sow - AIDE for Debian & Ubuntu Manual

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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Ed.	*Release*	*Comments*	
1	20210411	Initial release	sr
16	20210411	FAQ translation done	sr
22	20210406	Programming basics translation done	sr
23	20210419	Change Humanist 521 BT font to Airbus cockpit free font designed by Intactile[1]	sr
32	20210606	Updates about AIDE 2.14, many enhancements and typos fixed	sr
39	20210730	Updates about AIDE 2.15, major update, fix typos	sr
42			

[1] https://b612-font.com under Open Font License, replaced the Humanist 521 BT licensed by Monotype.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

❏ Author

Stéphane Rivière [Number Six] - stef@genesix.org [CTO Sowebio]

Suggestions in order to improve AIDE are welcome. AIDE means HELP in french.

This work is dedicated to Dino Risi, Vittorio Gasmann and Jean-Louis Trintignant for a sublime, profound and immensely light movie: The Boaster ("Le Fanfaron" in French or "Il Sorpasso" in Italian).

❏ Manual

Stéphane Rivière [Number Six] - stef@genesix.org [CTO Sowebio]
Stéphane Richard - About Ada section from 2005 AIDE 1.x manual.
Ludovic Brenta - Part of History section

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[2]. 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 42 - 2021-07-31

[2] 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* :]

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Contents

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Introduction

1 About AIDE

AIDE, Ada Instant Development Environment, make Ada integrated development environment setup a breeze. AIDE is intended to GNU/Linux Debian, Ubuntu and derivatives using Libre software. AIDE is written in Ada, as too GNATStudio, the Ada IDE and GNAT, the GCC Ada compiler.

We hope that AIDE will inspire new generations to create quality software. Ada is the best insurance to write reliable programs while being creative and having fun with an amazing language!

2 About the Ada Community

At first, thanks to the Ada Community, definitely one of the best.

2.1 Inspiration, ideas, help and more

AdaCore Ada compiler - https://www.adacore.com/community
Daniel Feneuille - df- http://d.feneuille.free.fr
Gautier de Montmollin - gdm - https://github.com/zertovitch
Jean-Pierre Rosen - jpr - https://adalog.fr
Pascal Pignard - pp - https://github.com/Blady-Com
Rolf Ebert - re - https://github.com/RREE

Special thanks to Ada gurus Daniel Feneuille, Gautier de Montmollin and Jean-Pierre Rosen. 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.

3 AIDE history

AIDE has its roots from 2002 (v0.5) to 2005 (v1.4), 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 with AIDE.

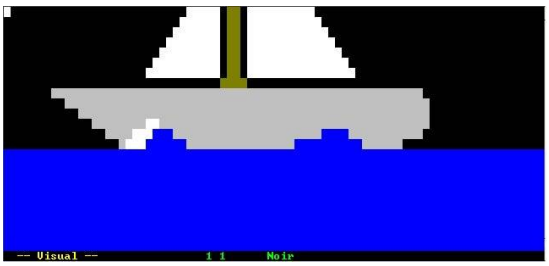
CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Let's hear from Ludovic Brenta[3], 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[4]! 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!"



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 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.

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

[4] Previous name of GNATStudio, GPS was renamed in 2020.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

4 About Ada

Some general thoughts about Ada.

4.1 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[5] and proves itself light enough to be usable on low class 8 bit processors[6].

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[7] as well as an international civil norm[8], 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 (A3xx civil series and A400 military), Alsthom (High speed train), Boeing (777 and 787), EADS (Eurofighter, Ariane, ATV, many spaces probes), STS (line 14 Meteor), NASA (Electric power supply of the International Space Station). The list goes on and on. Everywhere reliability and security must come first, Ada is the language of choice.

4.2 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

5 Thanks to it's representation clauses that obliterates the need to use bit masking for XORed for bit manipulation. This functionality *essential to system programming* is simply not there in pure C or even in Assembly language.

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

7 MIL-STD-1815

8 ISO/IEC 8652

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

high quality. Finally, the Ada community is very highly active and by now growing again.

4.3 The ending word

When Boeing decided, two decades ago, that all software for the 777[9] 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.

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.

[9] 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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Getting started

One can write neatly in any language, including C. One can write badly in any language, including Ada. But Ada is the only language where it is more tedious to write badly than neatly.
Jean-Pierre Rosen

1 Getting AIDE

You can get a pre-build ready to use AIDE at https://github.com/sowebio/aide-bin

1.1 What's in the box ?

Only two files !
- aide
- aide.pdf

1.2 Want more ?

Just launch AIDE !

You'll get:
– AIDE program sources (Ada code and manual) https://github.com/sowebio/aide;
– v20 library sources (Ada code and manual) https://github.com/sowebio/v20;
– GNAT Community Edition **2019** or **2020** or **2021** compiler from AdaCore, with GNATStudio IDE with extension association set;
– Ready to use libraries like: aws, ada-util, gnatcoll, xmlada, zlib;
– HAC, a very capable Ada subset interpreter;
– AdaControl, the well-known Source checker used by EuroControl and many high-tech companies (only with ASIS compliant GNAT CE 2019);

Additionally, you will benefit from:
– Full non interactive installation (unattended) mode for station and server targets.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

– Server target mode for the terminal world. GNAT CE installer is a real pain with distant servers in console mode. With this mode, we can build Ada programs on servers or any small text-mode only systems with the most up to date Ada compiler from AdaCore without hassles;
– Three ways to get install files:
  - Local generated packages;
  - Internet AIDE repository packages;
  - Adacore repository installers.
– A package command to create your own GNAT CE 2019 2020 & 2021 ultra compressed packages (.xz format) to be independent from AdaCore AWS repositories.

➪ AIDE comes with tons of well-written manuals in PDF and HTML formats to support you !

➪ The IDE comes with all cross references facilities and search tools helping you to browse easily in the most huge projects and the the Ada run-time system too.

2 Installing AIDE on an Ubuntu station

➪ AIDE is a real standalone program[10].

No need to install *tons of bullshit and dependencies*, just run it ! Launched without parameter, AIDE installs in /home/user/opt/gnat-2021 (CE 2021) in station mode (full install with IDE, SPARK Discovery and more).

AIDE is intended to *Debian, Ubuntu and derivatives Linux distributions*.

AIDE take care of *apt, apt-get and aptitude package managers* and handles *station sudo users or root server users*.

AIDE needs - and *automatically installs* - some system packages: automake, curl, git, libtool, libcurl4, libcurl4-openssl-dev, libssl-dev and perl.

AIDE may install additional packages. At first, you should update and upgrade your system. Updating and upgrading your own system are *strictly of your duty*. AIDE will never took that responsibility as this may have sides effects that *only an human* could anticipate. Definitely, AIDE is *not* AI software :)

Installing creates:
– ~/opt/gnat-2020 (directory)
– ~/opt/gnat-2020-download (directory)
– ~/.local/share/applications/gps2020.desktop (launcher)
– ~/.local/share/mime/packages/application-x-adagpr.xml (MIME type)
– ~/.local/share/mime/application/x-adagpr.xml (created by the Gnome database update)
– ~/.bashrc updated to remove the PATH to ~/opt/gnat-2020

AIDE in target station mode also handles:

[10] Technically speaking, AIDE doesn't need any external shared library to work as it is statically build, i.e. all libraries are included in the program.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

– Gnome database update;
– The association of the extension .gpr with GNATStudio.

GNATStudio personal settings are created at the first launch of GNATStudio.

The total execution time is displayed at the end of the log, non-root users are managed, the application launcher and the MIME type for project files (.gpr) are created. The Gnome databases are updated to take the .gpr extension into account immediately. The environment is ready as soon as the installation is finished.

