ADA USER JOURNAL

Volume 44
Number 1
March 2023

Contents

Page

Editorial Policy for Ada User Journal 2
Editorial 3
Quarterly News Digest 4
Conference Calendar 25
Forthcoming Events 33
Proceedings of the "HILT'22 - Supporting a Rigorous Approach to Software Development Workshop"
H. Ausden. *"Achieving 100% Availability in the ERAM Air Traffic Control System"* 35
D. Hardin. *"Hardware/Software Co-assurance for the Rust Programming Language Applied to Zero Trust Architecture Development"* 38
D. Larraz, A. Viswanathan, C. Tinelli, M. Laurent. *"Beyond Model Checking of Idealized Lustre in KIND 2"* 44
J. Hatcliff, D. Stewart, J. Belt, F. Robby, A. Schwerdfeger. *"An AADL Contract Language Supporting Integrated Model- and Code-Level Verification"* 49
T. Bordis, T. Runge, A. Kittelmann, I. Schaefer. *"Correctness-by-Construction: An Overview of the CorC Ecosystem"* 59
Proceedings of the "ADEPT: AADL by its practitioners Workshop" of AEiC 2022
J. C. Roger, P. Dissaux. *"AADL Modelling with SysML v2"* 63
X. Xu, S. Wang, B. Zhan, X. Jin, N. Zhan, J-P. Talpin. *"Unified Graphical Co-modeling, Analysis and Verification of Cyber-physical Systems by Combining AADL and Simulink/Stateflow"* 67
Z. Yang, Z. Qiu, Y. Zhou, Z. Huang, J-P. Bodeveix, M. Filali. *"C2AADL_Reverse: A Model-Driven Reverse Engineering Approach for Development and Verification of Safety-Critical Software"* 71
A. Bombardelli, A. Bonizzi, M. Bozzano, R. Cavada, A. Cimatti, A. Griggio, M. Nazaria, E. Nicolodi, S. Tonetta, G. Zampedri. *"COMPASTA: Integrating COMPASS Functionality into TASTE"* 75
P. Denzler, D. Ramsauer, D. Scheuchenstuhl, W. Kastner. *"Experiences Modeling a OPC UA / DDS Gateway in AADL in the Context of Fog Computing"* 79
G. Bardaro, M. Matteucci. *"Modelling Robot Architectures with AADL"* 80
E. Senn, L. W. J. Bourdon. *"Modeling ROS Based Applications with AADL"* 84
D. Blouin, P. Crisafulli, C. Maxim, F. Caron. *"An Introduction to ALISA and Its Usage for an Industrial Railway System Case Study"* 88
Ada-Europe Associate Members (National Ada Organizations) 92
Ada-Europe Sponsors Inside Back Cover

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

Preface by the News Editor 4
Ada-related Events 4
Ada and Education 7
Ada-related Resources 7
Ada-related Tools 7
Ada and Other Languages 8
Ada Practice 11
Ada in Jest 24

[Messages without subject/newsgroups are replies from the same thread. Messages may have been edited for minor proofreading fixes. Quotations are trimmed where deemed too broad. Sender's signatures are omitted as a general rule. —arm]

Preface by the News Editor

Dear Reader,

We are fast approaching the main Ada-Europe yearly event, the AEiC conference. You can get the gist of this year's plans in this post: [1].

For the technically minded, several intricate aspects of the language are discussed in, e.g., [2] and [3]. It seems one never stops learning new details about Ada!

Finally, from the 'Ada-not-the-language' department (usually very quiet), several resources that promise lots of fun: An opera based on the wonderful work of Sydney Padua, the steampunk comic where Ada Lovelace and Charles Babbage team up to fight crime (!) is in the making [4]. Based on the same duo of scientists, you can already get your hands on an educational tabletop game [5]. Lastly, you may want to check an impressive Ada Lovelace cosplay sewn entirely from scratch [6].

[1] "AEiC 2023 - Ada-Europe Conference - Final Deadline Approaching", in Ada-related Events.

[2] "Ada Array Contiguity", in Ada Practice.

[3] "Assignment Access Type with Discriminants", in Ada Practice.

[4] "Babbage & Lovelace - The Opera", in Ada-related Events.

[5] "Table Game", in Ada and Education.

[6] "Ada Lovelace Cosplay", in Ada in Jest.

Sincerely,
Alejandro R. Mosteo.

Ada-related Events

Babbage & Lovelace - The Opera

From: Simon Wright <simon@pushface.org>
Subject: Babbage & Lovelace - The Opera
Date: Mon, 16 Jan 2023 18:30:56 +0000
Newsgroups: comp.lang.ada

https://guerillaopera.org/repertoire/thrilling-adventures

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Thu, 19 Jan 2023 12:17:08 -0000

> https://guerillaopera.org/repertoire/thrilling-adventures

As I reacted on Twitter when I saw Sydney Padua @sydneypadua announcing this opera adaption of her graphic novel "The Thrilling Adventures of Lovelace and Babbage":

-------start-quote-------

The late Robert Dewar, of #AdaProgramming language fame, would have loved this. For some history, look for "The Maiden and the Mandate" in https://ada-europe.org/archive/auj/auj-41-1-withcovers.pdf and https://adacore.com/adacore25...

-------end-quote-------

Those were hilariously funny fully staged musical performances at several ACM SIGAda and Ada-Europe conferences, which I was lucky enough to attend. It would be great if AdaCore could put online one of the video recordings that were made at the time.

Dirk

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

* 27th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2023)

* June 13-16, 2023, Lisbon, Portugal, www.ada-europe.org/conference2023

Ada Stand at FOSDEM 2023

[Past event for the record. —arm]

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada Stand at FOSDEM 2023 - Sat 4 & Sun 5 Feb (was: No Ada DevRoom in FOSDEM 2023, alternative DevRooms and Ada-Europe) support
Date: Thu, 2 Feb 2023 13:13:26 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Reminder: FOSDEM 2023 takes place this weekend, Sat 4 and Sun 5, in Brussels, Belgium. See www.fosdem.org.

Even though we didn't manage to get an Ada DevRoom this year […], the Ada FOSDEM team has an Ada stand in the "Education" group on level 2 of building K at the ULB site, with theme "It's time to learn Ada!"

Looking forward to meet many Adaists!

Dirk Craeynest, Ada FOSDEM team

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

AEiC 2023 - Ada-Europe Conference - Final Deadline Approaching

[For the record, as the deadline is past. —arm]

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: AEiC 2023 - Ada-Europe conference - Final Deadline Approaching
Date: Thu, 16 Feb 2023 09:39:33 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

FINAL Call for Contributions

27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023)

13-16 June 2023, Lisbon, Portugal

www.ada-europe.org/conference2023

*** FINAL submission DEADLINE 27 February 2023 ***

Organized by Ada-Europe in cooperation with ACM SIGAda (approval pending) and the Ada Resource Association (ARA)

#AEiC2023 #AdaEurope #AdaProgramming

General Information

The 27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023) will take place in Lisbon, Portugal. The conference schedule comprises a journal track, an industrial track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops.

* Journal-track submissions present research advances supported by solid theoretical foundation and thorough evaluation.

* Industrial-track submissions highlight the practitioners' side of a challenging case study or industrial project.

* Work-in-progress-track submissions illustrate a novel research idea that is still at an initial stage, between conception and first prototype.

* Tutorial submissions guide attenders through a hands-on familiarization with innovative developments or with useful features related to reliable software.

