sow - v20 Ada Library User 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

Ed.	*Release*	*Comments*	
1	20210324	Initial release	sr
8	20210402	First review	sr
9	20210404	New Shell_Execute procedure	sr
15	20210412	Refactoring and extend API	sr
23	20210419	Change Humanist 521 BT font to Airbus cockpit free font designed by Intactile[1]	sr
27	20210606	Updates about AIDE 2.14, many enhancements and typos fixed	sr
31			

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

❑ Author

Stéphane Rivière (Number Six) - stef@genesix.org (CTO Sowebio)

❑ Manual

Stéphane Rivière (Number Six) - stef@genesix.org (CTO Sowebio)

The "Excuse me I'm French" speech - The main author of this manual is a Frenchman with basic English skills. Frenchmen are essentially famous as frog eaters[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 31 - 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* :)

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

Contents

Introduction

1 About v20

v20 is a Ada Library dedicated to Linux service daemons and console programs, primary designed to be used in Genesix, a cluster manager for High Availability virtual instances on GNU/Linux Debian/Xen servers.

However, v20 is a general purpose library, KISS[3] oriented and very efficient to create any command line program.

v20 is a modular library with components designed to work together. Naming and conventions are consistent. Currently, v20 is composed of nine packages in charge of unbounded strings, program and OS functions, console and text files, logging and configuration files handling. At least six other packages are planned, related to databases and web APIs, without being limited to these aspects only.

2 About the Ada Community

At first, all our warmly 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] Keep It Simple, Stupid - https://en.wikipedia.org/wiki/KISS_principle - In memory of http://www.nasonline.org/publications/biographical-memoirs/memoir-pdfs/johnson-clarence.pdf the genius father of titanium Blackbirds.

3 v20 history

We own the copyrights for v89, v90, v93, v95, v04 and v20. Some work in v20 is derived from theses.

Ver.	*Langages*	*Proc.*	*Système*	*Context*	*Copyright*	*Users*
v87	Clipper	i386	MsDos	ST Formation	Proprietary	CEA-DAM CEA EDF
v89	Clipper/C/Asm	i386	MsDos	Atlansys	Proprietary	ETDE SAMU EDF
v90	Clipper/C/Asm	i386	MsDos	Atlansys	Proprietary	Military NGO EDF
v93	C++	i386	Windows	Atlansys	Proprietary	Research
v95	Delphi	i386	Windows	Astriane	Proprietary	Military NGO
v96	Asm	st62xx	Embedded	MRT	Proprietary	Military Civilian
v97	Asm	pic17c44	Embedded	MRT	Proprietary	Military Civilian
v04	Ada	i386	Windows	AIDE v1	GMGPL	Education
v20	Ada	All	Linux	AIDE v2	GPL v3	General Purpose

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 v20 Distribution

1.1 Directories

v20 comes with some inner directories:

Packages	***Description***
bin	test binary place, with dontdelete.me test file for trailing comments preservation
doc	place of sow - v20 Ada Library User Manual.pdf and others documentation files
doc-generated	API doc generated by GNATStrudio with GNATDoc
obj/debug obj/fast obj/small	build directories
src	sources of v20
src/sys	specials system files as s-memory.adb, the GNATColl memory monitory hook
src-tests	sources of v20 tests programs

1.2 Key files

Key files are located in the main directory.

v20.gpr project file for building v20 with GNAT

2 Get an Ada compiler

Just use AIDE: https://github.com/sowebio/aide-bin

3 Get v20

You can get v20 at https://github.com/sowebio/v20

4 v20 build

❑ Compilation

Assuming you wish to install v20 under <your path> with a GNAT compiler allready installed, do the following from a command line interpreter. Open a terminal:

```
user@system: cd <your path>
user@system: git clone https://github.com/sowebio/v20
user@system: cd v20
user@system: gprbuild -P v20
user@system: cd bin
user@system: ./test
```

v20 at work

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

1 Basic template

<<<TODO>>>

v20 API

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

1 Introduction

1.1 Concepts

The developer is a writer. The writer's courtesy is clarity;

Clarity and ease of use are prioritized over speed and efficiency.

The performance of a compiled language such as Ada as well as the hardware capabilities of current systems justify these choices.

On a simple loop, let's recall that if HAC [Ada subset interpreter] is [among others] 7 times faster than Bash, HAC itself is 300 times slower than Ada.

1.2 Conventions

To ease developers:

- All strings constants and function only returns VString typed.

- All strings parameters accept both String and VString types.

1.3 Usage

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.

2 v20

Base package.

2.1 Get_Version

- Description

Returns the Library name and formatted version: “<space>v.minor.major”.

- Usage

function Get_Version return VString

- Example

```
Log.Msg ("Library version: " & v20.Get_Version);
```

2.2 Raise_Exception

- Description

Raise an exception for reporting test and <program_Name.err> file creation.

In addition to the usual trace, a v20 exception give some extra information like: exception time, program uptime, program & library names & versions, start & home directories and Ada and all languages memory allocation, current & maximum (peak) values.

- Usage

procedure Raise_Exception

- Example

```
Raise_Exception;

--------------------------------------------------------------------------------

Exception time       : 20210402 160834
Program uptime       : 0h00m00s
Program name & version: test v0.2
Library name & version: v20 v0.1
Start directory      : /home/sr/Seafile/Sowebio/informatique/dev/ada/prj/v20/bin
Home directory       : /home/sr
Ada memory allocations: Ada Cur: [ 2272 ] Max: [ 201912 ]
All memory allocations: All Cur: [ 3465216 ] Max: [ 3465216 ]

raised V20.RAISE_EXCEPTION.V20_EXCEPTION_TEST : v20.adb:47
[./test]
V20.Raise_Exception at v20.adb:47
Test at test.adb:311
Main at b__test.adb:375
0x475937 __libc_start_main at ???
0x4053c8 _start at ???

--------------------------------------------------------------------------------
```

3 Cfg - Configuration files

3.1 Close

- Description

Close Cfg file. For sanity only as each setting is instantly flushed to disk.