2.1 Help screen

At first, let's take a look at the AIDE arguments and options using -h or --help switch:

```
user@system: ./aide --help

AIDE - Ada Instant Development Environment - aide v2.15
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

Usage: aide [install]|activate|deactivate|update|remove|list|package [options]

 -t, --target=TARGET     server|[station] with graphic IDE
 -i, --install=DIRECTORY [~/opt/gnat-YEAR] or home based custom sub-directory
 -y, --year=YEAR         2019|2020|[2021] GNAT CE Year edition
 -c                      Check .err trace raising an exception

Running AIDE without option will create a station install of GNAT CE
2021 with IDE to ~/opt/gnat-2021 with docs, tools and libraries.

AIDE is intended to be used on Debian, Ubuntu & derivatives
distributions. You should first UPDATE & UPGRADE your system before
using AIDE, as some additional packages could be installed.
```

The arguments or switches order are not important, nor the case used.

Install is the default argument and default parameter switch --year is the last usable year in AIDE.

Commands are:

– install install a GNAT CE --=year=YEAR edition
– activate switch on the GNAT CE --=year=YEAR installation
– deactivate switch off the GNAT CE --=year=YEAR Year installation
– update update a GNAT CE --=year=YEAR edition
– remove remove a GNAT CE --=year=YEAR edition
– list list all GNAT CE editions installed
– package package generation for further install GNAT CE 2019 2020 & 2021 editions

Options are:
– t/target station or server target
– i/install installation directory (persistence not yet implemented, don't use it)
– y/year year edition
– c generate an exception to check .err trace recording (dev only)

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

2.2 Install GNAT CE 2021

Running on an Ubuntu Station 18.04 LTS desktop station (Nuc Intel 2016 i5 SSD).

Install free space: less than 8 Go
Minimal free space: 3,7 Go

❑ First step : setting PATH (if not present)

Launch AIDE:

```
user@system: ./aide

AIDE - Ada Instant Development Environment - aide v2.15
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

You are not logged as root.
Your password will be asked if new packages are needed.

Do you wish to INSTALL GNAT CE 2021 for STATION target ?
Press any key to continue or [Ctrl-C] to abort...

20210730 184312 - INIT    - MSG - ------------------------------------------------------------
20210730 184312 - INSTALL - MSG - GNAT path /home/sr/opt/gnat-2021 set in /home/sr/.bashrc
20210730 184312 - INSTALL - MSG - ------------------------------------------------
20210730 184312 - INSTALL - MSG - To taking account of the PATH update:
20210730 184312 - INSTALL - MSG - - Close all terminals, including this one
20210730 184312 - INSTALL - MSG - - Run a fresh terminal with the updated PATH inside
20210730 184312 - INSTALL - MSG - /!\ Not follow this advice could rising problems.
20210730 184312 - INSTALL - MSG - ------------------------------------------------
20210730 184312 - INSTALL - MSG - Run again AIDE to resume and finish the installation.
20210730 184312 - INSTALL - MSG - ------------------------------------------------
20210730 184312 - EXIT    - MSG - Total execution time: 0h00m01s
20210730 184312 - EXIT    - MSG - ------------------------------------------------------------
```

❑ Second step : install AIDE

Open a new console and relaunch AIDE[11]:

```
user@system: ./aide

AIDE - Ada Instant Development Environment - aide v2.15
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

You are not logged as root.
Your password will be asked if new packages are needed.

Do you wish to INSTALL GNAT CE 2021 for STATION target ?
Press any key to continue or [Ctrl-C] to abort...

20210730 185030 - INIT    - MSG - ------------------------------------------------------------
20210730 185030 - INSTALL - MSG - GNAT path already exists in /home/sr/.bashrc
20210730 185030 - INSTALL - MSG - GNAT path /home/sr/opt/gnat-2021 already in PATH
20210730 185030 - INSTALL - MSG - GNAT path OK, no need to start a new console
20210730 185030 - INSTALL - MSG - Check system packages dependencies.
20210730 185030 - INSTALL - MSG - Package automake already installed.
20210730 185030 - INSTALL - MSG - Package make already installed.
20210730 185030 - INSTALL - MSG - Package curl already installed.
20210730 185030 - INSTALL - MSG - Package git already installed.
20210730 185030 - INSTALL - MSG - Package libtool already installed.
20210730 185030 - INSTALL - MSG - Package tar already installed.
20210730 185031 - INSTALL - MSG - Package xz-utils already installed.
20210730 185031 - INSTALL - MSG - Package libcurl4 already installed.
```

11 The CE 2021 package was already on disk, so no download was performed

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
20210730 185031 - INSTALL - MSG - Package libcurl4-openssl-dev already installed.
20210730 185031 - INSTALL - MSG - Package libssl-dev already installed.
20210730 185031 - INSTALL - MSG - Package perl already installed.
20210730 185031 - INSTALL - MSG - Delete previous GNAT unfinished install
20210730 185031 - INSTALL - MSG - Try to install GNAT from AIDE local package repository.
20210730 185032 - INSTALL - MSG - Decompress: gnat-2021-20210519-linux64-station.tar.xz
20210730 185128 - INSTALL - MSG - GNAT installation from archive.
20210730 185147 - INSTALL - MSG - GNAT install done.
20210730 185147 - INSTALL - MSG - Building GNAT debug runtime.
20210730 185245 - INSTALL - MSG - GNAT debug run-time build sucessfully.
20210730 185245 - INSTALL - MSG - GNAT debug ready RTS done.
20210730 185245 - INSTALL - MSG - GNAT install complete.
20210730 185245 - INSTALL - MSG - Desktop launcher file already deleted.
20210730 185245 - INSTALL - MSG - Deleting MIME type package file
20210730 185245 - INSTALL - MSG - Deleting MIME type application file
20210730 185245 - INSTALL - MSG - Updating MIME and Desktop databases.
20210730 185246 - INSTALL - MSG - Deleting desktop launcher file.
20210730 185246 - INSTALL - MSG - MIME type package file already deleted.
20210730 185246 - INSTALL - MSG - Deleting MIME type application file
20210730 185246 - INSTALL - MSG - Updating MIME and Desktop databases.
20210730 185246 - INSTALL - MSG - Desktop launcher file already deleted.
20210730 185246 - INSTALL - MSG - MIME type package file already deleted.
20210730 185246 - INSTALL - MSG - Deleting MIME type application file
20210730 185246 - INSTALL - MSG - Updating MIME and Desktop databases.
20210730 185246 - INSTALL - MSG - GNAT path /home/sr/opt/gnat-2021 set in /home/sr/.bashrc
20210730 185246 - INSTALL - MSG - GNAT path /home/sr/opt/gnat-2021 already in PATH
20210730 185246 - INSTALL - MSG - GNAT path OK, no need to start a new console
20210730 185246 - INSTALL - MSG - Generating MIME Type file.
20210730 185246 - INSTALL - MSG - Generating GPS launcher file.
20210730 185246 - INSTALL - MSG - Updating MIME and Desktop databases.
20210730 185246 - INSTALL - MSG - Check .gpr assoc. with application/x-adagpr MIME type.
20210730 185246 - INSTALL - MSG - Check reg. app. for application/x-adagpr MIME Type.
20210730 185246 - INSTALL - MSG - Download file: AdaWebServer
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
100 4249k  100 4249k    0     0  1291k      0  0:00:03  0:00:03 --:--:-- 2271k
20210730 185250 - INSTALL - MSG - Ada Web Server docs & examples installation.
20210730 185250 - INSTALL - MSG - Download Utilada.
20210730 185254 - INSTALL - MSG - Utilada installation.
20210730 185417 - INSTALL - MSG - Download file: Zlib
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  593k  100  593k    0     0   304k      0  0:00:01  0:00:01 --:--:--  304k
20210730 185419 - INSTALL - MSG - Zlib installation.
20210730 185424 - INSTALL - MSG - Download HAC.
20210730 185436 - EXIT    - MSG - Total execution time: 0h04m15s
20210730 185436 - EXIT    - MSG - ------------------------------------------------------------
```

Put GNATStudio IDE in the favorites of your dock or menu bar. You can now jump to the *GNATStudio at work* chapter to compile your first program !

3 Installing AIDE on a Debian server

Running on a Xen/Debian 10 infrastructure server with 1 Gbps connectivity.

3.1 Install GNAT CE 2021

Install free space: less than 2 Go
Minimal free space: 1,3 Go

In this mode, installation is minimal:
– No GNATStudio (no graphic IDE on console server);
– No rts-native-debug RTS (no debug session on server);
– Delete all documentation files, examples and download files on server.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

❑ First step : setting PATH [if not present]

Launch AIDE:

```
user@system: aide -t server

AIDE - Ada Instant Development Environment - aide v2.15
Copyright [C] Sowebio SARL 2020-2021, according to GPLv3.

