https://github.com/AdaCore/gnatstudio/releases/latest.

1.1.1 26.0w release

From your $HOME:

```
user@system : cd $HOME

user@system :  wget  https://github.com/AdaCore/gnatstudio/releases/download/gnatstudio-cr-20250417/gnat-studio-26.0w-20250417-x86_64-linux-bin.tar.gz

user@system : tar -xzvf gnatstudio-26.0w-20250417-x86_64-linux-bin.tar.gz

user@system : cd ./gnatstudio-26.0w-20250417-x86_64-linux-bin

user@system : ./doinstall

Change the default installation PATH :

Default install: /home/<your $HOME>/adel

user@system : echo >> ~/.bashrc; echo 'export PATH=$HOME/adel/bin:$PATH' >> ~/.bashrc

Reload environment :

source ~/.bashrc

Check PATH:

user@system : printenv PATH
```

```
Delete installation files:

user@system : cd $HOME

user@system : rm -R ./gnatstudio-26.0w-20250417-x86_64-linux-bin

user@system : rm gnatstudio-26.0w-20250417-x86_64-linux-bin.tar.gz
```

1.1.2 Desktop launcher

Create $HOME/.local/share/applications/gnatstudio.desktop:

```
[Desktop Entry]
Name=GnatStudio
Icon=/home/<your $HOME account>/adel/share/gnatstudio/icons/hicolor/32x32/apps/gnatstudio_logo.png
Exec=adel/bin/gnatstudio
Terminal=false
Type=Application
MimeType=application/x-adagpr
Categories=Development;
StartupWMClass=gnatstudio_exe
```

Then, you may create a shortcut to your dock or desktop.

A PATH not starting with / is relative to $HOME.

1.1.3 Console launcher

‘e’ is a utility as small as it’s useful. At any Alire root project, just type ‘e’ to launch GNAT Studio without blocking your terminal.

‘e’ is a GNAT Studio launcher through Alire with detach from console without the need of pressing Enter key while avoiding the creation of nohup.out file by nohup.

Create and edit $HOME/adel/bin/e:

```
#!/bin/env bash
# e : A GNAT Studio launcher through Alire with detach from console without the
#     need of pressing Enter key while avoiding the creation of nohup.out file.
nohup alr edit >/dev/null 2>&1 &
cat <<EOF
EOF
```

Add execution rights:

```
user@system : chmod +x $HOME/adel/bin/e
```

1.1.4 Special cases

- Python 3.8 dependence missing for GNAT Studio 24w

If, when launching a debug session, this error appears:

```
Welcome to GNAT Studio 24.0w (20230501) hosted on x86_64-pc-linux-gnu
(c) 2001-2023 AdaCore
```

```
[2023-12-04 15:10:12] MI protocol is not supported by GDB, switching to CI mode
[2023-12-04 15:10:12] /home/sr/.cache/alire/toolchains/gnat_native_13.2.1_788a01f9/bin/gdb: error while
loading shared libraries: libpython3.8.so.1.0: cannot open shared object file: No such file or directory
```

Python 3.8 dates from 2019. The 22.04 LTS contains 3.10 by default. Close but no cigar. Strangely enough, we don't have this problem with the much older gnat_native 11 crate. On Ubuntu, we'll solve it with the Dead Snakes PPA:

```
user@system : sudo add-apt-repository ppa:deadsnakes/ppa -y
user@system : sudo apt update
user@system : sudo apt install python3.8
user@system : sudo apt install python3.8-dev (c'est là qu'on aura la lib manquante)
```

As an alternative, David Sauvage [from AdaLabs Ltd] suggests creating a symbolic link between libpython3.8.so.1.0 and the Python library available on the system, whatever its version.

1.2 Setup

Launch GNAT Studio.

⇨ At this stage, you can't use 'e' yet. You'll have to wait for Alire to be installed.

The very first time, a configuration wizard is displayed. Set the color theme of your choice and click on [Skip & Use Defaults] at the upper right window corner.

⇨ Only the relevant commands are mentioned, whether they are left at their default value or not.

Menu > Edit > Preferences...

1.2.1 General

- Main

```
Behavior
[x] Display splash screen (default)
[x] Auto save (default)
[x] Save desktop on exit (default)
[ ] Display welcome window

Default Builder
(o) Gprbuild (default)

Charsets
Character set: Unicode UTF-8[11] (instead of Western/Latin-1 (ISO-8859-1)

Clipboard
Clipboard size: 50 (instead of 10)
```

- Color Theme

Depending on your taste.

[11] GNAT Studio uses Unicode internally

- Custom styles

```
Theme: Adwaita (default)
Default font: DejaVu Sans 9 (default)
Monospace font: DejaVu Sans Mono 8 (default)
Command window background: white (default)
Toolbar style: Small Icons (default)
```

- Key Shortcuts

```
Build > Alire Build All > [Add] > F9 >  [Remove]
Editor > Center Line > [Add] > Alt + C
Editor > Comment lines Ctrl + / > [Remove] > [Add] > Ctrl + Shift + >
Editor > Delete line > [Add] > Ctrl + Y > [Remove]
Editor > Subprogram box > [Add] > F10
Editor > Uncomment lines Ctrl + ? > [Remove] > [Add] > Ctrl + < [Remove]
```

1.2.2 Editor

- Ada

```
Indentation: [ ] Indent comments (instead of [x]
Auto indentation: (o) Simple indentation (instead of extended)

It should be wised to not change other options.
```

1.2.3 External commands

- General

```
List processes: sh -c """(ps x 2> /dev/null || ps -u \$USER 2> /dev/null || ps) | cat""" (default)
Execute command: xterm -hold -e (default)
Print command: a2ps (default)
```

You may find useful to hardcode your browser path if GNAT Studio can’t find it: HTML browser: /usr/bin/firefox %u [check path with command ‘which firefox’].

1.2.4 Windows

```
Floating Windows
You may prefer to use GNATStudio with floating windows:
[  ] or [x] All floating

Notebook Tabs
You may find this settings useful using a large screen:
Notebook tabs position: Right
Notebook tabs position: Horizontal
```

1.2.5 Build targets

A setting page of interest.

1.2.6 Plugins

You may wish to add theses plugins:

```
To be used with -bargs -E switch
[x] Addr2line
[x] Auto Locate File
[x] Build and run all
[x] Copy Paste
[x] Copy Paste Toolbar
[x] Cov Export
Important for your comfort
[x] Enter
Mandatory if you want to respect the Ada RTS Style
[x] Highlight Column with margin Column at 80
Depending of your choice but highly recommended
[x] Prevent Project Edition
[x] Separate
[x] Treemove
```

1.2.7 Shortcuts

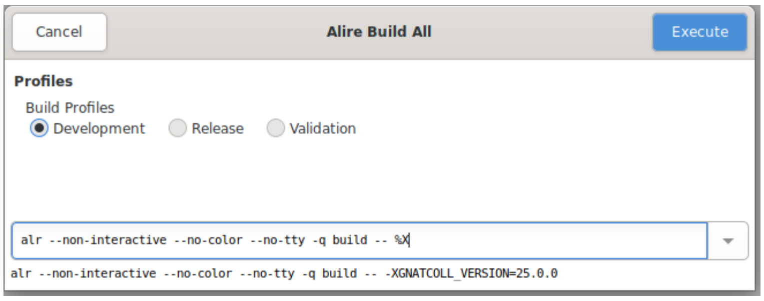
- Comment box for subprograms

  [F10] will generate a comment box with the same name above the subprogram declaration:

```
--------------------
-- Process_A_File --
--------------------

   procedure Process_A_File (TXT_Name: String) is
   -- Process a bank statement
```

- Build all [for Alire projects]

  [F9] triggers the build all window:



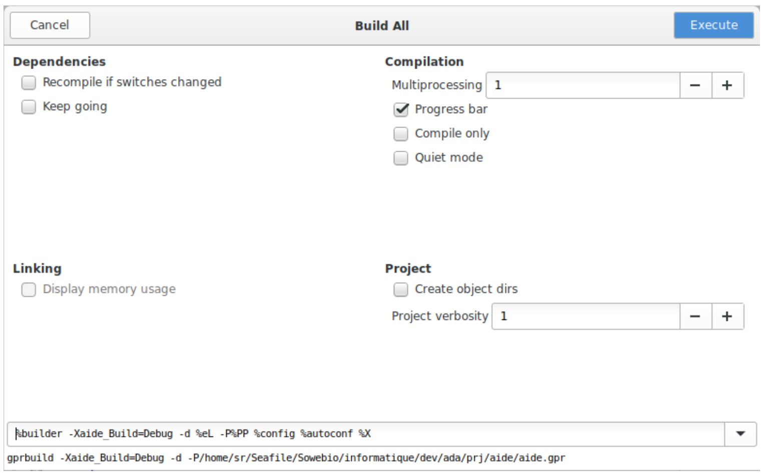
Build Profiles
[✓ Development] Release Validation

```
alr --non-interactive --no-color --no-tty -q build -- %X
alr --non-interactive --no-color --no-tty -q build -- -XGNATCOLL_VERSION=25.0.0
```

Recommended command line: alr --non-interactive --no-color --no-tty build

- Build all [for non Alire projects]

  [F9] triggers the build all window:



Dependencies
- [ ] Recompile if switches changed
- [ ] Keep going

Compilation
Multiprocessing 1
- [x] Progress bar
- [ ] Compile only
- [ ] Quiet mode

Linking
- [ ] Display memory usage

Project
- [ ] Create object dirs

Project verbosity 1

```
%builder -Xaide_Build=Debug -d %eL -P%PP %config %autoconf %X
gprbuild -Xaide_Build=Debug -d -P/home/sr/Seafile/Sowebio/informatique/dev/ada/prj/aide/aide.gpr
```

Default command line [at the window bottom]: %builder -d %eL -P%PP %config %autoconf %X

- Comment or comment a block

  [Ctrl] + [Shift] + [>] Comment the selected block.
  [Ctrl] + [<] Uncomment the selected block.

- Debug - Step

  [F5]

- Debug - Step out

  [F6] Execute the program until the next source line stepping over subprograms calls

- Debug - Finish

  [F7] Continue execution until selected stack frame returns

- Debug - Run

  [F8] Continue execution until next breakpoint

- Delete a line

  [Ctrl] + [Y] Remember Wordstar[12]

2 Alire

2.1 Install

Pick the latest at https://github.com/alire-project/alire/releases.

Use the most recent release. You'll need to update the version below if necessary.

12 https://en.wikipedia.org/wiki/WordStar

```
user@system :  wget  https://github.com/alire-project/alire/releases/download/v2.1.0/alr-2.1.0-bin-x86_64-
linux.zip

user@system : unzip -j -o  alr-2.1.0-bin-x86_64-linux.zip bin/alr -d $HOME/adel/bin

Archive:   alr-2.1.0-bin-x86_64-linux.zip
  inflating: /usr/local/bin/alr

user@system : rm alr-2.1.0-bin-x86_64-linux.zip
```

⇨ Congrats: you have completed the Alire install!

You may continue to Station compiler.

2.2 Use

2.2.1 List all crates

```
user@system : alr search --list
```

2.2.2 List all native GNAT compiler releases

```
user@system : alr search --full gnat_native

NAME            STATUS  VERSION  DESCRIPTION                                       NOTES  MATCHES
gnat_native             14.2.1   The GNAT Ada compiler - Native                           Name
gnat_native             14.1.3   The GNAT Ada compiler - Native                           Name
gnat_native             13.2.2   The GNAT Ada compiler - Native                           Name
gnat_native             13.2.1   The GNAT Ada compiler - Native                           Name
gnat_native             13.1.0   The GNAT Ada compiler - Native                           Name
gnat_native             12.2.1   The GNAT Ada compiler - Native                           Name
gnat_native             12.1.2   The GNAT Ada compiler - Native                           Name
gnat_native             12.1.1   The GNAT Ada compiler - Native                           Name
gnat_native             11.2.4   The GNAT Ada compiler - Native                           Name
gnat_native             11.2.3   The GNAT Ada compiler - Native                           Name
gnat_native             11.2.2   The GNAT Ada compiler - Native                           Name
gnat_native             11.2.1   The GNAT Ada compiler - Native                           Name
gnat_native             10.3.2   The GNAT Ada compiler - Native                           Name
gnat_native             10.3.1   The GNAT Ada compiler - Native                           Name
libadalang2xml           1.0.0    Generates XML for Ada source, using libadalang.         Long_Descrip-
tion
```

2.2.3 List all GNAT compiler packages

```
user@system : alr search --full --external-detect gnat_ | grep Name

gnat_arm_elf                   14.2.1        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   14.1.3        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   13.2.1        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   13.1.0        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   12.2.1        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   12.1.2        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   12.1.1        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   11.2.4        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   11.2.3        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   11.2.2        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   11.2.1        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   10.3.2        The GNAT Ada compiler - ARM cross-compiler
gnat_arm_elf                   10.3.1        The GNAT Ada compiler - ARM cross-compiler
gnat_avr_elf                   14.2.1        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   14.1.3        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   13.2.1        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   13.1.0        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   12.2.1        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   12.1.2        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                   12.1.1        The GNAT Ada compiler - AVR cross-compiler
gnat_avr_elf                    11.2.4       The GNAT Ada compiler - AVR cross-compiler
gnat_external            EU     external     GNAT is a compiler for the Ada programming language
gnat_native                     14.2.1       The GNAT Ada compiler - Native
gnat_native                     14.1.3       The GNAT Ada compiler - Native
gnat_native                     13.2.2       The GNAT Ada compiler - Native
gnat_native                     13.2.1       The GNAT Ada compiler - Native
gnat_native                     13.1.0       The GNAT Ada compiler - Native
gnat_native                     12.2.1       The GNAT Ada compiler - Native
gnat_native                     12.1.2       The GNAT Ada compiler - Native
gnat_native                     12.1.1       The GNAT Ada compiler - Native
gnat_native                     11.2.4       The GNAT Ada compiler - Native
gnat_native                     11.2.3       The GNAT Ada compiler - Native
gnat_native                     11.2.2       The GNAT Ada compiler - Native
gnat_native                     11.2.1       The GNAT Ada compiler - Native
gnat_native                     10.3.2       The GNAT Ada compiler - Native
gnat_native                     10.3.1       The GNAT Ada compiler - Native
gnat_riscv64_elf                14.2.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                14.1.3       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                13.2.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                13.1.0       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                12.2.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                12.1.2       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                12.1.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                11.2.4       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                11.2.3       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                11.2.2       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                11.2.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                10.3.2       The GNAT Ada compiler - RISC-V cross-compiler
gnat_riscv64_elf                10.3.1       The GNAT Ada compiler - RISC-V cross-compiler
gnat_xtensa_esp32_elf           14.2.1       The GNAT Ada compiler - ESP32 cross-compiler
```

2.2.4 List some specific packages

```
user@system : alr search --full --external-detect avr (all versions)

user@system : alr search --external-detect avr (last versions)
```

2.2.5 Change default toolchain

```
user@system : alr toolchain --select

Welcome to the toolchain selection assistant

In this assistant you can set up the default toolchain to be used with any crate
that does not specify its own top-level dependency on a version of gnat or
gprbuild.

If you choose "None", Alire will use whatever version is found in the
environment.

ⓘ Currently configured: gnat_native=14.2.1

Please select the gnat version for use with this configuration
  1. gnat_native=14.2.1
  2. None
  3. gnat_arm_elf=14.2.1
  4. gnat_avr_elf=14.2.1
  5. gnat_riscv64_elf=14.2.1
  6. gnat_xtensa_esp32_elf=14.2.1
  7. gnat_arm_elf=14.1.3
  8. gnat_avr_elf=14.1.3
  9. gnat_native=14.1.3
  0. gnat_riscv64_elf=14.1.3
  a. (See more choices...)
Enter your choice index (first is default):
> a
Please select the gnat version for use with this configuration
  1. gnat_native=13.2.2
  2. gnat_arm_elf=13.2.1
  3. gnat_avr_elf=13.2.1
  4. gnat_native=13.2.1
  5. gnat_riscv64_elf=13.2.1
  6. gnat_arm_elf=13.1.0
  7. gnat_avr_elf=13.1.0
  8. gnat_native=13.1.0
  9. gnat_riscv64_elf=13.1.0
  0. gnat_arm_elf=12.2.1
  a. (See more choices...)
Enter your choice index (first is default):
> 4
ⓘ Selected tool version gnat_native=13.2.1

ⓘ Choices for the following tool are narrowed down to releases compatible with just selected gnat_na-
tive=13.2.1

ⓘ Currently configured: gprbuild=22.0.1

Please select the gprbuild version for use with this configuration
  1. gprbuild=22.0.1
  2. None
  3. gprbuild=21.0.2
  4. gprbuild=21.0.1
Enter your choice index (first is default):
> 1
ⓘ Selected tool version gprbuild=22.0.1

Check selected version

user@system : alr toolchain --select

Welcome to the toolchain selection assistant

In this assistant you can set up the default toolchain to be used with any crate
that does not specify its own top-level dependency on a version of gnat or
gprbuild.

If you choose "None", Alire will use whatever version is found in the
environment.

ⓘ Currently configured: gnat_native=13.2.1
```

Check the new compiler version:

```
user@system : alr settings --list --global

last_build_profile=gnat_native=RELEASE,gnatcoll=RELEASE,libgpr=RELEASE,v22utf8=DEVELOPMENT,vss=RELEASE,x
mlada=RELEASE
toolchain.external.gprbuild=false
editor.cmd=gnatstudio -P ${GPR_FILE}
index.last_update=237558937
toolchain.assistant=false
index.auto_update_asked=true
toolchain.use.gnat=gnat_native=13.2.1
toolchain.use.gprbuild=gprbuild=22.0.1
toolchain.external.gnat=false
```

➪ The default compiler is not necessarily the one that will be used, depending on the constraints expressed by the main project or its dependencies.