Schedule

[CLOSED] Extended submission deadline for journal-track papers

27 February 2023: Submission deadline for industrial-track and work-in-progress-track papers, tutorial & workshop proposals

20 March 2023: First round notification for journal-track papers, acceptance notification for other submission types

13-16 June 2023: Conference

Scope and Topics

The conference is a leading international forum for providers, practitioners, and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development, and maintenance of long-lived, high-quality software systems for a challenging variety of application domains. The program will allow ample time for keynotes, Q&A sessions and discussions, and social events. Participants include practitioners and researchers from industry, academia, and government organizations active in the promotion and development of reliable software technologies.

The topics of interest for the conference include but are not limited to:

- Formal and Model-Based Engineering of Critical Systems;
- Real-Time Systems;
- High-Integrity Systems and Reliability;
- Ada Language;
- Applications in a variety of domains.

More specific topics are described on the conference web page.

Call for Journal-track Submissions

Following a journal-first model, this edition of the conference again includes a journal track, which seeks original and high-quality papers that describe mature research work on the conference topics. Accepted journal-track papers will be published in the "Reliable Software Technologies (AEiC2023)" Special Issue of JSA -- the Journal of Systems Architecture (Scimago Q1 ranked, impact factor 5.936).

[Submission details removed. Call is closed now.]

Authors who have successfully passed the first round of review will be invited to present their work at the conference. Please note that the AEiC 2023 organization committee will waive the Open Access fees for the first four accepted papers, which do not already enjoy OA from personalized bilateral agreements with the Publisher. Subsequent papers will follow JSA regular publishing track.

Call for Industrial-track Submissions

The conference seeks industrial practitioner presentations that deliver insight on the challenges of developing reliable software. Especially welcome kinds of submissions are listed on the conference web site. Given their applied nature, such contributions will be subject to a dedicated practitioner-peer review process. Interested authors shall submit a one-to-two pages abstract, by 27 February 2023, via EasyChair at https://easychair.org/my/conference?conf=aeic2023, selecting the "Industrial Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstract of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be invited to expand their contributions into full-fledged articles for publication in the Ada User Journal, which will form the proceedings of the industrial track of the Conference. Prospective authors may direct all enquiries regarding this track to its chairs Alexandre Skrzyniarz (alexandre.skrzyniarz at fr.thalesgroup.com) and Sara Royuela (sara.royuela at bsc.es).

Call for Work-in-Progress-track Submissions

The work-in-progress track seeks two kinds of submissions: (a) ongoing research and (b) early-stage ideas. Ongoing research submissions are 4-page papers describing research results that are not mature enough to be submitted to the journal track. Early-stage ideas are 1-page papers that pitch new research directions that fall within the scope of the conference. Both kinds of submissions must be original and shall undergo anonymous peer review. Submissions by recent MSc graduates and PhD students are especially sought. Authors shall submit their work by 27 February 2023, via EasyChair at https://easychair.org/my/conference?conf=aeic2023, selecting the "Work in Progress Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstract of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be offered the opportunity to expand their contributions into 4-page articles for publication in the Ada User Journal, which will form the proceedings of the WiP track of the Conference. Prospective authors may direct all enquiries regarding this track to the corresponding chairs Bjorn Andersson (baandersson at sei.cmu.edu) and José Cecílio (jmcecilio at fc.ul.pt).

Awards

Ada-Europe will offer an honorary award for the best technical presentation, to be announced in the closing session of the conference.

Call for Tutorials

The conference seeks tutorials in the form of educational seminars on themes falling within the conference scope, with an academic or practitioner slant, including hands-on or practical elements. Tutorial proposals shall include a title, an abstract, a description of the topic, an outline of the presentation, the proposed duration (half-day or full-day), the intended level of the contents (introductory, intermediate, or advanced), and a statement motivating attendance. Tutorial proposals shall be submitted by e-mail to Tutorial and Education Chair, Luís Miguel Pinho (lmp at isep.ipp.pt), with subject line: "[AEiC 2023: tutorial proposal]". Tutorial proposals shall be submitted by 27 February 2023. The authors of accepted full-day tutorials will receive a complimentary conference registration, halved for half-day tutorials. The Ada User Journal will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

The conference welcomes satellite workshops centred on themes that fall within the conference scope. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the AEiC conference. Workshop organizers shall also commit to producing the proceedings of the event, for publication in the Ada User Journal. Workshop proposals shall be submitted by e-mail to the Workshop Chair, Frank Singhoff (singhoff at univ-brest.fr), with subject line: "[AEiC 2023: workshop proposal]". Workshop proposals shall be submitted at any time but no later than the 27 February 2023. Once submitted, each workshop proposal will be evaluated by the conference organizers as soon as possible.

Call for Exhibitors

The conference will include a vendor and technology exhibition. Interested providers should direct inquiries to the Exhibition & Sponsorship Chair, Ahlan Marriott (ahlan at Ada-Switzerland.ch).

Venue

The conference will take place at the Hotel Fénix Lisboa, near downtown Lisbon, Portugal. June is full of events in Lisbon, including the festivities in honour of St. António (June 13 is the town holiday), with music, grilled sardines, and popular parties in Alfama and Bairro Alto neighbourhoods. There's plenty to see and visit in Lisbon, so plan in advance!

Organizing Committee

- Conference Chair

António Casimiro,
University of Lisbon, Portugal
casim at ciencias.ulisboa.pt

- Journal-track Chair

Elena Troubitsyna,
KTH Royal Inst. of Technology, Sweden
elenatro at kth.se

- Industrial-track Chairs

Alexandre Skrzyniarz,
Thales, France
alexandre.skrzyniarz at fr.thalesgroup.com

Sara Royuela,
Barcelona Supercomputing Center, Spain
sara.royuela at bsc.es

- Work-In-Progress-track Chairs

Bjorn Andersson,
Carnegie Mellon University, USA
baandersson at sei.cmu.edu

José Cecílio,
University of Lisbon, Portugal
jmcecilio at fc.ul.pt

- Tutorial and Education Chair

Luis Miguel Pinho,
ISEP, Portugal
lmp at isep.ipp.pt

- Workshop Chair

Frank Singhoff,
University of Brest, France
singhoff at univ-brest.fr

- Exhibition & Sponsorship Chair

Ahlan Marriott,
White Elephant GmbH, Switzerland
ahlan at Ada-Switzerland.ch

- Publicity Chair

Dirk Craeynest,
Ada-Belgium & KU Leuven, Belgium
Dirk.Craeynest at cs.kuleuven.be

- Webmaster

Hai Nam Tran,
University of Brest, France
hai-nam.tran at univ-brest.fr

Previous Editions

Ada-Europe organizes annual international conferences since the early 80's. This is the 27th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08), Brest, France ('09), Valencia, Spain ('10), Edinburgh, UK ('11), Stockholm, Sweden ('12), Berlin, Germany ('13), Paris, France ('14), Madrid, Spain ('15), Pisa, Italy ('16), Vienna, Austria ('17), Lisbon, Portugal ('18), Warsaw, Poland ('19), online from Santander, Spain ('21), and Ghent, Belgium ('22).

Information on previous editions of the conference can be found at

http://www.ada-europe.org/confs/ae.

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2023 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 27th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2023)
* June 13-16, 2023, Lisbon, Portugal, www.ada-europe.org/conference2023

(V3.1)

Ada-Europe Conference - 6 March Extended Final Deadline

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Conference - 6 March Extended Final Deadline
Date: Sat, 25 Feb 2023 14:33:00 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

The recently posted reminder for the Ada-Europe 2023 Conference triggered several requests for extra time. To give all authors the same opportunity to further refine their submission, the organizers decided that the deadline for industrial- and work-in-progress-track abstracts, and for tutorial and workshop proposals will be extended by 1 week until Monday, 6 March 2023. 1+ week remains!