Do you wish to INSTALL GNAT CE 2021 for SERVER target ?
Press any key to continue or [Ctrl-C] to abort...

20210730 194531 - INIT    - MSG - ------------------------------------------------------------
20210730 194531 - INSTALL - MSG - GNAT directory created: /root/opt/gnat-2021
20210730 194531 - INSTALL - MSG - GNAT path /root/opt/gnat-2021 set in /root/.bashrc
20210730 194531 - INSTALL - MSG - ------------------------------------------------
20210730 194531 - INSTALL - MSG - To taking account of the PATH update:
20210730 194531 - INSTALL - MSG - - Close all terminals, including this one
20210730 194531 - INSTALL - MSG - - Run a fresh terminal with the updated PATH inside
20210730 194531 - INSTALL - MSG - /!\ Not follow this advice could rising problems.
20210730 194531 - INSTALL - MSG - ------------------------------------------------
20210730 194531 - INSTALL - MSG - Run again AIDE to resume and finish the installation.
20210730 194531 - INSTALL - MSG - ------------------------------------------------
20210730 194531 - EXIT    - MSG - Total execution time: 0h00m04s
20210730 194531 - EXIT    - MSG - ------------------------------------------------------------
```

❑ Second step : installing

Open a new console and relaunch AIDE:

```
user@system: ./aide -t server

AIDE - Ada Instant Development Environment - aide v2.15
Copyright [C] Sowebio SARL 2020-2021, according to GPLv3.

Do you wish to INSTALL GNAT CE 2021 for SERVER target ?
Press any key to continue or [Ctrl-C] to abort...

20210730 195550 - INIT    - MSG - ------------------------------------------------------------
20210730 195550 - INSTALL - MSG - GNAT directory created: /root/opt/gnat-2021
20210730 195550 - INSTALL - MSG - GNAT path already exists in /root/.bashrc
20210730 195550 - INSTALL - MSG - GNAT path /root/opt/gnat-2021 already in PATH
20210730 195550 - INSTALL - MSG - GNAT path OK, no need to start a new console
20210730 195550 - INSTALL - MSG - Check system packages dependencies.
20210730 195550 - INSTALL - MSG - Package automake already installed.
20210730 195550 - INSTALL - MSG - Package make already installed.
20210730 195550 - INSTALL - MSG - Package curl already installed.
20210730 195550 - INSTALL - MSG - Package git already installed.
20210730 195550 - INSTALL - MSG - Package libtool already installed.
20210730 195550 - INSTALL - MSG - Package tar already installed.
20210730 195550 - INSTALL - MSG - Package xz-utils already installed.
20210730 195550 - INSTALL - MSG - Package libcurl4 already installed.
20210730 195551 - INSTALL - MSG - Package libcurl4-openssl-dev already installed.
20210730 195551 - INSTALL - MSG - Package libssl-dev already installed.
20210730 195551 - INSTALL - MSG - Package perl already installed.
20210730 195551 - INSTALL - MSG - Delete previous GNAT unfinished install
20210730 195551 - INSTALL - MSG - Try to install GNAT from AIDE distant package repository.
20210730 195551 - INSTALL - MSG - Download file: gnat-2021-20210519-linux64-server.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  192M  100  192M    0     0  46.0M      0  0:00:04  0:00:04 --:--:-- 46.0M
20210730 195555 - INSTALL - MSG - Decompress: gnat-2021-20210519-linux64-server.tar.xz
/root/opt/gnat-2021/gnat-2021-20210519-linux64-server.tar.xz [1/1]
  100 %      192.7 MiB / 1322.1 MiB = 0.146    59 MiB/s       0:22
20210730 195617 - INSTALL - MSG - GNAT installation from archive.
20210730 195620 - INSTALL - MSG - GNAT install done.
20210730 195620 - INSTALL - MSG - Server target: no need to deregister.
20210730 195620 - INSTALL - MSG - GNAT path already exists in /root/.bashrc
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
20210730 195620 - INSTALL - MSG - GNAT path /root/opt/gnat-2021 already in PATH
20210730 195620 - INSTALL - MSG - GNAT path OK, no need to start a new console
20210730 195620 - INSTALL - MSG - Server target: no need to register.
20210730 195620 - INSTALL - MSG - Download file: AdaWebServer
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
100 4249k  100 4249k    0     0  1304k      0  0:00:03  0:00:03 --:--:-- 19.7M
20210730 195623 - INSTALL - MSG - Ada Web Server docs & examples installation.
20210730 195623 - INSTALL - MSG - Download Utilada.
20210730 195625 - INSTALL - MSG - Utilada installation.
20210730 195811 - INSTALL - MSG - Download file: Zlib
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  593k  100  593k    0     0   383k      0  0:00:01  0:00:01 --:--:--  383k
20210730 195812 - INSTALL - MSG - Zlib installation.
20210730 195818 - INSTALL - MSG - Download HAC.
20210730 195835 - INSTALL - MSG - /root/opt/gnat-2021/share/doc removed
20210730 195835 - INSTALL - MSG - /root/opt/gnat-2021share/examples removed
20210730 195835 - INSTALL - MSG - /root/opt/gnat-2021-downloads removed
20210730 195835 - EXIT    - MSG - Total execution time: 0h02m46s
20210730 195835 - EXIT    - MSG - ------------------------------------------------------------
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

AIDE at work

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

1 Checks

Check AIDE:

```
user@system: ./aide list

AIDE - Ada Instant Development Environment - aide v2.12
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

20210414 192858 - INIT    - MSG - ------------------------------------------------------
20210414 192858 - LIST    - MSG - GNAT Compiler    Activated  Location
20210414 192858 - LIST    - MSG - AdaCore CE 2020  Yes        /root/opt/gnat-2020
20210414 192858 - EXIT    - MSG - ------------------------------------------------------
```

We have GNAT compiler AdaCore CE 2020 activated.
Right-click on a .gpr file: GNATStudio 2020 is associated in the file manager.

Check the GNAT compiler:

```
user@system: gnat ls -v

GNATLS Community 2020 (20200429-93)
Copyright (C) 1997-2020, Free Software Foundation, Inc.

Source Search Path:
   <Current_Directory>
   /root/opt/gnat-2020/lib/gcc/x86_64-pc-linux-gnu/9.3.1/rts-native/adainclude


Object Search Path:
   <Current_Directory>
   /root/opt/gnat-2020/lib/gcc/x86_64-pc-linux-gnu/9.3.1/rts-native/adalib


Project Search Path:
   <Current_Directory>
   /root/opt/gnat-2020/x86_64-pc-linux-gnu/lib/gnat
   /root/opt/gnat-2020/x86_64-pc-linux-gnu/share/gpr
   /root/opt/gnat-2020/share/gpr
   /root/opt/gnat-2020/lib/gnat
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Check the HAC interpreter:

```
user@system: hac

HAC: command-line compilation and execution for HAC [HAC Ada Compiler]
Compiler version: 0.095 dated 13-Apr-2021.
URL: https://hacadacompiler.sourceforge.io/

Usage: hac [options] main.adb [command-line parameters for main]

Options: -h     : this help
         -v, v1 : verbose
         -v2    : very verbose
         -a     : assembler output in asm_dump.pca
         -d     : dump compiler information in compiler_dump.lst

Caution: HAC is not a complete Ada compiler.
Note: HAC [this command-line tool] accepts source files with shebang's,
      for instance:   #!/usr/bin/env hac     or     #!/usr/bin/hac

| This software is free and open-source.
| It is provided "as is", WITHOUT WARRANTY OF ANY KIND.
| For the full license wording, see the header [copyright & MIT license]
| appearing on top of this software's source files.
| In doubt, check the file: hac_sys.ads
```

2 Activate a GNAT CE

To activate CE 2019:

```
user@system: ./aide -y 2019 activate

AIDE - Ada Instant Development Environment - aide v2.12
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

You wish to ACTIVATE GNAT CE 2019 for STATION target ?
Press any key to continue or [Ctrl-C] to abort...

