ADA USER JOURNAL

Volume 40
Number 3
September 2019

Contents

Quarterly News Digest

Alejandro R. Mosteo

Centro Universitario de la Defensa de Zaragoza, 50090, Zaragoza, Spain; Instituto de Investigación en Ingeniería de Aragón, Mariano Esquillor s/n, 50018, Zaragoza, Spain; email: amosteo@unizar.es

Contents

Ada-related Organizations

Additional Comment Period for Upcoming Ada Revision

From: "Randy Brukardt" <randy@rrsoftware.com>
Subject: Additional Comment Period for Upcoming Ada Revision
Date: Fri, 26 Jul 2019 21:53:30 -0500
Newsgroups: comp.lang.ada

ISO/IEC JTC 1/SC 22/WG 9 (WG 9) is responsible for the maintenance and revision of the Ada Programming Language and associated standards and technical reports. As part of the language maintenance activity, WG 9 has established a group of Ada experts as the Ada Rapporteur Group (ARG). The ARG receives input from the Ada community at large to consider for inclusion in revision to the Ada programming language standard. The WG 9 has produced a number of revisions to the language in accordance with ISO policy and to address the evolution of technology (Ada 83, Ada 95, Ada 2005 and Ada 2012).

Presently, the ARG is nearing completion on a revision to Ada 2012 (known for now as Ada 202x) which includes new contracts and lightweight parallelism features. Concern has been raised that these new proposals have not been prototyped nor has the suitability for diverse target environments been assessed.

Therefore, the ARG is seeking comments, based on prototyping and review, on the new features (focused on the parallelism features) incorporated within the current draft of the Ada 202X standard. Comments should be submitted to ada-comment@ada-auth.org as described in the Ada Reference Manual Introduction (http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-0-3.html#p58). Please include the draft number with any Ada Reference Manual references in your comment. Comments should be sent by 1 June 2020 in order to be considered for the revision. (Note: While not required, joining the mailing list as described at http://www.ada-auth.org/comment.html is recommended so that you receive any queries on or responses to your comment.)

The draft revision can be found at http://www.ada-auth.org/standards/ada2x.html.

A list of issues addressed in Ada 202x can be found at http://www.ada-auth.org/ai-files/grab_bag/2020-Amendments.html.

(You can find an on-line version of this announcement at https://www.adaic.org/2019/07/additional-comment-period-for-upcoming-ada-revision/.)

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 26 Jul 2019 22:02:39 -0500n

To translate this announcement into plain English, the completion date of Ada 202x has been pushed back a year and a half in order to get more feedback on the proposed changes. Most of the major features went from rough outlines last fall to a completed standard with detailed wording by May. This rate of completion was just too much for most interested parties outside of the ARG to keep up with.

Rather than standardize something under-baked that might have to be changed in a few years, we're dialing back the amount of work and letting the Ada community catch up.

This comment period is not intended to introduce additional new features; such comments are always welcome but most will be deferred until the following revision. (Of course, additional features related to the ones already intended for the revision are possible.)

From: "Yannick Moy" <moy@adacore.com>
Date: Mon, 29 Jul 2019 02:44:22 -0700

I would add that participation in the new Ada/SPARK RFC website hosted by AdaCore is very welcome for anyone who wants to influence the future of Ada and/or SPARK:

https://github.com/AdaCore/ada-spark-rfcs

Participation can come in many flavors:

- signal your opinion on Pull Requests (PR) by adding a thumb-up/thumb-down on the first message of the PR
- comment on a PR to refine your opinion
- propose an RFC as a PR for others to comment

Ada-related Events

[To give an idea about the many Ada-related events organized by local groups, some information is included here. If you are organizing such an event, feel free to inform us as soon as possible. If you attended one such event, please consider writing a small report for the Ada User Journal.]

Ada-Europe 2019 Final Call for Participation

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Subject: Press Release - Reliable Software Technologies, Ada-Europe 2019
Date: Tue, 4 Jun 2019 22:21:31 -0000
Newsgroups: comp.lang.ada

FINAL Call for Participation

*** UPDATED Program Summary ***

24th International Conference on

Reliable Software Technologies - Ada-Europe 2019

11-14 June 2019, Warsaw, Poland

http://www.ada-europe.org/conference2019

**Check out tutorials and workshop! **

http://www.ada-europe.org/conference2019/tutorials.html

http://www.ada-europe.org/conference2019/workshops.html

*** Exhibition Opening & Welcome Aperitif on Tuesday ***

*** Full Program available on conference web site ***

*** Register now! ***

Press release:

24th Ada-Europe Conference on Reliable Software Technologies

International experts meet in Warsaw

Warsaw, Poland (5 June 2019) - Ada-Europe together with EDC (the Engineering Design Center, a partnership of General Electric and the Institute of Aviation), organize from 11 to 14 June 2019 the "24th International Conference on Reliable Software Technologies - Ada-Europe 2019" in Warsaw, Poland. The event is in cooperation with the Ada Resource Association (ARA), and with ACM's Special Interest Groups on Ada (SIGAda), on Embedded Systems (SIGBED) and on Programming Languages (SIGPLAN).

[...]

This year's conference offers tutorials and a workshop, two keynotes, a technical program of refereed papers and industrial presentations, an industrial exhibition and vendor presentations, and a social program.

Two tutorials are scheduled on Tuesday, targeting different audiences: "An Introduction to Ada", for those who want to understand the benefits of using Ada; and "Controlling I/O Devices with Ada, using the Remote I/O Protocol", for those willing to develop Ada programs that control external hardware devices. On Friday the conference hosts for the 6th consecutive year the workshop on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" (DeCPS 2019): registration is complementary for conference participants.

The industrial exhibition opens Tuesday mid-afternoon in the networking area and runs until the end of Thursday afternoon. Exhibitors include AdaCore, PTC Developer Tools, Rapita Systems, Vector, and Ada-Europe. All tutorial and conference participants are invited to the exhibition opening, as well as to the Welcome Aperitif afterwards.

Two eminent keynote speakers have been invited to open each day of the core conference program: Michael Klemm (OpenMP, Germany), on "OpenMP API: A Story about Threads, Tasks and Devices"; and Tucker Taft (AdaCore, USA), on "A 2020 View of Ada".