- Usage

procedure Close

- Example

<<<TODO>>>

3.2 Comment

- Description

Insert a comment Text after the last line of the config file.

- Usage

procedure Comment (Text : String)

- Example

<<<TODO>>>

3.3 Delete

- Description

Delete parameter in section. If no other parameter in this section, delete section too. Avoid reserved chars *[ ] = #* inside parameters.

- Usage

procedure Delete (Section : String; Parameter : String)

- Example

<<<TODO>>>

3.4 Get

- Description

Return parameter in section or empty string if not found. Avoid reserved chars *[ ] = #* inside parameters.

- Usage

function Get (Section : String; Parameter : String) return VString

- Example

<<<TODO>>>

3.5 Open

- Description

Open and load if exist a configuration file. Create blank if non existent. Default configuration file name is “program name” followed by “.cnf” extension and created in the program start directory.

- Usage

function Open (Cfg_File_Read_In : String := "") return Boolean

- Example

<<<TODO>>>

3.6 Set

- Description

Create or replace an existing parameter in a section. If this latter does not exist, also creating it. New setting is persistent even program quits unexpectedly after. Avoid reserved chars *[ ] = #* inside parameters. If reserved chars are passed, the procedure does nothing. An optional trailing comment can also be added.

- Usage

procedure Set (Section : String; Parameter : String; Value : String; Comment : String := "")

- Example

<<<TODO>>>

4 Fls - Files

4.1 Copy_File

- Description

Copy a Source_Name file to a Target_Name file destination.
Copy_Form is “preserve=all_attributes,mode=overwrite” (full attributes preservation and overwrite file if exists).

- Usage

```
procedure Copy_File (Source_Name, Target_Name : String)
procedure Copy_File (Source_Name, Target_Name : VString)
procedure Copy_File (Source_Name : VString; Target_Name : String)
procedure Copy_File (Source_Name : String;  Target_Name : VString)
```

- Example

<<<TODO>>>

4.2 Create_Directory_Tree

- Description

Create a directory tree Dir_Tree. Each non-existent directory named by Dir_Tree is created (possibly including other intermediate directories). Return False if operation is unsuccessfull (i.e. if base directory tree is unconsistent or already exist or still don't exist after the creating attempt). Return True of directory tree already exists.

Extra inner slashes are processed i.e. a directory like /home/sr/opt/ytr.lkj///// kjghgh will be valid. and will create, from /home/sr/opt :
- Directory ytr.lkj
- And then inner directory kjghgh

- Usage

function Create_Directory_Tree (Dir_Tree : String) return Boolean
function Create_Directory_Tree (Dir_Tree : VString) return Boolean

- Example

<<<TODO>>>

4.3 Delete_Directory_Tree

- Description

Delete a directory tree Dir_Tree. The directory and all of its contents (possibly including other directories) are deleted. Return True if Dir_Tree is successfully deleted or was already deleted. Return False if operation is unsuccessful (i.e. if base directory tree was non existent or still exists after the deleting attempt).

/! This function uses Ada.Directories.Delete_Tree, which raises an exception if the directory tree to delete contains a *broken* symbolic link (a file like any other). This latter is seen as *non-existent* and, when the parent directory is deleted, an exception occurs : raised ADA.IO_EXCEPTIONS.USE_ERROR : directory tree rooted at <directory tree> could not be deleted (because *not empty*). Funny, but not so much. Pure C code problem in Ada RTS. Stacked C calls in russian puppet mode until a logical problem arises.

- Usage

function Delete_Directory_Tree (Dir_Tree : String) return Boolean
function Delete_Directory_Tree (Dir_Tree : VString) return Boolean

- Example

<<<TODO>>>

4.4 Delete_File

- Description

Delete a Name file only if this latter exists. No exception will be raised if the file to delete does not exists.

- Usage

```
procedure Delete_File (Name : String)
procedure Delete_File (Name : VString)
```

- Example

<<<TODO>>>

4.5 Delete_Lines

- Description

Search and remove file lines matching Pattern in File_Name.

- Usage

```
procedure Delete_Lines (File_Name, Pattern : String)
procedure Delete_Lines (File_Name, Pattern : VString)
procedure Delete_Lines (File_Name : String; Pattern : VString)
procedure Delete_Lines (File_Name : VString; Pattern : String)
```

- Example

<<<TODO>>>

4.6 Download_File

- Description

Download a file from Url to Dlfile. Do nothing if Dlfile already exists with its size equals Dlsize. Name is purely informational and used to named file in text messages.

Return True is Dlfile present at the right size, False otherwise.

- Usage

```
function Download_File (Url : VString;
                        Dlfile : VString;
                        Name : VString;
                        Dlsize : Integer := 0) return Boolean;
```

- Example

<<<TODO>>>

4.7 Exists

- Description

Returns True if file or directory Name exists.

- Usage

function Exists (Name : String) return Boolean
function Exists (Name : VString) return Boolean

- Example

```
if Exists (HAC_Dir & "/hac") then
   Put_Line ("HAC installation is done :)");
end if;
```

4.8 File_Size

- Description

Return size of Name file.

- Usage

function File_Size (Name : String) return Integer
function File_Size (Name : VString) return Integer

- Example

<<<TODO>>>

4.9 Get_Directory

- Description

Returns current directory.

- Usage

function Current_Directory return String
function Current_Directory return VString

- Example

<<<TODO>>>

4.10 Rename

- Description

Rename an Old_Name file or directory to a New_Name file or directory. If exists a file New_File, it will be overwritten.

- Usage

```
procedure Rename (Old_Name, New_Name : String)
procedure Rename (Old_Name, New_Name : VString)
procedure Rename (Old_Name : VString; New_Name : String)
procedure Rename (Old_Name : String;  New_Name : VString)
```

- Example

<<<TODO>>>

4.11 Search_Lines

- Description

Search at least a line matching Pattern in File_Name and return true if found.