FINAL UPDATED Call for Contributions

27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023)

13-16 June 2023, Lisbon, Portugal

*** EXTENDED FINAL submission DEADLINE 6 March 2023 ***

Industrial- and Work-in-Progress-track: submit via
https://easychair.org/my/conference?conf=aeic2023
select "Industrial Track" or "Work in Progress Track"

Tutorials: submit to Tutorial and Education Chair,
Luís Miguel Pinho <lmp @ isep.ipp.pt>
subject "[AEiC 2023: tutorial proposal]"

Workshops: submit to Workshop Chair, Frank Singhoff
<singhoff @ univ-brest.fr>
subject "[AEiC 2023: workshop proposal]"

For more information please see the full Call for Papers at
www.ada-europe.org/conference2023

#AEiC2023 #AdaEurope #AdaProgramming

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2023 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 27th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2023)*
* June 13-16, 2023, Lisbon, Portugal, www.ada-europe.org/conference2023

(V4.1)

Ada and Education

Table Game

From: Mockturtle <framefritti@gmail.com>
Subject: Table game
Date: Tue, 17 Jan 2023 05:56:31 -0800
Newsgroups: comp.lang.ada

Really?!?

https://www.amazon.com/Artana-AAX14001-Lovelace-Babbage/dp/B07WHMG5Y8

[The link is for a tabletop game with the blurb "Play as a pioneer of early computing, like Ada Lovelace or Charles Babbage, to build a program that solves problems for famous patrons like Charles Darwin, Mary Shelley, and more!". It is priced at 19.98$ and has 4.5/5 stars rating with 50 reviews at the time of this writing. —arm]

Ada-related Resources

[Delta counts are from February 12th to April 5th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 5 Apr 2023 17:36 CET[b]
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 8_349 (+58) members [1]
- LinkedIn: 3_436 (+18) members [2]
- Stack Overflow: 2_323 (+14) questions [3]
- Telegram: 160 (+1) users [4]
- Gitter: 219 (+68*) people [5]
- Ada-lang.io: 107 (+6) users [6]
- Libera.Chat: 74 (-8) concurrent users [7]
- Twitter: 22 (-10) tweeters [8]
  44 (-5) unique tweets [8]

* Gitter has migrated its messaging to the Matrix open standard. The [5] reference has been updated accordingly.

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

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

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

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

[5] https://app.gitter.im/#/room/#ada-lang_Lobby:gitter.im

[6] https://forum.ada-lang.io/u

[7] https://netsplit.de/channels/details.php?room=%23ada&net=Libera.Chat

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 5 Apr 2023 17:45 CET[c]
To: Ada User Journal readership

Rosetta Code: 924 (+4) examples [1]
40 (+1) developers [2]

GitHub: 763* (=) developers [3]

Alire: 337 (+13) crates [4]

Sourceforge: 240 (=) projects [5]

Open Hub: 214 (=) projects [6]

Codelabs: 54 (=) repositories [7]

Bitbucket: 31 (=) repositories [8]

* This number is unreliable due to GitHub search limitations.

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

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

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

[4] https://alire.ada.dev/crates.html

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: 5 Apr 2023 17:36 CET[d]
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. —arm]

- TIOBE Index: 28 (-5) 0.42% (-0.18%) [1]
- PYPL Index: 19 (-2) 0.83% (-0.11%) [2]
- IEEE Spectrum (general): 35 (=) Score: 1.16 [3]
- IEEE Spectrum (jobs): 33 (=) Score: 0.79 [3]
- IEEE Spectrum (trending): 32 (=) Score: 3.95 [3]

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

[2] http://pypl.github.io/PYPL.html

[3] https://spectrum.ieee.org/top-programming-languages/

Ada-related Tools

Embedded AVR Ada Setup - Linux Edition

From: Stéphane Rivière <stef@genesix.org>
Subject: ANN: Embedded AVR Ada Setup - Linux edition
Date: Thu, 12 Jan 2023 11:21:30 +0100
Newsgroups: comp.lang.ada

Hi all,

Embedded AVR Ada Setup - Linux edition.

Thanks to the work of Rolf Ebert (AVR-Ada and AVR-Ada to Alire conversion), Fabien Chouteau and AdaCore (GNAT-AVR, GNAT-AVR to Alire conversion, Alire promotion) and their friendly help, here is a tutorial to get the most pleasant environment to develop in Ada on 8-bit AVR targets under Linux.

Based on Alire and GNAT Studio 23 it allows real-time debugging in GNAT Studio as if you were in a native X86_64 environment.

This was an opportunity to get acquainted with Alire while keeping our usual GNAT Studio based environment, which integrates perfectly with Alire. Thanks to the author Alejandro R. Mosteo, who also wrote a very interesting presentation of Alire in AUJ Vol 39, Number 3, Sept 2018, P 189.

This work is part of a more general desire to empower the Ada community with respect to the defunct GNAT CE. We therefore adhere to this new policy of Adacore. Between this new direction, the arrival of Alire, the availability of many Crates, the first successes of the community in building GNAT Studio independently, the arrival of Rust which is good for the visibility of our favorite language, Ada is certainly entering a new era :)

https://github.com/sowebio/adam-doc (GNAT Studio & project example additional files)

https://github.com/sowebio/adam-doc/blob/master/Ada%20Development%20on%20AVR%20Microcontroller.pdf

Feedback and criticism are welcome.

Short Video on Getting Started with GtkAda in 2023

From: Stephen Merrony <merrony@gmail.com>
Subject: A Short Video on Getting Started with GtkAda in 2023
Date: Sat, 14 Jan 2023 01:01:13 -0800
Newsgroups: comp.lang.ada

I made a quick video showing how easy it is to get started writing a Gtk application in Ada these days...

https://youtu.be/IofrV5hsUvg

[Video running time is 11:03 minutes. —arm]

Gnu Emacs Ada Mode 8.0.4 Released

From: Stephen Leake stephen.leake84@gmail.com
Subject: Gnu Emacs Ada mode 8.0.4 released.
Date: Wed, 25 Jan 2023 05:27:57 -0800
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 8.0.4 is now available in GNU ELPA.

All Ada mode executables can now be built with Alire (https://alire.ada.dev/); this greatly simplifies that process.

gpr-query and gpr-mode are split out into separate GNU ELPA packages. You must install them separately (Emacs install-package doesn't support "recommended packages" like Debian does).

Ada mode can now be used with Eglot; this is controlled by new variables:

ada-diagnostics-backend - one of wisi, eglot, none

ada-face-backend - one of wisi, eglot, none

ada-indent-backend - one of wisi, eglot, none

ada-statement-backend - one of wisi, eglot, none

ada-xref-backend - one of GNAT, gpr_query, eglot, none

The diagnostic, face, indent, and statement backends default to wisi if the wisi parser is found in PATH, to eglot if the Ada LSP server is found, and none otherwise. The xref backend defaults to gpr_query if the gpr_query executable in PATH, to GNAT otherwise.

ada-diagnostics-backend controls the source of compilation error messages while editing.

ada-statement-backend controls statement motion; forward-sexp, wisi-goto-statement-end, etc. ada-xref-backend controls wisi-goto-spec/body and Emacs xref commands.

In addition, name completion is provided by eglot if any of the other backends are using eglot; eglot completion is always better than wisi.

The current AdaCore language server (version 23) supports face but not indent. The current version of eglot (1.10) does not support face. The Language Server Protocol does not support statement motion. So for now, eglot + ada_language_server only provides xref and completion.