The technical program on Wednesday and Thursday presents 9 refereed technical papers and 8 industrial presentations in sessions on Assurance Issues in Critical Systems, Tooling Aid for Verification, Best Practices for Critical Applications, Uses of Ada in Challenging Environments, Verification Challenges, and Real-Time Systems. Also included is a speaker's corner on "Experience from 40 years of teaching Ada", and vendor presentations. Peer-reviewed papers will be published in an open-access journal, industrial presentations and tutorial abstracts in the Ada User Journal, the quarterly magazine of Ada-Europe.

The social program includes on Tuesday evening a Welcome Aperitif on the terrace of the Institute of Aviation, enjoying a wonderful view of the Warsaw airport and city center, accompanied by drinks and typical Polish snacks. On Wednesday evening will be the traditional Ada-Europe Conference Banquet, with Polish cuisine, drinks, and live piano music, in the restaurant "Przepis na kompot" in the town where Chopin was born.

The Best Paper Award will be presented during the Conference Banquet, the Best Presentation Award during the Closing session.

The full program is available on the conference web site. [...]

Latest updates:

The 12-page "Final Program" is available at http://www.ada-europe.org/conference2019/AE-2019-Final-Program.pdf

Check out the tutorials in the PDF program, or in the schedule at http://www.ada-europe.org/conference2019/tutorials.html.

[...]

A printed Conference Booklet with abstracts of all technical papers and industrial presentations will be included in every conference handout.

Help promote the conference by advertising for it:

http://www.ada-europe.org/conference2019/promotion.html

Put up the poster at

http://www.ada-europe.org/conference2019/picts/AE2019_poster.pdf

Recommended Twitter hashtags: #AdaEurope and/or #AdaEurope2019.

For more info and latest updates see the conference web site at http://www.ada-europe.org/conference2019.

Update about Ada-Europe Conferences 2019 and 2020

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Subject: Update about Ada-Europe Conferences 2019 and 2020
Date: Sat, 29 Jun 2019 13:53:33 -0000
Newsgroups: comp.lang.ada

The 24th edition of Ada-Europe's International Conference on Reliable Software Technologies took place on 11-14 June in Warsaw, Poland, with considerable success.

The conference, graciously hosted by the Institute of Aviation, had nearly 100 participants, enjoyed a rich technical and social program, and saw much active interaction between participants, presenters, and exhibitors.

For your information, the following material is now available online:

- the "Conference Booklet" in PDF, which contains the abstracts of all presentations in the core program (see first section on [1]);
- copies of conference presentations (see "Download" links in "Conference Core Schedule" table on [1]);
- copies of DeCPS workshop presentations (see "Download links in "Program" table on [2]);
- pictures of the exhibition booths (see final part of [3]).

[1] www.ada-europe.org/conference2019/overview.html

[2] www.ada-europe.org/conference2019/workshops.html

[3] www.ada-europe.org/conference2019/sponsors.html

As announced in Warsaw, next year's conference will be held in Santander, Spain, in the week of 8-12 June 2020.

The preliminary Call for Contributions is already available on the (mini) conference web site at [4]. More details will follow later.

[4] www.ada-europe.org/conference2020/

On this occasion, the Ada-Europe Board announces a slight update of the name of its conference series:

- the complete name is "25th Ada-Europe International Conference on Reliable Software Technologies";
- the short name is "Ada-Europe Conference 2020";
- the acronym is "AEiC 2020".

Hence on social media when referring to the Ada-Europe organization we'll use #AdaEurope, and when referring to next year's Ada-Europe Conference we'll use #AEiC2020.

Ada-related Resources

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 2019/Aug/06
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 2_848 (+35) members [1]
- Reddit: 2_307 (+64) members [2]
- StackOverflow: 1_685 questions [3]

- Freenode: 76 (-11) users [4]
- Gitter: 42 (=) people [5]
- Telegram: 45 (-2) users [6]
- Twitter: 32 (+26) tweeters [7]
  36 unique tweets [7]

[1] https://www.linkedin.com/groups/114211/

[2] http://www.reddit.com/r/ada/

[3] http://stackoverflow.com/questions/tagged/ada

[4] #Ada on irc.freenode.net

[5] https://gitter.im/ada-lang

[6] https://t.me/ada_lang

[7] http://bit.ly/adalang-twitter

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 2019/Aug/06
To: Ada User Journal readership

GitHub: 573 (-30) developers [1]
Rosetta Code: 666 (+2) examples [2]
36 (=) developers [3]
Sourceforge: 270 (=) projects [4]
Open Hub: 209 (=) projects [5]
Bitbucket: 87 (=) repositories [6]
Codelabs: 47 (+1) repositories [7]
AdaForge: 8 (=) repositories [8]

[1] https://github.com/search?q=language%3AAda&type=Users

[2] http://rosettacode.org/wiki/Category:Ada

[3] http://rosettacode.org/wiki/Category:Ada_User

[4] https://sourceforge.net/directory/language:ada/

[5] https://www.openhub.net/tags?names=ada

[6] https://bitbucket.org/repo/all?name=ada&language=ada

[7] https://git.codelabs.ch/?a=project_index

[8] http://forge.ada-ru.org/adaforge

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Thu May 23 2019
To: Ada User Journal readership

Note: positive ranking changes means to go down in the ranking.

- TIOBE Index: 37 (+1) 0.296% (-0.03%) [1]
- IEEE Spectrum (general): 42 (-4) [2]
- IEEE Spectrum (embedded): 13 (=) [2]

[1] https://www.tiobe.com/tiobe-index/

[2] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

Ada-related Tools

Pure Ada libraries for Artificial Intelligence

From: Daniel <danielnorberto@gmail.com>
Subject: Artificial Intelligence libraries in Ada
Date: Wed, 10 Jul 2019 00:25:48 -0700
Newsgroups: comp.lang.ada

Does anybody knows pure Ada libraries for AI?

Specially, I'm interested in Decission Trees, but I can't find anything on internet.

In case of a negative answer, does anybody knows a good CPU performance AI C/C++ Library working good binded to Ada code?