- Usage

```
function Search_Lines (File_Name, Pattern : String) return Boolean
function  Search_Lines (File_Name, Pattern : VString) return Boolean
function  Search_Lines (File_Name : String; Pattern : VString) return Boolean
function  Search_Lines (File_Name : VString; Pattern : String) return Boolean
```

- Example

<<<TODO>>>

4.12 Set_Directory

- Description

Change to a directory Directory. Create Directory if this latter does not exist, return False if operation failed.

- Usage

```
function Set_Directory (Directory : String) return Boolean
function Set_Directory (Directory : VString) return Boolean
```

- Example

<<<TODO>>>

5 Log - Logging

5.1 Dbg

- Description

Log a debug message. 45 characters max before truncation with a maximum line length of 79.

- Usage

procedure Dbg (Message : in String)
procedure Dbg (Message : in VString)

- Example

<<<TODO>>>

5.2 Err

- Description

Log an error message. 45 characters max before truncation with a maximum line length of 79.

- Usage

procedure Err (Message : in String)
procedure Err (Message : in VString)

- Example

<<<TODO>>>

5.3 Get_Debug

- Description

Return true if debug status is on.

- Usage

function Get_Debug return Boolean

- Example

<<<TODO>>>

5.4 Line

- Description

Log a blank line.

- Usage

procedure Line

- Example

<<<TODO>>>

5.5 Log_Dir

- Description

Returns log file directory.

- Usage

```
function Log_Dir return VString
```

- Example

<<<TODO>>>

5.6 Msg

- Description

Log a message. 45 characters max before truncation with a maximum line length of 79.

- Usage

```
procedure Msg (Message : in String)
procedure Msg (Message : in VString)
```

- Example

<<<TODO>>>

5.7 Set_Debug

- Description

Set debug messages status on/[off].

- Usage

```
procedure Set_Debug (Action : Boolean)
```

- Example

<<<TODO>>>

5.8 Set_Disk

- Description

Log to disk on/[off].

- Usage

```
procedure Set_Disk (Action : Boolean)
```

- Example

<<<TODO>>>

5.9 Set_Header

- Description

Line header on/[off].

- Usage

procedure Set_Header (Action : Boolean)

- Example

<<<TODO>>>

5.10 Set_Log_Dir

- Description

Set log file directory.

- Usage

procedure Set_Log_Dir (Dir_In : String)
procedure Set_Log_Dir (Dir_In : VString)

- Example

<<<TODO>>>

5.11 Set_Task

- Description

Set new current log task name. 7 characters max before truncation.

- Usage

function Log_Dir return String

- Example

<<<TODO>>>

5.12 Title

- Description

Log a title. 45 characters max before truncation with a maximum line length of 79.

- Usage

procedure Title (Message : in String);
procedure Title (Message : in VString);

- Example

<<<TODO>>>

6 Prg - Program

6.1 Command

- Description

Constant storing program command (Arg 0).

- Usage

Command : constant VString

- Example

Tio_Line (Command);

/home/sr/Seafile/Sowebio/informatique/dev/ada/app/gnx/src/gnx-instance

---

6.2 Duration_Stamp

- Description

Returns a duration as HHhMMmSSs since Start_Time.

- Usage

function Duration_Stamp (Start_Time : Ada.Calendar.Time) return VString

- Example

<<<TODO>>>

6.3 Get_Version

- Description

Returns program name and formatted program version : "<space>v.minor.major".

- Usage

function Get_Version return VString

- Example

<<<TODO>>>

6.4 Is_User_Not_Root

- Description

Returns true if program user's not root.

- Usage

function Is_User_Not_Root return Boolean

- Example

<<<TODO>>>

6.5 Name

- Description

Return program name.

- Usage

function Name return VString

- Example

```
sr@ro8  ~/Seafile/Sowebio/informatique/github/aide/bin > aide
aide
```

6.6 Path

- Description

Return program path.

- Usage

function Path return String

- Example

```
sr@ro8  ~/Seafile/Sowebio/informatique/github/aide/bin > aide
/home/sr/Seafile/Sowebio/informatique/github/aide/bin
```

6.7 Set_Exit_Status

- Description

Set errorlevel return code. Each call is cumulative. Four calls with 1, 2, 4 and 8 set 15 ie msb-00001111-lsb. Can be used everywhere in the program without special call at its end.
Convention : 1 = no or bad command, 128 = runtime exception (8th bit).

- Usage

procedure Set_Exit_Status (Code : Natural)

- Example

<<<TODO>>>

6.8 Set_Version

- Description

Set program version.

- Usage

procedure Set_Version (Major : Natural; Minor : Natural)

- Example

<<<TODO>>>

6.9 Start_Dir

- Description

Constant storing current directory at start.

- Usage

Start_Dir : constant VString

- Example

<<<TODO>>>

6.10 Start_Time

- Description

Constant storing current directory at start.

- Usage

Start_Time : constant Ada.Calendar.Time

- Example

<<<TODO>>>

6.11 Time_Stamp

- Description

Returns current timestamp as YYYYMMDD-HHMMSS

- Usage

function Time_Stamp return VString

- Example

<<<TODO>>>

7 Sys - System

7.1 Get_Alloc_Ada

- Description

Return current and max allocations done from Ada excluding others languages. Format of returned string : Ada Cur: [868] Max: [1600].

- Usage

function Get_Alloc_Ada return String;

- Example

Prg.Get_Alloc_Ada;

Ada Cur: [868] Max: [1600]

7.2 Get_Alloc_All

- Description

Return current and max allocations done from all languages including Ada. Format of returned string: Ada Cur: [868] Max: [1600]. This uses system calls to find out the program's resident size (RSS) information, both the peak and the current size.