The AdaCore language server ada_language_server is installed with GNATStudio (which ada-mode will find by default), or can be built with Alire. If you build it with Alire, either put it in PATH, or set gnat-lsp-server-exec.

I have not tested ada-mode with lsp-mode. You can set ada-*-backend to 'other to experiment with that, or tree-sitter, or some other backend. tree-sitter will be fully supported in the next ada-mode release.

The required Ada code requires a manual compile step, after the normal list-packages installation:

```
cd ~/.emacs.d/elpa/ada-mode-7.3beta*

./build.sh

./install.sh
```

If you have Alire installed, these scripts use it.

Ada and Other Languages

Carbon New Language

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Subject: Carbon
Date: Fri, 22 Jul 2022 14:13:08 -0700
Newsgroups: comp.lang.ada

[This thread is a bit dated as it was deemed less of a priority due to space constraints in past issues. —arm]

Next attempt to replace C/C++ without really replacing it: Carbon!

You will notice, as usual, a few aspects borrowed from Ada - and one point inspired by Ada 83 (which was relaxed in a later Ada version) :-)

https://mybroadband.co.za/news/software/453410-googles-carbon-programming-language-aims-to-replace-c.html

https://devclass.com/2022/07/20/google-brands-carbon-language-as-experimental-successor-to-c/

https://9to5google.com/2022/07/19/carbon-programming-language-google-cpp/

https://thenewstack.io/google-launches-carbon-an-experimental-replacement-for-c/

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Sat, 23 Jul 2022 09:09:57 -0000

I read that stuff yesterday and, yet again, shook my head in disbelief :-(

The bit where I laughed was where it was claimed that C++ is building technical debt because it's not changing quickly enough; C++ is currently a mess because it's changing too quickly! Half-baked, and half-implemented ideas are going into 'standards' in the full knowledge that they'll change again in the next one. Even g++ doesn't provide 100% support for C++17 (https://gcc.gnu.org/projects/cxx-status.html#cxx17)!

Carbon is likely to be even worse; every 'new' language that promises the earth, without being designed in a rigorous way, ends up with the same problems. Java - I started playing with that in the 90s and got frustrated that every update brought more and more depreciation warnings in. Python - 2.x -> 3.0 was a massive jump (and took years to gain traction) because the 'designers' just hadn't done a very good job to start with! Rust? Mmm

As for the 'reuse C++ syntax'; why the obsession with that? C++ syntax is really bad! (Semantics, in some cases, are another level - how many languages need a book like "C++ Gotchas"?!).

Aaaaarrrrgghhh!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 23 Jul 2022 15:14:15 +0200

> Next attempt to replace C/C++ without really replacing it: Carbon!

We have just learned how dangerous carbon is for our climate. Yet these few privileged keep on pumping it up! (:-))

From: Stéphane Rivière <stef@genesix.org>
Date: Sat, 23 Jul 2022 15:49:05 +0200

> We have just learned how dangerous carbon is for our climate. Yet these few privileged keep on pumping it up! (:-))

Carbon language bad, green language good

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 24 Jul 2022 10:38:58 +0100

> Next attempt to replace C/C++ without really replacing it: Carbon!

Saw this last week and immediately thought they'd failed on one of their "design goals," i.e. to be "readable".

> You will notice, as usual, a few aspects borrowed from Ada - and one point inspired by Ada 83 (which was relaxed in a later Ada version) :-)

What did they take from Ada?

From: John Mccabe <john@mccabe.org.uk>
Date: Tue, 26 Jul 2022 10:31:42 -0700

> What did they take from Ada?

Certainly not the approach to making life easier and less error-prone for developers.

I've got involved in a couple of discussions on their forum, and I'm inclined to think they just want C++ but taken out of the control of ISO/IEC WGs steering committees.

They're pretty much not considering changing any of the aspects of C++ that make it such a heap of junk (IMO, of course), including, but not limited to:

1. arrays
2. enums
3. (both of the above when used together :-))
4. symbols - overuse, duplication, inconsistency
5. implicit stuff
6. pretend strong typing
7. forcing developers to deal manually with numeric values that don't fit into an n-byte range, where n is a whole number

It really is shockingly soul-destroying watching all that. What's worse is that, from what I've seen over the years, the new languages that have been developed in a more 'relaxed' way than Ada (well, evolved, really, like Java, Python etc) and have become relatively successful have taken a good 10 years or so to get to that point, yet the discussions on the Carbon forum are all about how to appeal to _current_ developers who're used to C++; not _future_ developers who, ideally, would _never_ be used to C++!

From: Nasser M. Abbasi <nma@12000.org>
Date: Thu, 28 Jul 2022 18:48:49 -0500

Since Ada has solved these problems a long time ago, then why are people still reinventing the wheel? Why are they not just using Ada? Ada is free software.

Maybe there is something in Ada that prevents it from being widely adopted and used? [...]

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Fri, 29 Jul 2022 11:03:36 -0000

> why are people still reinventing the wheel?

Possibly for the same reason that I was so anti-Ada in my early years; it takes getting used to and people are lazy.

Looking at some of the languages that have come out in recent years, it's obvious that people can't be bothered to type much; "fn"/"def" (or, even, nothing!) instead of "function"/"procedure", "{"/"}" instead of "begin"/"end", "&&" instead of "and", "||" instead of "or" (!!!) etc.

From what I can see, some of the "moderators" on that Carbon group don't have much real professional software development experience, so I suspect they really have no clue about what they could achieve with Ada, and have little understanding of some of the constraints that embedded, especially bare-metal, systems impose on what you can and can't include in a program. I'm thinking here of things like heap-unfriendly container classes, such as (in Swift) arrays that are automatically expandable when you append a new item, rather than being fixed size etc.

There also seems to be a bit of an obsession with the time between "empty editor window" and "executable available", rather than "empty editor window" and "executable that actually does what you want"!

Also, as Devin says, compiler availability is an issue, from the point of view of actually _using_ Ada.

However, from the point of view of creating a new language, the fact that so many people clearly think it _has_ to be the C/C++ way is quite disturbing, especially since, as I think I mentioned, it's going to be a number of years until any new language really makes its mark, so new languages should be taking future developers into account, not just pandering to the laziness of existing ones!

At this point I think I should make it clear that, although I think Ada has some great features (and I regularly espouse them amongst my colleagues), I don't use Ada in the software I'm developing. I'd like to, but it would take me a lot of time to get back to a level in Ada where I'd be comfortable creating a relatively substantial codebase from scratch. The alternative would be to go and join a team that's already using Ada, but every Ada job I've seen come up locally is to support code that was written in Ada 95; I'd rather be looking at Ada 2005 -> if I was to make that jump.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Fri, 29 Jul 2022 11:59:21 -0700

[...] IMHO the only way to make Ada more popular is to create popular applications with it.

From: Dennis Knorr <dennis.knorr@gmx.net>
Date: Sat, 6 Aug 2022 16:18:12 +0200

> Maybe there is something in Ada that prevents it from being widely adopted and used?

An opinion from a bystander who wants to like Ada, this is only after I looked the resources and the community up a bit two years ago again. you do not have to agree, it's just my experience and sometimes gut feeling.

* Bad to no marketing
* sometimes elitism by members of the community/Ada fans
* no modern feeling toolchain (Even Lazarus+Pascal or Gambas has a more modern feeling toolchain, and that says a lot)
* not much free software built with it
* not much free software for the toolchain available
* not much libraries which are ready and easy to use as a beginner
* no modern/up2date books and articles (especially in other languages than in English) seem to be available.
* the free Ada Compiler seems slow and a while back it generated relatively big binaries and the result was not very fast.