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Wed, 10 Jul 2019 09:39:39 +0200

There is FannAda (https://sourceforge.net/projects/lfa/), a binding to the Fann neural network library. No idea what it's worth.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 10 Jul 2019 12:52:40 +0200

http://www.dmitry-kazakov.de/ada/fuzzy_ml.htm

This includes decision trees both fuzzy and crisp. It is 100% Ada, except the database persistence back ends.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Wed, 10 Jul 2019 18:13:14 +0200

I guess you're not interested in neural networks, but there's an implementation of REM NNs in the PragmAda Reusable components.

https://github.com/jrcarter/PragmARC

[...] It's NNs with the REM 2nd-order learning algorithm.

http://pragmada.x10hosting.com/REM_Eq.pdf

Gnu Emacs Ada mode 6.1.1

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode 6.1.1 released.
Date: Fri, 12 Jul 2019 11:10:22 -0700
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 6.1.1 is now available in GNU ELPA. This is a minor feature and bug fix release; partial file parsing is now supported for `which-function-mode', and error correction is improved. See the NEWS files in ~/.emacs.d/elpa/ada-mode-6.1.1 and wisi-2.1.1, or at http://www.nongnu.org/ada-mode/, for more details.

The process parser requires a manual compile step, after the normal list-packages installation:

cd ~/.emacs.d/elpa/ada-mode-6.1.1

./build.sh

This requires AdaCore gnatcoll packages which you may not have installed; see ada-mode.info Installation for help in installing them.

dcf-ada 2.0.0 Library for Document Container Files

From: onox <denkpadje@gmail.com>
Subject: ANN: dcf-ada 2.0.0 -- A library for document container files, a Zip-based archive format
Date: Tue, 23 Jul 2019 14:15:03 -0700
Newsgroups: comp.lang.ada

An Ada 2012 library for document container files, a Zip-based archive format standardized in ISO/IEC 21320-1:2015. Document container files are Zip files with several restrictions:

* Only "store" (uncompressed) and "deflate" compression methods are allowed
* Archives may not be encrypted or contain digital signatures
* Archives may not span multiple volumes or be segmented

This library is based on the Zip-Ada library, with extensive modifications:

* Binary and Windows-specific files have been removed with The BFG Repo Cleaner
* Reformatted code to Ada default style guide
* Removed obsolescent features and implementation-defined extensions
* All packages except one that uses Ada.Calendar are preelaborated
* Removed features prohibited by ISO standard
* Removed lots of duplicated code and simplified the API, reducing SLOC from 12k to 4.5k

Although the tools can (un)zip basic .zip files, the purpose of the library is to be able to read container files, including a future binary storage format for 3D meshes.

See the README.md at https://github.com/onox/dcf-ada on how to list or extract files from an archive.

Qt5Ada 5.13.0

From: leonid.dulman@gmail.com
Subject: Announce : Announce : Qt5Ada version 5.13.0 (594 packages) release 01/07/2019 free edition
Date: Sat, 3 Aug 2019 05:09:13 -0700
Newsgroups: comp.lang.ada

Qt5Ada is Ada-2012 port to Qt5 framework (based on Qt 5.13.0 open source final)

Qt5ada version 5.13.0 open source and qt5c.dll(win64),libqt5c.so(x64) built with Microsoft Visual Studio 2017 x64 in Windows, gcc x86-64 in Linux.

Package tested with gnat gpl 2012 Ada compiler in Windows 64bit, Linux x86-64 Debian 9.4.

It supports GUI, SQL, Multimedia, Web, Network, Touch devices, Sensors, Bluetooth, Navigation and many others thinks.

My configuration script to build Qt 5.13.0 is: configure -opensource -release -nomake tests -opengl dynamic -qt-zlib -qt-libpng -qt-libjpeg -openssl-linked OPENSSL_LIBS="-lssleay32 -llibeay32" -plugin-sql-mysql -plugin-sql-odbc -plugin-sql-oci -icu -prefix "e:/Qt/5.13"

As a role Ada is used in embedded systems, but with QTADA(+VTKADA) you can build any desktop applications with powerful 2D/3D rendering and imaging (games, animations, emulations) GUI, Database connection, server/client, Internet browsing , Modbus control and many others thinks.

Qt5Ada and VTKAda for Windows, Linux (Unix) is available from https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/

The full list of released classes is in "Qt5 classes to Qt5Ada packages relation table.docx"

VTKAda version 8.2.0 is based on VTK 8.2.0 (OpenGL2) is fully compatible with Qt5Ada 5.13.0.

Qt5AVAda

From: leonid.dulman@gmail.com
Subject: Announce : QtAVAda version 1.12.0 release 01/08/2019 free edition
Date: Sat, 3 Aug 2019 05:09:13 -0700
Newsgroups: comp.lang.ada

Qt5AVAda is ada-2012 port to QtAV multimedia playback framework based on Qt + FFmpeg. Cross platform. High performace. Easy to use and base on QtAV 1.12 developed by wang-bin https://github.com/wang-bin/QtAV.

QtAVAda build widgets inside Qt5Ada application(5.13.1 release 01/08/2019).

QtAVAda for Windows, Linux (Unix) is available from https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio

If you have any problems or questions, tell me know.

String edit v3.5

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: String edit v3.5 released
Date: Sun, 4 Aug 2019 16:56:40 +0200
Newsgroups: comp.lang.ada

The library provides various means for editing and formatting strings:

http://www.dmitry-kazakov.de/ada/strings_edit.htm

This release adds implementations of some standards actively used in communication RFC 3061, 4514; ISO 8601.

Changes to the previous version:

- Added the package Strings_Edit.Long_Floats, an instance of String_Edit.Floats with Long_Float;
- The package Strings_Edit.UTF8.ITU_T61 provides ITU T.61 encoding conversions;
- The package Strings_Edit.Object_Identifiers provides implementation of RFC 3061 object identifiers (OID);
- The package Strings_Edit.Distinguished_Names provides implementation of RFC 4514 distinguished names (DN);
- The package Strings_Edit.ISO_8601 provides ISO 8601 representations of time and duration;
- Encoding and decoding Base64 streams were added to the package Strings_Edit.Base64.