- Usage

function Get_Alloc_All return String;

- Example

```
Prg.Get_Alloc_All;

All Cur: [ 2514944 ] Max: [ 2514944 ]
```

7.3 Get_Env

- Description

Returns VString value of VString or String environment variable Name

- Usage

```
function Get_Env (Name : String) return VString
function Get_Env (Name : VString) return VString
```

- Example

<<<TODO>>>

7.4 Get_Home

- Description

Returns HOME path without trailing slash.

- Usage

```
function Get_Home return VString
```

- Example

```
Get_Home – for user 'sr'

"/home/sr"
```

7.5 Get_Memory_Dump

- Description

Dump information about memory usage. Size is the number of the biggest memory users we want to show. Report indicates which sorting order is used, depending of the following options:
- Prg.All_Reports;
- Prg.Memory_Usage;
- Prg.Allocations_Count;
- Prg.Sort_Total_Allocs;
- Prg.Marked_Blocks;

You must activate memory monitor with Set_Memory_Monitor before using this function.

- Usage

procedure Get_Memory_Dump (Size : Positive; Report_View : Report := Memory_Usage)

Prg.Get_Memory_Dump (1);

- Example

Displaying all report options :

Prg.Get_Memory_Dump (1);

```
Traceback elements allocated:  2480
Validity elements allocated:  1

Ada Allocs: 60608 bytes in 1258 chunks
Ada Free: 60008 bytes in 1248 chunks
Ada Current watermark:  600 in 10 chunks
Ada High watermark:  1600

1 biggest memory users at this time:
Results include bytes and chunks still allocated
Traceback elements allocated:  2480
Validity elements allocated:  1
```

Prg.Get_Memory_Dump (1, Prg.Allocations_Count);

```
Traceback elements allocated:  2798
Validity elements allocated:  1

Ada Allocs: 68456 bytes in 1419 chunks
Ada Free: 67588 bytes in 1405 chunks
Ada Current watermark:  868 in 14 chunks
Ada High watermark:  1600

 1 biggest number of live allocations:
Results include bytes and chunks still allocated
 5.5%: 48 bytes in 1 chunks at 0x000000000040C509 0x000000000040C33B 0x000000000043B74A
0x000000000043D42F   0x000000000042B7A7   0x0000000000407090   0x000000000040C2BE
0x0000000000474D27 0x00000000004053C8
```

Prg.Get_Memory_Dump (1, Prg.Sort_Total_Allocs);

```
Traceback elements allocated:  3106
Validity elements allocated:  1

Ada Allocs: 75816 bytes in 1573 chunks
Ada Free: 74948 bytes in 1559 chunks
Ada Current watermark:  868 in 14 chunks
Ada High watermark:  1600

 1 biggest number of allocations:
Results include total bytes and chunks allocated,
even if no longer allocated - Deallocations are ignored
```

Prg.Get_Memory_Dump (1, Prg.Marked_Blocks);

```
Traceback elements allocated:  3414
Validity elements allocated:  1

Ada Allocs: 83192 bytes in 1727 chunks
```

```
Ada Free: 82324 bytes in 1713 chunks
Ada Current watermark:  868 in 14 chunks
Ada High watermark:  1600

Special blocks marked by Mark_Traceback
  0.0%:  0  chunks  /  1  at  0x000000000040C509  0x000000000040C33B  0x0000000000043B74A
0x0000000000043DB1E    0x00000000004126A5    0x0000000000041AC80    0x0000000000041ED3D
0x0000000000405B71 0x000000000040C2BE 0x0000000000474D27 0x00000000004053C8
```

7.6 Install_Packages

- Description

Install system packages for Debian, Ubuntu or derivatives distributions.

- Usage

function Install_Packages (Packages_List : String) return Boolean

- Example

```
if not Sys.Install_Packages ("curl, libtool, libcurl4, "libcurl4-openssl-dev, libssl-dev") then
   Log.Err ("Can't install system packages.");
end if;
```

7.7 Reset_Memory_Monitor

- Description

Reset all internal data (i.e. reset all displayed counters. This is in general not needed, unless you want to know what memory is used by specific parts of your application.

➪ You must activate memory monitor with Set_Memory_Monitor before using this function.

- Usage

procedure Reset_Memory_Monitor

- Example

```
Reset_Memory_Monitor;
```

7.8 Set_Memory_Monitor

- Description

If Activate_Monitor is true, the program will monitor all memory allocations and deallocations, and through the Get_Memory_Dump procedure below be able to report the memory usage. The overhead is almost null when the monitor is disabled.

- Usage

procedure Set_Memory_Monitor (State : Boolean := True)

- Example

Activate memory monitor :

```
Prg.Set_Memory_Monitor;
```

Disable memory monitor :

```
Prg.Set_Memory_Monitor (False);
```

7.9 Shell_Execute

- Description

Executes shell command. Return the exit code if passed from the executed command. Without Output parameter, the command console output is displayed by default but can be redirected. If Output is used, then the executed command output is return in this parameter.

- Usage

procedure Shell_Execute (Command : String)
procedure Shell_Execute (Command : VString)
procedure Shell_Execute (Command : String; Result : out Integer)
procedure Shell_Execute (Command : VString; Result : out Integer)
procedure Shell_Execute (Command : String; Result : out Integer; Output : out VString)
procedure Shell_Execute (Command : VString; Result : out Integer; Output : out VString)

- Example

```
---------------------
declare
  SE_Result : Integer := 0;
begin
  Sys.Shell_Execute ("find test.cfg", SE_Result);
  Tio.Put_Line(SE_Result);
  Tio.Line;
end;

0 <- found

---------------------
declare
  SE_Result : Integer := 0;
begin
  Sys.Shell_Execute ("find i.dont.exist", SE_Result);
  Tio.Put_Line(SE_Result);
```

```
  Tio.Line;
end;

1 <- not found

--------------------
declare
  SE_Result : Integer := 0;
  SE_Output : VString := +"";
begin
  Sys.Shell_Execute ["cat test.cfg", SE_Result, SE_Output];
  if SE_Result = 0 then
    Tio.Put_Line [SE_Output];
    Tio.Line;
  end if;
end;

[Section_1]
Parameter_11 = Value_11
[Section_2]
Parameter_21 = Value_21
[Section_3]
Parameter_31 = Value_31

...which is the content of test.cfg.
```

8 Tio - Text console

```
Max_Row  : constant Natural := 24;
Max_Column : constant Natural := 79;

subtype Row is Natural range 0..Max_Row;
subtype Column  is Natural range 0..Max_Column;
```

8.1 Beep

- Description

Send a beep.