2.2.6 Inspect project dependencies

```
user@system : alr show --solve

srp01_joe=0.1.0-dev: Sowebio Ressource Plannner
Origin: path /home/sr/Sowebio/informatique/dev/prv/srp/srp01_joe
Properties:
   Author: Stéphane Rivière
   Build Switches:
   Description: Sowebio Ressource Plannner
   Executable: prg/srp01_joe
   License: LGPL-3.0-or-later
   Maintainer: Sowebio SARL <development@soweb.io>
   Maintainers_Logins: github-username
   Name: srp01_joe
   Version: 0.1.0-dev
   Website:
```

```
Dependencies (direct):
   gnatcoll^25.0.0
Dependencies (solution):
   gnat=14.2.1 (gnat_native)
   gnatcoll=25.0.0
   libgpr=25.0.0
   xmlada=25.0.0
Dependencies (graph):
   gnatcoll=25.0.0     --> gnat=14.2.1 (gnat_native) (>=13)
   gnatcoll=25.0.0     --> libgpr=25.0.0 (~25.0.0)
   libgpr=25.0.0       --> gnat=14.2.1 (gnat_native) (/=2020)
   libgpr=25.0.0       --> xmlada=25.0.0 (~25.0.0)
   srp01_joe=0.1.0-dev --> gnatcoll=25.0.0 (^25.0.0)
   xmlada=25.0.0       --> gnat=14.2.1 (gnat_native) (>=11)
```

2.2.7 Install a specific crate version

```
user@system : alr search --full gprbuild | grep gprbuild

gprbuild U 24.0.1 The GPRBuild Ada/multilanguage build tool  Description, Name
gprbuild   22.0.1 The GPRBuild Ada/multilanguage build tool  Description, Name
gprbuild   21.0.2 The GPRBuild Ada/multilanguage build tool  Description, Name
gprbuild   21.0.1 The GPRBuild Ada/multilanguage build tool  Description, Name

user@system : alr get gprbuild=22.0.1
```

Please note of theses attributes:
E: the release is externally provided.
S: the release is available through a system package.
U: the release is not available in the current platform.
X: the release has dependencies that cannot be resolved.

3 Station compiler

The installation and build process needs at least:

- 6 GB of free memory to be secured (compilation of libadalang may fail with 4Go of free memory).

- 2 GB of free disk space.

3.1 Install as a breeze

The fast and furious full install:

```
user@system : sudo apt install libgmp-dev

user@system : alr index --reset-community (1)

user@system : alr install gnat_native gprbuild
user@system :  oi (2)
user@system : alr install gnatcov gnatdoc (3)
user@system : alr edit --select-editor (4)

(1) This step may be useful to wipe a previous Alire community index. This is a good practice to follow
when installing a new system if you don't know whether an old Alire installation already exists.
(2) choose defaults pressing <enter> twice
(3) gnatdoc will install useful others crates: adasat gnatcoll gnatcoll_gmp gnatcoll_iconv langkit_sup-
port libadalang libgmp libgpr libgpr2 markdown prettier_ada vss xmlada
(4) This step must be skipped for a non graphic instance but it is mandatory to select the preferred ed-
itor on a graphic station: then choose GNAT Studio
```

⇨ Congrats: you have completed the Ada compiler install!

3.2 Detailed install

The same as above but heavily detailed.

3.2.1 Install system packages

```
user@system : sudo apt install libgmp-dev
```

3.2.2 Install GNAT native compiler and GprBuild crates

```
user@system : alr install gnat_native gprbuild

           ⓘ Computing solutions...
Clonage dans '/root/.config/alire/indexes/community/repo'...
remote: Enumerating objects: 11913, done.
remote: Counting objects: 100% (1195/1195), done.
remote: Compressing objects: 100% (491/491), done.
remote: Total 11913 (delta 906), reused 720 (delta 699), pack-reused 10718 (from 3)
Réception d'objets: 100% (11913/11913), 2.32 Mio | 20.30 Mio/s, fait.
Résolution des deltas: 100% (6472/6472), fait.
✓ Installation targets fully solved
ⓘ Deploying gnat_native=14.2.1...
###################################################################### 100.0%
ⓘ Installing gnat_native=14.2.1...
ⓘ Deploying gprbuild=22.0.1...
###################################################################### 100.0%
ⓘ Installing gprbuild=22.0.1...
✓ Install to /root/.alire finished successfully in 14.78 seconds.
```

3.2.3 Select toolchain

⇨ This step is mandatory to select the right compiler and gprbuild release.

```
user@system : alr toolchain --select

Welcome to the toolchain selection assistant

In this assistant you can set up the default toolchain to be used with any crate
that does not specify its own top-level dependency on a version of gnat or
gprbuild.

If you choose "None", Alire will use whatever version is found in the
environment.

ⓘ gnat is currently not configured. (alr will use the version found in the environment.)

Please select the gnat version for use with this configuration
  1. gnat_native=14.2.1
  2. None
  3. gnat_arm_elf=14.2.1
  4. gnat_avr_elf=14.2.1
  5. gnat_riscv64_elf=14.2.1
  6. gnat_xtensa_esp32_elf=14.2.1
  7. gnat_arm_elf=14.1.3
  8. gnat_avr_elf=14.1.3
  9. gnat_native=14.1.3
  0. gnat_riscv64_elf=14.1.3
  a. (See more choices...)
Enter your choice index (first is default):
> <enter>
ⓘ Selected tool version gnat_native=14.2.1

ⓘ Choices for the following tool are narrowed down to releases compatible with just selected gnat_na-
tive=14.2.1

ⓘ gprbuild is currently not configured. (alr will use the version found in the environment.)
```

```
Please select the gprbuild version for use with this configuration
  1. gprbuild=22.0.1
  2. None
  3. gprbuild=21.0.2
  4. gprbuild=21.0.1
Enter your choice index (first is default):
> <enter>
ⓘ Selected tool version gprbuild=22.0.1
ⓘ Deploying gprbuild=22.0.1...
############################################################################ 100,0%
ⓘ gprbuild=22.0.1 installed successfully.
ⓘ Deploying gnat_native=14.2.1...
############################################################################ 100,0%
ⓘ gnat_native=14.2.1 installed successfully.
```

3.2.4 Install other components

Depending on your projects, you may also want to install these crates :

```
user@system : alr install gnatcov gnatdoc

ⓘ Computing solutions...
✓ Installation targets fully solved
ⓘ Skipping already installed gnatcov=22.0.1
ⓘ Starting deployment of gnatdoc=25.0.0 to fulfill gnatdoc* with solution:
   Dependencies (solution):
      adasat=25.0.0
      gnat=14.2.1 (gnat_native)
      gnatcoll=25.0.0
      gnatcoll_gmp=25.0.0
      gnatcoll_iconv=25.0.0
      gnatdoc=25.0.0
      langkit_support=25.0.0
      libadalang=25.0.0
      libgmp=6.2.1
      libgpr=25.0.0
      libgpr2=25.0.0
      markdown=25.0.0
      prettier_ada=25.0.0
      vss=25.0.0
      xmlada=25.0.0
ⓘ Deploying gnatdoc=25.0.0...
#=#=-  #      #
ⓘ Starting installation of gnatdoc=25.0.0...
ⓘ Building adasat=25.0.0/adasat.gpr...
ⓘ Building vss=25.0.0/gnat/vss_text.gpr (1/5)...
ⓘ Building vss=25.0.0/gnat/vss_json.gpr (2/5)...
ⓘ Building vss=25.0.0/gnat/vss_regexp.gpr (3/5)...
ⓘ Building vss=25.0.0/gnat/vss_xml.gpr (4/5)...
ⓘ Building vss=25.0.0/gnat/vss_xml_templates.gpr (5/5)...
ⓘ Building xmlada=25.0.0/distrib/xmlada.gpr (1/6)...
ⓘ Building xmlada=25.0.0/dom/xmlada_dom.gpr (2/6)...
Compile
   [Ada]          input_sources-strings.adb
   [Ada]          input_sources-file.adb

.../...

   [Ada]          dom-core-nodes.adb
   [Ada]          dom.ads
Build Libraries
   [gprlib]       xmlada_unicode.lexch
   [gprlib]       xmlada_input_sources.lexch
   [archive]      libxmlada_unicode.a
   [index]        libxmlada_unicode.a
   [gprlib]       xmlada_sax.lexch
   [archive]      libxmlada_input_sources.a
   [index]        libxmlada_input_sources.a
   [gprlib]       xmlada_dom.lexch
   [archive]      libxmlada_sax.a
   [index]        libxmlada_sax.a
   [archive]      libxmlada_dom.a
   [index]        libxmlada_dom.a
ⓘ Building xmlada=25.0.0/sax/xmlada_sax.gpr (3/6)...
ⓘ Building xmlada=25.0.0/input_sources/xmlada_input.gpr (4/6)...
ⓘ Building xmlada=25.0.0/schema/xmlada_schema.gpr (5/6)...
Compile
```

```
   [Ada]          schema-simple_types.adb
   [Ada]          schema-dom_readers.adb
   [Ada]          schema-schema_readers.adb
   [Ada]          schema-date_time.adb
   [Ada]          schema-validators-xsd_grammar.adb
   [Ada]          schema-readers.adb
   [Ada]          schema-validators.adb
   [Ada]          schema-decimal.adb
   [Ada]          schema.adb
Build Libraries
   [gprlib]       xmlada_schema.lexch
   [archive]      libxmlada_schema.a
   [index]        libxmlada_schema.a
ⓘ Building xmlada=25.0.0/unicode/xmlada_unicode.gpr (6/6)...
ⓘ Building libgpr=25.0.0/gpr/gpr.gpr...
Compile
   [Ada]          schema-simple_types.adb
   [Ada]          schema-dom_readers.adb

.../...

   [Ada]          unicode-names-kanbun.ads
   [Ada]          unicode-names-ornamental_dingbats.ads
Build Libraries
   [gprlib]       xmlada_unicode.lexch
   [gprlib]       xmlada_input_sources.lexch
   [archive]      libxmlada_unicode.a
   [index]        libxmlada_unicode.a
   [gprlib]       xmlada_sax.lexch
   [archive]      libxmlada_input_sources.a
   [index]        libxmlada_input_sources.a
   [gprlib]       xmlada_dom.lexch
   [archive]      libxmlada_sax.a
   [index]        libxmlada_sax.a
   [gprlib]       xmlada_schema.lexch
   [archive]      libxmlada_dom.a
   [index]        libxmlada_dom.a
   [archive]      libxmlada_schema.a
   [index]        libxmlada_schema.a
   [gprlib]       gpr.lexch
   [archive]      libgpr.a
   [index]        libgpr.a
ⓘ Building markdown=25.0.0/gnat/markdown.gpr...
ⓘ Building gnatcoll=25.0.0/gnatcoll.gpr (1/4)...
Build Libraries
   [gprlib]       gnatcoll_projects.lexch
   [archive]      libgnatcoll_projects.a
   [index]        libgnatcoll_projects.a
ⓘ Building gnatcoll=25.0.0/projects/gnatcoll_projects.gpr (2/4)...
ⓘ Building gnatcoll=25.0.0/core/gnatcoll_core.gpr (3/4)...
ⓘ Building gnatcoll=25.0.0/minimal/gnatcoll_minimal.gpr (4/4)...
ⓘ Building gnatcoll_gmp=25.0.0/gmp/gnatcoll_gmp.gpr...
ⓘ Building gnatcoll_iconv=25.0.0/iconv/gnatcoll_iconv.gpr...
ⓘ Building prettier_ada=25.0.0/prettier_ada.gpr...
ⓘ Building langkit_support=25.0.0/langkit/support/langkit_support.gpr...
Build Libraries
   [gprlib]       langkit_support.lexch
   [bind SAL]     langkit_support
   [Ada]          b__langkit_support.adb
   [objcopy]      p__langkit_support_0.o
   [archive]      liblangkit_support.a
   [index]        liblangkit_support.a
ⓘ Building libgpr2=25.0.0/gpr2.gpr...
Build Libraries
   [gprlib]       gpr2.lexch
   [archive]      libgpr2.a
   [index]        libgpr2.a
ⓘ Building libadalang=25.0.0/libadalang.gpr...
Compile
   [Ada]          libadalang-implementation.adb
   [Ada]          libadalang-implementation-c.adb

.../...

   [Ada]          libadalang-generic_introspection.adb
   [Ada]          libadalang-unparsers.ads
Build Libraries
   [gprlib]       langkit_support.lexch
   [bind SAL]     langkit_support
   [Ada]          b__langkit_support.adb
   [objcopy]      p__langkit_support_0.o
   [archive]      liblangkit_support.a
   [index]        liblangkit_support.a
   [gprlib]       adalang.lexch
   [bind SAL]     adalang
   [Ada]          b__adalang.adb
   [objcopy]      p__adalang_0.o
   [archive]      libadalang.a
   [index]        libadalang.a
ⓘ Building gnatdoc=25.0.0/gnat/gnatdoc.gpr...
Setup
   [mkdir]        object directory for project LibGNATdoc
   [mkdir]        exec directory for project GNATdoc
Compile
   [Ada]          gnatdoc-driver.adb
   [Ada]          vss-xml-implementation-xmlada_attributes.adb

../...

   [Ada]          gnatdoc-comments-builders-subprograms.adb
   [Ada]          gnatdoc-comments-utilities.adb
Build Libraries
   [gprlib]       langkit_support.lexch
   [bind SAL]     langkit_support
   [Ada]          b__langkit_support.adb
   [objcopy]      p__langkit_support_0.o
   [archive]      liblangkit_support.a
   [index]        liblangkit_support.a
   [gprlib]       adalang.lexch
   [gprlib]       vss-xml-xmlada.lexch
   [archive]      libvss-xml-xmlada.a
   [index]        libvss-xml-xmlada.a
   [bind SAL]     adalang
   [Ada]          b__adalang.adb
   [objcopy]      p__adalang_0.o
   [archive]      libadalang.a
   [index]        libadalang.a
Bind
   [gprbind]      gnatdoc-driver.bexch
   [Ada]          gnatdoc-driver.ali
Link
   [link]         gnatdoc-driver.adb
Using built-in specs.
COLLECT_GCC=/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/gcc
COLLECT_LTO_WRAPPER=/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../libexec/gcc/
x86_64-pc-linux-gnu/14.2.0/lto-wrapper
Target: x86_64-pc-linux-gnu
Configured with: ../src/configure --prefix=/home/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-
linux/gcc/install   --with-build-time-tools=/home/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-
linux/binutils/install/bin --enable-languages=c,ada,c++ --enable-libstdcxx --enable-libstdcxx-threads --
enable-libada  --disable-nls  --without-libiconv-prefix  --disable-libstdcxx-pch  --enable-lto  --disable-
multilib  --enable-threads=posix  --with-gnu-ld  --with-gnu-as  --with-mpfr=/home/runner/work/GNAT-FSF-
builds/GNAT-FSF-builds/sbx/x86_64-linux/mpfr/install   --with-gmp=/home/runner/work/GNAT-FSF-builds/GNAT-
FSF-builds/sbx/x86_64-linux/gmp/install     --with-mpc=/home/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/
sbx/x86_64-linux/mpc/install     --with-isl=/home/runner/work/GNAT-FSF-builds/GNAT-FSF-builds/sbx/x86_64-
linux/isl/install --build=x86_64-pc-linux-gnu
Thread model: posix
Supported LTO compression algorithms: zlib
gcc version 14.2.0 (GCC)
COMPILER_PATH=/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../libexec/gcc/x86_64-
pc-linux-gnu/14.2.0/:/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../libexec/
gcc/:/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/x86_64-pc-linux-
gnu/14.2.0/../../../../x86_64-pc-linux-gnu/bin/
LIBRARY_PATH=/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/x86_64-pc-
linux-gnu/14.2.0/:/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/:/
root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/lib64/../lib64/:/root/.local/share/
alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../
lib64/:/lib/x86_64-linux-gnu/:/lib/../lib64/:/usr/lib/x86_64-linux-gnu/:/usr/lib/../lib64/:/
root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/lib64/:/root/.local/share/alire/
toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../x86_64-pc-
linux-gnu/lib/:/root/.local/share/alire/toolchains/gnat_native_14.2.1_06bb3def/bin/../lib/gcc/x86_64-pc-
linux-gnu/14.2.0/../../../:/lib/:/usr/lib/
COLLECT_GCC_OPTIONS='-v'  '-L/run/user/0/alr-wzra.tmp/gnatdoc_25.0.0_8c5dade2/.objs/'  '-L/run/user/0/alr-
wzra.tmp/gnatdoc_25.0.0_8c5dade2/.objs/' '-L/root/.local/share/alire/builds/libgpr2_25.0.0_70fe0fcf/

.../...

langkit_support.gpr:7:17:  warning:  linker  options  section  not  found  in  langkit_support.a,  using  de-
faults.
ⓘ Installing adasat=25.0.0/adasat.gpr...
ⓘ Skipping gnat_native=14.2.1 without project files...
ⓘ Skipping libgmp=6.2.1 without project files...
ⓘ Installing vss=25.0.0/gnat/vss_text.gpr...
ⓘ Installing vss=25.0.0/gnat/vss_json.gpr...
ⓘ Installing vss=25.0.0/gnat/vss_regexp.gpr...
ⓘ Installing vss=25.0.0/gnat/vss_xml.gpr...
ⓘ Installing vss=25.0.0/gnat/vss_xml_templates.gpr...
ⓘ Installing xmlada=25.0.0/distrib/xmlada.gpr...
ⓘ Installing xmlada=25.0.0/dom/xmlada_dom.gpr...
ⓘ Installing xmlada=25.0.0/sax/xmlada_sax.gpr...
ⓘ Installing xmlada=25.0.0/input_sources/xmlada_input.gpr...
ⓘ Installing xmlada=25.0.0/schema/xmlada_schema.gpr...
ⓘ Installing xmlada=25.0.0/unicode/xmlada_unicode.gpr...
ⓘ Installing libgpr=25.0.0/gpr/gpr.gpr...
ⓘ Installing markdown=25.0.0/gnat/markdown.gpr...
ⓘ Installing gnatcoll=25.0.0/gnatcoll.gpr...
ⓘ Installing gnatcoll=25.0.0/projects/gnatcoll_projects.gpr...
ⓘ Installing gnatcoll=25.0.0/core/gnatcoll_core.gpr...
ⓘ Installing gnatcoll=25.0.0/minimal/gnatcoll_minimal.gpr...
ⓘ Installing gnatcoll_gmp=25.0.0/gmp/gnatcoll_gmp.gpr...
ⓘ Installing gnatcoll_iconv=25.0.0/iconv/gnatcoll_iconv.gpr...
ⓘ Installing prettier_ada=25.0.0/prettier_ada.gpr...
ⓘ Installing langkit_support=25.0.0/langkit/support/langkit_support.gpr...
ⓘ Installing libgpr2=25.0.0/gpr2.gpr...
ⓘ Installing libadalang=25.0.0/libadalang.gpr...
ⓘ Installing gnatdoc=25.0.0/gnat/gnatdoc.gpr...
✓ Install to /root/.alire finished successfully in 364.07 seconds.
```