Simple Components for Ada v4.41

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components for Ada v4.41 released
Date: Mon, 5 Aug 2019 13:57:16 +0200
Newsgroups: comp.lang.ada

The new release is focused on ASN.1 support. The implementation does not require ASN.1 compiler. It is based on reflection of Ada attributes. The objects corresponding to ASN.1 objects are put together into record types and the encoding is deduced from the placement. The implementation provides arena pool to allocate data associated with ASN.1 objects. This allows to handle very large and indefinite ASN.1 objects without allocating maximum possible memory in advance. This also enables sharing memory between ASN.1 CHOICE alternatives as well as recursively defined ASN.1 objects. Implementations of LDAP and X.509 certificates based on ASN.1 are provided.

http://www.dmitry-kazakov.de/ada/components.htm

Changes to the previous version:

- The package OpenSSL was extended;
- Added implementation of ASN.1 encoding;
- X.509 ASN.1 certificates implementation added;
- LDAP implementation added.

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 5 Aug 2019 07:22:43 -0700

Wow!

This is incredible news, especially for things like the Wasabee browser project.

There was someone who was working on an Ada/SPARK ASN.1 compiler (Peter Chapin?) and I think the people doing this project -- https://github.com/ttsiodras/asn1scc -- which *is* an ASN.1 compiler.

WRT the OpenSSL dependency, how much work would it be to get rid of it?

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 5 Aug 2019 17:56:43 +0200

> Wow!
>
> This is incredible news, especially for things like the Wasabee browser project. There was someone who was working on an Ada/SPARK ASN.1 compiler (Peter Chapin?) and I think the people doing this project [...]

I am aware of ASN1SCC, but I wanted an alternative approach that does not require code generator and can handle constraints dynamically.

ASN.1 specifications are infested with objects defined up to "MAX" items. E.g. the LDAP filter is a variable record (CHOICE) with disjunctive and conjunctive forms as alternatives containing the LDAP filter recursively as terms. The number of terms is an unspecified MAX and the depth of recursion is kind of infinite. I have no idea how the generators handle this mess. If compiled literally, e.g. with MAX=256 depth=32, it would take a huge amount of memory while in reality it is bounded from above just by the message length.

> WRT the OpenSSL dependency, how much work would it be to get rid of it?

There is no dependency on OpenSSL.

OpenSSL and GNUTLS are two back-ends used in the corresponding implementations of the secure connection handler. Both are separate gpr-projects.

All network stacks are designed to work with any handler implementation. Should Ada TLS become available I would use it in yet another implementation of.

Ada-related Products

Embedded Boards for Ada

From: Ricardo Brandão <rbrandao.br@gmail.com>
Subject: Which embedded devices do you use?
Date: Tue, 4 Jun 2019 08:01:50 -0700
Newsgroups: comp.lang.ada

I worked with embedded systems for a long time.

I started with Z-World devices on late 80's. And now I'm working mainly with ESP32 boards.

I'm learning Ada and I'd like to use it on my new projects. So, I'd like to know what boards/processors you guys are using.

Normally, my projects need Digital IOs, Analog Inputs, and any way to wireless communication: Bluetooth, BLE, WiFi...

And I'm used to work with I2C devices as well (OLED displays, sensors, RTC, and so on).

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 4 Jun 2019 17:14:33 +0200

On 2019-06-04 17:01, Ricardo Brandão wrote:

> So, I'd like to know what boards/processors you guys are using.

ARM-based boards with a Linux on it.

> Normally, my projects need Digital IOs, Analog Inputs, and any way to wireless communication: Bluetooth, BLE, WiFi...

For quality analogue I/O we are using EitherCAT or ModBus terminals. For digital I/O on board GPIO could serve but usually it is terminals as well. CAN and Serial is used too.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 4 Jun 2019 17:56:39 +0200

On 2019-06-04 17:26, Ricardo Brandão wrote:

> So, it could be a good idea use Beaglebone as a start point?

Yes. We are using BB a lot, for prototyping etc.

From: Optikos <optikos@verizon.net>
Date: Tue, 4 Jun 2019 08:55:21 -0700

I like Marvell's ESPRESSObin board, as distributed in the USA by Globalscale Technologies (shipped direct from PRChina).

http://ESPRESSObin.net

With an Armada 3720 SOC, it is capable of doing some serious telecom/datacom high-speed packet processing with some hardware assist (instead of slow software-processor speed) on its 2 LAN and 1 WAN Ethernet ports. (Of course better would be the 7000 or 8000 series Armadas which have full-fledged SR-IOV on their SOC, but hey there is always room for improvement in the future.)

There is also the ESPRESSObin's baby brother (with fewer Ethernet ports): the new Sheeva64 in wall-wart form-factor, continuing the venerable SheevaPlug family.

https://www.GlobalscaleTechnologies.com/p-86-sheeva64.aspx

What is nice about the ESPRESSObin and Sheeva is that they are embrace Yocto-Project Linux, so you are not tied to any one Linux distro. Instead, Yocto Project requires that you roll your own Linux distro from near-scratch (e.g., mimicking whichever distro or bleeding edge referent* that you prefer).

* e.g., Linus Torvalds' git repository

https://www.YoctoProject.org

Each ARM hobbyist SBC community has a different specialty. I wouldn't do high-packet-rate telecom/datacom processing on a Raspberry Pi, for example. That is what the Marvell Armada line is better suited for.

Btw, Marvell's Armada series is the descendent whose ancestors include the DEC StrongARM and the Intel XScale, so in some ways this is one of the “main trunks” in the ARM-processor community, especially for industrial usage–not some twig on a branch.

https://www.TheRegister.co.uk/2006/06/27/intel_sells_xscale

Plus, Marvell's MoChi (modular chip multi-die SOCs) technology (•not• in the Armada 3720) is one of the industry leaders in DARPA's MoChi endeavors in recent years. DARPA is trying to seed some of the major SOC processor manufacturers with MoChi. Getting on board with Marvell now likely prepares you for the aggressive MoChi future as the 1st-generation-MoChi 7000 and 8000 series eventually migrates into the hobbyist SBCs, and then aggressive-MoChi successors follow after that in coming years.

https://www.marvell.com/architecture/mochi

From: Olivier Henley <olivier.henley@gmail.com>
Date: Tue, 4 Jun 2019 11:51:10 -0700

You can dig here:

- https://github.com/ohenley/awesome-ada#Runtimes (the bb-runtimes repo by AdaCore)
- https://github.com/ohenley/awesome-ada/blob/master/README.md#Hardware-and-Embedded (The main repo to check is ada-drivers-library. Adacore is behind and they are of great assistance.)
- https://github.com/ohenley/awesome-ada/blob/master/README.md#Books Do not forget to check the book about embedded by Maciej Sobczak.