20210414 151523 - INIT    - MSG - ------------------------------------------------------------
20210414 151523 - 19 ON   - MSG - Desktop launcher file already deleted.
20210414 151523 - 19 ON   - MSG - MIME type package file already deleted.
20210414 151523 - 19 ON   - MSG - MIME type application file already deleted.
20210414 151523 - 19 ON   - MSG - Updating MIME and Desktop databases.
20210414 151523 - 19 ON   - MSG - Desktop launcher file already deleted.
20210414 151523 - 19 ON   - MSG - MIME type package file already deleted.
20210414 151523 - 19 ON   - MSG - MIME type application file already deleted.
20210414 151523 - 19 ON   - MSG - Updating MIME and Desktop databases.
20210414 151523 - 19 ON   - MSG - Generating MIME Type file.
20210414 151523 - 19 ON   - MSG - Generating GPS launcher file.
20210414 151523 - 19 ON   - MSG - Updating MIME and Desktop databases.
20210414 151523 - 19 ON   - MSG - Check .gpr association with application/x-adagpr MIME
type.
20210414 151523 - 19 ON   - MSG - Check default reg. app. for application/x-adagpr MIME
Type.
20210414 151523 - 19 ON   - MSG - GNAT path set in /home/sr/.bashrc
20210414 151523 - 19 ON   - MSG - ------------------------------------------------
20210414 151523 - 19 ON   - MSG - Launch a new terminal or execute the line below.
20210414 151523 - 19 ON   - MSG - export PATH=/home/sr/opt/gnat-2019/bin:$PATH
20210414 151523 - 19 ON   - MSG - And run again script to resume the installation.
20210414 151523 - 19 ON   - MSG - /!\ Not follow this advice could rising problems.
20210414 151523 - 19 ON   - MSG - ------------------------------------------------
20210414 151523 - EXIT    - MSG - Total execution time: 0h00m01s
20210414 151523 - EXIT    - MSG - ------------------------------------------------------------
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Right-click on a .gpr file: GNATStudio 2019 is now associated! It is immediately taken into account because the mime database has been updated by AIDE during the activate operation.

AIDE ask user to launch a new terminal because the current system PATH in the still opened terminal is already the old one: /home/sr/opt/gnat-2020/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

But stay in this terminal anyway, you'll see why later.

Check installed compilers:

```
user@system: ./aide list

AIDE - Ada Instant Development Environment - aide v2.12
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

20210414 162207 - INIT    - MSG - ------------------------------------------------------------
20210414 162207 - LIST    - MSG - GNAT Compiler    Activated  Location
20210414 162207 - LIST    - MSG - AdaCore CE 2019  Yes        /home/sr/opt/gnat-2019
20210414 162207 - LIST    - MSG - AdaCore CE 2020             /home/sr/opt/gnat-2020
20210414 162207 - EXIT    - MSG - ------------------------------------------------------------
```

The GNAT compiler AdaCore CE 2019 is well activated.

3 Delete a GNAT CE

Remove CE 2020:

```
user@system: ./aide -y 2020 remove

AIDE - Ada Instant Development Environment - aide v2.12
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

You wish to REMOVE GNAT CE 2020 for STATION target ?
Press any key to continue or [Ctrl-C] to abort...

20210414 162757 - INIT    - MSG - ------------------------------------------------------------
20210414 162757 - REMOVE  - MSG - Deleting: /home/sr/opt/gnat-2020
20210414 162759 - REMOVE  - MSG - Deleting /home/sr/opt/gnat-2020-downloads
20210414 162800 - REMOVE  - MSG - Desktop launcher file already deleted.
20210414 162800 - REMOVE  - MSG - Deleting MIME type package file
20210414 162800 - REMOVE  - MSG - Deleting MIME type application file
20210414 162800 - REMOVE  - MSG - Updating MIME and Desktop databases.
20210414 162800 - REMOVE  - MSG - ------------------------------------------------
20210414 162800 - REMOVE  - MSG - To taking account of the PATH update:
20210414 162800 - REMOVE  - MSG - - Close all terminals, including this one
20210414 162800 - REMOVE  - MSG - - Run a fresh terminal with the updated PATH inside
20210414 162800 - REMOVE  - MSG - /!\ Not follow this advice could rising problems.
20210414 162800 - REMOVE  - MSG - ------------------------------------------------
20210414 162800 - EXIT    - MSG - Total execution time: 0h00m05s
20210414 162800 - EXIT    - MSG - ------------------------------------------------------------
```

Uninstalling removes everything that was previously installed:
- ~/opt/gnat-2020 (directory)
- ~/opt/gnat-2020-download (directory)
- ~/.local/share/applications/gps2020.desktop (launcher)
- ~/.local/share/mime/packages/application-x-adagpr.xml (MIME type)

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

– ~/.local/share/mime/application/x-adagpr.xml (created by the Gnome database update)
– ~/.bashrc updated to remove the PATH to ~/opt/gnat-2020
– Gnome database update

The association of the extension .gpr is deleted.
The personal settings are kept.

4 Install a GNAT CE

But, just after doing that, we change our mind and now we prefer to reinstall it !

Note PATH is already the old one: /home/sr/opt/gnat-2020/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