- Usage

procedure Beep

- Example

<<<TODO>>>

8.2 Clear_Screen

- Description

Clear the screen.

- Usage

procedure Clear_Screen

- Example

<<<TODO>>>

8.3 Cursor_Line_Backward

- Description

Move the cursor backward X rows.

- Usage

procedure Cursor_Line_Backward (X : Row)

- Example

<<<TODO>>>

8.4 Cursor_Line_ Erase

- Description

Erase the current line from the current cursor position to the end of the line.

- Usage

procedure Cursor_Line_Erase (X : Row)

- Example

<<<TODO>>>

8.5 Cursor_Line_ Forward

- Description

Move the cursor forward X rows.

- Usage

procedure Cursor_Line_Forward (X : Row)

- Example

<<<TODO>>>

8.6 Cursor_Line_ Move

- Description

Move the cursor at the specified X,Y coordinates.

- Usage

procedure Cursor_Move (X : Row; Y : Column)

- Example

<<<TODO>>>

8.7 Cursor_Restore

- Description

Restore the previous saved cursor position.

- Usage

procedure Cursor_Restore

- Example

<<<TODO>>>

8.8 Cursor_Save

- Description

Save the current cursor position.

- Usage

procedure Cursor_save

- Example

<<<TODO>>>

8.9 Line

- Description

Create a new blank line, or more than one when Spacing is passed.

- Usage

procedure New_Line (Spacing : Positive)

- Example

<<<TODO>>>

8.10 Get_Immediate

- Description

Get a character validated by [Enter]

- Usage

procedure Get_Immediate (C : out Character)

- Example

```
procedure Pause is

Dummy : Character;

begin

  Put_Line (+"Press any key to continue...");
  Get_Immediate(Dummy);

end Pause;
```

8.11 Pause

- Description

Displays *Press any key to continue or [Ctrl-C] to abort...* waiting for user input.

- Usage

procedure Pause

- Example

```
procedure Test_Pause is
begin

  Pause;
```

end Test_Pause;

8.12 Put

- Description

Print to the console.

- Usage

procedure Put (C : Character)
procedure Put (S : String);
procedure Put (V : VString);

- Example

<<<TODO>>>

8.13 Put_Line

- Description

Print to the console then add a new line.

- Usage

procedure Put_Line (C : Character);
procedure Put_Line (S : String);
procedure Put_Line (V : VString);

- Example

<<<TODO>>>

9 Tio - Text files

subtype File is Ada.Text_IO.File_Type;
Copy_Form : constant String := "preserve=no_attributes,mode=overwrite";

9.1 Append

- Description

Append a file.
File mode is “Out” (write mode).

- Usage

procedure Append (Handle : in out File; Name : String)
procedure Append (Handle : in out File; Name : VString)

- Example

```
Append (File_Tmp_Handle, +"./toto");
while not End_Of_File (File_Tmp_Handle) loop
  Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Close (File_Tmp_Handle);
```

9.2 Close

- Description

Close a file.

- Usage

procedure Close (Handle : in out File)

- Example

```
Open (File_Tmp_Handle, +"./toto");
```

```
while not End_Of_File (File_Tmp_Handle) loop
  Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Close (File_Tmp_Handle);
```

9.3 Create

- Description

Create a file.
File mode is “Out” (write mode).

- Usage

procedure Create (Handle : in out File; Name : String)
procedure Create (Handle : in out File; Name : VString)

- Example

```
Create (File_Tmp_Handle, +"./toto");
while not End_Of_File (File_Tmp_Handle) loop
  Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Close (File_Tmp_Handle);
```

9.4 End_Of_Line

- Description

Return true if end of line is reached.

- Usage

function End_Of_Line (Handle : File) return Boolean
function End_Of_Line (Handle : File) return Boolean

- Example

<<<TODO>>>

9.5 End_Of_File

- Description

Return true if end of file is reached.

- Usage

function End_Of_File (Handle : File) return Boolean
function End_Of_File (Handle : File) return Boolean

- Example

<<<TODO>>>

9.6 Flush

- Description

Flush file buffer to disk.

- Usage

procedure Flush (Handle : in File)

- Example

<<<TODO>>>

9.7 Get

- Description

Get the current line.

- Usage

procedure Get (Handle : File; C : out Character)
procedure Get (Handle : File; S : out String)
procedure Get (Handle : File; I : out Integer);
procedure Get (Handle : File; F : out Real);

- Example

```
Create (File_Tmp_Handle, +"./toto");

while not End_Of_File (File_Tmp_Handle) loop

  Get (File_Tmp_Handle, Line_Buffer);
  Skip_Line;

end loop;

Close (File_Tmp_Handle);
```

9.8 Get_Line

- Description

Get the current line and then move the file pointer to the next line.

- Usage

procedure Get_Line (Handle : File; V : out VString)

- Example

```
Create (File_Tmp_Handle, +"./toto");
while not End_Of_File (File_Tmp_Handle) loop
  Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Close (File_Tmp_Handle);
```

9.9 Is_Open

- Description

Returns true if Handle file is open.

- Usage

function Is_Open (Handle : in File) return Boolean

- Example

<<<TODO>>>

9.10 Line

- Description

Create a new blank line, or more when Spacing is passed.

- Usage

procedure New_Line (Handle : File; Spacing : Positive)

- Example

<<<TODO>>>

9.11 Open_Read

- Description

Open a file. File mode is "In" (read mode).