Hope it helps and any PR/Suggestions to refactor the list is welcome.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Tue, 4 Jun 2019 22:14:12 +0300

The AdaCore "Make with Ada" competition entries use a wide range of hardware. See https://www.hackster.io/contests/adacore/submissions#challengeNav.

(As for myself, I've recently used Ada for embedded systems only in space applications, so only on made-for-space computers, usually with SPARC processors and a high price tag.)

From: Philip Munts <philip.munts@gmail.com>
Date: Wed, 5 Jun 2019 01:33:09 -0700

BeagleBone (more and better I/O) and Raspberry Pi (faster). Both running my own embedded Linux distribution:

https://github.com/pmunts/muntsos

Debian and Raspbian are fine general purpose operating systems, but IMHO they are wretched for embedded systems.

Anything on mains power should be running Linux. The networking capabilities and development tools are just so far beyond microcontrollers.

I'm especially fond of the PocketBeagle and the Raspberry Pi Zero Wireless. Running Ada programs, of course.

Janus/Ada 3.2.1

From: "Randy Brukardt" <randy@rrsoftware.com>
Subject: Janus/Ada 3.2.1 Released!
Date: Wed, 26 Jun 2019 00:12:16 -0500
Newsgroups: comp.lang.ada

A new version of Janus/Ada has finally made it to release. This version includes recognition of the full Ada 2012 syntax, null exclusions, private with, a number of language-defined libraries from both Ada 2005 and 2012, and code quality warnings to detect likely bugs early.

Read the full announcement at

http://www.rrsoftware.com/html/blog/ja-321a-rel.html.

Existing customers with a current support agreement (including those in their first 90 days of ownership) can download the new version and use their existing key to unlock it. For everyone else, see our website for pricing: http://www.rrsoftware.com/html/companyinf/prices.htm.

Randy Brukardt.

P.S. I apologize to anyone that would rather not see the blatant ad. I try not to

do this more often than once per year, and the information ought to be relevant to those who sometimes forget that there are other, actively developed Ada compilers out there.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Wed, 26 Jun 2019 08:53:02 +0200

Good news. I see that the website still refers to the compiler as Janus/Ada 95. How much additional work is needed before you have a full Ada-12 compiler?

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 26 Jun 2019 17:40:42 -0500

Probably more years than I have left on the planet. While I've mapped out a design for most new features, a few things have been pretty much ignored (esp. interfaces and real-time stuff).

If I was able to find a business plan that made sense, it could get done faster, but as it stands I don't expect to ever break even with it and as such one can't really spend $$$ (as opposed to time) on it.

From: Optikos <optikos@verizon.net>
Date: Wed, 26 Jun 2019 08:41:54 -0700

On Wednesday, June 26, 2019 at 3:52:51 AM UTC-5, Dmitry A. Kazakov wrote:

[...]

> P.S. I hope Janus will target Linux someday. It could be a Windows-hosted cross. I think many would buy that thing.

I concur, but the highest-RoI would be for Janus/Ada to have the LLVM backend in one fell swoop. Then we as users would naturally get various object-file formats (e.g., ELF, XCOFF) and ISAs (e.g., Apple ARM) and debug formats (e.g., gdb's; lldb's)—both native and cross-compiled—inherited as a by-product, killing multiple birds with one stone.

Randy, would putting Janus/Ada's front end on

0) LLVM backend

be more difficult than any major target feature listed above alone (e.g.:

1) Janus/Ada as-is without LLVM plus ELF on x86;

2) Janus/Ada as-is without LLVM plus PE-on-ARM for the forthcoming ARM-based bendable/foldable Surface Phone thingy-whatever-it-will-be-called, deriving from Andromeda & Courier prototypes with Composable-Shell and Windows Core OS)?

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 26 Jun 2019 17:36:27 -0500

[Replying to the numbered list in the previous post:]

These are almost completely orthogonal: the existing code generator would work for Linux, and the (old) Unix JLink did ELF. The issue with Linux is updating the runtime to use Linux system calls (these are different than the ones from the old Unix).

OTOH, attaching LLVM is a totally different level of work, and I don't know enough about LLVM to say how easy or hard it would be. OTOH, we did something similar of Unisys, so we already have most of the ability available.

But again, note that a code generator is a small (and usually easiest) part of porting to a new target. Making a usable runtime (that is, exception handling, finalization, overflow checking, divide-by-zero traps, basic I/O, and most of all, tasking) is generally a bigger job.

AdaControl 1.21r3

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: [Ann] AdaControl version 1.21r3 released
Date: Thu, 11 Jul 2019 14:42:04 +0200
Newsgroups: comp.lang.ada

Adalog is pleased to announce version 1.21r3 of AdaControl. There are now 71 rules, 579 subrules.

This version includes new checks to ease the transition to Ada 2012 (like for-in loops that can be changed to for-of loops), improvements to the auto-fixing features, extensions to existing rules (like use-package that can be changed to use use-type or use-all-type), bug fixes... See file HISTORY for the complete list of improvements.

The pre-compiled version uses now GNAT Community 2019.

Available from http://www.adacontrol.fr

Enjoy!

Ada and Operating Systems

GNAT CE 2019 and Impending Changes on MacOS

From: Simon Wright <simon@pushface.org>
Subject: Re: GNAT CE 2019 macOS
Date: Thu, 30 May 2019 20:34:44 +0100
Newsgroups: comp.lang.ada

[The following post discusses an issue with missing system libraries during linking in MacOS, due to changes in the operating system SDK.]