But stay in this terminal anyway, you'll see why now. AIDE will check the current PATH before ask again to launch a new terminal !

```
user@system: ./aide -y 2020 install

AIDE - Ada Instant Development Environment - aide v2.12
Copyright (C) Sowebio SARL 2020-2021, according to GPLv3.

You are not logged as root.
Your password will be asked if new packages are needed.

You wish to INSTALL GNAT CE 2020 for STATION target ?
Press any key to continue or [Ctrl-C] to abort...

20210414 164002 - INIT    - MSG - ------------------------------------------------------------
20210414 164002 - INSTALL - MSG - GNAT path set in /home/sr/.bashrc
20210414 164002 - INSTALL - MSG - Check system packages dependencies.
20210414 164002 - INSTALL - MSG - Package automake already installed.
20210414 164002 - INSTALL - MSG - Package curl already installed.
20210414 164002 - INSTALL - MSG - Package git already installed.
20210414 164002 - INSTALL - MSG - Package libtool already installed.
20210414 164002 - INSTALL - MSG - Package libcurl4 already installed.
20210414 164002 - INSTALL - MSG - Package libcurl4-openssl-dev already installed.
20210414 164002 - INSTALL - MSG - Package libssl-dev already installed.
20210414 164002 - INSTALL - MSG - Package perl already installed.
20210414 164002 - INSTALL - MSG - Download GNAT file.
20210414 164802 - INSTALL - MSG - GNAT path already exists in /home/sr/.bashrc
20210414 164802 - INSTALL - MSG - Check system packages dependencies.
20210414 164802 - INSTALL - MSG - Package automake already installed.
20210414 164802 - INSTALL - MSG - Package curl already installed.
20210414 164802 - INSTALL - MSG - Package git already installed.
20210414 164802 - INSTALL - MSG - Package libtool already installed.
20210414 164802 - INSTALL - MSG - Package libcurl4 already installed.
20210414 164802 - INSTALL - MSG - Package libcurl4-openssl-dev already installed.
20210414 164802 - INSTALL - MSG - Package libssl-dev already installed.
20210414 164802 - INSTALL - MSG - Package perl already installed.
20210414 164802 - INSTALL - MSG - Download GNAT file.
20210414 165151 - INSTALL - MSG - GNAT installation.
20210414 165151 - INSTALL - MSG - Generate GNAT installation script.
20210414 165151 - INSTALL - MSG - GNAT installation script is ready.
20210414 165151 - INSTALL - MSG - Change to GNAT installation directory.
20210414 165312 - INSTALL - MSG - Delete GNAT installation script.
20210414 165312 - INSTALL - MSG - GNAT install done.
20210414 165312 - INSTALL - MSG - Building GNAT debug runtime.
20210414 165423 - INSTALL - MSG - GNAT debug run-time build sucessfully.
20210414 165423 - INSTALL - MSG - GNAT debug ready RTS done.
20210414 165423 - INSTALL - MSG - Desktop launcher file already deleted.
20210414 165423 - INSTALL - MSG - MIME type package file already deleted.
20210414 165423 - INSTALL - MSG - MIME type application file already deleted.
20210414 165423 - INSTALL - MSG - Updating MIME and Desktop databases.
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
20210414 165424 - INSTALL - MSG - Desktop launcher file already deleted.
20210414 165424 - INSTALL - MSG - MIME type package file already deleted.
20210414 165424 - INSTALL - MSG - MIME type application file already deleted.
20210414 165424 - INSTALL - MSG - Updating MIME and Desktop databases.
20210414 165424 - INSTALL - MSG - GNAT path set in /home/sr/.bashrc
20210414 165424 - INSTALL - MSG - Generating MIME Type file.
20210414 165424 - INSTALL - MSG - Generating GPS launcher file.
20210414 165424 - INSTALL - MSG - Updating MIME and Desktop databases.
20210414 165424 - INSTALL - MSG - Check .gpr assoc. with application/x-adagpr MIME type.
20210414 165424 - INSTALL - MSG - Check reg. app. for application/x-adagpr MIME Type.
20210414 165424 - INSTALL - MSG - Download AdaWebServer file.
20210414 165424 - INSTALL - MSG - Ada Web Server docs & examples installation.
20210414 165427 - INSTALL - MSG - Download Utilada.
20210414 165427 - INSTALL - MSG - Utilada installation.
20210414 165540 - INSTALL - MSG - Download Zlib file.
20210414 165540 - INSTALL - MSG - Zlib installation.
20210414 165550 - INSTALL - MSG - Download HAC.
20210414 165550 - INSTALL - MSG - HAC installation.
20210414 165601 - EXIT    - MSG - Total execution time: 0h08m00s
20210414 165601 - EXIT    - MSG - ------------------------------------------------------------
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

GNATStudio at work

The last bug isn't fixed until the last user is dead.
Sidney Markowitz

1 Introduction

As stated by https://www.adacore.com/gnatpro/toolsuite/gnatstudio:

The GNAT Studio is a powerful and intuitive IDE that supports the full development workflow, from coding to system integration, testing, debugging, and code analysis. GNAT Studio is versatile and customizable and gives you easy access to the GNAT Pro technologies.

Getting started video: https://www.youtube.com/watch?v=oMQn_M-9Kmw
Understanding Code with GPS: https://www.youtube.com/watch?v=plOUcXJMHAI
Debugging with GPS: https://www.youtube.com/watch?v=pohu-dHPLyk

2 Setting up GNATStudio

Launch GNATStudio

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...

2.1 General

❑ Main

```
Behavior
[x] Auto save (default)
[x] Save desktop on exit (default)

Default Builder
(o) Gprbuild (default)
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
Charsets
Character set: Unicode UTF-8[12] (instead of Western/Latin-1 (ISO-8859-1)

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

❑ 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 > 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]
```

2.2 Editor

❑ Ada

```
(o) Simple indentation (instead of extended)
[  ] Indent comments (instead of [x]
It should be wised to not change other options.
```

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 GNATStudio can't find it:
HTML browser: /usr/bin/firefox %u

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
```

12 GNATStudio uses Unicode internally

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

2.5 Build targets

A setting page of interest.

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
```

3 Shortcuts

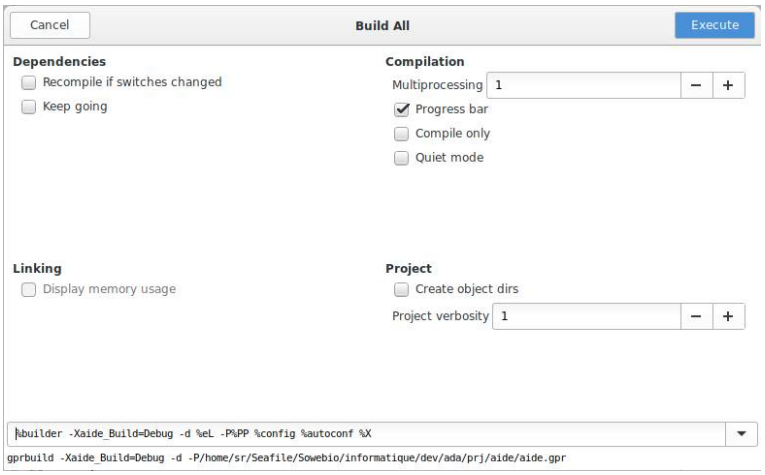
3.1 Comment box for subprograms

[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
```

3.2 Build all

[F9] triggers the build all window:



CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

3.3 Comment or comment a block

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

3.4 Debug - Step

[F5]

3.5 Debug - Step out

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

3.6 Debug - Finish

[F7] Continue execution until selected stack frame returns

3.7 Debug - Run

[F8] Continue execution until next breakpoint

3.8 Delete a line

[Ctrl] + [Y] Remember Wordstar[13]

4 Build an example

The HAC runtime is located in the ./v20/src directory.

Use ./v20/v20.gpr as a stub for your own projects.

Use ./v20/src-tests/test.adb as an template to integrate the appropriate v20 with and use clauses.

5 AIDE

5.1 Open the project

Click on aide.gpr to open the project in GNATStudio.

5.2 Build AIDE

The standard build string is:

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

Without hacking aide.gpr, from the build all window, you may choose between some build flavors:

[13] https://en.wikipedia.org/wiki/WordStar

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

- **Debug**: standard build during development, make binary big and debugger friendly.
- **Fast**: production build for specific developments.
- **Small**: production build, which can be followed by an UPX pass.
- **Style**: production build with very strict style checker used to build the GNAT Run-Time system.

Just insert *-Xaide_Build=**Debug** | **Fast** | **Small** | **Style*** in the standard build string.

After pressing [F9] or click on  and choose to make a production build:

```
%builder -Xaide_Build=Small -d %eL -P%PP %config %autoconf %X
```

Then press or click on [Execute]:

```
gprbuild  -Xaide_Build=Small  -d  -P/home/sr/Seafile/Sowebio/informatique/dev/ada/prj/aide/
aide.gpr
Compile
   [Ada]          aide.adb
   [C]            link_max.c
   [C]            set_std_prefix.c
   [C]            executable_path.c
   [C]            objlist_file.c
   [C]            update_path.c
   [C]            run_path_option.c
   [C]            gnatcoll_support.c
   [C]            separate_run_path_option.c
   [C]            getRSS.c
   [C]            terminals.c
   [Ada]          v20.adb
   [Ada]          v20-fls.adb
   [Ada]          v20-log.adb
   [Ada]          v20-prg.adb
   [Ada]          v20-sys.adb
   [Ada]          v20-tio.adb
   [Ada]          v20-vst.adb
   [Ada]          gnatcoll.ads
   [Ada]          gnatcoll-memory.adb
   [Ada]          s-memory.adb
Bind
   [gprbind]      aide.bexch
   [Ada]          aide.ali
Link
   [archive]      libaide.a
   [index]        libaide.a
   [link]         aide.adb
[2021-04-13 17:50:31] process terminated successfully, elapsed time: 07.16s
```

6 v20 library

This library has its own manual. You can refer to it as a source of information and introduction to Ada.

v20 does not use any advanced Ada concepts and is therefore a good introduction to the language.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Learn Ada

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

1 Introduction

Ada is not just programming.

Ada is software engineering.

<<<TODO>>>

2 Requirements

AIDE :]

3 Manuals in AIDE

Where they are located.

Book summary.