Just a few concrete examples to back that up:

* Is there a web playground or repl shell trying or learning/trying Ada or some of its prominent modules?
* There's no modern book in German about modern Ada and its libraries
* There's no syntax highlighting package in vim for Ada
* No exercises like for example Ruby Koans
* It *Looks* like there are no libraries which make it easy use Ada for programming (think json/document formats, http/mail/mime protocols, algorithms or cryptography libraries)

I know there are libraries out there, but they are hard to find, not promoted/marketed and I saw developers (also in other languages, I admit that) talking like, if you do not understand it, you should go back to toy languages like python.

I also know that not all bullet points above are really true to the fullest, but most of them from the outside look like it and also have at least some grain of truth in there.

If someone would write a book in German, how to write Ada and use $cryptolibrary, $networklibrary and how to integrate it in one's favorite development software, this surely would be very interesting to many.

The ONLY thing where I see Ada Marketing in the free software world is FOSDEM. But it is in its own Room. Ada people would need to go out and say: hey, look we also can do good stuff, look, an https server with letsencrypt support with library in 30 lines.

To be honest, I am curious how the community here will react to it. I mean, I got the Book "Programming in Ada 2005" as a present and I liked it, but after reading the introduction (first 2-3 chapters I think) back then (was like over 15 years ago) I saw no libraries which I can use. and I was not that big a programmer to write them myself.

From: A.J. <ianozia@gmail.com>
Date: Sat, 6 Aug 2022 10:48:00 -0700

I agree with you on some of these points. Ada never seemed to be big on marketing, at least outside of specific niches, and from a learning & resources aspect, it took me reading Barnes' Programming in Ada 2012 cover-to-cover to properly grok the language. With that being said, things have been changing a lot in the last two years.

https://learn.adacore.com is a decent resource in that it gives you a little Ada interpreter with code snippets you can test out yourself right in the browser. It's not exactly a "web playground or repl shell" but it's pretty good and seems to support the standard library.

From a library and tooling standpoint, I would check out Alire. It takes a matter of minutes to get from not having any Ada compilers installed at all to compiling your own hello example and there's a lot of libraries already supported (https://alire.ada.dev/crates.html). To bring, for example, Gnatcoll_sqlite, into your project, you would simply just type "alr with gnatcoll_sqlite" while in that directory.

[...]

Then of course there's the awesome-ada repository that has some nice resources, albeit they seem to mostly be in English: https://github.com/ohenley/awesome-ada

From: G.B.
<bauhaus@notmyhomepage.invalid>
Date: Sun, 7 Aug 2022 11:08:43 +0200

> * There's no modern book in German about modern Ada and its libraries

What's the competition, considering C#, Swift, Java or C? I.e., an original work written by a German author, bought and studied by many? There used to be a number of books on Ada written in German when the market had developed ideas of a government mandate, the ideas producing corresponding opportunities.

> * There's no syntax highlighting package in vim for Ada

:syntax enable

(Does vim feature in a modern feeling tool chain, though?)

> * No exercises like for example Ruby Koans

> * It *Looks* like there are no libraries which make it easy use Ada for programming (think json/document formats, http/mail/mime protocols,

AWS, GNATColl, $ alr with json.

> algorithms or cryptography libraries

Just use one that you can trust. If you need it to be more Ada-ish, ChaCha20 cipher and Poly1305 digest have just been mentioned a few postings ago. If algorithms can address securing the entire computation...

There used to be the PAL, which is the Public Ada Library, easy to find. A bit dated, and reflecting the hype back then, I guess.

I gather that, currently, and in the past, Ada tools are also focusing on topics of embedded computers, a fairly large and attractive market. JSON or MIME, perhaps even interpreters are present, but I think not central to control stuff near sensors and actuators. How does one compute deterministic responses before a deadline using Node.js?

[...]

From: Dennis Knorr
<dennis.knorr@gmx.net>
Date: Mon, 8 Aug 2022 23:38:59 +0200

> What's the competition, considering C#, Swift, Java or C?

From the absolute amount in English, these languages or Python or Rust have more books. Hell, even Raku has more books.

Python, Kotlin(!), C# have more german books and also more current ones. I bet in five years from now there will be more German books about Carbon than about Ada, even if you include the old ones. [...]

>> * There's no syntax highlighting package in vim for ada

> :syntax enable

Okay, that I did not know.

> (Does vim feature in a modern feeling tool chain, though?)

Well, okay, Intellij is called more modern of course or VSCode, but you still can craft modern tooling onto vim and it works well.

[...]

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 8 Aug 2022 23:12:44 -0500

> P.S. Nobody writes Ada books these days because they do not sell.

Do *any* programming books really sell? If so, why? :-)

There are plenty of free, on-line resources for pretty much any programming language. Why pay for something you can get free?

When someone starts talking about books, I think they're a troll. I can understand complaints about having trouble finding stuff (although Google should find AdaIC.org pretty easily, it's usually pretty high in Ada results, and most of the good stuff is linked from there), and lack of hype, and so on. But there's lots of good stuff if one looks (or asks here - if someone knows about here they're ready to use Ada).

AdaIC has an Ada-specific search engine which hopefully makes it easier to find Ada stuff than a general engine like Google.

From: Paul Rubin
<no.email@nospam.invalid>
Date: Mon, 08 Aug 2022 23:05:12 -0700

> When someone starts talking about books, I think they're a troll.

I don't know of any online Ada docs that I'd call helpful past the beginner level (Ada Distilled). Someone here recommended a book to me a year or two ago and I bought a copy. It looks good but has just been sitting around waiting for me to read it. I haven't done that because I haven't had any occasion to mess with Ada, and have too many other pending projects. One of these days.

From: John Mccabe
<john@nospam.mccabe.org.uk>
Date: Tue, 9 Aug 2022 07:22:13 -0000

>There are plenty of free, on-line resources for pretty much any programming language. Why pay for something you can get free?

FWIW, I may be 'old school', but I buy loads of programming books. That obviously doesn't qualify me to answer the question of whether "*any* programming books really sell", but the main reasons I like books are that they tend to be more constrained and cohesive than jumping around websites (at least, the decent ones are :-)). Also for those times when I want to flick back and forth between sections quickly, when I don't want to be staring at a screen and so on. One particular reason is that, unless I've actually got a block of free time to be experimenting with stuff, using a Web browser presents multiple, frustrating distractions, and it's often the case that an example of something you might want to do has no explanation about how it works (books, especially Ada As A Second Language, if I remember correctly, are generally fairly good at that bit), so that leads to more searching, more jumping about webpages and, nowadays, a helluva lot of stale and misleading information.

So, basically, that's why I pay for books.

From: John Perry <devotus@yahoo.com>
Date: Tue, 9 Aug 2022 18:19:47 -0700

> Do *any* programming books really sell? If so, why? :-)

Having recently left a university, I can attest that schoolbooks are still a thing, and that includes textbooks on computer programming. I recently inherited from a member of this forum a nice textbook on Data Structures in Ada, but it was based on Ada 95, and I'm not sure it's in print anymore. In fact, and alas, only three of the Ada-based textbooks I find "easily" on Amazon date from the early- to mid-90s, and of the three recent ones I find, only the Barnes book is of good quality.

I'd be delighted if someone would prove me wrong.

From: Paul Rubin <no.email@nospam.invalid>
Date: Tue, 09 Aug 2022 23:20:43 -0700

> of the three recent ones I find, only the Barnes book is of good quality. I'd be delighted if someone would prove me wrong.