Bill Findlay <findlaybill@blueyonder.co.uk> writes:

>> gnatlink /Users/wf/mekhos/MacOSX/e.ali -funwind-tables -fdata-sections -ffunction-sections -mtune=native -fno-stack-check -fomit-frame-pointer -flto -O3

> ./quad_div.o -Wl,-dead_strip -Wl,-dead_strip -flto

>

>> ld: library not found for -lSystem

>> collect2: error: ld returned 1 exit status

>> gnatmake: *** link failed.

>

> -lSystem ??

I've had a discussion about this with AdaCore.

The problem they are addressing is that Apple are moving towards having system includes only in the SDKs rather than in /usr/include; see [1], which says "As a workaround, an extra package is provided which will install the headers to the base system. In a future release, this package will no longer be provided".

"this package" is the one I reference at [2].

AdaCore's approach is to build the compiler with a "system root" that references the SDK in situ; the actual link takes place with

/usr/bin/ld -syslibroot

/Library/Developer/CommandLineTools/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/

and, unfortunately for us, that's the full Xcode and not the CommandLineTools subset; so if you only have the CommandLineTools, ld looks for libSystem.dylib in a non-existent directory.

One approach is to build with -largs -Wl,-syslibroot,/

Another one is to install the full Xcode.

I guess Xcode is the way to go.

For the future

I don't think it's possible to have multiple syslibroots.

I don't think the GCC developers would be happy with building knowledge of xcode-select into the compiler, so it could make the same runtime choices as Apple tools.

Since the SDKs really only impact the includes, at any rate as long as you're on macOS and not iOS, I'm wondering whether it'd be possible to add both SDK include paths to GCC's include paths and avoid the syslibroot impact on libraries.

Nothing yet about this on the GCC mailing lists, that I can see.

[1] https://developer.apple.com/documentation/xcode_release_notes/xcode_10_release_notes#3035624

[2] https://forward-in-code.blogspot.com/2018/11/mojave-vs-gcc.html

From: Simon Wright <simon@pushface.org>
Date: Tue, 18 Jun 2019 18:00:52 +0100

I did something on this, written up here:

https://forward-in-code.blogspot.com/2019/06/macos-software-development-kit-changes.html

Ada in Genode OS

From: Kay-Uwe Genz <kug1977@web.de>
Subject: Genode OS Framework 19.05 goes SPARK
Date: Mon, 17 Jun 2019 03:43:40 -0700
Newsgroups: comp.lang.ada

you might be interested to see, that Genode OS Framework 19.05 is integrating Ada/SPARK runtime and SPARK-based cryptography

Spunky: A kernel using Ada - Part 1: RPC

For me these news were new.
https://www.osnews.com/story/130141/ada-spark-on-genode/

From: Kay-Uwe Genz <kug1977@web.de>
Date: Wed, 19 Jun 2019 07:30:34 -0700

> I don't understand why they put c++ on one end and SPARK on the other... Don't they know "normal" Ada includes quite enough "non-static" features? Or is that compatibility with existing libraries the problem? Not really said in that article.

Most of the L4 development which is where Genode OS Framwork came from is done in C++ and Ada/SPARK is more a hobbiest project, I guess. The Muen kernel is focussed 100% on Ada/SPARK.

Ada and other Languages

Specification/Body Separation in Ada

From: John Perry <john.perry@usm.edu>
Subject: Why .ads as well as .adb?
Date: Sat, 1 Jun 2019 17:48:16 -0700
Newsgroups: comp.lang.ada

I understand that Ada, like Modula-2 and Modula-3, and arguably like C++, requires a definition file (.ads) as well as an implementation file (.adb). With Oberon, Wirth moved away from definition files, using a symbol to indicate which module identifiers should be exported. (Someone else may have done this before him; it's just that I'm most familiar with this history.) Most languages I'm familiar with these days do something similar, either via public/private or some other mechanism.

As far as I can tell, though, Ada has stuck with the two separate files, rather than, say, generating an .ads from an .adb with export markup.

Is there a reason Ada hasn't moved to this simpler structure?

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Sun, 2 Jun 2019 07:42:58 +0200

[...]

One of the main (huge) benefits of Ada is in being able to use specifications even before the body exists. You can:

1) write the specification, compile it to make sure that it make sense

2) write the code that uses the specification, to make sure that the specification meets the needs of the using code

3) write the body, with the assurance that what you do is the right thing.

You can even add:

2.5) write a prototype body to check that the behaviour is correct, before writing the full body that meets all requirements.

[...]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sun, 2 Jun 2019 08:39:23 +0200

It is general design principle of separation specifications from implementations. [...] [It] has evident advantages for code base maintenance, team development, testing, separate compilation etc. BTW, you can stuff bodies and specifications in the same file. It is purely compiler's business. See gnatchop for GNAT. [...]

On 2019-06-02 02:48, John Perry wrote:

> As far as I can tell, though, Ada has stuck with the two separate files, rather than, say, generating an .ads from an .adb with export markup.

That is not possible. You cannot generate specification from implementation and conversely. In both cases there is additional information missing. It could be two different languages. Even in the languages that confuse these things, declarations have syntax different from definitions. [...]

From: Maciej Sobczak <see.my.homepage@gmail.com>
Date: Tue, 4 Jun 2019 01:03:26 -0700

[Written by J-P. Rosen <rosen@adalog.fr>]

> If you have a body with many subprograms, how can you tell which ones are intended to be exported, and which ones are private to the body?

By annotating them appropriately? Keywords "private" or "export" or similar are commonly used for this purpose.

Please note that your question could also refer to the concept of DLLs, which is not directly addressed by Ada (nor C++). Yet, somehow we do manage to solve this problem.

[...]

From: Keith Thompson <kst-u@mib.org>
Date: Mon, 03 Jun 2019 12:51:14 -0700

"Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de> writes:

> No. Specification describes a class of implementations. You cannot deduce class from its single member.

I suspect the point is that you *could* have an Ada-like language in which specifications could be unambiguously generated from implementations. You'd need some kind of additional annotation to specify whether a given declaration is to be exported.

You can't in Ada as it is, because Ada isn't designed that way.

[Editor's note: the following subthread discusses the readability concerns of separating specifications, but also that clarity and separation may be achieved not only via specifications.]