<<<TODO>>>

4 Essential reading

4.1 English books

Adacore books

AIDE v1.4 books repository

<<<TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

4.2 English courses

<<<TODO>>>

4.3 French books

Rosen Book

<<<TODO>>>

4.4 French courses

Feneuille courses

<<<TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Coding examples

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

1 HAC is good for you

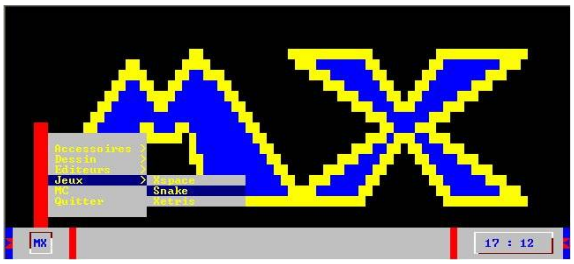
If you're not an experienced programmer, we invite you to use the HAC interpreter included in AIDE. HAC comes with a lot of basic to less simple code, for all tastes.

2 GNATStudio Examples

<<<TODO>>>

3 Programs from the MX Team

3.1 Mx



- 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.

- Build

The sources of Mx are available in:

<<<TODO>>>

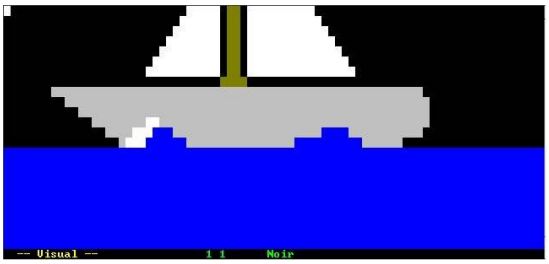
CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

❑ Usage

• General commands

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

3.2 Visual



❑ 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.

❑ Build

The sources of Visual are available in:

<<<TODO>>>

❑ 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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

- [F10] White
- [Ctrl] + [F2] Light blue
- [Ctrl] + [F3] Light green
- [Ctrl] + [F4] Light cyan
- [Ctrl] + [F5] Light red
- [Ctrl] + [F6] Light magenta

3.3 Updates from original 2004 release

❑ 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>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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!

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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.

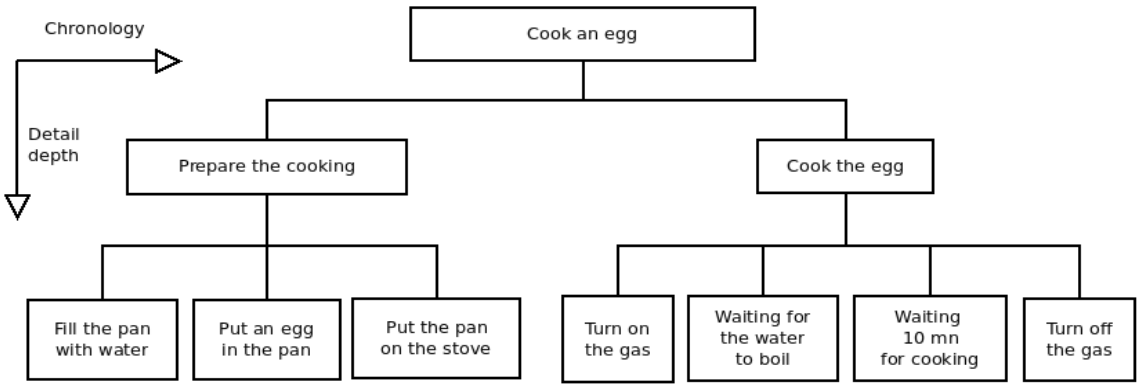
❑ Problem's decomposition

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 *SPD*, the *Structured Program Diagram*, sometimes called the *JSP diagram*, after its inventor[15]:



This SPD works in two dimensions:
- In the vertical plane, we go down from the most complex to the simplest;
- In the horizontal plane, the direction of the reading represents naturally, chronologically, the tasks to be performed.

The SPD 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 DSP allows you to go directly to the second phase: *the pseudo-code*!

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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

In creating this SPD, 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.

❑ Pseudo-code

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

The SPD 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 SPD 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 SPD 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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

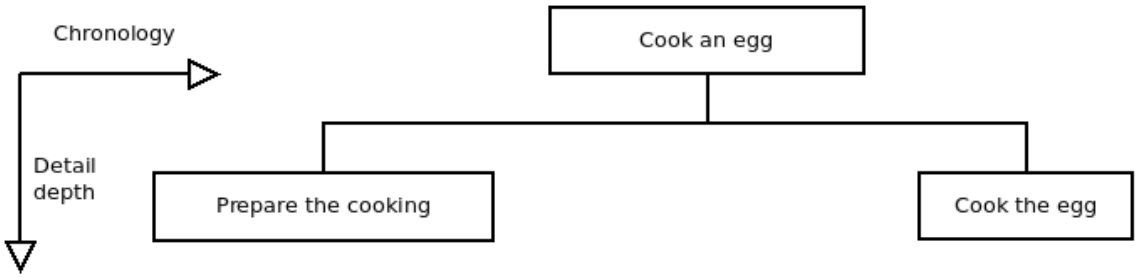
in a module. Leaving the analysis as it is, without changing anything, would result in making the program more complex than it deserves to be.

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 SPD 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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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 SPD 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's Structured Diagram

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

❑ Process detailed

The process of creating the SPD 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.

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 SPD, all boxes of a lower rank are connected by lines to the box of the higher rank.

SPDs are always written and read:

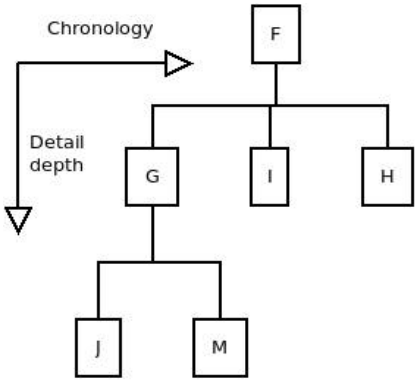
[16] PGMS in french, as "Programmation Générale, Modulaire et Structurée"

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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

- Top to bottom, for level of detail;
- From left to right, for chronological steps.

Example:



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

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

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

3.3 Pseudo-code

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

➪ We repeat: one SPD box to one module in the pseudo-code.

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

❑ Main module

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

Here is the pseudo-code, also called PC, of the previous SPD, 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 ***.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

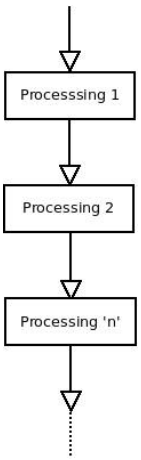
❑ Other modules

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

```
begin *** G ***

 do *** J ***

 if Q [if Q is true]
  do *** M ***
 end if

end *** G ***
```

❑ Sequence

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

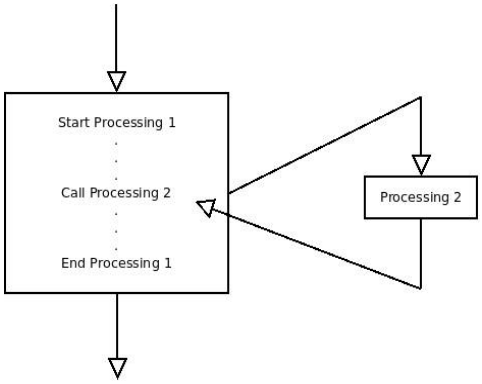


❑ Module call

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.