<<<TODO>>>

- Usage

procedure Open_Read (Handle : in out File; Name : String)

procedure Open_Read (Handle : in out File; Name : VString)

- Example

```
Open_Read (File_Tmp_Handle, +"./toto");
while not End_Of_File (File_Tmp_Handle) loop
  Get_Line (File_Tmp_Handle, Line_Buffer);
end loop;
Close (File_Tmp_Handle);
```

9.12 Put

- Description

Write to a file

- Usage

procedure Put (Handle : File; C : Character)
procedure Put (Handle : File; S : String)
procedure Put (Handle : File; V : VString)

- Example

<<<TODO>>>

9.13 Put_Line

- Description

Write a file and then add a new line

- Usage

procedure Put_Line (Handle : File; C : Character)
procedure Put_Line (Handle : File; S : String)
procedure Put_Line (Handle : File; V : VString)

- Example

<<<TODO>>>

9.14 Reset

- Description

Reset the file pointer to the start of the file

- Usage

procedure Reset (Handle : in out File)

- Example

<<<TODO>>>

10 Vst - VStrings

Variable-size string type

Null_VString : VString

10.1 Element

- Description

Return the Character in Index position of the Vstring argument.
Index starts at one.

- Usage

function Element (Source : VString; Index : Positive) return Character

- Example

<<<TODO>>>

10.2 Ends_With

- Description

Check if VString Item ends with another VString or String Pattern.

- Usage

function Ends_With (Item : VString; Pattern : Character) return Boolean;
function Ends_With (Item : VString; Pattern : String) return Boolean
function Ends_With (Item : VString; Pattern : VString) return Boolean

- Example

```
-- Check VString with String pattern
if Ends_With (+"package",  "age") then
  Put_Line ("Match !");
end if;

-- Check VString with VString pattern
if Ends_With (+"package",  +"age") then
  Put_Line ("Match !");
end if;
```

10.3 Head

- Description

Extract a VString between the beginning to Count Value to a VString.

Count starts at one.

- Usage

function Head (Source : VString; Count : Natural) return VString

- Example

```
Put_Line (Head (+"ABCDEFGH", 4));

"ABCD"
```

10.4 Index

- Description

Returns Natural start position of String or VString Pattern in the target Vstring Source, From a starting index.
Natural is zero if not found.
Natural starts at one.

- Usage

function Index (Source : VString; Pattern : Character) return Natural;
function Index (Source : VString; Pattern : String) return Natural
function Index (Source : VString; Pattern : VString) return Natural
function Index_Backward (Source : VString; Pattern : Character; From : Positive) return Natural;
function Index (Source : VString; Pattern : String; From : Natural) return Natural
function Index (Source : VString; Pattern : VString; From : Natural) return Natural

- Example

```
if Index (+"ABCDABCD", +"BC") = 2 then
   Put_Line ("Match !");
end if;

if Index (+"ABCDEFGH", +"BC", 4) = 6 then
   Put_Line ("Match !");
end if;
```

10.5 Index_backward

- Description

From the end of the target Vstring Source, returns Natural start position of String or VString Pattern in the target Vstring Source, From a backward starting index.
Natural is zero if not found.
Natural starts at one.

- Usage

function Index_Backward (Source : String; Pattern : String) return Natural;
function Index_Backward (Source : VString; Pattern : String) return Natural
function Index_Backward (Source : VString; Pattern : VString) return Natural
function Index_Backward (Source : VString; Pattern : String; From : Natural) return Natural
function Index_Backward (Source : VString; Pattern : VString; From : Natural) return Natural

- Example

```
if Index_Backward (+"abcdefabcdef", +"cd") = 9 then
   Put_Line ("Match !");
end if;

if Index_Backward (+"abcdefabcdef", +"cd", 8) = 3 then
   Put_Line ("Match !");
end if;
```

10.6 Length

- Description

Returns the length of the VString represented by Source.

- Usage

function Length (Source : VString) return Natural

- Example

```
Put (Length (+"ABCDEFGH"));

8
```

10.7 Slice

- Description

Returns a Vstring portion of the Vstring represented by Source delimited by From and To.
From and To start at one.

- Usage

function Slice (Source : VString; From : Positive; To : Natural) return VString

- Example

```
Put_Line (Slice (+"ABCDEFGH", 2,4));
```

```
"BCDE"
```

10.8 Starts_With

- Description

Check if Vstring Item starts with another VString or String Pattern.

- Usage

```
function Starts_With (Item : VString; Pattern : Character) return Boolean;
function Starts_With (Item : VString; Pattern : String) return Boolean
function Starts_With (Item : VString; Pattern : VString) return Boolean
```

- Example

```
-- Check VString with String pattern
if Ends_With (+"package",  "pac") then
  Put_Line ("Match !");
end if;

-- Check VString with VString pattern
if Ends_With (+"package",  +"pac") then
  Put_Line ("Match !");
end if;
```

10.9 Tail

- Description

Extract a VString from Source between its end to backward Count Value.
Count starts at one (backward).

- Usage

```
function Tail (Source : VString; Count : Natural) return VString
```

- Example

```
Put_Line (Tail (+"ABCDEFGH", 4));

"EFGH"
```

10.10 Tail_After_Match

- Description

Extract a VString from Source starting from Pattern+1 position to the end.

- Usage

```
function Tail_After_Match (Source : VString; Pattern : Character) return VString;
```

function Tail_After_Match (Source : String; Pattern : String) return VString;
function Tail_After_Match (Source : VString; Pattern : String) return VString;
function Tail_After_Match (Source : VString; Pattern : VString) return VString;

- Examples

```
Put_Line (Tail_After_Match (Path, '/'));
"gnx-startup"

Put_Line (Tail_After_Match (Path, "ix"));
"/gnx-startup"

Put_Line (Tail_After_Match (Path, "gene"));
"six/gnx-startup"

Put_Line (Tail_After_Match (Path, "etc/genesix/gnx-startu"));
"p"

Put_Line (Tail_After_Match (Path, "/etc/genesix/gnx-startu"));
"p"

Put_Line (Tail_After_Match (Path, "/etc/genesix/gnx-startup"));
empty string

Put_Line (Tail_After_Match (Path, +"/etc/genesix/gnx-startupp"));
empty string

Put_Line (Tail_After_Match (Path, +"/etc/geneseven"));
empty string
```

10.11 To_Lower

- Description

Convert a Character or a VString to lower case.