3.2.5 Select Editor

This step must be skipped for a non graphic instance.
This step is mandatory to select the preferred editor on a graphic station.

```
user@system : alr edit --select-editor
ⓘ The current editor command is: 'code ${CRATE_ROOT}'
Please select your prefered editor or 'other' to enter a custom command
  1. VS Code
  2. GNAT studio
  3. Other
Enter your choice index (first is default):
> 2
ⓘ 'gnatstudio -P ${GPR_FILE}' is now set as the editor command.
ⓘ You can change editors by running the following command:
ⓘ `alr edit --select-editor`
```

3.2.6 Check selected toolchain

```
user@system : alr settings --global

toolchain.external.gprbuild=false
editor.cmd=gnatstudio -P ${GPR_FILE}
index.last_update=237558937
toolchain.assistant=false
index.auto_update_asked=true
toolchain.use.gnat=gnat_native=14.2.1
toolchain.use.gprbuild=gprbuild=22.0.1
toolchain.external.gnat=false
```

4 GNAT Remote

GNAT Remote is a GNAT Studio remote build and deployment utility for Alire projects.

GNAT Remote can also manage Linux services via systemd, making it an outstanding fast deployment tool.

```
GNAT Remote - GNAT Studio remote utility
Copyright (C) Sowebio SARL 2023-2024
```

```
gnatremote v0.8 - v22 v0.6 - build 2024-11-24 18:26:55

This is the short help text
Usage: gnatremote [switches] [arguments] overview

 -a, --action=ACTION COMMAND dev_save_copy_build_fast_restart|
                             prod_save_copy_build_fast_restart|
                             dev_save_copy_build_full_restart|
                             prod_save_copy_build_full_restart

 -e, --check-error-trace     Check .err trace raising an exception
```

After saving all opened sources, copies program and libraries sources from a local project to a remote host, GNAT Remote builds the program on the remote host and, if necessary, deploys the program as a service, then restarts it after the update.

For maximum efficiency, copies of sources and library sources are made with rsync and unnecessary directories are not copied.

Better still, depending of FAST or FULL build, the main binary and the entire object directory are deleted, ensuring a correct build timestamp. These two precautions also eliminate any possibility of dependency conflicts from the local system.

Two branches, DEV and PROD, are also managed for testing and production.

On a reasonably fast host located in a data center with a good decent and a fiber-optic or Starlink internet connection, the speed of the overall process is remarkable.

All these functions are controlled via the additional GNAT Studio Remote menu.

4.1 Install

Download GNAT Remote at https://github.com/sowebio/adel-gnatremote:

```
user@system : git clone https://github.com/sowebio/adel-gnatremote

user@system : git clone https://github.com/sowebio/v22
user@system : git clone https://github.com/sowebio/v22-sqlite ./v22/lib/sqlite

user@system : cd ./adel-gnatremote

user@system : alr build

user@system : cp ./prg/gnatremote $HOME/adel/bin
```

⇨ A valid SSH public key for login automation is mandatory by GNAT Remote.

Using instance.domain.tld, if you get the message “The authenticity of host 'instance.domain.tld [*.*.*.*]' can't be established.” you may use the command:

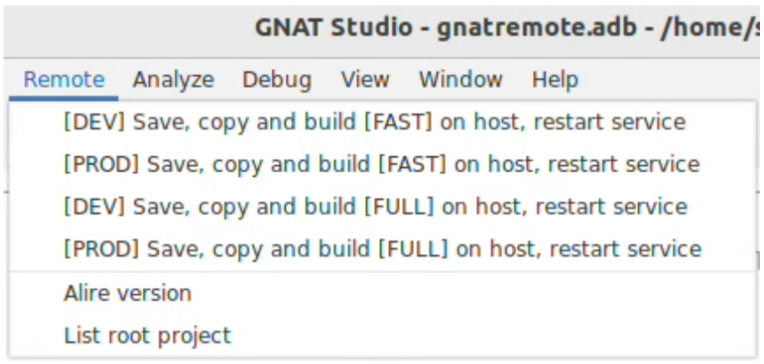
```
user@system: ssh-copy-id instance.domain.tld
```

4.1.1 Extra menu

If you deal with remote developments, you need a better utility than the soon deprecated[13] “Files>Open project from Host…”.

On the GNAT Studio side, an extra menu is created to ease the remote Ada development workflow.

- Extra menu xml file


GNAT Studio - gnatremote.adb - /home/s
Remote
Analyze
Debug
View
Window
Help
[DEV] Save, copy and build [FAST] on host, restart service
[PROD] Save, copy and build [FAST] on host, restart service
[DEV] Save, copy and build [FULL] on host, restart service
[PROD] Save, copy and build [FULL] on host, restart service
Alire version
List root project

Create $HOME/.gnatstudio/plug-ins/remote.xml:

```
<?xml version="1.0" ?>

<!--
GNAT Studio menu extender for distant development utilities

<shell>MDI.save_all false</shell>: through a dialog, ask user which files to save
<shell>MDI.save_all true</shell>: save all files without user interaction

gnatcheck.py : GPS.MDI.save_all(force=True)
expanded_code.py : GPS.MDI.save_all(False)
gnatpp.py : <shell lang="python">GPS.MDI.save_all()</shell>
-->

<GNAT_Remote>

  <action name="dev_save_copy_build_fast_restart" output="default output">
    <shell>MDI.save_all true</shell>
    <external output="Remote" >gnatremote --action=dev_save_copy_build_fast_restart</external>
  </action>

  <action name="prod_save_copy_build_fast_restart" output="default output">
    <shell>MDI.save_all true</shell>
    <external output="Remote" >gnatremote --action=prod_save_copy_build_fast_restart</external>
  </action>

  <action name="dev_save_copy_build_full_restart" output="default output">
    <shell>MDI.save_all true</shell>
    <external output="Remote" >gnatremote --action=dev_save_copy_build_full_restart</external>
  </action>

  <action name="prod_save_copy_build_full_restart" output="default output">
    <shell>MDI.save_all true</shell>
    <external output="Remote" >gnatremote --action=prod_save_copy_build_full_restart</external>
  </action>

  <action name="alireversion_remote">
    <external>alr version</external>
  </action>
```

[13] https://docs.adacore.com/live/wave/gps/html/gps_ug/remote.html

```
  <action name="projectdir_remote">
    <shell>pwd</shell>
    <external>ls -lpX --time-style=long-iso</external>
  </action>

  <submenu>

    <title>Remote</title>


      <menu after ="Build" action="dev_save_copy_build_fast_restart">
        <title>[DEV] Save, copy and build [FAST] on host, restart service</title>
      </menu>

      <menu action="prod_save_copy_build_fast_restart">
        <title>[PROD] Save, copy and build [FAST] on host, restart service</title>
      </menu>

      <menu action="dev_save_copy_build_full_restart">
        <title>[DEV] Save, copy and build [FULL] on host, restart service</title>
      </menu>

      <menu action="prod_save_copy_build_full_restart">
        <title>[PROD] Save, copy and build [FULL] on host, restart service</title>
      </menu>

      <menu><title/></menu>

      <menu action="alireversion_remote">
        <title>Alire version</title>
      </menu>
      <menu action="projectdir_remote">
        <title>List root project</title>
      </menu>

  </submenu>

</GNAT_Remote>
```

4.1.2 Static build

Some built versions of Alire on github are not statically compiled. For example, at a given moment, they are compatible with Ubuntu 22.04 LTS but not Debian 11.

A static build is therefore very useful with GNAT Remote for deployment.

Alr dependencies:

```
user@system : alr search --list
alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.34' not found (required by alr)
alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.32' not found (required by alr)
alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.33' not found (required by alr)
```

Dependencies are standard:

```
user@system : ldd alr
./alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.34' not found (required by ./alr)
./alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.32' not found (required by ./alr)
./alr: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.33' not found (required by ./alr)
    linux-vdso.so.1 (0x00007ffe43bad000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007faf09a54000)
    /lib64/ld-linux-x86-64.so.2 (0x00007faf09c32000)
```

Download sources:

```
user@system : git clone https://github.com/alire-project/alire
```

At the very end of Alire_Common.gpr, set static build for Linux (bolded line):

```
   package Linker is
      case Host_OS is
         -- Link statically on Windows to avoid DLL dependencies
         when "windows" => for Switches ("Ada") use ("-static");
         when "linux" => for Switches ("Ada") use ("-static");
         when others    => null;
      end case;
   end Linker;
```

Build Alire:

```
user@system : alr --force build

--force is needed as https://github.com/alire-project/alire/issues/1750
```

Check the build:

```
user@system : ldd bin/alr

    is not a dynamic executable
```

Size difference is not significant:

```
alr static: 38 MB
alr dynamic: 34 MB
```

5 Remote compiler

Choosing the Install section, proceed as for Station compiler, but in the root user directory /root, without typing the last command (as there is no GNAT Studio installation on the remote side):

```
ser@system : alr edit --select-editor <- Don't type it.
```

Station workflow

There are 10 types of people in the world: those who understand binary and those who don't.
Anonymous

1 Project example Hello from Alire repository

1.1 Get Hello

```
user@system : cd $HOME/opt/alire

user@system : alr get hello

Clonage dans '/home/sr/.config/alire/indexes/community/repo'...
remote: Enumerating objects: 7784, done.
remote: Counting objects: 100% (94/94), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 7784 (delta 29), reused 72 (delta 16), pack-reused 7690
Réception d'objets: 100% (7784/7784), 1.33 Mio | 9.30 Mio/s, fait.
Résolution des deltas: 100% (4228/4228), fait.
ⓘ Deploying hello=1.0.2...
Clonage dans '/home/sr/opt/alire-test/alr-olkw.tmp'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 34 (delta 0), reused 10 (delta 0), pack-reused 24
Réception d'objets: 100% (34/34), 5.44 Kio | 5.44 Mio/s, fait.
Résolution des deltas: 100% (5/5), fait.
ⓘ Deploying libhello=1.0.1...
Clonage dans '/home/sr/opt/alire-test/hello_1.0.2_5715870b/alire/cache/dependencies/alr-lwuj.tmp'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 30 (delta 0), reused 12 (delta 0), pack-reused 17
Réception d'objets: 100% (30/30), 5.20 Kio | 5.20 Mio/s, fait.
Résolution des deltas: 100% (3/3), fait.

hello=1.0.2 successfully retrieved.
Dependencies were solved as follows:

+ libhello 1.0.1 (new)

user@system : ls -l

drwxrwxr-x 11 dv dv 4096 mai   29 11:51 gnat_native_12.2.1_11f3b811
drwxrwxr-x  6 dv dv 4096 mai   29 11:49 gprbuild_22.0.1_24dfc1b5
drwxrwxr-x  8 dv dv 4096 mai   29 12:03 hello_1.0.2_5715870b

user@system : cd hello_1.0.2_5715870b
```

1.2 Build

At the root of your alire project, open a terminal and execute:

```
user@system : alr edit
```

- Clean all

  GNAT Studio: Build > Clean > Clean all

- Build all

  GNAT Studio: Build > Project > Build all [F9]

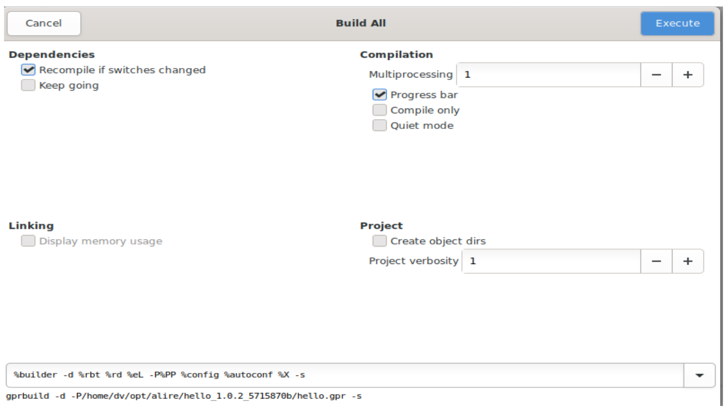
  In the build window, check the build string:

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X
```

  Which is automatically translated by GNAT Studio as:

```
gprbuild -d -P/home/dv/opt/alire/hello_1.0.2_5715870b/hello.gprlash_led.gpr
```

  Press [Execute] to build:



Dependencies
- ✓ Recompile if switches changed
- ☐ Keep going

Compilation
- Multiprocessing 1
- ✓ Progress bar
- ☐ Compile only
- ☐ Quiet mode

Linking
- ☐ Display memory usage

Project
- ☐ Create object dirs
- Project verbosity 1

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X -s
gprbuild -d -P/home/dv/opt/alire/hello_1.0.2_5715870b/hello.gpr -s
```

  Build log:

```
gprbuild  -d  -P/home/dv/opt/alire/hello_1.0.2_5715870b/hello.gpr  -XLIBHELLO_LIBRARY_TYPE=static  -XLI-
BRARY_TYPE=static -XADAFLAGS=
Compile
   [Ada]          hello.adb
   [Ada]          libhello_config.ads
   [Ada]          libhello.adb
Build Libraries
   [gprlib]       Libhello.lexch
   [archive]      libLibhello.a
   [index]        libLibhello.a
Bind
   [gprbind]      hello.bexch
   [Ada]          hello.ali
Link
   [link]         hello.adb
[2023-05-30 13:56:27] process terminated successfully, elapsed time: 01.10s
```

2 Create native Alire project

2.1 Create Alire project

Select a root working directory:

```
user@system : cd $HOME/Sowebio/devgpl/stagiaires/arthur
```

Initialize Alire project in prog01 sub-directory:

```
user@system : alr init --bin prog01

Alire needs some user information to initialize the crate author and maintainer,
for eventual submission to the Alire community index. This information will be
interactively requested now.

You can edit this information at any time with 'alr config'

Please enter your GitHub login: (default: 'github-username')
>
Using default: 'github-username'
✓ prog01 initialized successfully.
```

2.2 Build

- Launch

At the root of this alire project, open a terminal and execute:

```
user@system : cd $HOME/Sowebio/devgpl/stagiaires/arthur/prog01

user@system : alr edit
```

- Build Alire project

GNAT Studio: Build > Project > Build all [F9]

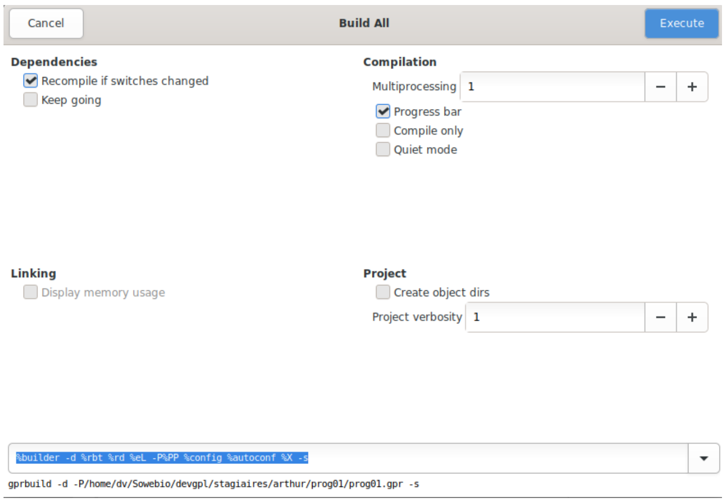
In the build window, check the build string:

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X -s
```

Which is automatically translated by GNAT Studio as:

```
gprbuild -d -P/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/prog01.gpr -s
```

Press [Execute] to build:



[✓] Recompile if switches changed
[ ] Keep going

Multiprocessing 1
[✓] Progress bar
[ ] Compile only
[ ] Quiet mode

[ ] Display memory usage

[ ] Create object dirs
Project verbosity 1

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X -s
gprbuild -d -P/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/prog01.gpr -s
```

Build log:

```
gprbuild -d -P/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/prog01.gpr -XADAFLAGS= -s
Setup
   [mkdir]        object directory for project Prog01
Compile
   [Ada]          prog01.adb
prog01.adb:20:16: (style) bad casing of "Text_IO" declared at a-textio.ads:58
prog01.adb:23:01: (style) trailing spaces not permitted
prog01.adb:25:01: (style) trailing spaces not permitted
prog01.adb:29:01: (style) trailing spaces not permitted
prog01.adb:31:04: (style) space required
Bind
   [gprbind]      prog01.bexch
   [Ada]          prog01.ali
Link
   [link]         prog01.adb
[2023-06-05 16:54:22] process terminated successfully, elapsed time: 01.50s
```

Builder results:

```
    /home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/src/prog01.adb
        20:16 (style) bad casing of "Text_IO" declared at a-textio.ads:58
        23:1 (style) trailing spaces not permitted
        25:1 (style) trailing spaces not permitted
        29:1 (style) trailing spaces not permitted
        31:4 (style) space required (2 spaces mandatory after --)
```

2.3 Run

GNAT Studio: Build > Run > Run Main Prog01 [Maj]+[F2]

In the run window, check the box:

```
[x] Run in executables directory
```

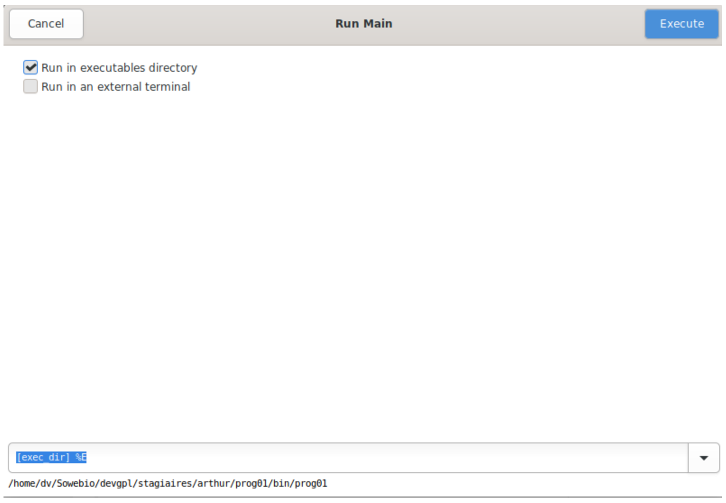
In the Run Main window, check the run string:

```
[exec_dir] %E
```

Which is automatically translated by GNAT Studio as:

```
/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/bin/prog01
```

Press [Execute] to run Prog01:



[✓] Run in executables directory
[ ] Run in an external terminal

```
[exec_dir] %E
/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/bin/prog01
```

Run Main results:

```
/home/dv/Sowebio/devgpl/stagiaires/arthur/prog01/bin/prog01
Hello World!
[2023-06-05 16:53:31] process terminated successfully, elapsed time: 00.31s
```

3 Convert a project to Alire

Kalle is a small indexed btree database from Wasiliy W. Molostoff with 35 files and 5 test programs.

Kalle is available at: https://github.com/sowebio/aide-repository

In directory: examples/kalle

3.1 Repository conversion

At the root of your alire repository, open a terminal and execute:

```
user@system : cd $HOME/opt/alire
user@system : git clone https://github.com/sowebio/kalle
```

Then, at the root of this new repository, execute:

```
user@system : alr init --bin --in-place kalle
```

```
Alire needs some user information to initialize the crate author and maintainer,
for eventual submission to the Alire community index. This information will be
interactively requested now.

You can edit this information at any time with 'alr config'

Please enter your GitHub login: (default: 'github-username')
>
Using default: 'github-username'
Please enter your full name: (default: 'Your Name')
> Stéphane Rivière
Please enter your email address: (default: 'example@example.com')
> sriviere@soweb.io
✓ kalle initialized successfully.
```

Thus, new files were generated:

```
user@system : tree -L 1

kalle/
├── alire
├── alire.toml
├── bin
├── config
├── kalle.gpr
├── kalle.txt
├── obj
├── share
└── src
```

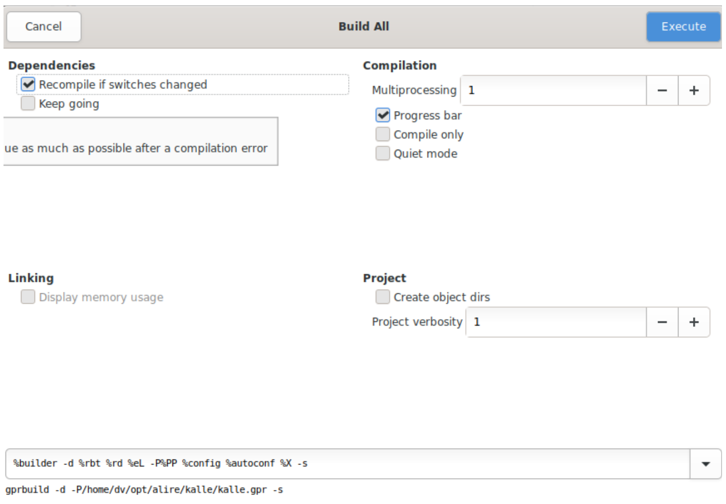
3.2 Repository customization

Let’s customize it:

```
user@system : rm ./alire/kalle/src/kalle.adb

user@system : sed -i 's/("src\/",/("src\/**",/g' ./alire/kalle/kalle.gpr
user@system :    sed    -i    's/("kalle.adb")/("test_ah.adb",    "test_btav.adb",    "test_btpa.adb",
"test_dbase.adb", "test_fs.adb")/g' ./alire/kalle/kalle.gpr
user@system :  sed  -i  's/Ada_Compiler_Switches/Ada_Compiler_Switches  \&  "-gnat95"/g'  ./alire/kalle/
kalle.gpr
```

3.3 Build

```
user@system : cd $HOME/opt/alire/kalle

user@system : alr edit
```



Dependencies

- [✓] Recompile if switches changed
- [ ] Keep going

ue as much as possible after a compilation error

Compilation

Multiprocessing 1

- [✓] Progress bar
- [ ] Compile only
- [ ] Quiet mode

Linking

- [ ] Display memory usage

Project

- [ ] Create object dirs

Project verbosity 1

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X -s
gprbuild -d -P/home/dv/opt/alire/kalle/kalle.gpr -s
```

3.4 Test programs

```
user@system : ~/opt/alire/kalle/bin$ ./test_ah

--- try-suffix:  true: ordinary
--- try-insert:  true: with expand for 1 elem
--- try-insert:  true: with amend for 1 elem
--- try-insert:  true: with delete for 1 elem
--- try-insert:  true: with delete
--- try-insert:  true: with expand
--- try-expand:  true: ordinary
--- try-delete:  true: ordinary
--- try-locate:  true: full subpattern matching

user@system : ~/opt/alire/kalle/bin$ ./test_btav

--- try-insert:  true: inserting values in empty tree
--- try-get_ge:  true: try to get value over the upper bound
--- try-get_ge:  true: find values by less argument
--- try-get_ge:  true: find values by equal argument
--- try-get_le:  true: try to get value over the lower bound
--- try-get_le:  true: find values by equal argument
--- try-get_le:  true: find values by greater argument
--- try-get_lt:  true: try to get value over the lower bound
--- try-get_lt:  true: find values by greater argument
--- try-get_lt:  true: find values by equal argument
--- try-get_gt:  true: try to get value over the upper bound
--- try-get_gt:  true: find values by less argument
--- try-get_gt:  true: find values by equal argument
--- try-delete:  true: 1st part
--- try-delete:  true: 2nd part

user@system : ~/opt/alire/kalle/bin$ ./test_btpa

--- try-insert:  true: inserting values in empty tree
--- try-get_ge:  true: try to get value over the upper bound
--- try-get_ge:  true: find values by less argument
--- try-get_ge:  true: find values by equal argument
--- try-get_le:  true: try to get value over the lower bound
--- try-get_le:  true: find values by equal argument
--- try-get_le:  true: find values by greater argument
--- try-get_lt:  true: try to get value over the lower bound
--- try-get_lt:  true: find values by greater argument
--- try-get_lt:  true: find values by equal argument
--- try-get_gt:  true: try to get value over the upper bound
--- try-get_gt:  true: find values by less argument
--- try-get_gt:  true: find values by equal argument
--- try-delete:  true: 1st part
--- try-delete:  true: 2nd part

user@system : ~/opt/alire/kalle/bin$ ./test_dbase
```

```
user@system : ~/opt/alire/kalle/bin$ ./test_fs

 [ 40: 50: 60]
 [ 40: 50: 0 60]
]%
```

4 Convert a library to Alire

V20 is an example of library which has a dependency with another library called GNATColl.
V20 is available at : https://github.com/sowebio/v20.

4.1 Alire repository conversion

To get the repository, open a terminal and execute:

```
user@system : cd $HOME/opt/alire
user@system : git clone https://github.com/sowebio/v20
```

Then, at the root of this new repository, execute:

```
user@system : alr init --lib --in-place --no-skel v20

Alire needs some user information to initialize the crate author and maintainer,
for eventual submission to the Alire community index. This information will be
interactively requested now.

You can edit this information at any time with 'alr config'

Please enter your GitHub login: (default: 'github-username')
>
Using default: 'github-username'
Please enter your full name: (default: 'Your Name')
> Stéphane Rivière
Please enter your email address: (default: 'example@example.com')
> sriviere@soweb.io
✓ v20 initialized successfully.
```

The standard Alire tree is generated alongside previous files:

```
user@system : tree -L 1
v20-alire
├── alire
├── alire.toml
├── bak
├── bin
├── config
├── doc-generated
├── nohup.out
├── obj
├── README.md
├── src
├── src-out
├── src-tests
├── v20.aru
├── v20.copyrights
├── v20.dbg
├── v20.gpr
├── v20_html
├── v20.txt
└── v20.udb
```

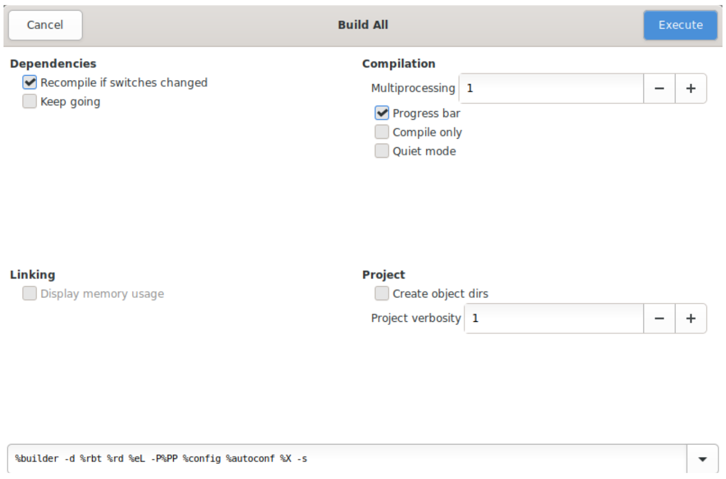
4.2 Setup dependencies with Alire

V20 requires the GNATColl library, which can be linked using the following command:

```
user@system : alr with gnatcoll
```

4.3 Edit and build Alire library

```
user@system : alr edit
```



Dependencies
- [✓] Recompile if switches changed
- [ ] Keep going

Compilation
- Multiprocessing 1
- [✓] Progress bar
- [ ] Compile only
- [ ] Quiet mode

Linking
- [ ] Display memory usage

Project
- [ ] Create object dirs
- Project verbosity 1

```
%builder -d %rbt %rd %eL -P%PP %config %autoconf %X -s
```

4.4 Test library

```
user@system : cd bin && ./test

v20 library test program - test v0.6
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

20230609-114550 - INIT    - MSG - Ada Cur: [ 1388 ] Max: [ 1696 ]
20230609-114550 - INIT    - MSG - All Cur: [ 2969600 ] Max: [ 2969600 ]
...
```

Remote workflow

Networks always go down on a Friday.
John Karr law

1 Local side implementation

Let's consider, on the local side, a project names **hex01**, including three programs **hex01_net**, **hex01_tsk** and **hex01_web** using common dependencies (which are located in ~/hex01/lib - common project libraries):

```
Local build tree

   ...── hex01          [Project].Name = hex01 : Root project
         ├── hex01      [Project].Name = hex01 : Shared
         │   └── lib    Common project libraries
         ├── hex01_net  [Project].Name = hex01 & [Program].name = net : Net program folder
         │   ├── .git   git autogenerated files
         │   ├── alire  alire autogenerated files
         │   ├── config alire project autogenerated files
         │   ├── obj    objects files
         │   ├── prg    binary file
         │   └── src    sources files
         ├── hex01_tsk  [Project].Name = hex01 & [Program].name = tsk : Tsk program folder
         │   ├── .git
         │   ├── alire
         │   ├── config
         │   ├── obj
         │   ├── prg
         │   └── src
         └── hex01_web  [Project].Name = hex01 & [Program].name = web : Web program folder
             ├── .git
             ├── alire
             ├── config
             ├── obj
             ├── prg
             └── src
```

Theses three programs are run as services. At the root of each program is a configuration file gnatremote.cfg:

```
├── alire
├── config
├── obj
├── prg
├── src
├── alire.toml
├── gnatremote.cfg
├── hex01_web.gpr
├── pragmas.adc
└── README.md
```

2 Remote side implementation

2.1 With service handling

For service handling, meaning managing the stop, copy and restart sequences, create a gnatremote.cfg file with this content:

```
# -----------------------------------------------------------------------------
#  gnatremote.cfg - Configuration file
# -----------------------------------------------------------------------------
#
#  20231229 - sr - First release
#  20240322 - sr - Add common parameter for common sources between projects
#  20240521 - sr - Add dev & prod build management
#  20250209 - sr - Deep reworking and external libraries integration
#
# -----------------------------------------------------------------------------

# All paths are relative apart Build_Dir and Run_Dir

[Project]
Name = hex01

[Program]
Name = web
Sources_Dir = src
Objects_Dir = obj
Binary_Dir = prg

[Local]
Beep = bell

[Remote]
Host = host.domain.tld
User = root
Build_Dir = /root/build
Run_Dir = /opt
Dev_Dir = dev
Prod_Dir = prod

# -----------------------------------------------------------------------------
#  EOF
# -----------------------------------------------------------------------------
```

The examination of the local build tree above and the remote build and executions trees below will be helpful for the understanding of GNAT Remote configuration.

On the remote side, we found a build tree and an execution tree:

```
Remote build tree

  /root
  └── build
      └── hex01           [Project].Name = hex01 : Root project
          ├── hex01       [Project].Name = hex01 : Common project sources
          │   └── lib     Common project libraries
          ├── hex01_net   [Project].Name = hex01 & [Program].name = net : Net program folder
          │   ├── alire
          │   ├── config
          │   ├── obj
          │   ├── prg
          │   └── src
          ├── hex01_tsk   [Project].Name = hex01 & [Program].name = tsk : Tsk program folder
          │   ├── alire
          │   ├── config
          │   ├── obj
          │   ├── prg
          │   └── src
          └── hex01_web   [Project].Name = hex01 & [Program].name = web : Web program folder
              ├── alire
              ├── config
              ├── obj
              ├── prg
              └── src


  Remote execution tree

  /opt                      [Remote].Run_Dir = /opt
  └── hex01                 [Project].Name = hex01 : Root project
      ├── dev               [Remote].Dev_Dir = dev : development environment
      │   ├── css
      │   ├── html
      │   │   └── downloads
      │   ├── img
      │   ├── js
      │   └── sys           Systemd services files
      └── prod              [Remote].Prod_Dir = prod : production environment
          ├── css
          ├── html
          │   └── downloads
          ├── img
          ├── js
          └── sys           Systemd services files
```

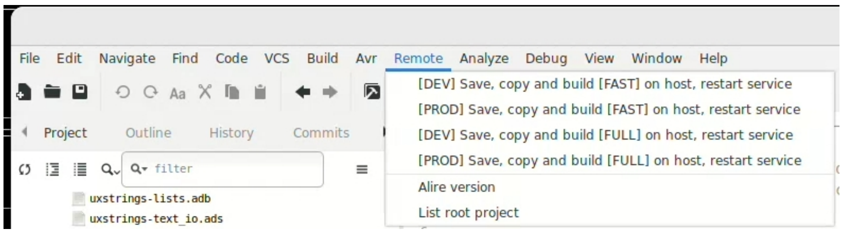
2.2 Without service handling

Without building a service, meaning not handling the stop, copy and restart sequences, create a gnatremote.cfg file with this content:

```
# -----------------------------------------------------------------------------
#  gnatremote.cfg - Configuration file
# -----------------------------------------------------------------------------

# All paths are relative apart Build_Dir and Run_Dir

[Project]
Name = hex01

[Program]
Name = web
Sources_Dir = src
Objects_Dir = obj
Binary_Dir = prg

[Local]
Beep = bell

[Remote]
Host = host.domain.tld
User = root
Build_Dir = /root/build
Run_Dir = none
Dev_Dir =
Prod_Dir =