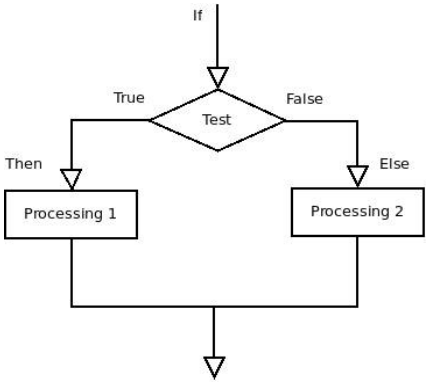
At the end of the called module, the latter returns to the calling module and the execution of the latter resumes at the line following the call which has just been executed:

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike



❑ 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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

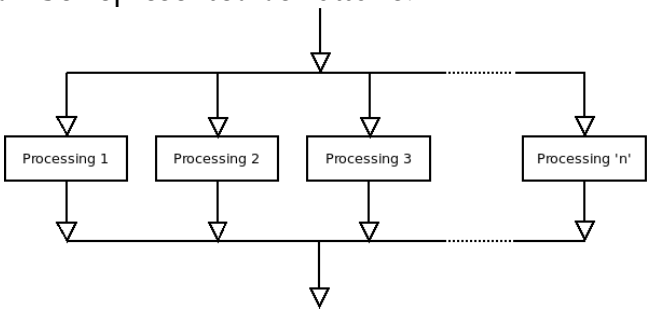
```
if test condition 1
 Processing 1
else
 if test condition 2
  Processing 2
 else
  if test condition 3
   Processing 3
  else
   Default processing
  end if
 end if
end if
```

- 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
```

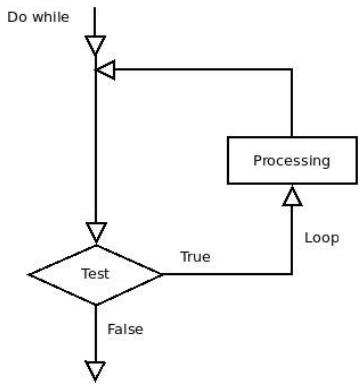
This structure can be represented as follows:



- 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*:

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

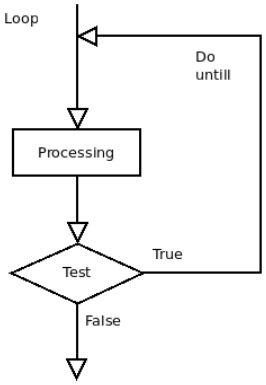


The pseudo code of such a structure is as follows:

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

❑ 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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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.

❑ 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
```

❑ 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]
```

❑ 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.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

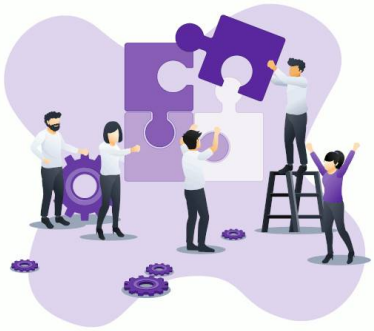
5.1 Initial reading & current reading in loops

<<<TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

FAQ

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



1 Issues & solutions

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

In message console :

```
Launching xdg-open to view file:///home/sr/opt/gnat-2020/share/doc/gnatstudio/html/tuto-
rial/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/index.html
Launching /usr/bin/firefox to view file:///home/sr/opt/gnat-2020/share/doc/gnatstudio/html/
users_guide/index.html
```

Sets the real path of your browser of choice:
Edit > Preferences > External Commands > Browser > HTML Browser :
/usr/bin/firefox %u

1.2 Error in .gpr project files association with GNATStudio

1.3 Association lost between .gpr project files and GNATStudio

- Case n°1

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

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

- Case n°2

Check directory ~/.local/share/applications.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

You must have only one gpsYYYY.desktop launcher.

If you have an old and original GNAT GPS launcher as gps.desktop or gnatstudio.desktop, you have to delete it.

However, this should not happen since AIDE checks and deletes the gps.desktop and gnatstudio.desktop files before creating or removing an association with the .gpr files

1.4 GNATStudio CE 2019 can't load a .gpr project file at start

After starting GNATStudio 2019, open the project with File > Open Project...

2 AIDE

2.1 Why AIDE use AdaCore GNAT Community Edition ?

On Debian, Ubuntu and derivatives, GNAT Ada compiler is part of the standard packages. But we recommend GNAT Community Edition from AdaCore.

GNAT CE is the most advanced Ada environment available. It is - apart from a few ancillary code quality control tools - exactly the same set used by high-speed train developers, rocket or satellite software designers, etc.

A nice detail too: unlike the FSF version, the post-mortem trace with the call stack and the source lines is directly available in debug mode.

AdaCore makes this environment available to free software developers. We thank them for that.

2.2 Why AIDE program seems bigger than a common Ada program ?

AIDE is a totally autonomous program, statically compiled and therefore without dependencies, with checking at runtime and embedding all the symbols necessary for debugging .AIDE does not use UPX to minimize its size.

Thanks to this, AIDE runs on every 64 bits Linux system and, before ending the program, a runtime error gracefully dumps a file trace including the call stack with source code lines.

2.3 Why AIDE delete deactivated GNATStudio desktop launcher ?

It would be nice to keep all GNATStrudio desktop launchers for all years. Not only the activated one!

All GNATStudio desktop launchers can't stay to ease users because they reference the mime type with .gpr extension. Then you would end up with several combinations of GNATStudio applications from different years pointing on only one .gpr extension, which should be avoided at all costs.

We need to associate the .gpr extension with one GNATStudio year edition at a time. Which is logically the one that is activated.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

2.4 Which components are in the GNAT Community Edition installer?

There are four components:
- com.adacore.gnat - The compiler;
- com.adacore.libadalang - The library for parsing and semantic analysis of Ada code;
- com.adacore.spark2014_discovery - The Ada subset for formal analysys;
- com.adacore.gnatstudio - The intuitive IDE that supports the full development workflow.

The AIDE's server target installs only the core component com.adacore.gnat.
The AIDE's station target installs the whole components.

2.5 Where are stored GNATStudio configuration files ?

Personal setting are located in:
~/.gps [2019]
~/.gnatstudio [2020]
~/.gnatstudio [2021]

2.6 How file association is processed in AIDE - i.e. conditions for file type handling

AIDE generates a GNATStudio launcher which specifies the MIME[18] type for the GNATStudio application:

~/.local/share/applications/gps2020.desktop

```
[Desktop Entry]
Name=Gnat Programming System 2020
Icon=/home/sr/opt/gnat-2020/share/gnatstudio/icons/hicolor/32x32/apps/gnatstudio_logo.png
Exec=/home/sr/opt/gnat-2020/bin/gnatstudio
Terminal=false
Type=Application
MimeType=application/x-adagpr
Categories=Development;
StartupWMClass=gnatstudio_exe
```

AIDE then generates the association file between extension .gpr and MIME type:

~/.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, AIDE updates the MIME and Desktop databases and, finally, check association[19]:

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

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

19 Not yet implemented.

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
user@system: update-desktop-database ~/.local/share/applications

Association test
user@system: gio mime application/x-adagpr

Application par défaut pour « application/x-adagpr » : gps2020.desktop
Applications inscrites :
   gps2020.desktop
Applications recommandées :
   gps2020.desktop
```

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>
```

3 Ada

3.1 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
```

3.2 Library integration with .gpr

❑ Check paths

```
user@system: gnat ls -v

GNATLS Community 2019 (20190517-83)
Copyright (C) 1997-2019, Free Software Foundation, Inc.

Source Search Path:
   <Current_Directory>
   /home/sr/opt/gnat-2019/lib/gcc/x86_64-pc-linux-gnu/8.3.1/rts-native/adainclude
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
Object Search Path:
   <Current_Directory>
   /home/sr/opt/gnat-2019/lib/gcc/x86_64-pc-linux-gnu/8.3.1/rts-native/adalib


Project Search Path:
   <Current_Directory>
   /home/sr/opt/gnat-2019/lib
   /home/sr/opt/gnat-2019/x86_64-pc-linux-gnu/lib/gnat
   /home/sr/opt/gnat-2019/x86_64-pc-linux-gnu/share/gpr
   /home/sr/opt/gnat-2019/share/gpr
   /home/sr/opt/gnat-2019/lib/gnat
```

❑ Environment variable

La variable d'environnement ADA_PROJECT_PATH permet d'étendre les chemins par défaut :

```
user@system: printenv ADA_PROJECT_PATH

/home/sr/opt/gnat-2019/lib
```

Elle peut être ajustée avec persistence dans ~/.bashrc :

~/.bashr

…/…

```
# Additional path for GNAT compiler
export PATH=/home/sr/opt/gnat-2019/bin:$PATH
export ADA_PROJECT_PATH=/home/sr/opt/gnat-2019/lib
```

❑ Libraries location

- Sources location

/home/<user>/opt/gnat-2019/include/<nom lib>

- .gpr location

/home/<user>/opt/gnat-2019/share/gpr

❑ GNAT CE libraries Gprbuild project integration

Une bibliothèque installée dans GNAT CE a son projet GprBuild dans /share/gpr (cf ci-dessus).