From: Brad Moore <bmoore.ada@gmail.com>
Date: Fri, 7 Jun 2019 07:10:11 -0700

On Friday, June 7, 2019 at 1:59:26 AM UTC-6, Maciej Sobczak wrote:

> 1. There *are* languages that don't use separate spec files. Java and Python are well known examples, representing both compiled and scripted approaches.

[...] I think it is a big mistake of languages that encourage the specification and implementation to be in the same source file, and very surprised to see that anyone would be arguing for that.

The separation of specification and implementation ties into the "separation of concerns" attributed to Dijkstra way back in 1974.

When wanting to make use of a 3rd party package in Ada, I value being able to generally understand how to use that package by looking at the specification without having to look at the implementation. You generally only need to look at the public part of a package specification, as you can rely on anything past that as being implementation details.

Even with C++, one cannot stop reading when you see a private: keyword in a class definition, because there can be many public and private sections in a class. You have to keep reading the class specification until to hit the end of the class specification, in case you missed more public parts.

[...]

> 2. Programs written in those languages do ***not*** need to be written in one giant file. Actually, Java is frequently criticized (it was even in this thread) for forcing the programmer to use too many (!) files. Even though it does not have separate specs.

Maybe Ada offers a benefit here. In languages like Java, there is a tendency to

want to put each class in a separate file. With Ada packages, it can make more sense to organize related types in the same package.

[Editor's note: another subthread explores the implications for "Programming in the large"]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 10 Jun 2019 17:07:38 -0500

[...]

"Maciej Sobczak" <see.my.homepage@gmail.com> wrote [...]

> What I don't accept is the religious attitude that Ada is the only language that got the software engineering right and (consequently) that everything else is broken.

The truth hurts. So far as I can tell, no other language has really tried to "get software engineering right". It's possible, of course, but everyone either is trying to graft engineering onto some preexisting base without it (C++, Java) or is building something that's more about fast construction than engineering (Python).

[...]

From: Optikos <optikos@verizon.net>
Date: Mon, 10 Jun 2019 17:32:36 -0700

There was only one other programming language that tried to "get software engineering right" and that achieved significant industrial usage and an open-source GCC compiler and that was ISO standardized: CHILL. While DoD & NATO were busy with their HOLWG effort for the military, ITU-T (in the United Nations) launched a somewhat competing effort for telecom in the EU (and AT&T steadfastly rejected both for the most part except for some monitoring of the 2 other efforts, so that AT&T pushed forward with C).

As can be seen in the following example CHILL source code, if Ada was envisioned as a Pascal/Wirth-esque-family language, CHILL was envisioned as a PL/1esque-family language. As such, Ada is beautiful & refined by comparison, whereas CHILL is rather abrupt & uncouth, as if it is most at home on an IBM mainframe with its fellow brethren CICS and JCL and of course PL/I. CHILL and Ada share many of the same goals and as such have some analogous language features that are absent in most other programming languages. Except for some maintenance of CHILL-based telecom equipment from Alcatel and Siemens, CHILL has become a dead language.

http://psc.informatik.uni-jena.de/languages/chill/chill.htm

[...]

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Tue, 11 Jun 2019 17:49:24 +0200

On 6/11/19 12:07 AM, Randy Brukardt wrote:

> So far as I can tell, no other language has really tried to "get software engineering right".

Precisely. It's important to remember that separation of spec and body have been part of Ada from the beginning, and Ada was designed to support the way S/W engineers think and work from the beginning. Like many Ada features, S/W engineers understand and like separation of spec and body, and coders don't. For me, much of this thread can be viewed simply as people saying "I'm a S/W engineer" or "I'm a coder".

From: John Perry <john.perry@usm.edu>
Date: Mon, 3 Jun 2019 06:37:43 -0700

Thanks to everyone for the replies. Personally, I find three of them especially compelling:

"As a teacher, I keep fighting with students who jump to writing bodies too early."

[I know exactly what this is like.]

"teams can work separated from each other as needed, without the project having to distribute all of the implementation to everyone"

[Having separate specification files against which one can *compile* would be useful, not just convenient, though I think it's arguable that one can do this in Oberon, too, via .smb files and documentation.]

"convenience"

[not a direct quote, but several people point to this, and until I read their explanations I thought the convenience ran in the other direction]

[...]

[Editor's note: the author proposes to follow-up with the impossibility of generating unambiguous specifications from bodies. If the conversation catches up, this will be reported in the next issue.]

Issues with Fortran Calling Convention

From: Chris M Moore <zmower@ntlworld.com>
Subject: Making the same mistake as the broken C interface to fortran
Date: Tue, 25 Jun 2019 00:33:39 +0100
Newsgroups: comp.lang.ada

Read this interesting article today:

https://lwn.net/SubscriberLink/791393/41d57555202e8cdb/

Synopsis: C interfaces to Fortran makes some assumptions about how to call fortran ABIs (I don't need to pass the hidden length parameter if it is a character*1) but now Gfortran has optimisations which assume a different calling convention (Thou shalt pass the hidden length).

There are work around (compile fortran with -fno-optimize-sibling-calls) but it seems that the proper fix is to pass the hidden length parameter.

I had a quick look at the LAPACK bindings and they both seem to use Ada characters. :/

[Editor's note: after some back and forth discussion, it seems Ada may be affected by the same issue. What follows is the last post in the thread with an Ada reproducer.]

From: Chris M Moore <zmower@ntlworld.com>
Date: Sun, 7 Jul 2019 17:33:46 +0100

I spoke too soon when I said

> I'm sure GNAT does the right thing if you're using Fortran_Character.

If I change callee.f to

```
subroutine callee (c)
character (len=*), intent (in) :: c
print *, 'parameter c is ', c
end
```

then STORAGE_ERROR is the order of the day no matter the call used. Looking at the assembler, this is because GNAT does not pass the length of the string.

I compared it to fcall.f:

```
program fcall
call callee("OK")
call callee("Oh noes")
stop
end
```

and this unsurprisingly does pass the lengths.

I've used the webform on the Community section of the GNAT website to provide feedback. I've pointed out that the issue also affects single character parameters.