- Usage

function To_Lower (Item : Character) return Character
function To_Lower (Item : String) return VString
function To_Lower (Item : VString) return VString

- Example

<<<TODO>>>

10.12 To_Upper

- Description

Convert a Character or a VString to upper case.

- Usage

function To_Upper (Item : Character) return Character
function To_Upper (Item : String) return VString
function To_Upper (Item : VString) return VString

- Example

<<<TODO>>>

10.13 Trim_Both

- Description

Returns an all trimmed spaces VString of VString Source.

- Usage

function Trim_Both (Source : VString) return VString

- Example

```
Put_Line (Trim_Right (+"  AB CD  "));
"AB CD"
```

10.14 Trim_Left

- Description

Returns a trimmed leading spaces VString of VString Source.

- Usage

function Trim_Left (Source : VString) return VString

- Example

```
Put_Line (Trim_Left (+"  ABCD  "));
"ABCD  "
```

10.15 Trim_Right

- Description

Returns a trimmed trailing spaces VString of VString Source.

- Usage

function Trim_Right (Source : VString) return VString

- Example

```
Put_Line (Trim_Right (+"  ABCD  "));
"  ABCD"
```

10.16 Trim_Slashes

- Description

Returns an all trimmed slahes VString of VString Source.

- Usage

function Trim_Slashes (Source : VString) return VString

- Example

```
Trim_Slashes ("/")
""

Trim_Slashes ("I")
"I"

Trim_Slashes ("/i")
"I"

Trim_Slashes ("//////i/////")
"i"
```

10.17 +

- Description

Cast a String to a VString.

- Usage

function "+" (C : Character) return VString renames To_VString;
function "+" (S : String) return VString

- Example

<<<TODO>>>

10.18 *

- Description

Duplicate a Character, String or VString Num times to a VString.

- Usage

function "*" (Num : Natural; Pattern : Character) return VString
function "*" (Num : Natural; Pattern : String) return VString
function "*" (Num : Natural; Pattern : VString) return VString

- Example

```
Put_Line (3 * "0");
```

```
"000"

Put_Line (3 * +"12");

"121212"
```

10.19 &

- Description

Concatenate a VString with a VString, String, Character, Integer and Real to a VString

- Usage

```
function "&" (V1, V2 : VString) return VString

function "&" (V : VString; S : String) return VString
function "&" (S : String; V : VString) return VString

function "&" (V : VString; C : Character) return VString
function "&" (C : Character; V : VString) return VString

function "&" (I : Integer; V : VString) return VString
function "&" (V : VString; I : Integer) return VString

function "&" (R : Real; V : VString) return VString
function "&" (V : VString; R : Real) return VString
```

10.20 =

- Description

Test equality between a VString and another VString or String.

- Usage

```
function "="  (Left, Right : VString) return Boolean
function "="  (Left : VString;  Right : String) return Boolean
function "="  (Left : String;  Right : VString) return Boolean
```

- Example

<<<TODO>>>

10.21 <

- Description

<<<TODO>>>

- Usage

```
function "<"  (Left, Right : VString) return Boolean
function "<"  (Left : VString;  Right : String) return Boolean
function "<"  (Left : String;  Right : VString) return Boolean
```

- Example

<<<TODO>>>

10.22 <=

- Description

<<<TODO>>>

- Usage

```
function "<=" (Left, Right : VString) return Boolean
function "<=" (Left : VString;  Right : String) return Boolean
function "<=" (Left : String;  Right : VString) return Boolean
```

- Example

<<<TODO>>>

10.23 >

- Description

<<<TODO>>>

- Usage

```
function ">"  (Left, Right : VString) return Boolean
function ">"  (Left : VString;  Right : String) return Boolean
function ">"  (Left : String;  Right : VString) return Boolean
```

- Example

<<<TODO>>>

10.24 >=

- Description

- Usage

```
function ">=" (Left, Right : VString) return Boolean
function ">=" (Left : VString;  Right : String) return Boolean
function ">=" (Left : String;  Right : VString) return Boolean
```

- Example

<<<TODO>>>

11 Type conversion

11.1 Chr

- Description

Convert an Integer to a Character.

- Usage

function Chr (I : Integer) return Character

- Example

<<<TODO>>>

11.2 To_Integer

- Description

Convert a String or VString to an Integer.

- Usage

function To_Integer (V : String) return Integer
function To_Integer (V : VString) return Integer

- Example

<<<TODO>>>

11.3 To_String

- Description

Convert a VString to an integer.

- Usage

function To_String (V : VString) return Integer

- Example

<<<TODO>>>

11.4 To_VString

- Description

Convert a Char or a String type into VString type.

- Usage

function To_VString (I : Integer) return VString
function To_VString (C : Char) return VString
function To_VString (S : String) return VString

- Example

```
Input : String := "ABC";
Result : VString;
Result := To_VString (Input);
```

v20 architecture

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

1 Introduction

<<<TODO>>>

2 Requirements

An Ada compiler from the GCC/GNAT family, preferably a GNAT CE 2020.
An Unix system, preferably a GNU/Linux Debian (or Debian based like Ubuntu or Mint).

3 Coding guidelines

3.1 General

Language: English
Source code length: 79 columns
Naming: Capitalize and user underscore with compound name. ex: Entry_Value

3.2 Messages

- Log.Msg ("Blahblah.")

Information messages starts with a capital and ends with a dot. Ending message with three dots are only allowed when a user input is waited.

- Log.Err ("v20.Fls.Function_Name - Can't do something.")