Par exemple, pour intégrer la bibliothèque GNATColl, qui est présente dans la distribution GNAT CE, il suffit de débuter le fichier projet avec la clause :

./exemple.gpr

```
with "gnatcoll";
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
project exemple is

…/…
```

3.3 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).

3.4 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 to proceed. It is 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 ;

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

- 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).

3.5 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:

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
-------------------------------------------------------------------------------
--
--  v20
--
--  @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 GNATStudio 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/gnat-
coll"];

   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
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
    "-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-Character-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<let-
ter> 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"  =>
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
         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-sec-
tions"];
      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;
------------------------------------------------------------------------------
```

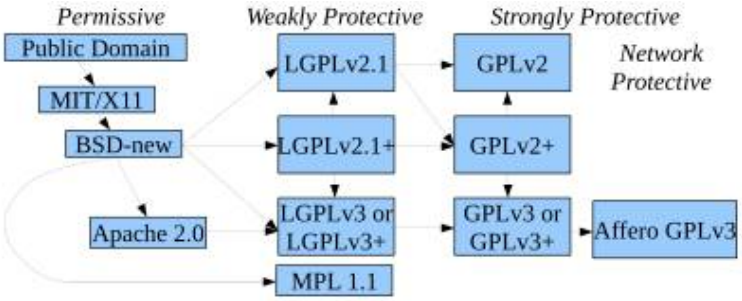

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Appendices

1 Copyrights & credits

1.1 Program license

AIDE is copyright Sowebio under GPL v3 license.



- 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

This manual is intended for AIDE, Ada Instant Development Environment. Copyright ©2001, 2002, 2003, 2004, 2005, 2020, 2021 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 Quality control

Check list

<<< TODO>>>

3 Release check list

Things to do to release to github

<<< TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

4 To-do list

4.1 AIDE --install=DIRECTORY

Activate, Deactive and List commands actually only works with default path /home/user/opt/gnat-YEAR. In order to work with custom paths, we need a persistence with a config file in /home/user/.config/.aide.cfg - this way we will be able to find the paths of existing installations.

4.2 AIDE list

Should also list libraries and tools installed. Needs persistence, so TBD with AIDE --install=DIRECTORY.

4.3 ALIRE Package manager

Integrate

4.4 Additional libraries install

Additional libraries: need option to force install in server target or not install in station target.

4.5 SERVER target

With this target, during install process, we should:
– Not install additional libraries by default (saving space but can be installed anyway with option);
– Not install rts-native-debug RTS (no debug session on server target);
– Delete all documentation files (no PDF or HTML reader on server target).

4.6 sed better usage

Use sed -i to make replaces inline (instead of create a .tmp file).

4.7 Doc

- The never-ending task

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

- Some OS patterns

```
cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"

cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.5 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
```


CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

```
PRETTY_NAME="Ubuntu 18.04.5 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/priv***
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

5 Issues

5.1 Launch from a SMB drive doesn't work

Launch from a SMB drive (through Samba) raise an exception. Investigate why and fix this bug.

5.2 Log file has only extension

AIDE program creates a .log file name instead of aide.log

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Bibliography

1 Books

<<<TODO>>>

2 Books - Data structures

2.1 File structures with Ada

Nancy E. Miller & Charles G. Petersen
Alan Apt
ISBN 0-8053-0440-1

<<<TODO>>> ref to aide-repository

2.2 Structures de données avec Ada

P. Lignelet
Masson Apt
ISBN 2-903-60780-X

2.3 Ada95 : Orientation objet, structures de données et algorithmes

Philippe Gabrini
De Boeck Université
ISBN 2-8041-3790-2
https://web.archive.org/web/20021203092652/http://grosmax.si.uqam.ca/Professeurs/Gabrini/Exemples

2.4 Algorithmes et structures de données avec Ada, C++ et Java

Abdelali Guerid, Pierre Breguet et Henri Röthlisberger
Presses Polytechniques et universitaires romandes
ISBN 2-88074-488-1
http://www.eivd.ch/Francais/Informations/École/Informatique

3 Papers – Data structures

<<<TODO>>> ref to aide-repository

3.1 [DOD69] Elements of Data Management Systems

George G. Dodd
Computer Technology Department
General Motors Research Laboratories
ACM, Computing Surveys, Vol 1, No 2, June 1969
1969-06-01_dodd_elements of data management systems.pdf

3.2 [BAC71] Organization and Maintenance of Large Orderer Indexes

R. Bayer and E. McCreight
Dept. of Computer Science

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Purdue University and Palo Alto Research Center
Acta Informatica, Vol 1, September 1971
1971-09-29_bayer_mccreight_organisation of large ordered indexes.pdf

3.3 [BAY72] Symetric Binary B-Trees: Data Structure and Maintenance

R. Bayer
Mathematisches Institut
Technischen Universität, München
Acta Informatica, Vol 1, January 1972
1972-01-24_bayer_symetric binary btree.pdf

3.4 [BAM76] On the Encipherment of Search Trees and Random Access Files

R. Bayer and J. K. Metzger
Institut für Informatik
Technischen Universität, München
ACM, Transactions on Database Systems, Vol 1, No 1, March 1976
1976-03-01_bayer_metzger_encipherment of search trees and random access files.pdf

3.5 [BAS76] Concurrency of Operations on B-Trees

R. Bayer and M. Schkolnick
IBM Research Laboratory
San José
Acta Informatica, Vol 9, June 1976
1976-06-10_bayer_schkolnick_concurrency of operations on btrees.pdf

3.6 [BAU77] Prefix B-Trees

R. Bayer and K. Unterauer
Technischen Universität,
München
ACM, Transactions on Database Systems, Vol 2, No 1, March 1977
1976-11-01_bayer_unterauer_prefix btrees.pdf

3.7 [COM78] The ubiquitous B-Tree

D. Comer
Computer Science Department
Purdue University
ACM, Computing Surveys, Vol 11, No 2, June 1979
1978-12-01_comer_the ubiquitous btree.pdf

3.8 [KEW91] Modularization of DADAISM Ada Database System Architecture

A. M. Keller and G. Wiederhold
Computer Science Department
Standford University
Sigmod Record, Vol 24, No 1, May 1991
1991-05-21_keller_wiederhold_modularization of dadaism.pdf

3.9 [JAN95] Implementing Deletion in B+-Trees

J. Jannink

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Computer Science Department
Standford University
Sigmod Record, Vol 24, No 1, January 1995

3.10 1995-01-01_jannink_implementing deletion in b+tree.pdf

3.11 [MAO95] Optimizing Jan Jannink's Implementation of B+Tree deletion

R. Maelbrancke and H. Olivié
Department of Computer Science
Katholieke Universiteit Leuven
Sigmod Record, Vol 24, No 3, June 1995
1995-06-01_maelbrancke_olivie_Optimizing Jan Jannink's Implementation of b+tree deletion.pdf

3.12 [RBE96] Embedded RT and Database: how do they fit together ?

M. B. Roark, M. Bohler and B. L. Eldridge
Lockheed Martin and Wright Laboratory USAF
STC 96
1996-04-25_roark_bohler_eldridge_embedded real-time and database.pdf

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Glossary

<<<TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Index

<<<TODO>>>

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

Ada, “it’s stronger than you”.
Tribute to Daniel Feneuille, legendary french Ada teacher

In Strong Typing We Trust !

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike

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

CC-by-nc-sa: Attribution + Noncommercial + ShareAlike