# -----------------------------------------------------------------------------
#  EOF
# -----------------------------------------------------------------------------
```

3 Running a remote build and run

Clic on Menu > Remote and choose the [DEV] or [PROD] build. The [FAST] menus rebuild only changed sources. The [FULL] menus rebuild all objects from all sources.


File Edit Navigate Find Code VCS Build Avr Remote Analyze Debug View Window Help
[DEV] Save, copy and build [FAST] on host, restart service
[PROD] Save, copy and build [FAST] on host, restart service
[DEV] Save, copy and build [FULL] on host, restart service
[PROD] Save, copy and build [FULL] on host, restart service
Alire version
List root project
Project
Outline
History
Commits
filter
uxstrings-lists.adb
uxstrings-text_io.ads

GNAT Studio console Remote output:

```
gnatremote --action=dev_save_copy_build_full_restart

GnatRemote - Gnatstudio remote utility
Copyright (C) Sowebio SARL 2023-2025
gnatremote v0.9 - v22 v0.6 - build 2025-02-09 14:16:01

20250209-182253.588 - INIT    - MSG - gnatremote.Init.App > Configuration file ../gnatremote.cfg loaded

20250209-182253.588 - ACTION  - --- - INPUT PARAMETERS ------------------------------
20250209-182253.591 - ACTION  - MSG - Action required: dev_save_copy_build_full_restart
20250209-182253.591 - ACTION  - MSG - Local project directory:   /home/sr/Sowebio/informatique/dev/prv/hex01
20250209-182253.591 - ACTION  - MSG - Local shared directory:    /home/sr/Sowebio/informatique/dev/prv/hex01/hex01
20250209-182253.592  -  ACTION   -  MSG  -  Local  program  directory:     /home/sr/Sowebio/informatique/dev/prv/hex01/
hex01_net
20250209-182253.592 - ACTION  - MSG - Remote build directory:    root@i188c1.genesix.org/root/build/hex01
20250209-182253.592 - ACTION  - MSG - Remote run dir directory:  root@i188c1.genesix.org/opt/hex01
20250209-182253.592 - ACTION  - MSG - Remote run dev directory:  root@i188c1.genesix.org/opt/hex01/dev
20250209-182253.593 - ACTION  - MSG - Remote run prod directory: root@i188c1.genesix.org/opt/hex01/prod

20250209-182253.593 - ACTION  - --- - GNATREMOTE.COPY_SHARED > COPY COMMON PROJECT SOURCES
20250209-182253.593 - ACTION  - MSG - Net.Rsync > Excluding: .*
20250209-182253.594 - ACTION  - MSG - Net.Rsync > Excluding: prg
20250209-182253.594 - ACTION  - MSG - Net.Rsync > Excluding: obj
20250209-182255.223 - ACTION  - MSG - Net.Rsync > Directory_Rx exists, no need to create it
sending incremental file list

sent 10.198 bytes  received 27 bytes  2.921,43 bytes/sec
total size is 13.522.374  speedup is 1.322,48
20250209-182258.424 - ACTION   - MSG - Net.Rsync > Copy: /home/sr/Sowebio/informatique/dev/prv/hex01/hex01/ to /root/
build/hex01/hex01 successful

20250209-182258.425 - ACTION  - --- - GNATREMOTE.COPY_SOURCES > COPY BUILD FILES AND PROGRAM SOURCES
20250209-182258.425 - ACTION  - MSG - Net.Rsync > Excluding: .*
20250209-182258.425 - ACTION  - MSG - Net.Rsync > Excluding: prg
20250209-182258.426 - ACTION  - MSG - Net.Rsync > Excluding: obj
20250209-182259.611 - ACTION  - MSG - Net.Rsync > Directory_Rx exists, no need to create it
sending incremental file list
./
alire.toml
          3.807 100%    0,00kB/s    0:00:00           3.807 100%    0,00kB/s    0:00:00 (xfr#1, to-chk=35/38)
gnatremote.cfg
          3.955 100%    3,77MB/s    0:00:00           3.955 100%    3,77MB/s    0:00:00 (xfr#2, to-chk=34/38)
gnatremote.log
          1.946 100%  633,46kB/s    0:00:00           1.946 100%  633,46kB/s    0:00:00 (xfr#3, to-chk=33/38)
hex01_net.gpr
          3.480 100%  261,42kB/s    0:00:00           3.480 100%  261,42kB/s    0:00:00 (xfr#4, to-chk=32/38)
alire/
alire/alire.lock
          5.247 100%  197,08kB/s    0:00:00           5.247 100%  197,08kB/s    0:00:00 (xfr#5, to-chk=27/38)
alire/build_hash_inputs
            400 100%    7,66kB/s    0:00:00             400 100%    7,66kB/s    0:00:00 (xfr#6, to-chk=24/38)
alire/settings.toml
            113 100%    1,49kB/s    0:00:00             113 100%    1,49kB/s    0:00:00 (xfr#7, to-chk=23/38)
alire/flags/
alire/flags/post_fetch_done
              0 100%    0,00kB/s    0:00:00 (xfr#8, to-chk=20/38)
alire/tmp/
config/
config/hex01_net_config.ads
            571 100%    6,56kB/s    0:00:00             571 100%    6,56kB/s    0:00:00 (xfr#9, to-chk=17/38)
config/hex01_net_config.gpr
          1.628 100%   18,07kB/s    0:00:00           1.628 100%   18,07kB/s    0:00:00 (xfr#10, to-chk=16/38)
config/hex01_net_config.h
            412 100%    4,33kB/s    0:00:00             412 100%    4,33kB/s    0:00:00 (xfr#11, to-chk=15/38)
src/

sent 22.664 bytes  received 2.657 bytes  7.234,57 bytes/sec
total size is 223.371  speedup is 8,82
20250209-182302.568 - ACTION   - MSG - Net.Rsync > Copy: /home/sr/Sowebio/informatique/dev/prv/hex01/hex01_net/ to /
root/build/hex01/hex01_net successful

20250209-182302.569 - ACTION  - --- - GNATREMOTE.BUILD > HEX01_NET -------------------
20250209-182304.605 - ACTION  - MSG - /root/build/hex01/hex01_net/obj successfully deleted
```

```
20250209-182307.184  -  ACTION   -  MSG  -  Remote  command:  rm  --force  /root/build/hex01/hex01_net/prg/hex01_net  on
root@i188c1.genesix.org successful
20250209-182307.185 - ACTION  - MSG - /root/build/hex01/hex01_net/prg/hex01_net successfully deleted
20250209-182415.556 - ACTION  - MSG - Remote command cd /root/build/hex01/hex01_net ; alr build ; ls -l ./prg on
root@i188c1.genesix.org successful
Note: Building hex01_net=0.1.0-dev/hex01_net.gpr...
Setup
   [mkdir]        object directory for project Hex01_Net
Compile
   [Ada]          hex01_net.adb
   [C]            sqlite3.c
   [Ada]          s-memory.adb
   [Ada]          ews.ads
   [Ada]          ews-http.adb
   [Ada]          gnoga.adb
   [Ada]          gnoga-server.adb
   [Ada]          gnoga-server-database.adb
   [Ada]          gnoga-server-database-mysql.adb
   [Ada]          hex01_def.adb
   [Ada]          uxstrings.adb
   [Ada]          v22.adb
   [Ada]          v22-cfg.adb
   [Ada]          v22-fls.adb
   [Ada]          v22-gui.adb
   [Ada]          v22-msg.adb
   [Ada]          v22-prg.adb
   [Ada]          v22-sql.adb
   [Ada]          v22-sys.adb
   [Ada]          v22-tio.adb
   [Ada]          v22-uxs.adb
   [Ada]          uxstrings-conversions.adb
   [Ada]          uxstrings-text_io.adb
   [Ada]          strings_edit.adb
   [Ada]          ews-reference_counted_pointers_g.adb
   [Ada]          ews-static.adb
   [Ada]          gnoga-types.adb
   [Ada]          gnoga-gui.ads
   [Ada]          gnoga-gui-base.adb
   [Ada]          uxstrings-lists.adb
   [Ada]          gnoga-application.adb
   [Ada]          gnoga-application-multi_connect.adb
   [Ada]          gnoga-gui-element.adb
   [Ada]          gnoga-gui-element-canvas.adb
   [Ada]          gnoga-gui-element-canvas-context_2d.adb
   [Ada]          gnoga-gui-element-canvas-context_2d-plotting.adb
   [Ada]          gnoga-gui-element-common.adb
   [Ada]          gnoga-gui-element-form.adb
   [Ada]          gnoga-gui-element-table.adb
   [Ada]          gnoga-gui-plugin.ads
   [Ada]          gnoga-gui-plugin-jqueryui.adb
   [Ada]          gnoga-gui-plugin-jqueryui-widget.adb
   [Ada]          gnoga-gui-window.adb
   [Ada]          uxstrings-hash.adb
   [Ada]          v22-gui-footer.adb
   [Ada]          v22-gui-header.adb
   [Ada]          v22-gui-main_menu.adb
   [Ada]          gnoga-gui-view.adb
   [Ada]          v22-net.adb
   [Ada]          gnoga-server-database-sqlite.adb
   [Ada]          strings_edit-integer_edit.adb
   [Ada]          strings_edit-utf8.adb
   [Ada]          ews-types.adb
   [Ada]          gnoga-server-connection.adb
   [Ada]          gnoga-gui-navigator.adb
   [Ada]          parsers.ads
   [Ada]          parsers-multiline_source.adb
   [Ada]          parsers-multiline_source-text_io.adb
   [Ada]          parsers-multiline_source-xpm.ads
   [Ada]          gnoga-types-colors.adb
   [Ada]          gnoga-gui-element-style_block.adb
   [Ada]          gnoga-gui-element-list.adb
   [Ada]          gnoga-client.ads
   [Ada]          gnoga-client-storage.adb
   [Ada]          gnoga-gui-document.adb
   [Ada]          gnoga-gui-location.adb
   [Ada]          v22-gui-breadcrumb.adb
   [Ada]          v22-gui-user_menu.adb
   [Ada]          gnoga-server-template_parser.adb
   [Ada]          gnoga-server-template_parser-simple.adb
   [Ada]          strings_edit-integers.ads
   [Ada]          parsers-generic_source.ads
   [Ada]          parsers-generic_source-get_cpp_blank.adb
   [Ada]          parsers-generic_source-get_text.adb
   [Ada]          parsers-generic_source-get_token.adb
   [Ada]          parsers-generic_source-keywords.adb
   [Ada]          parsers-generic_source-xpm.adb
   [Ada]          tables.adb
   [Ada]          tables-names.adb
   [Ada]          gnat-sockets-connection_state_machine.adb
   [Ada]          gnat-sockets-connection_state_machine-http_server.adb
   [Ada]          gnat-sockets-server.adb
   [Ada]          gnoga-server-connection-common.ads
   [Ada]          gnoga-server-mime.adb
   [Ada]          strings_edit-quoted.adb
   [Ada]          strings_edit-streams.adb
   [Ada]          gnoga-server-model.adb
   [Ada]          gnoga-server-model-queries.adb
   [Ada]          generic_unbounded_array.adb
   [Ada]          object-handle.adb
   [Ada]          object-handle-generic_unbounded_array.adb
   [Ada]          object.adb
   [Ada]          strings_edit-fields.adb
   [Ada]          gnat-sockets-connection_state_machine-big_endian.ads
   [Ada]          gnat-sockets-connection_state_machine-big_endian-unsigneds.adb
   [Ada]          strings_edit-base64.adb
   [Ada]          strings_edit-floats.ads
   [Ada]          strings_edit-time_conversions.adb
   [Ada]          gnat-sockets-connection_state_machine-expected_sequence.adb
   [Ada]          gnat-sockets-connection_state_machine-terminated_strings.adb
   [Ada]          stack_storage.adb
   [Ada]          strings_edit-float_edit.adb
   [Ada]          generic_unbounded_ptr_array.adb
Bind
   [gprbind]      hex01_net.bexch
   [Ada]          hex01_net.ali
Link
   [archive]      libhex01_net.a
   [index]        libhex01_net.a
   [link]         hex01_net.adb
Success: Build finished successfully in 67.66 seconds.
total 21828
-rwxr-xr-x 1 root root 22346416  9 fÃ©vr. 18:25 hex01_net

20250209-182415.556 - ACTION  - --- - GNATREMOTE.RESTART > HEX01_NET -----------------
20250209-182418.734 - ACTION  - MSG - Remote command systemctl stop hex01_net_dev on root@i188c1.genesix.org success-
ful
20250209-182419.925 - ACTION  - MSG - Remote command cp --force /root/build/hex01/hex01_net/prg/hex01_net /opt/
hex01/dev/hex01_net_dev on root@i188c1.genesix.org successful
20250209-182420.607 - ACTION  - MSG - Remote command systemctl start hex01_net_dev on root@i188c1.genesix.org suc-
cessful
20250209-182420.607 - ACTION  - MSG - Action successful

20250209-182421.721 - EXIT    - MSG - Total execution time: 0h01m28s
```

4 Notes

4.1 Service naming

Associated services must comply with a naming convention. See example for the hex01_net service.

- hex01_net.install

```
#! /bin/bash
#-----------------------------------------------------------------------------
#
#-----------------------------------------------------------------------------

systemctl enable /root/opt/hex01_net.service
systemctl start hex01_net

#--------------------------------------------------------------------------------
# EOF
#--------------------------------------------------------------------------------
```

- hex01_net.service

```
#-----------------------------------------------------------------------------
# Hex Systemd service conf
#-----------------------------------------------------------------------------

[Unit]
Description=Hex01_Net launcher
After=network.target

[Service]
Restart=always
ExecStart=/root/opt/services/hex01_net
```

```
WorkingDirectory=/root/opt/services

[Install]
WantedBy=multi-user.target

#-----------------------------------------------------------------------------
# EOF
#-----------------------------------------------------------------------------
```

4.2 Build timestamp

In order to keep application build timestamp accurate when using v22 framework function v22.Get_Build, one must always delete v22.o to force an exact build date.

This is implemented in GNAT Remote choosing one of the [FULL] options.

4.3 Using Samba

When the v22 library is located in another directory (as v22 is LGPL v3 licencing but hex01* programs are proprietary), it is advisable to create a symbolic link to avoid v22 sources duplication.

If the “local” directory is a share hosted on a Samba server, it will probably not be possible to create a symbolic link for the v22 library and its dependencies. To get around this problem, create the symbolic link *directly on the Samba server*. Here's an example:

With v22 on /srv/smb/sowebio/informatique/dev/gpl/github/v22:

```
root@i10rs3    cd /srv/smb/sowebio/informatique/dev/prv
root@i10rs3    /srv/smb/sowebio/informatique/dev/prv >ln -s ../gpl/github/v22
```

From a station point of view, even if this directory does not appear as a symbolic link, any alteration to this directory will be reflected on ../gpl/github/v22.

Learning Ada

Investment in C programs reliability will increase up to exceed the probable cost of errors or until someone insists on recoding everything in Ada.
Gilb's laws synthesis

1 Introduction

Ada is not just programming, Ada is software engineering.

Before study Object Oriented Programming, study first Structured Programming.

2 Requirements

GNAT Toolchain.

3 Historical books

Structured Programming – Dahl, Dijkstra, Hoare [1972]

Principle of Program Design – Jackson [1975]

A Structured programming Approach to Data – Coleman [1978]

4 Ada books

Ada avec le Sourire – Bergé, Donzelle, Olive, Rouillard [1989]

Méthodes de Génie Logiciel avec Ada 95 – Rosen [1995] - https://fr.wikibooks.org/wiki/M%C3%A9thodes_de_g%C3%A9nie_logiciel_avec_Ada

Ada Essentials: Overview, Examples and Glosssary – Crawford [2000]

Ada distilled – Riehle [2003] - https://www.sigada.org/wg/eduwg/pages/Ada-Distilled-07-27-2003-Color-Version.pdf

Adacore books: multiple authors - https://www.adacore.com/books

5 Ada courses

Ensemble pédagogique IUT - Feneuille (2003) - http://d.feneuille.free.fr/paquetage.htm

6 Ada links

Le langage Ada: https://www.adalog.fr/fr/faq_ada.html

https://this-page-intentionally-left-blank.org

Coding examples

Variables won't; Constants aren't.
Osborn Law

1 HAC Ada interpreter

If you're not an experienced programmer, you are invited to use HAC, an outstanding Ada subset interpreter.

HAC is available at: https://github.com/zertovitch/hac

HAC documentation (source) is available at: https://github.com/sowebio/hac-doc

2 GNAT Studio Examples

2.1 Drink dispenser

<<<TODO>>>

2.2 Sorting algorithm

<<<TODO>>>

3 GNAT Studio Examples (AVR 8 bits microcontroller)

Ada Development on 8 bits AVR Microcontroller (ADAM) is based on the latest GNAT, Alire and GNAT Studio releases and allows real-time AVR debugging in GNAT Studio.

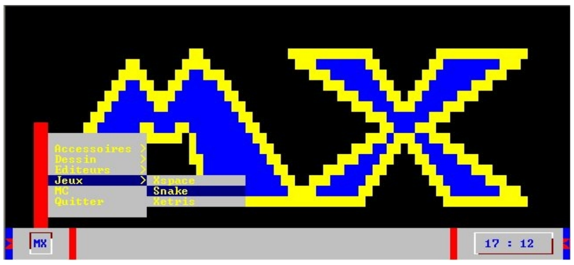
ADAM is available at: https://github.com/sowebio/adam-doc

4 Programs from the MX Team

Theses programs are available at: https://github.com/sowebio/aide-repository

In directory: examples/aide/projects/mx-team

4.1 Mx



4.1.1 Overview

Mx was coded by Xavier, 13 years old in 2004, when he discovered programming and Ada five months before. Mx is an application launcher.

The Start" button, named here "Mx" is in relief. The menus are nested. Mx uses the '.vsl' resource files created by Visual. Visual also use Mx as a main program.

4.1.2 Build

The sources of Mx are available in:

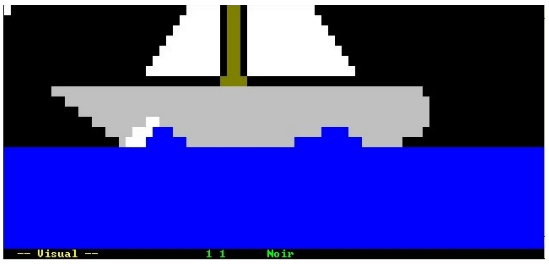
<<<TODO>>>

4.1.3 Usage

- General commands

- [Esc]	Exit	
- [Enter ↲]		Validation
- [←] [↑] [→] [↓]	Move	

4.2 Visual



4.2.1 Overview

Visual was coded by Martin, 13 years old in 2004, when he discovered programming and Ada five months before.

Visual is a text-based screen editor. The created images can be saved in screen image files with the extension '.vsl'. These files can be used directly as external resources by third party applications.

4.2.2 Build

The sources of Visual are available in:

<<<TODO>>>

4.2.3 Usage

- General commands

- [Esc] or [Alt] + [F4]	Exit
- [Ctrl] + S	Save to file
- [Ctrl] + O	Open a file
- [Ctrl] + N	New file

- Selection of the "brush" colors

- [F1]	Black
- [F2]	Blue
- [F3]	Green
- [F4]	Cyan
- [F5]	Red
- [F6]	Magenta
- [F7]	Brown
- [F8]	Grey
- [F9]	Yellow
- [F10]	White
- [Ctrl] + [F2]	Light blue
- [Ctrl] + [F3]	Light green
- [Ctrl] + [F4]	Light cyan
- [Ctrl] + [F5]	Light red
- [Ctrl] + [F6]	Light magenta

4.3 Updates from original 2004 release

4.3.1 Overview

MX team programs were developed in 2004 and tested under Windows 2K only, using some functions from the v04 library, a console multi-platform library for Windows and ANSI console.

v04 library has been resurrected and then simplified to use ANSI console only. Some Windows special features were hardcoded in MX team programs to get graphic effects. They have been slightly modified to handle this new environment.

<<<TODO>>>

https://this-page-intentionally-left-blank.org

Programming basics

Weinberg's Second Law : If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.
Gerald Weinberg

Ada is very well suited for educational purposes. If your are not an experienced programmer mastering a procedural method programming as a tool, you may find this chapter useful. Your creative spirit's the limit.

If your are not an experienced programmer mastering a method programming as a tool, you may find this chapter useful. Your creative spirit's the limit.

This chapter deals with top-down analysis and modular programming method. Understanding and assimilating the following will already make you a very good developer.

This matter is not an end but a foundation to go further. Like understanding the differences between object programming by classification or by composition. And why, for many projects, object programming should be avoided and for others, it should really be adopted.

So, no object methods will be discussed. It's beyond the scope of this manual. Most developers using object-oriented languages have not learned any methods, using wrong tools with no thinking. We know the result.

To be good at object-oriented development, you must already understand the basics of analysis and modular and structured programming.

One step after the other :]

1 Tools

To create a program, you must:

- Master an analytical method;

- Know Boolean algebra;

- Use a programming language;

- Have a good general culture and know-how.

Of these four elements, the first one is the most difficult to acquire, but I hope that the following lines will help you in this field.

I could have added: paper, a pencil and an eraser, because these three objects are always the basis of a good program and you should not rush to code.
always the basis of a good program and that one should not rush to code.

You will notice that the knowledge of a language comes after the theory. This is normal. As analysis precedes writing, mastering design precedes mastering a language. Finally...

The joy of programming must remain the driving force of your motivation.

2 Analysis

2.1 Methods overview

The main classes of methods are:

- Modular and structured programming ;

- Object method by composition

- Object method by classification.

The modular and structured programming method is still used in many fields as the main programming method.

It is also used in object methods, at least in the following contexts:

- In the main startup and finalization module;

- In the functions (methods) of the objects.

Object method can be divided into object methods by composition or by classification.

The object method by classification (hierarchical) is the most known object method *and yet the least relevant*, except for developing a graphical interface or any other project clearly requiring the inheritance tool.

Object method programming is beyond the scope of this manual.

2.2 Top-Down example

The top-down analysis approach is one among many. It is intuitive and efficient. One can fly rockets with it but it is good that you know that other ways exist.

Everyone programs, the car mechanic, the postal worker and the cook. Didn't you know that? So let's start by cooking an egg!

Mastering an analysis method allows to *analyze a problem*, even a very complex one, and break it down by *successive refinements*, into a sum of problems, one by one so obvious to solve, that one stops the analysis by declaring it is finished!

So we're going to cook an egg, a hard-boiled egg to be precise. But could you detail such a seemingly simple process without hesitation? Let's see it together.

2.2.1 Problem's decomposition

We could, for example, start by breaking down, by *refining*, the action of cooking an egg into two main steps two main steps: *preparation* and *cooking*.

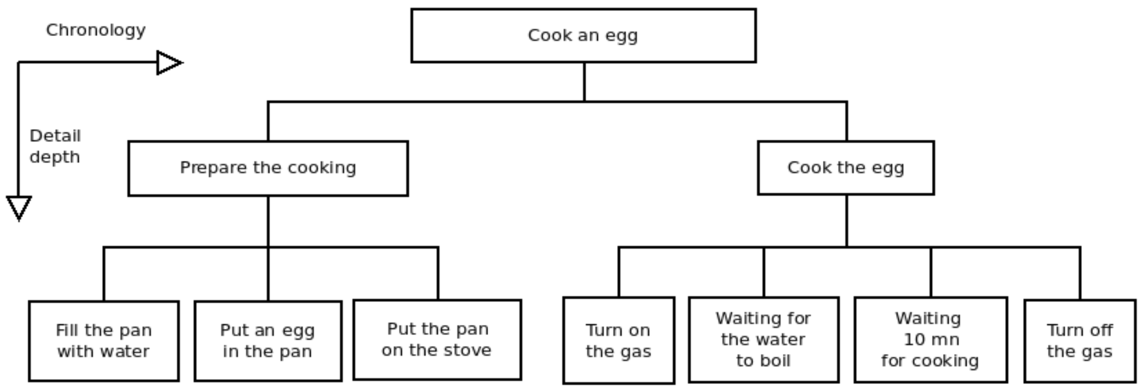
Then we could take these two main steps and refine them again:

- The preparation is to fill the pan with water, put an egg in it and put the pan on the stove;

- Cooking is turning on the gas, waiting for the water to boil, wait 10 minutes for cooking[14] , then turn off the gas.

This approach is known as *decomposition by successive refinements*.

Once this decomposition is completed, it is essential to represent it visually, thanks to the *PSD*, the *Program Structure Diagram*, sometimes called the *JSP Structure Diagram*, after its inventor[15]:



This PSD works in two dimensions:

- In the vertical plane, we go down from the most complex to the simplest;

[14] It's a lot, but not a problem, unless you like them soft. The shell will come off more easily.

[15] It is difficult to determine the origin of these concepts. Many researchers worked on them at the same time. One of Jackson's merits was to promote the notion of initial read-current read in loop processing. https://en.wikipedia.org/wiki/Jackson_structured_programming

- In the horizontal plane, the direction of the reading represents naturally, chronologically, the tasks to be performed.

The PSD has a dual purpose:

- In the first instance, it allows you to gain an *overview of the problem at hand* and ensure that your analysis is *consistent and complete* ;

- Secondly, since each box represents an action that is so simple to solve that it does not require further analysis, the PSD allows you to go directly to the second phase: *the pseudo-code*!

In creating this PSD, we have *modularized* our problem. We have *decomposed* our problem into a series of *elementary modules*. When writing the *pseudo-code*, we will describe the functioning of each *module* using *structures*. These *structures* form the basic building blocks of *structured* programming, without goto or spaghetti code.

2.2.2 Pseudo-code

The pseudo-code is the computer translation, as structures, of the already written PSD.

The PSD and the pseudo-code are linked. They must be consistent with each other.

It is often while checking this consistency, at the time of writing the pseudo-code, that one realizes that the level of detail of the PSD is incorrect. If the level of detail is too high, the pseudo-code contains useless modules that do not contain any processing that deserves to be modularized. On the other hand, if the level of detail is not high enough, the pseudo-code contains modules that are far too big.

Before going into the details of the general writing of a pseudo-code, let's see a small example,
with our hard-boiled egg, just to get a taste of it.

```
begin *** cook an egg ***

do *** prepare the cooking ***
do *** cook the egg ***

end *** cook an egg ***
```

In this first pseudo-code, representing the main module of the "cook an egg" program, the analogy between PSD and pseudo-code is clear.

The term *do* before *prepare the cooking* represents the call to the module *prepare the cooking*. Each module starts with *start *** module name **** and ends with *end *** module name ****.

Let's move on to writing prepare the cooking module:

```
begin *** prepare the cooking ****
```

```
 do while “pan is not filled”
  fill with water
 end do while

 do *** put the pan on the stove ***

 do *** put an egg in the pan ***

end *** prepare the cooking ***
```

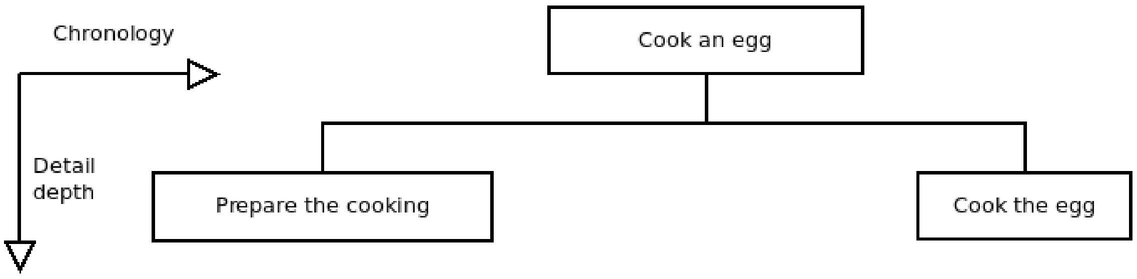
This is when a problem arises. The module putting the pan on the stove is a really very simple action. A so simple one that it does not, in fact, deserve to be isolated in a module. Leaving the analysis as it is, without changing anything, would result in making the program more complex than it deserves to be.

So we will simplify the pseudo-code:

```
begin *** prepare the cooking ****

 do while “pan is not filled”
  fill with water
 end do while

 put the pan on the stove

 put an egg in the pan

end *** prepare the cooking ***
```

So it appeared that the level of detail in the PSD was too high. The actions of the last rank: *pan on the fire*, *fill with water*, etc. did not deserve, by themselves, a separate module.

They should be grouped together in the modules of higher rank: *prepare the cooking* and *cook the egg*.



The analysis of *the cooking of the egg* ends with the pseudo-code of the last module:

```
begin *** cook the egg ***

 turn on the gas

 do while “water does not boil”
  wait
 end do while

 do while “not 10 minutes elapsed”
  wait
 end do while

 turn off the gas
```

```
end *** cook the egg ***
```

After this example, we now take a closer look at this analysis method.

3 Modular and structured programming method

3.1 Introduction

This modular and structured programming approach is generic to dozens of methods invented in the 1980s to make software execution more reliable and improve maintenance.

These methods differed essentially in the symbols, vocabulary and aesthetics of the diagrams. They are still relevant today as the indispensable basis of the methods used by a good developer.

The method illustrated here is GMSP: General, Modular and Structured Programming[16]. It comes from the teaching provided by the french Control Data Institute, located in Paris, which has now disappeared, with the help of PLATO[17], a Computer Aided Learning sytem. Graphical extensions to these methods exist, for example SADT or its real-time extension SART.

The author does not really appreciate graphical representations (which make nice drawings for IT managers) in analysis methods. Flowcharts, flow diagrams, SADT or UML graphs generally bring more confusion than information.

However, some graphical representations, such as the PSD or the HOOD method diagrams, are good tools. They are the first steps of the written specifications, which can be found, strictly speaking, in the specifications of an Ada package.

3.2 Program Structure Diagram

Writing a PSD - *Program Structure Diagram* - means identifying, decomposing and prioritizing functions in a coherent whole, in order to allow the writing of the program pseudo-code.

3.2.1 Process detailed

The process of creating the PSD is an iterative one, which loops around itself, to identify all the tasks to be carried out, until the possibilities of refinement are exhausted, i.e. until the problem to be solved can no longer be detailed.

This approach is called a *top-down approach*, in order to show that we start from the global problem, at *the top of the diagram*, and work our way down to the smallest detail, *towards the bottom of the diagram*. Each time we add a level of detail, we create a new line.

[16] PGMS in french, as "Programmation Générale, Modulaire et Structurée"

[17] Programmed Logic for Automatic Teaching Operations - https://en.wikipedia.org/wiki/PLATO_(computer_system)

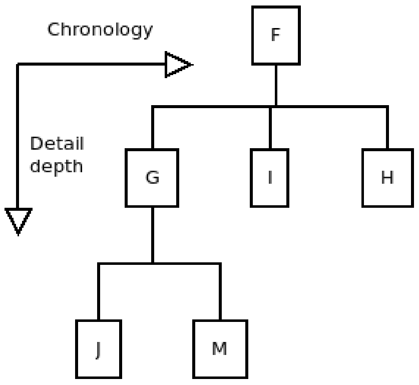
For each detail level, the identified tasks are written in the reading direction, in order of execution. They are placed in *boxes*. For clarity of the PSD, all boxes of a lower rank are connected by lines to the box of the higher rank.

PSDs are always written and read:

- Top to bottom, for level of detail;

- From left to right, for chronological steps.

Example:



In no case does a PSD show the tests and other low-level actions that are the responsibility of programming.

A PSD is both the overview and the backbone of the analysis.

Writing the PSD is the most difficult part of the analysis.

3.3 Pseudo-code

The pseudo-code writing is done from the PSD. Each *box* of the PSD will correspond to a module in the *pseudo-code*.

⇨ We repeat: one PSD box to one module in the pseudo-code.

The writing of a pseudo-code is done from elementary bricks, which we will examine now.

3.3.1 Main module

A program *starts* and *ends* at the master module.

Here is the pseudo-code, also called PC, of the previous PSD, describing the master module of program F:

```
begin *** F ***

 do *** G ***

 do while P (while P is true)
  do *** I ***
 end do while
 do *** H ***

end *** F ***
```

The beginning of a module is represented by begin *** module name *** and the end of a module is represented by end *** module name ***.

The name of the main module is the name of the program.

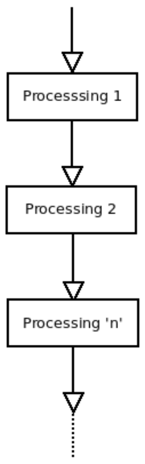
3.3.2 Other modules

Other modules are written the same way. Here is the pseudo-code of module G of the previous PSD, describing the program G:

```
begin *** G ***

 do *** J ***

 if Q (if Q is true)
  do *** M ***
 end if

end *** G ***
```

3.3.3 Sequence

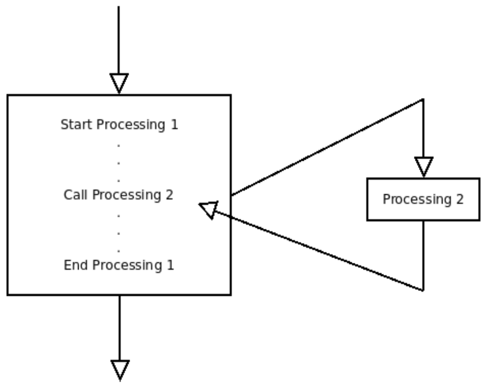
Sequence is the simplest form of pseudo-code. It just represents the sequence of several processes, which are executed one after the other:



3.3.4 Module call

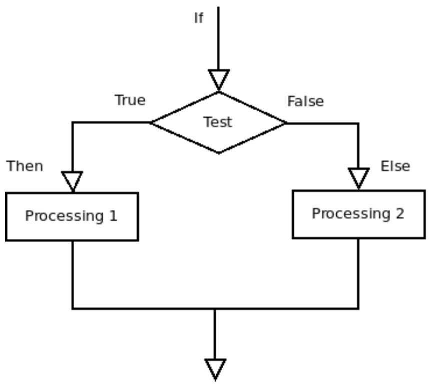
Module call is represented by *do *** module name ****. The processing of the calling module stops at the line of the call and the called module executes.

At the end of the called module, the latter returns to the calling module and the execution of the latter resumes at the line following the call which has just been executed:



3.3.5 If... else... end if

The alternative is the simplest test of a pseudo-code. Depending on the truth of the test condition, the program flow is directed to one processing or another:



The alternative is represented in pseudo-code as follows:

```
if test condition (is true)
 Processing 1
else
 Processing 2
end if
```

❑ If... elsif... else... endif

This structure is an extension of the alternative:

```
if test condition 1
 Processing 1

elsif test condition 2
 Processing 2

elsif test condition 3
 Processing 3

else
 Default proessing
end if
```

The default processing is executed when no test condition has been checked.

This structure is equivalent to a nesting of alternatives. But these nestings are much less readable, as shown in the example below:

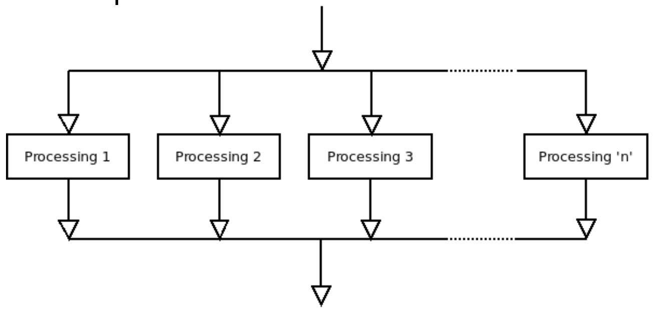
```
if test condition 1
 Processing 1
else
 if test condition 2
  Processing 2
 else
  if test condition 3
   Processing 3
  else
   Default processing
  end if
 end if
end if
```

3.3.6 Case… when… else… end case

The selection is a different form of the alternative because the test is no longer Boolean (true or false) but depends on the content of the tested value. A pseudo-code is more meaningful:

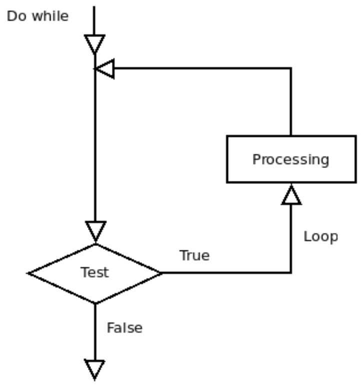
```
selection value to test

 when value 1
  Processing 1

 when value 2
  Processing 2

 when value 3
  Processing 3

 when others
  Default processing

end selection
```

This structure can be represented as follows:



3.3.7 Do while… end do

This loop structure is useful when you want the program flow *to avoid processing in the loop if the condition is false* at the *first pass in the loop*:

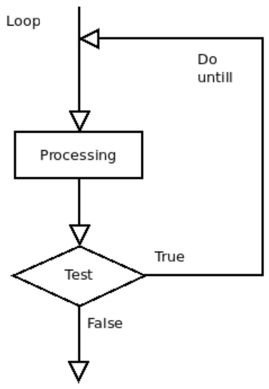


The pseudo code of such a structure is as follows:

```
do while test (is true)
 process
end do while
```

3.3.8 Loop… until

This loop structure differs from the previous one because the processing in the loop is done once before the loop condition is tested. Thus, one will always pass at least once in this type of loop:



Here is the notation of the loop... until in pseudo-code:

```
loop
 process
until condition test (is true)
```

It is clear that the test is performed after a first pass in the loop.

3.4 Functions

➪ One point of entry, one point of exit. No anticipated exit. Never. We repeat: never :]

All parameters will be named and, if the language - such as Ada - allows it, the parameter names will be used in the function calls.

4 Boole algebra

Here is a practical summary about Boolean algebra, which should be known by all developers.

4.1 Identities, properties and De Morgan's laws

Two conventions are used:

– ≡ for equivalence. A ≡ B means that A and B are two equivalent conditions and that they are interchangeable;

– NOT A for the negation of A. If A is true, NOT A is false.

4.1.1 Identities

```
A OR 0 ≡ A                          NOT (NOT A) ≡ 1
A OR 1 ≡ A                          A OR A ≡ A
A OR (NON A) ≡ 1                    A AND A ≡ A
A AND (NON A) ≡ 0
```

4.1.2 Properties

```
A AND B ≡ B AND A
A OR B ≡ B OR A
A AND (B AND C) ≡ (A AND B) AND C
A OR (B OR C) ≡ (A OR B) OR C
A AND (B OR C) ≡ (A OR B) ET (A OR C)
A AND (B OR C) ≡ (A AND B) ET (A AND C)
A OR (B AND C) ≡ (A OR B) ET (A OR C)
```

4.1.3 De Morgan's law

```
NOT (A OR B) ≡ (NOT A) AND (NOT B)
NOT (A AND B) ≡ (NOT A) OR (NOT B)
```

4.2 Practical advises

In your current language manual, you will certainly find the description of priorities in the evaluation of logical expressions.

The following is an example of evaluation priorities:

1. Expressions located in the innermost brackets;

2. Negation;

3. AND and OR [In the Ada language, these two operators are on an equal footing, which is not the general rule in other languages where AND usually has a higher priority than OR];

4. With equal priority, evaluate expressions from left to right.

⇨ One might be tempted to take these priorities into account to write the shortest possible test condition, but *this should be avoided at all costs* for reasons of clarity.

Here are three basic rules *to follow in all circumstances*:

1. Never hesitate to *use parentheses* to increase readability and reliability.

2. To work on or reverse a complex condition, you must *first restore the implicit parentheses*.

3. A simplification of a complex condition is *done by applying the De Morgan's laws*.

5 Basics algorithms

5.1 Initial reading & current reading in loops

<<<TODO>>>

FAQ

With the Wildebeest and the Penguin, there's no Bull.
Number Six

1 System & compiler

1.1 Error when compiling

In the event of an incomprehensible error, for example following a major refactoring, the first action should be to switch to the command line to execute the following commands, at the project's root:

```
user@system : cd <your alire project root>
user@system : rm -r alire config obj
user@system : alr build

ⓘ Synchronizing workspace...
Dependencies automatically updated as follows:

   +   gnat      15.1.2 (new,gnat_native,binary)
   +   gnatcoll 25.0.0 (new)
   +   libgpr    25.0.0 (new,indirect)
   +   xmlada    25.0.0 (new,indirect)

ⓘ Building libreframe=0.1.0-dev/libreframe.gpr...
Setup
   [mkdir]         object directory for project LibreFrame
Compile
   [Ada]           testapi.adb
   [Ada]           s-memory.adb
   [Ada]           libreframe.adb
   [Ada]           libreframe-config.adb
   .../...
   [Ada]           gid-buffering.adb
   [Ada]           gid-color_tables.adb
   [Ada]           gid-decoding_png-huffman.adb
Bind
   [gprbind]       testapi.bexch
   [Ada]           testapi.ali
Link
   [link]          testapi.adb
✓ Build finished successfully in 27.97 seconds.
```

Most of the time, all errors will have disappeared and the build will be successful.

⇨ Alire and config directories are always created and kept up to date by Alire.

1.2 Error when trying to reading documentation: No HTML browser specified

Q: I see theses errors in message console :

```
Launching xdg-open to view file:///home/sr/opt/gnat-2020/share/doc/gnatstudio/html/tutorial/index.html
[2021-03-14 21:45:08] No HTML browser specified
[2021-03-14 21:45:17] No HTML browser specified
[2021-03-14 21:45:37] No HTML browser specified
[2021-03-14 21:49:04] No source file selected
Launching /usr/bin/firefox to view file:///home/sr/opt/gnat-2020/share/doc/gnatstudio/html/tutorial/in-
dex.html
Launching /usr/bin/firefox to view file:///home/sr/opt/gnat-2020/share/doc/gnatstudio/html/users_guide/
index.html
```

A: Sets the real path of your browser of choice:

Edit > Preferences > External Commands > Browser > HTML Browser : /usr/bin/firefox %u

1.3 No GNAT Studio icon in dock

Check ~/.local/share/applications/gnatstudio.desktop or /*usr*/share/applications/gnatstudio.desktop :

```
[Desktop Entry]
Name=GnatStudio
Icon=/opt/gnatstudio/share/gnatstudio/icons/hicolor/32x32/apps/gnatstudio_logo.png
Exec=/opt/gnatstudio/bin/gnatstudio
Terminal=false
Type=Application
MimeType=application/x-adagpr
Categories=Development;
StartupWMClass=gnatstudio_exe
```

If missing, create it.

1.4 Association lost between .gpr project files and GNAT Studio

Check GNAT Studio icon in dock [see above].

If a .gpr file has been opened with a program other than GNAT Studio, the association between .gpr project files and GNAT Studio may have been lost.

Right-click on a .gpr file, choose Properties and go to the “Open With” ta, select GNAT Studio b and then click the [Reset] button. The original association with GNAT Studio is restored.

If GNAT Studio choise does not appear, check file: ~/.local/share/mime/packages/x-adagpr.xml

```
<?xml version="1.0" encoding="utf-8"?>
<mime-type xmlns="http://www.freedesktop.org/standards/shared-mime-info" type="application/x-adagpr">
  <!--Created automatically by update-mime-database. DO NOT EDIT!-->
  <comment>GNAT Gprbuild file</comment>
  <glob pattern="*.gpr"/>
</mime-type>
```

Check file: ~/.local/share/mime/packages/application-x-adagpr.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mime-info xmlns="http://www.freedesktop.org/standards/shared-mime-info">
```

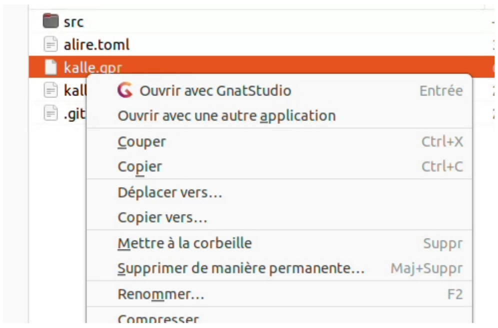
```
  <mime-type type="application/x-adagpr">
    <comment>GNAT Gprbuild file</comment>
    <glob pattern="*.gpr"/>
  </mime-type>
</mime-info>
```

If one of these files are missing, create application-x-adagpr.xml and run:

```
user@system : update-mime-database
```

x-adagpr.xml will be created automatically.

Right click on a .gpr file, you must see “Open with GNAT Studio”:


src
alire.toml
kalle.gpr
kall
.git
Ouvrir avec GnatStudio   Entrée
Ouvrir avec une autre application
Couper   Ctrl+X
Copier   Ctrl+C
Déplacer vers…
Copier vers…
Mettre à la corbeille   Suppr
Supprimer de manière permanente…   Maj+Suppr
Renommer…   F2
Compresser

1.5 GNAT Runtime help tree is altered in GNAT Studio

Q: Instead of having the package tree directly, you have to go through a whole path of intermediate menus before reaching the package menu.

A: You have initiated a run-time debugging session by uncomment the line

```
for Runtime ("Ada") use "/home/sr/opt/gnat-YYYY/lib/gcc/x86_64-pc-linux-gnu/X.Y.Z/rts-native-debug";
```

in the .gpr project file. This has the side effect to alter Menu > Help > GNAT Runtime tree package help files.

1.6 How file association is processed by the system

This is related to previous section above and list conditions summary for file type handling.

A GNAT Studio launcher specifies the MIME[18] type for the GNAT Studio application:

```
~/.local/share/applications/gnatstudio.desktop

[Desktop Entry]
Name=GnatStudio
Icon=/opt/gnatstudio/share/gnatstudio/icons/hicolor/32x32/apps/gnatstudio_logo.png
Exec=/opt/gnatstudio/bin/gnatstudio
```

[18] MIME [IANA Types] stands for Multipurpose Internet Mail Extensions. For more information refers to: https://datatracker.ietf.org/doc/html/rfc6838.

```
Terminal=false
Type=Application
MimeType=application/x-adagpr
Categories=Development;
StartupWMClass=gnatstudio_exe
```

The association file between extension .gpr and MIME type is done by this file:

```
~/.local/share/mime/packages/application-x-adagpr.xml

<?xml version="1.0" encoding="UTF-8"?>
<mime-info xmlns="http://www.freedesktop.org/standards/shared-mime-info">
  <mime-type type="application/x-adagpr">
    <comment>GNAT Gprbuild file</comment>
    <glob pattern="*.gpr"/>
  </mime-type>
</mime-info>
```

Finally, we update the MIME and Desktop databases:

```
MIME & Desktop DB updates
user@system: update-mime-database ~/.local/share/mime
user@system: update-desktop-database ~/.local/share/applications
```

By the way, a file is automatically generated:

```
~/.local/share/mime/application/x-adagpr.xml

<?xml version="1.0" encoding="utf-8"?>
<mime-type xmlns="http://www.freedesktop.org/standards/shared-mime-info" type="application/x-adagpr">
  <!--Created automatically by update-mime-database. DO NOT EDIT!-->
  <comment>GNAT Gprbuild file</comment>
  <glob pattern="*.gpr"/>
</mime-type>
```

Association test:

```
user@system: gio mime application/x-adagpr

Application par défaut pour « application/x-adagpr » : gnatstudio.desktop
Applications inscrites :
   gnatstudio.desktop
Applications recommandées :
   gnatstudio.desktop
```

1.7 Where are stored GNAT Studio configuration files ?

Personal setting are located in:

```
~/.gps (2019)
~/.gnatstudio (2020 and later)
```

2 Ada

2.1 Ada.Containers.Vectors with records

2.1.1 Declarations

<<<TODO>>>

2.1.2 Write

<<<TODO>>>

2.1.3 Read

```
DB_Index := Databases.First_Index;
Msg.Dbg ("Test DB_Properties: " & Databases(DB_Index).Name);
```

2.1.4 Iterates

```
  DB_Cursor : Databases_List.Cursor := Databases_List.No_Element;
begin
  while DB_Cursor /= Databases_List.No_Element loop
    DB_Cursor := Next (DB_Cursor);
  end loop;
  DB_Index := To_Index (DB_Cursor);

  for I of Databases loop
    if I.Name = DB_Name then
      exit;
    end if;
  end loop;
```

2.1.5 Search

```
- Test
  DBI := Sql.DB_Index ("v23");
  if DBI /= Sql.Databases_List.No_Index then
     Msg.Dbg ("DB_Index found: " & From_Latin_1 (DBI'Image));
     Msg.Dbg ("URI from Index: " & Sql.Databases(DBI).URI);
  else
     Msg.Err ("DB_Index not found: v23");
  end if;

- DB_Index not implemented in the API, as it required the Databases container instance
to be public

function DB_Index (DB_Name : String) return Databases_List.Extended_Index is
   use Databases_List; -- for operators
   DB_Index : Databases_List.Extended_Index := Databases_List.No_Index;
begin

   for C in Databases.Iterate loop
      --Msg.Dbg ("Index: " & From_Latin_1 (Databases_List.Extended_Index'Image (To_In-
dex (C))));
   if  Databases(C).Name = DB_Name then
        DB_Index := Databases_List.Extended_Index (To_Index (C));
      exit;
      end if;
   end loop;
   return DB_Index;
end DB_Index;

function DB_Properties (DB_Name : String) return Database_Line is
   use Databases_List; -- for operators
   DB_Index : Databases_List.Extended_Index := Databases_List.No_Index;
   DB_Record : Database_Line;
begin
   for C in Databases.Iterate loop
      --Msg.Dbg ("Index: " & From_Latin_1 (Databases_List.Extended_Index'Image (To_In-
dex (C))));
   if  Databases(C).Name = DB_Name then
       DB_Record := Databases(C);
      exit;
      end if;
   end loop;
  return DB_Record;
end DB_Properties;
```

2.2 Check calls to external libraries

Use the LDD utility:

```
user@system: ldd ./test

linux-vdso.so.1 (0x00007ffcb9dd9000)
libz.so.1    =>    /home/sr/Seafile/Sowebio/informatique/dev/ada/lib/zlib-1211/contrib/ada/bin/./../../../
libz.so.1 (0x00007f3fcf111000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f3fcef0d000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3fceb1c000)
/lib64/ld-linux-x86-64.so.2 (0x00007f3fcf32c000)
```

The line in bold is a link to a specific library.

If the program is statically linked:

```
user@system: ldd ./test

          is not a dynamic executable
```

2.3 Converting reminder

2.3.1 Converting Integer to String with Character'Val and Integer'Image

65 is ASCII code for 'A' :

```
Tio.Put_Line (Integer'Image (65));
The string "65"

Tio.Put_Line (Character'Val(65));
The string "A"
```

2.3.2 Converting a character to its ASCII value

65 is ASCII code for 'A' :

```
Tio.Put_Line (Character'Pos('A'));
The string “65”
```

2.3.3 Converting String from and to Long_Integer

```
procedure Test is

      Test_String : String := "";
      Test_Long_Integer : Long_Integer := 10737418240;

   begin

      Test_String := To_String (Test_Long_Integer); -- Test_Long_Integer´Image

      Msg.Std (Test_String);

      Msg.Std (Test_Long_Integer);

      -- Non qualified expression conversion
      Test_Long_Integer := Long_Integer'Value ("10737418240");

      -- Qualified expression conversion
      Test_Long_Integer := Long_Integer'Value (String ' ("10737418240"));

end Test;
```

2.4 Library integration with .gpr

This is for your information as that's not the way we do things with Alire.