Analysable Real-Time Systems: Programmed in Ada by Prof. Alan Burns is from 2016 and looks pretty good. It is the book I mentioned that I got on the recommendation of someone here. I've flipped through it but still haven't read it.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 17 Aug 2022 20:02:53 -0500

The Ada 2012 book by Peter Chapin looks promising, although I don't think he's finished it (https://github.com/pchapin/tutorialada). There is a PDF version of it available on-line.

Otherwise, I recommend Ada Distilled (https://www.adaic.org/wp-content/uploads/2010/05/Ada-Distilled-24-January-2011-Ada-2005-Version.pdf) [Ada 2005], and the Craft of Object Oriented Programming (http://archive.adaic.com/docs/craft/craft.html) [Ada 95], depending on the programmer's level. These are all written in the textbook style, and are all available for free on the Internet. I don't think you miss too much learning with an Ada 95 book first (most of the newer stuff is fairly obvious, or needs a textbook of its own.

The Wikibook is also a good choice

(http://en.wikibooks.org/wiki/Ada_Programming), but you do have to be on-line to use it (the others are downloadable and thus usable locally).

I find the Barnes book to be too much of a good thing (sorry, John!). When John gave me a copy at a Paris ARG meeting, I put it on my lap to look through it (since my hotel room was tiny), my legs got numb after not very long. I know better than to put it on a body part again. :-) I'd recommend it as a reference for serious Ada programmers, since it tries to cover everything (the latest version has an Ada 2022 appendix).

Ada Practice

Working with Library Versions

[This thread splinter veered off from the announcement of ada-lang-io towards library management. Other topics have been pruned out. —arm]

From: Paul Jarrett <jarrett.paul.young@gmail.com>
Date: Tue, 11 Oct 2022 21:21:31 -0700
Date: Sun, 09 Oct 2022 09:13:11 -0700
Newsgroups: comp.lang.ada

> Adahome.com is sort of like that, but it is run by some company and hasn't been updated in forever.

https://ada-lang.io/ is designed to be updateable for a long time and open to community contributions by being completely open source. There're already multiple people who have permissions to merge changes to help ensure longevity.

ada-lang.io is indexed using Algolia, so the entire site (including the Ada 2022 draft RM) is searchable.

Someone else wrote a tool for searching through all code in Alire crates at https://search.synack.me/

> I am not sure if package manager is a good idea if it does not refer to the target system's packaging tools, e.g. DEB, RPM, MSI etc. The main problem with that stuff is usually architectural. Most of it is plain aggregation of source code, which is utterly wrong. The very idea to rebuild everything each time on each client is atrocious both with regard to wasting computing resources as well as testing, safety, consistency, interoperability inside the target.

Alire can do additional build steps and other things.

As an application developer, having the code available helps in auditing third-party software for security reasons, build it in a debug configuration for troubleshooting, and also provides the means to locally fix bugs or adapt the library if needed. Isolating libraries and including them with a package manager on a per project basis eases setup also by not making developers have to look up or use multiple installers.

I've seen inconsistencies in builds when developers who rely on the system libraries (installed by things like apt) join the project at different times -- the earliest developers might be on libfoo-1.2 whereas newer developers are on libfoo-1.4. You don't run into this problem if the repo points to the applicable dependencies and everyone builds everything locally. It also avoids other problems such as if your system's package manager doesn't have a particular library version, and the project builds that library from source. It's not perfect and there's other problems that you run into, but it often does help understanding what is being built in the project more clearly. Alire even takes this an entire step further by being able to install and manage the toolchain as well (gprbuild and GNAT).

Package managers also simplify having multiple projects using the same library, but different and possibly incompatible versions on the same system. You get a snapshot in time and a more consistent path to get a build working for new developers, or on a new system. There are limitations due to what systems open source library writers have available to test on, so you shouldn't just blanket trust code you pull in though, and you should be careful how you use it.

Overall, Alire makes the experience building and developing in Ada for me on Windows, Mac and Linux, considerably simpler and more efficient, by providing the same interface for use across all of these systems.

With the beautiful site styling done by onox, someone pointed to ada-lang.io should be able to download Alire, install a toolchain, make a project and build in less than 15 minutes or so (depending on download and install time). The work done by Fabien and Alejandro, and everyone else who has contributed to Alire to make this happen within the last couple years is absolutely incredible. Combined with Maxim's fantastic work on the Ada language plugin for Visual Studio Code, it's a great experience for first-time users of the language.

[...]

> Maybe a web forum would be a good idea, because many people nowadays see Usenet newgroups as an outdated thing. So the fact that the community mostly relies on comp.lang.ada may turn them off.

There's a dedicated forum now at https://forum.ada-lang.io/

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 12 Oct 2022 17:06:26 -0700

>> I am not sure if package manager is a good idea if it does not refer to the target system's packaging tools, e.g. DEB, RPM, MSI etc.

Alire can define crates that import system libraries, using those tools. They are subject to the same version checks as other Alire crates.

>> The very idea to rebuild everything each time on each client is atrocious both with regard of wasting computing resources as well as testing, safety, consistency, interoperability inside the target.

Actually, it's better for consistency; that's why Alire does it.

I don't understand what you mean by "testing" here; how does compiling from source affect testing?

Same for "interoperability".

> I've seen inconsistencies in builds when developers who rely on the system libraries [...]

More precisely, an Alire crate can specify precisely which version of each dependency it requires/is compatible with.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 13 Oct 2022 08:58:16 +0200

> Actually, it's better for consistency; that's why Alire does it.

Consistency is easier to enforce on pre-built deployments, obviously. Moreover libraries usually provide integrated checks and/or have some target platform policy, e.g. naming and placement conventions.

> I don't understand what you mean by "testing" here; how does compiling from source affect testing?

Because one can run tests on pre-built packages impossible to run on the sources. For example, network/hardware protocols. In order to test a protocol implementation one needs complex mock setups the client simply does not have. Such tests may run for many hours etc.

> Same for "interoperability".

See above. You cannot run integration tests on the client, it is just silly.

>> [...] You don't run into this problem if the repo points to the applicable dependencies and everyone builds everything locally.

No difference whether deployment is in source or pre-built. Dependencies must be enforced regardless. However it is far easier to do with pre-built packages.

> More precisely, an Alire crate can specify precisely which version of each dependency it requires/is compatible with.

It seems so. Multiple versions at once are not supported. E.g. when you are working on two projects both dependent on different versions of another project:

B -> A.1

C -> A.2

Or even the same project, e.g. when doing some migration from one version to another.

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Fri, 14 Oct 2022 01:41:32 -0700

> Multiple versions at once are not supported. [...]

Yes of course, different crates can depend on different versions of the same crate.

> Or even the same project, e.g. when doing some migration from one version to another.

Not sure how you would do that? Link two different versions of the same library in an executable? That's not going to work.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 14 Oct 2022 12:05:00 +0200

> Yes of course, different crates can depend on different versions of the same crate.

It is about whether both A's can be installed and coexist on the same machine.

> Not sure how you would do that? Link two different versions of the same library in an executable? That's not going to work.

Same as above. You have B.1 -> A.1 and B.* -> A.2. You want to install both A.1 and A.2 and work on B.* while checking on B.1.

In the long gone time of common sense, a project code management system would use a virtual file system and map different parts of the project's graph onto a structure of folders arranged by versions. Today one would use something ugly like a virtual machine or incredibly ugly like a docker.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 14 Oct 2022 04:19:05 -0700

> It is about whether both A's can be installed and coexist on the same machine.