Ada Practice

References vs. Access Types

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Subject: References vs access types
Date: Fri, 31 May 2019 17:44:34 +0200
Newsgroups: comp.lang.ada

So, part of the point of reference types is to be able to return an item "by reference" without being able to store the pointer:

```
type Item;
type Item_Access is access Item;

type Reference (Ptr : access Item) is
      limited null record;
function Get (...) return Reference; -- (1)
```

In Gem #107 this is said as advantageous against, for example,

```
function Get (...) return Item_Access;
      -- (2)
```

because "access discriminants are unchangeable. The discriminant also cannot be copied to a variable [like Item_Access]" [1].

Now, without thinking much about it, while fighting old bugs, I have sometimes replaced a problematic Reference with

```
function Get (...) return access Item;
      -- (3)
```

And here comes the question: besides losing the ability to use aspects on the Reference type, or using it for some fancy refcounting, does (3) give the same safeties wrt to copying as (1)? Are there any other hidden traps in (3) (assuming the pointee thread-safety/lifetime is properly managed)?

Or, put it another way, is (1) always preferable? Or may (3) suffice for simple uses?

[1] https://www.adacore.com/gems/gem-107-preventing-deallocation-for-reference-counted-types/

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 31 May 2019 18:55:53 +0200

My preferences list would be:

#1 - Never, visually ugly, semantically questionable, lacking transparent access to the target object and technically not a reference at all, plus unstable with GNAT compilers.

#2 - Construction of new stand-alone objects (frequently class-wide), implementation-dependent stuff.

#3 - Access to a component of an existing object.

As for hidden traps, only #3 is safe upon inheritance, if primitive operation and thus covariant.

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Fri, 31 May 2019 16:55:01 -0700

I'm quite opposed to Dmitry ['s statement about #1].

I admit that #1 is clumsy. But see Gem 123 to learn how this syntax may be improved with some aspects.

(Compiler problems are never an argument to avoid some feature forever.)

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 31 May 2019 16:33:18 -0500

(1) and (3) have the same accessibility rules, so they have the same safety from copying (no more and no less). However, since (3) is returning an access type, one can directly assign the function result into an access type, and that will work as the function will then have the accessibility of the access value. (But of course, you might get an accessibility failure inside the function in that case.)

An important part of the reference mechanism is the use of aliased parameters. For a function, those are required to have the same accessibility as the function result. This makes most problematic calls illegal. For instance, in:

```
function Get (Obj : aliased in out
      Some_Type) return access
      Some_Other_Type;

Ptr : Some_Access_Type;

procedure Whatever is
  Local: Some_Type;
begin
  Ptr := Get (Local); -- Illegal.
  Get (Local).all := ...;
end Whatever;
```

The first call to Get here is illegal as the actual parameter is more nested than the level of the function call (which is that of Ptr). This prevents Get from keeping a pointer longer than the object exists. The second call to Get is legal because the level of that call is local, and therefore the object lives long enough.

Create and Append_File

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Subject: Create and Append_File
Date: Thu, 6 Jun 2019 22:45:09 +0200
Newsgroups: comp.lang.ada

You can call Create with mode Append_File. I'm trying to figure out what that's supposed to do (as opposed to what compilers do). I've read ARM A.7, A.8.2, and A.10.2, and am still not sure.

It seems there are 2 likely interpretations:

1. Create creates a file, so this is the same as using mode Out_File
2. Since mode Append_File was given, it means to open the file in append mode if it exists, or create it as for mode Out_File if it doesn't

If 1., then why allow Append_File for Create? A subtype excluding it could be defined for Create.

Of course, you can also Create a file with mode In_File, which I presume means to create an empty file and open it for reading, which doesn't seem very useful, so maybe I shouldn't expect these to make sense.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 6 Jun 2019 16:24:52 -0500

I believe it means the same as Out_File. Other requirements in RM (not very clear ones, I'm afraid) require the file opened by Create to be empty, whether or not the file previously existed. So, if Create allows (re)creating an existing file (it doesn't have to, it could raise Use_Error), that file will be empty. In that case, Out_File and Append_File are the same.

As you note, Create (In_File) is already nonsense, so Create (Append_File) might as well be nonsense as well (it's ***less*** nonsense in any case, since a modeless Reset preserves the mode, and the file wouldn't necessarily be empty at that point).

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Fri, 7 Jun 2019 17:59:39 +0200

On 6/7/19 10:01 AM, Simon Wright wrote:

> I guess I should add this to my StackOverflow answer which may have been the trigger for this question. I have No Idea why I thought it sensible to Create the file in Append_File mode.

Yes, I saw Create with Append_File and wondered what that should do. It seemed reasonable that it would open the file in append mode if it existed, and create it in output mode otherwise, but that's not what GNAT does, so here we are.

Conventions Applied to Entity Views

From: Jere <jhb.chat@gmail.com>
Subject: Convention Question related to access types
Date: Thu, 6 Jun 2019 18:51:29 -0700
Newsgroups: comp.lang.ada

The RM in section B.1 talks about Ada Standard requirements for convention compatibility. In it however it doesn't mention anything about private types, full views, etc.

Say you are wanting to bind to an opaque type in C:

```
package Bindings is
   type Opaque_Type(<>) is limited
                           private;
   type Binding is access Opaque_Type
         with Convention => C;

   procedure Some_Procedure(
   Value : Binding)  with Import,
   Convention => C;

private

   type Opaque_Base is limited null
        record with Convention => C;
   type Opaque_Type is new
        Opaque_Base;

end Bindings;
```

GNAT happily accepts that, but I am unsure if that is because of the "The implementation permits T as an L-compatible type." part or because Opaque_Base is a proper convention compatible type and Opaque_Type derives from it and is thus convention

compatible as well, even though it is a private type.

I couldn't find anything dictating whether the convention compatibility rules applied to the full view or the public view.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sat, 8 Jun 2019 00:11:08 -0500

Conventions apply to *entities*. See 6.3.1(2/1): "a convention can be specified for an entity". Views like a partial view is *of* an entity, not an entity itself. Thus there is only a single convention for a type. Where it is specified doesn't matter outside of Legality Rules. Thus the rules in B.1 only need to talk about types, not views.

I just had this argument about "entity" with other ARG members vis-a-vis a different topic (I lost :-), so I'm very certain this is correct.