Check paths:

```
user@system: Cd ~/opt/alire/gnat_native_12.2.1_11f3b811/bin

user@system: gnat ls -v

GNATLS 12.2.0
Copyright (C) 1997-2022, Free Software Foundation, Inc.

Source Search Path:
   <Current_Directory>
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/lib/gcc/x86_64-pc-linux-gnu/12.2.0/adainclude


Object Search Path:
   <Current_Directory>
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/lib/gcc/x86_64-pc-linux-gnu/12.2.0/adalib


Project Search Path:
   <Current_Directory>
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/x86_64-pc-linux-gnu/lib/gnat
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/x86_64-pc-linux-gnu/share/gpr
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/share/gpr
   /home/dv/opt/alire/gnat_native_12.2.1_11f3b811/lib/gnat
```

2.5 Program calls analysis

Practical calls analysis. Useful to know which library is really called, and after which attempts. One will be surprised to see how many attempts a program can make before finding (or not) the wanted library:

```
user@system: sudo apt install strace ltrace

user@system: strace -o sortie.txt ./programme

user@system: strace -c ./programme

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 38.64    0.004999           3      1451           write
 33.58    0.004344           2      2718           read
 10.69    0.001383           7       202        20 openat
  6.89    0.000892           5       182           close
  3.97    0.000513           1       363           fstat
  2.93    0.000379           4       102           brk
  2.27    0.000294           2       177           getcwd
  0.39    0.000050          50         1           munmap
  0.32    0.000042           8         5           rt_sigaction
  0.19    0.000024           3         8           mprotect
  0.07    0.000009           9         1           sigaltstack
  0.06    0.000008           8         1           lseek
  0.00    0.000000           0        16        14 stat
  0.00    0.000000           0        10           mmap
  0.00    0.000000           0         5         5 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           readlink
  0.00    0.000000           0         1           arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.012937                  5245        39 total
```

In our case, we wanted to understand why the example program did not compile and therefore did not use the zlib library. The contributor to the demo program considered that the zlib library was installed by default at the system level. In the case of Ubuntu, via the package zlib1g (l=one).

2.6 Statically link an external library to an executable

To statically link zlib, you need to put the options below in the right order:

– First the search paths;

– Then the library or libraries.

Copy the static library libz.a to the current directory is allowed (or to ./obj if the Object_Dir use 'obj' clause is used), but this is not a very clean way to proceed. It's better to use the path specification parameter -L.

The usage is, however, tricky:

– This parameter will not support any spaces or dots in the path;

– If both versions - shared and static - of the library exist in the same directory, the shared library libz.so will always be chosen over the static library libz.a;

– To force the choice of the static version, you must then specify by name the library to be statically linked with the -l:libz.a option instead of -lz.

Example:

```
-- gprbuild -d -P./zlib.gpr
```

```
project Zlib is

   for Languages   use ("Ada");

   for Source_Dirs use ("src"); -- Avec parenthèses
   for Object_Dir  use  "obj";  -- Sans parenthèses

   for Main use ("test.adb", "mtest.adb", "read.adb", "buffer_demo");

   -- gnatmake
   --
   -- -gnat w cfilopru         Warnings management
   -- -gnat V cdfimorst        Validity checking mode
   -- -gnat y abcefhiklmnoprst Style checks

   package Compiler is
      for Default_Switches ("ada") use
("-gnatwcfilopru", "-gnatVcdfimorst", "-gnatyabcefhiklmnoprst");
   end Compiler;

   -- ld
   --
   -- -L    Library path (for libz.a)
   --       avoid space(s) and dot(s) in names, accept full qualified and relative paths
   -- -l    Library name (for libz.a)

   package Linker is

   -- valid full qualified path - .so shared lib first
   -- for Default_Switches ("ada")
   -- use ("-L/home/sr/Seafile/Sowebio/informatique/dev/ada/lib/zlib-1211","-lz");
   -- valid relative path - .so shared lib first
   -- for Default_Switches ("ada") use ("-L../../","-lz");

   -- valid relative path - specify libz.a static lib
      for Default_Switches ("ada") use ("-L../../","-l:libz.a");

   end Linker;

   -- gprbuild
   --
   -- -s       Recompile if compiler switches have changed
   -- -gnatQ   Don't quit, write ali/tree file even if compile errors

   package Builder is
      for Default_Switches ("ada") use ("-s", "-gnatQ");
   end Builder;

end Zlib;
```

One can check that the program size has increased by about the same amount as the static library size. One can also check it visually with strace (the call to the library is pathless).

2.7 Statically linked executable embedding the run-time system

To statically link the runtime, you have to put the "-static" option in the binder and the linker, as in the AIDE build file below:

```
------------------------------------------------------------------------------
--
--
--
--
--  @file      aide.gpr
--  @copyright See authors list below and aide.copyrights file
--  @licence   GPL v3
--  @encoding  UTF-8
------------------------------------------------------------------------------
--  @summary
--  aide library project file
--
--  @description
--  Build application and documentation
```

```
--
--  @authors
--  Stéphane Rivière - sr - sriviere@soweb.io
--
--  @versions
--  20210317 - 0.1 - sr - initial release
--  20210331 - 0.2 - sr - Add Style and GNATColl builds
------------------------------------------------------------------------------

--  (0) invert comments for the 3 related lines to unlink gnatcoll sources
--      in order to generate pertinent documentation and true metrics

--  with "gnatcoll"; --  (0)
project aide is

   --  for Languages use ("Ada"); --  (0)
   for Languages use ("Ada", "C");

   type aide_Build_Type is ("Style", "Debug", "Fast", "Small");

   --  Add -Xaide_Build=Style in the GNAT Studio build all window...
   --  %builder -Xaide_Build=Style -d %eL -P%PP %config %autoconf %X
   --  ...to directly control the build behaviour

   aide_Build: aide_Build_Type := external ("aide_Build", "Debug");

   --  for Source_Dirs use ("src/**", "../v20/src/**"); --  (0)
   for Source_Dirs use ("src/**", "../v20/src/**", "/home/sr/opt/gnat-2020/include/gnatcoll");

   case aide_Build is
      when "Style" =>
         for Object_Dir use "obj/style";
      when "Debug" =>
         for Object_Dir use "obj/debug";
         --  Use runtime with debug capabilities
           for Runtime ("Ada") use "/home/sr/opt/gnat-2020/lib/gcc/x86_64-pc-linux-gnu/9.3.1/rts-native-
debug";
      when "Fast"  =>
         for Object_Dir use "obj/fast";
      when "Small" =>
         for Object_Dir use "obj/small";
   end case;

   for Exec_Dir use "bin";
   for Create_Missing_Dirs use "True";

   for Main use ("aide.adb");

   Common_Compiler_Options := (
    -- General
    "-gnatW8",            --  Both brackets and UTF-8 encodings will be recognized (1)
    -- Warnings & Errors
    "-gnatU",             --  Enable unique tag for error messages
    "-gnatf",             --  Full errors. Verbose details, all undefined references
    "-gnatq",             --  Don't quit, try semantics, even if parse errors
    "-gnatQ",             --  Don't quit, write ali/tree file even if compile errors
    "-gnatVaep",          --  Enable selected validity checking mode (2)
    "-gnatw.eDH.Y",       --  Enable selected warning modes (3)
    -- "-Wall",           --  Enable most warning messages
    -- Style
    "-gnatyaefhkM160npr" --  Enable selected style checks (4)
   );

   Style_Compiler_Options := (
   "-gnatg"              --  RTS Style (6)
   );

   Debug_Compiler_Options := (
    "-gnata",             --  Assertions enabled
    "-gnato",             --  Enable overflow checking in STRICT mode
    "-gnateE",            --  Generate extra information in exception messages
    "-gnateF",            --  Check overflow on predefined Float types
    "-gnatVa",            --  Enable all validity checking options
    "-fstack-check",
    "-fno-inline",
    --
    "-gnatec=" & project'Project_Dir & "aide.dbg",
    "-g"                  --  Generate debugging information
   );

   Fast_Compiler_Options := (
    "-O2",
    "-gnatpn",
    "-fipa-cp-clone", "-fgcse-after-reload",
    "-funroll-loops", "-fpeel-loops", "-funswitch-loops",
    "-ftracer", "-fweb", "-ftree-vectorize",
    "-frename-registers", "-ffunction-sections",
    "-g"
   );

   Small_Compiler_Options := (
    "-Os"
   );

   --  (1)
   --  https://gcc.gnu.org/onlinedocs/gcc-4.8.5/gnat_ugn_unw/Character-Set-Control.html
    --   https://gcc.gnu.org/onlinedocs/gcc-4.8.5/gnat_ugn_unw/Wide-Character-Encodings.html#Wide-Charac-
ter-Encodings
   --  (2)
   --  a turn on all validity checking options
   --  e turn on checking for elementary components
   --  p turn on checking for parameters
   --  (3)
   --  .e turn on every optional info/warning (no exceptions)
   --  D  turn off warnings for implicit dereference (default)
   --  H  turn off warnings for hiding declarations (default)
   --  .Y turn off info messages for why pkg body needed (default)
   --  (4)
   --  a  check attribute casing
   --  e  check end/exit labels present
   --  f  check no form feeds/vertical tabs in source
   --  h  no horizontal tabs in source
   --  k  check casing rules for keywords
   --  Mn check line length <= n characters
   --  n  check casing of package Standard identifiers
   --  p  check pragma casing
   --  r  check casing for identifier references
   --  (5)
   --  Options starting with -g, -f, -m, -O, -W, or --param are automatically passed on to the various
sub-processes
   --  invoked by gcc.  In order to pass  other options on to these processes the -W<letter> options
must be used.
   --  (6) All warnings and style messages are treated as errors. -gnatg implies -gnatw.ge and -gnatyg
so that all
   --  standard warnings and all standard style options are turned on. All warnings and style messages
are treated
   --  as errors.'

   --  gnatmake options
   package Compiler is
      case aide_Build is
      when "Style"   =>
         for Default_Switches ("ada") use Common_Compiler_Options & Style_Compiler_Options;
      when "Debug" =>
         for Default_Switches ("ada") use Common_Compiler_Options & Debug_Compiler_Options;
         for Switches ("s-memory.adb") use ("-gnatg");
      when "Fast"  =>
         for Default_Switches ("ada") use Common_Compiler_Options & Fast_Compiler_Options;
         for Switches ("s-memory.adb") use ("-gnatg");
      when "Small" =>
         for Default_Switches ("ada") use Common_Compiler_Options & Small_Compiler_Options;
         for Switches ("s-memory.adb") use ("-gnatg");
      end case;
   end Compiler;

   Common_Binder_Options := ("-static");

   --  gnatbind options
   package Binder is
      case aide_Build is
      when "Small" => for Default_Switches ("ada") use Common_Binder_Options;
         --  -Es: Store tracebacks in exception occurrences, and enable symbolic tracebacks
      when others  => for Default_Switches ("ada") use Common_Binder_Options & ("-Es");
      end case;
   end Binder;

   Common_Linker_Options := ("-static");

   --  ld options
   package Linker is
      -- Static link with external C libs
      -- for Switches ("ada") use ("-L/home/sr/Seafile/Sowebio/informatique/dev/ada/lib/zlib-1211", "-
lz");
      case aide_Build is
      when "Style" =>
         for Default_Switches ("ada") use Common_Linker_Options;
      when "Debug" =>
         for Default_Switches ("ada") use Common_Linker_Options & ("-g");
      when "Fast"  =>
         for Default_Switches ("ada") use Common_Linker_Options & ("-g", "-Wl,--gc-sections");
      when "Small" =>
         for Default_Switches ("ada") use Common_Linker_Options & ("-Wl,--gc-sections");
      end case;
   end Linker;

  --  gprbuild options
   package Builder is
      -- -d   Display compilation process
      -- -j0  Use num processes to compile 0=all platform cores are used
      -- -s   Recompile if compiler switches have changed
      for Default_Switches ("ada") use ("-d", "-j0", "-s");
   end Builder;

   -- gnatdoc options
   package Documentation is -- gnatdoc options
      for Documentation_Dir use "doc-generated";
   end Documentation;

   --  gnatpp option
   package Pretty_Printer is
      for Default_Switches ("ada") use ("-M120", "-W8", "--comments-unchanged");
   end Pretty_Printer;

   --  gps options (to be reworked with appropriate options)
   --  package Ide is
   --   for Default_Switches ("adacontrol") use ("-f", "aide.aru", "-r");
   --  end Ide;

-------------------------------------------------------------------------------
end aide;
-------------------------------------------------------------------------------
```

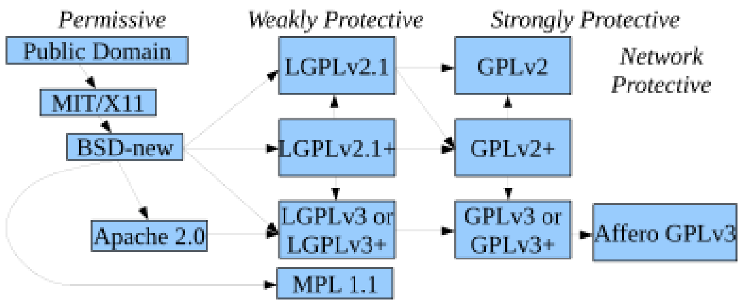
https://this-page-intentionally-left-blank.org

Appendices

1 Copyrights & credits

1.1 Library Licence

v20 is copyright Sowebio under GPL v3 license.



1.1.1 GPL v3 compatibility with others licenses

https://en.wikipedia.org/wiki/License_compatibility: MIT licence is compatible with GPL and can be re-licensed as GPL. European Union Public Licence (EUPL) is *explicitly compatible* with GPL v2 v3, OSL v2.1 v 3, CPL v1, EPL v1, CeCILL v2 v2.1, MPL v2, LGPL v2.1 v3, LiLIQ R R+ AGPL v3.

1.2 Manual license

Copyright © 2023 Stéphane Rivière. This document may be copied, in whole or in part, in any form or by any means, as is or with alterations, provided that alterations are clearly marked as alterations and this copyright notice is included unmodified in any copy.

2 To-do list Documentation

Hunt **<<<TODO>>>** tags :)

3 To-do list Software

<<<TODO>>>

4 Issues

4.1 Compiler bug reporting

Historic and still working report email: report@gnat.com
Since the beginning of the XXIth century: report@adacore.com

4.1.1 GNAT CE 2019 - Exception with Delete_Tree

Ada.Directories.Del_Tree crashes if a broken symbolic link in a directory of the tree to be deleted exists: raised ADA.IO_EXCEPTIONS.USE_ERROR: directory tree rooted at

"/home/sr/opt/gnat-2019/lib/xmlada/xmlada_input.relocatable" could not be deleted.

- Demo

```
Ada.Directories.Delete_Tree > Is_Valid_Path_Name > Is_Directory Ada >
is_Directory C > adaint.c > __gnat_is_directory >
__gnat_reset_attributes > __gnat_is_directory_attr >
*__gnat_stat_to_attr* > __gnat_stat > GNAT_STAT

Around More_Entries > Fetch_Next_Entry > readdir_gnat > Match

Here a broken symbolic link libxmlada_input_sources.so which is declared “non exis-
tent” by File_Exists_Attr (C_Full_Name'Address,Attr'Access); en 776 which is
__gnat_file_exists_attr in 1668 of adaint.c with a reference to a structure in adain-
t.h:

-----------------------------
struct file_attributes {
  int           error;

/* Errno value returned by stat()/fstat(). If non-zero, other fields
should be considered as invalid.  */

  unsigned char exists;

  unsigned char writable;
  unsigned char readable;
  unsigned char executable;

  unsigned char symbolic_link;
  unsigned char regular;
  unsigned char directory;
-----------------------------

Who calls  *__gnat_stat_to_attr* and test a file descriptor at -1, (broken symbolic
link I guess). Then __gnat_stat returns 2 to __gnat_stat_to_att with a test at line
1124 of adaint.c

 if (error == 0 || error == ENOENT)
    attr->error = 0;

In s-oscons.ads ENOENT: constant := 2;  --  File not found !

<shadok> So if you can't find the file, there's no error. </shadok>
```

- Conclusion

  The rest becomes clear... The broken symbolic link libxmlada_input_sources.so is declared to not exist, the routine exits of the current directory (which it is believed to be empty, to delete it, and then crashes when it tries to erase this directory, which is empty but isn't...

- Solving

  We could re-code this recursive function in a simpler way. However, the best thing to do would be to correct the anomaly which is probably in _gnat_stat, so that this function returns the correct value and doesn't confuse 'doesn't exist' (the file to which the symbolic link points) with 'doesn't exist' (the symbolic file).

4.1.2 GNAT GCC 13 - UXStrings

Under Linux, using GNAT GCC 13 and UXStrings creates huge memory leaks while using GNAT GCC 14 and UXStrings works flawlessly (which has been verified by billions of iterations on files of hundreds of GB).

5 Links

5.1 Ada

https://en.wikibooks.org/wiki/Ada_Programming

5.2 Others

Avr Freaks : http://www.avrfreaks.net

Adacore Github : https://github.com/AdaCore
Adacore Papers : https://www.adacore.com/papers
Adacore Gems : https://www.adacore.com/gems
Adacore Books : https://www.adacore.com/books
Adacore GPL : https://www.adacore.com/community

http://www.tldp.org/HOWTO/Avr-Microcontrollers-in-Linux-Howto/x207.html

http://www.avrfreaks.net/forum/i-didnt-know-you-could-get-ada-avr

5.3 People

5.3.1 Ludovic Brenta

Ada Debian Maintainer, and a good friend too.

https://people.debian.org/~lbrenta/debian-ada-policy.html

5.3.2 Stéphane Carrez

Member of Ada-France.

Blog: https://blog.vacs.fr
Sources repository: https://github.com/stcarrez

5.3.3 Gautier de Montmollin

Prolific Ada program author (HAC, Lea, Azip, Gwindows, TexCad, among others), and a good friend too.

Blog: https://gautiersblog.blogspot.com
Sources repository: https://github.com/zertovitch

Ada, « it’s stronger than you ».
Tribute to Daniel Feneuille, a legendary french Ada teacher (and much more)[19]

The link below is kept here for future use...

https://this-page-intentionally-left-blank.org

19 http://d.feneuille.free.fr