In Alire, "installed" means "checked out the source code into a local directory".

If A depends on a system library that is a shared object file, and those are different versions, then it depends on the OS; Debian can handle this nicely, Windows only via separate directories and search paths.

> Same as above. You have B.1 -> A.1 and B.* -> A.2. You want to install both A.1 and A.2 and work on B.* while checking on B.1.

And the solution is the same as well.

> [...] a project code management system would use a virtual file system and map different parts of the project's graph onto a structure of folders arranged by versions.

What prevents that now?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 14 Oct 2022 15:05:06 +0200

> What prevents that now?

Nothing except that it is to be done manually. Why not download a source archive and bother with anything? It is Turing-complete, after all... (:-))

The advantage of a file system is that developing image will be automated and consistent. And you would not need to move any files physically. Alire is extremely slow because it must pull all files [and then compile them on top of that].

Furthermore, a virtual file system shares duplicates of the same version of the same file. When you work with naked Git you must have as many copies as you have projects. Same applies to virtual machines and dockers. It is a huge overhead for nothing.

Moreover, a virtual file system is instant so long you do not access a file for read or write. Which is the case for gprbuild, make and other tools which use timestamps and then never look into files.

With a virtual file system you can automatically check in all files on closing if it was open for write and never worry about command-line mess or plug-ins. Any tool will work out of the box.

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Sun, 16 Oct 2022 10:54:57 +0200

> Furthermore, a virtual file system shares duplicates of the same version of the same file. When you work with naked Git you must have as many copies as you have projects. Same applies to virtual machines and dockers. It is a huge overhead for nothing.

Inasmuch as versions are subject to business, software configuration management is just work that requires resources to get it done. Problem solved. (Well, not for the small shop on a budget, granted.)

To what extent can static linking make B.1 and B.2 exist on the same system?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 16 Oct 2022 11:20:33 +0200

> Inasmuch as versions are subject to business, software configuration management is just work that requires resources to get it done.

Yes, human resources especially. It is a self-feeding system that exists in each organization. It creates problems in order to justify its continuous growth. Modern time tools excel at wasting and perfect outright meaninglessness.

> Problem solved. (Well, not for the small shop on a budget, granted.)

I cannot say that ClearCase, which did things more or less right 20 years ago, was for small business either. (:-)) AFAIK it is still available and GNAT Studio supports it. However, IBM (Rational, actually) fulfills its existential end goal of wasting personal and hardware resources by other, no less efficient, techniques... (:-))

> To what extent can static linking make B.1 and B.2 exist on the same system?

To a full extent! (:-))

Sorry, I do not understand your question...

Arrays with Discriminated Task Components

From: Adamagica <christ-usch.grein@t-online.de>
Subject: Arrays with discriminated task components
Date: Sat, 24 Dec 2022 03:44:27 -0800
Newsgroups: comp.lang.ada

I've got a task type with a discriminant:

```
type Index is range 1 .. N;
task type T (D: Index);
```

Now I want an array of these tasks, where each task knows its identity (the index) via the discriminant, an iterated_component_association:

```
Arr: array (Index) of T :=
  (for I in Index => ???);
```

How can I do this?

This works with access, but I find this extremely ugly:

```
Arr: array (Index) of access T :=
 (for I in Index => new T (I));
```

Alternatively, I could use the traditional method with a Start entry with the index as parameter:

```
task type T is
  entry Start (D: Index);
end T;
```

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 24 Dec 2022 20:05:19 +0200

One way is to give the discriminant a default value that is a function call that returns a new identifier on each call:

```
Next_Index : Index := Index'First;
-- The value returned by the next call
-- of New_Index.
function New_Index return Index
-- Returns a unique Index value (up to N).
is
   Result : constant Index := Next_Index;
begin
   if Next_Index < Index'Last then
      Next_Index := Next_Index + 1;
   -- else report error?
   end if;
   return Result;
end New_Index;
task type T (D: Index := New_Index);
```

Then you can declare the array without any initial value:

```
Arr: array (Index) of T;
```

and the initialization of each task in the array makes its own call to New_Index and gets its own identifier value.

A bit sneaky but has the advantage that it extends automatically to two arrays of tasks, or one array and some separate single declarations of tasks, etc.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 24 Dec 2022 23:41:34 +0100

> One way is to give the discriminant a default value that is a function call that returns a new identifier on each call:

No, this does not guarantee that the task's discriminant is its index in the array, which is a requirement of the question.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 25 Dec 2022 18:16:56 +0200

This seems to work with GNAT, but I'm not entirely sure if it is legal (could there be a conflict between the default value of the task discriminant, which is the same for all tasks in the array, and the actual discriminants which are different for each task in the array?):

```
N : constant := 10;
type Index is range 1 .. N;
task type T (D: Index := Index'First);
-- A default value for D is needed to make
-- the type constrained, as
-- required by the Arr declaration below.
function New_T (I : in Index)
return T
is
begin
  return TI : T (D => I)
  do
    null;
  end return;
end New_T;
Arr: array (Index) of T := (for I in Index =>
    New_T(I));
```

Whether this is any less ugly than the heap allocation method is doubtful.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Mon, 26 Dec 2022 08:39:23 -0800

Thanx, Niklas and Jeffrey. I just didn't think of the generator function.

Sockets, Streams, and Element_Arrays

From: Mark Gardner <magardner2017@gmail.com>
Subject: Sockets, Streams, and Element_Arrays: Much confusion
Date: Sat, 31 Dec 2022 14:11:55 +0200
Newsgroups: comp.lang.ada

Hello, I've been having a bit of difficulty doing some UDP socket programming in Ada. As outlined in my stackoverflow question here (https://stackoverflow.com/q/74953052/7105391), I'm trying to reply to messages I am getting over UDP.

GNAT.Sockets gives me a Stream_Element_Array, which I can't find any documentation on how to make use of other than "You should also be able to get a Stream, which you should use instead" (About ten years ago, on this very newsgroup, somebody said not to use streams with UDP, or at least not GNAT.Sockets.Stream).

Adasockets gives me a String, which I can work with, except it throws away the data recvfrom gives it, apparently making it impossible to reply to the querying address.

At this point, I'm half-tempted to make my own binding, but as I've never done that sort of thing before, I thought I'd ask the wisdom of the Usenet if there is a way to convert a Stream_Element_Array into the exotic types of Unsigned_16 and String.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 31 Dec 2022 14:11:11 +0100

> GNAT.Sockets gives me a Stream_Element_Array [...]

Stream_Element_Array is declared in Ada.Streams as

```
type Stream_Element_Array is
   array(Stream_Element_Offset range <>)
of aliased Stream_Element;
```

For communication purposes it is an array of octets. Your datagram is represented as a Stream_Element_Array or a slice of.

As for streams, yes, it does not make sense to use them for networking, unless you override all stream primitives. The reasons for that are

- non-portability of predefined primitives

- low efficiency for complex data types

- encoding inefficiency as well

You will need to handle some application protocol artifacts, checksums, counters, strange encodings, sequence numbers etc. It is easier to do this directly on the Stream_Element_Array elements.

And, well, do not use UDP, except for broadcasting. There is no reason to use it. For multicast consider delivery-safe protocols like PGM. For single cast use TCP/IP. (If you need low latency see the socket NO_DELAY option)

From: Mark Gardner <magardner2017@gmail.com>
Date: Sat, 31 Dec 2022 15:50:29 +0200

> For communication purposes it is an array of octets.

According to RM 13.13.1, "Stream_Element is mod implementation-defined" which to me says there is no guarantee that they will be octets, unless this is specified elsewhere?