Error messages starts with the library or program hierarchy following by a dash and then the error message.

3.3 Naming

We tried to avoid few naming or consistency flaws of the original Ada runtime:

- The text mode *Open* function of v20 now logically opens in *File_In* mode [read mode];
- If the procedures *Put* and *Put_Line* are named like this, then *New_Line* should be called *Line* :]

4 Design

v20 is designed as a KISS working library . It does not attempt to reproduce the outstanding granularity of the Ada runtime.

<<<TODO>>>

4.1 Types

Name	***Packages***	***Description***
Character	Base	
String	VString	Unbounded string subtyping derived from HAC runtime by Gautier de Montmollin
VString	Program	Program and OS related
Integer	Text I/O	Text Input/Output related
Boolean	Logging	Log - Terminal and file log - on top of Tio
BCD		Financial computing
Float		Scientific computing
Geo		Geo. Coords.
	handling ok	

4.2 Packages

Name	***Packages***	***Description***
v20	Base	
Bio	Binary I/O	Binary IO: Binary files, locking, etc.
Cfg	Configuration files	Simple and user friendly config files handling
Dbf	Multiusers btree DB	Data base files: indexed btree with locks management - on top of Bio
Eml	Email	Pop3/Smtp
Fls	File system	
Log	Logging	Log - Terminal and file log - on top of Tio
Pdf	Pdf handling	See Gautier de Montmollin package
Prg	Program	Program and user related
Prt	Printer package	Print to local network duplex A3 & A4 printer [see previous works: v90, psrc and a2ps]
Rts	Run Time System	AVR embedded
Ser	Serial handling	Tx, Rx and spying

Name	Packages	Description
Sys	System	Operating System related
Tio	Text I/O	Text Input/Output related
Usb	Usb handling	Tx, Rx and spying
Vst	VString	Unbounded string subtyping derived from HAC runtime by Gautier de Montmollin
	Already coded	

4.3 Functions

About strings, v20 functions always [*should* actually] return VString [never String type].

FAQ

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

1 Conventional exit codes

v20 should returns:
- 1 if bad or no commands;
- 128 if an exception occurs during execution.

<<<TODO>>>

2 Log has too long separators lines

Programs examples

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

1 test.adb

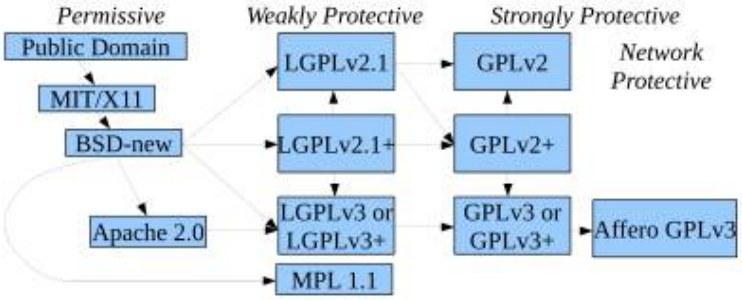
<<<TODO>>>

Appendices

1 Copyrights & credits

1.1 Library Licence

v20 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

1.3 v20 Packages copyrights & credits

Vst - Variable Strings from HAC runtime - gdm sr : HAC is copyright Gautier de Montmollin.

2 To-do list

2.1 v20

About strings, v20 functions ***should*** return VString, **never** String type.
Add Tio.Cursor_On and Cursor_Off (need a little C binding to a Linux function call).
Add function [enter] or [quit]
Add function [Yes] or [no] with Yes/No default choice

2.2 Doc

- The never-ending task

Hunt **<<<TODO>>>** tags :]

3 Quality control

Check list

<<< TODO>>>

4 Release check list

Things to do to release to github

<<< TODO>>>

5 Issues

5.1 Compiler bug reporting

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

- Exception with Delete_Tree dealing with broken symbolic links

In french only: Ada.Directories.Del_Tree explose en présence d'un lien symbolique cassé dans un répertoire de l'arborescence à effacer: raised ADA.IO_EXCEPTION-S.USE_ERROR: directory tree rooted at "/home/sr/opt/gnat-2019/lib/xmlada/xm-lada_input.relocatable" could not be deleted

- Demo

L'empilement général est

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

Du coté de More_Entries > Fetch_Next_Entry > readdir_gnat > Match

On arrive à un /lien symbolique cassé/ libxmlada_input_sources.so qui
est /déclaré ne pas exister/ par File_Exists_Attr (C_Full_Name'Address,
Attr'Access); en 776 qui est en fait __gnat_file_exists_attr en 1668 de
adaint.c qui fait référence à une structure dans adaint.h:

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

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

  unsigned char exists;
```

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

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

Qui appelle  *__gnat_stat_to_attr*

Qui teste un file descripteur à -1, lien symbolique cassé je suppose...

Puis __gnat_stat qui renvoie 2 à __gnat_stat_to_att

Avec le test suivant en 1124 de adaint.c

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

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

<shadok> Donc si on trouve pas le fichier, c'est qu'il n'y a pas
d'erreur. </shadok>

La suite devient alors compréhensible... Le lien symbolique cassé
libxmlada_input_sources.so est déclaré ne pas exister, la routine sort
du répertoire courant (qu'elle croit donc vidé) pour l'effacer et
explose alors quand elle tente d'effacer ce répertoire vide mais qui ne
l'est pas...
```

- Solving

On pourrait re-coder cette fonction récursive plus simplement. Cru voir en traçant que la fonction C d'effacement récursif existe déjà... Toutefois, le mieux serait de corriger l'anomalie qui est probablement dans _gnat_stat, afin que cette fonction retourne la bonne valeur et ne confonde pas 'n'existe pas' (le fichier sur lequel pointe le lien symbolique) avec 'n'existe pas' (le fichier symbolique).

Ada, “it’s stronger than you”.

Tribute to Daniel Feneuille, legendary french Ada teacher

In Strong Typing We Trust !

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