> You will need to handle some application protocol artifacts, checksums, counters, strange encodings, sequence numbers etc. It is easier to do this directly on the Stream_Element_Array elements.

So, how would I do this directly on the elements? I mean, if it is an octet-array to a string, I expect an element-to-element copy, or type conversion to work, but what about integers? Do I need to do something like My_Int:=Unsigned_8(octet(1))+2**8* Unsigned_8(octet(2)); or whatever endianness demands? Or is this the time to learn how to use Unchecked_Conversion?

> And, well, do not use UDP, except for broadcasting.

Well, my use case just so happens to be broadcasting, and re-broadcasting data across a binary-tree-like p2p network.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 31 Dec 2022 15:16:05 +0100

> According to RM 13.13.1, "Stream_Element is mod implementation-defined"

GNAT.Sockets is GNAT-specific. All GNAT compilers have Stream_Element 8 bits. I can imagine some DSP implementation with Stream_Element of 32 bits. But realistically add

```
pragma Assert (Stream_Element'Size >= 8);
```

and be done with that.

[...]

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 31 Dec 2022 16:18:50 +0100

> According to RM 13.13.1, "Stream_Element is mod implementation-defined"

The ARM has always tried to ensure that the language could be implemented on any kind of processor. Thus you have implementation-defined separate definitions of Storage_Element and Stream_Element, which need not be the same, and no guarantee that Interfaces contains declarations of Integer_8 or Unsigned_8.

But these days almost everything is byte oriented, so unless you need what you're writing to work on some unusual H/W, you can presume that both of these are bytes, and that Interfaces contains those declarations.

From: Simon Wright <simon@pushface.org>
Date: Sat, 31 Dec 2022 17:39:07 +0000

> About ten years ago, on this very newsgroup, somebody said not to use streams with UDP, or at least not GNAT.Sockets.Stream.

The reasoning behind the recommendation not to use streams with UDP was as follows (there's a faint possibility that it no longer applies!)

If the data type you want to send is e.g.

```
type Message is record
   Id  : Integer;
   Val : Boolean;
end record;
```

and you create a datagram socket and from that a stream, then use Message'Write to the stream, GNAT will transmit each component of Message separately in canonical order (the order they're written in the type declaration). This results in two datagrams being sent, one of 4 bytes and one of 1 byte.

If you take the same approach at the destination, Message'Read reads one datagram of 4 bytes, and one of 1 byte, and it all looks perfect from the outside. If the destination is expecting a 5 byte record, of course, things won't work so well.

The approach we adopted was to create a 'memory stream', which is a chunk of memory that you can treat as a stream (see for example ColdFrame.Memory_Streams at [1]). With Ada2022, you should be able to use Ada.Streams.Storage.Bounded [2].

Message'Write the record into the memory stream; transmit the written contents as one datagram.

To read, create a memory stream large enough for the message you expect; read a datagram into the memory stream; Message'Read (Stream => the_memory_stream, Item => a_message);

You can use gnatbind's switch -xdr to "Use the target-independent XDR protocol for stream oriented attributes instead of the default implementation which is based on direct binary representations and is therefore target-and endianness-dependent".

[1] https://github.com/simonjwright/coldframe/blob/master/lib/coldframe-memory_streams.ads

[2] http://www.ada-auth.org/standards/22rm/html/RM-13-13-1.html#p25

From: Mark Gardner <magardner2017@gmail.com>
Date: Sat, 31 Dec 2022 21:36:40 +0200

> The approach we adopted was to create a 'memory stream'

Wait, so if I know what shape my data is, and use a memory_stream (like the one in the Big Online Book of Linux Ada Programming chapter 11 [1]), I'm fine using Stream, in conjunction with Get_Address? That's wonderful. Not at all frustrated that I just wasted approximately three working days looking for a solution to a problem that didn't exist.

> Message'Write the record into the memory stream; transmit the written contents as one datagram.

I'm guessing with Memory_Stream'Write(Socket_Stream, Buffer);?

> To read, create a memory stream large enough for the message you expect

Does this second buffer need to be added? If the datagram arrives (UDP), shouldn't GNAT.Sockets.Stream() be able to handle it?

> You can use gnatbind's switch -xdr to "Use the target-independent XDR protocol for stream oriented attributes [...]

Oh fun, I didn't think of that aspect. Thanks! Would I have to pass it as a command line flag, or would there be some kind of pragma I could use?

Thanks for the help so far, and happy new year!

[1] http://www.pegasoft.ca/resources/boblap/11.html#11.12

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 31 Dec 2022 21:16:18 +0100

> I'm guessing with Memory_Stream'Write(Socket_Stream, Buffer);?

No, you create a memory stream object. Then you write your packet into it:

```
My_Message'Write
(My_Memory_Stream'Access);
```

Once written you use the accumulated stream contents to write it into the socket. An implementation of a memory-resident stream is very simple. E.g. see: http://www.dmitry-kazakov.de/ada/strings_edit.htm#Strings_Edit.Streams

My advice would be not to do this. It is wasting resources and complicated being indirect when 'Write and 'Read are compiler-generated. If you implement

'Write and 'Read yourself, then why not call these implementations directly. It just does not make sense to me. I always wonder why people always overdesign communication stuff.

Build messages directly in a Stream_Element_Array. Use system-independent ways to encode packet data. E.g. chained codes for integers. Mantissa + exponent for real numbers. If you have Booleans and enumerations it is a good idea to pack them into one or two octets to shorten the packets. All this is very straightforward and easy to implement.

You can also consider using some standard data representation format, e.g. ASN.1. An Ada ASN.1 implementation is here:
http://www.dmitry-kazakov.de/ada/components.htm#ASN.1

You describe your message in ASN.1 as an Ada tagged type derived from building blocks. Then you can encode and decode it directly from Stream_Element_Array. I would not recommend that either. ASN.1 is quite overblown.

Happy New Year!

From: philip...@gmail.com <philip.munts@gmail.com>
Date: Sat, 31 Dec 2022 14:32:17 -0800

> And, well, do not use UDP, except for broadcasting

I have to disagree here. UDP is perfectly fine for RPC-like (Remote Procedure Call) transactions on a local area network. And it is orders of magnitude easier to implement on microcontrollers than TCP. An Ada program using UDP to communicate with data collecting microcontrollers makes perfect sense in some contexts. I use it for my Remote I/O Protocol.

The only trick is that the server (or responder, as I like to call it) and client (or initiator) can't quite use the same code.

Here is my generic package for UDP with fixed length messages:

https://github.com/pmunts/libsimpleio/blob/master/ada/objects/messaging-fixed-gnat_udp.ads

https://github.com/pmunts/libsimpleio/blob/master/ada/objects/messaging-fixed-gnat_udp.adb

Getting between Stream_Element_Array and a byte array is a pain and I wound up just looping over arrays, copying one byte at a time. If somebody has a better idea, let me know.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 31 Dec 2022 23:49:33 +0100

> Getting between Stream_Element_Array and a byte array is a pain and I wound up just looping over arrays, copying one byte at a time. If somebody has a better idea, let me know.

You should be able to use Unchecked_Conversion for that.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 31 Dec 2022 23:55:07 +0100

> I have to disagree here. UDP is perfectly fine for RPC-like (Remote Procedure Call) transactions on a local area network.

RPC and other synchronous exchange policies should be avoided as much as possible.

Having said that, implementation of RPC on top of streams is incomparably easier than on top of UDP.

> And it is orders of magnitude easier to implement on microcontrollers than TCP.

Not at all. You need:

- Safe transmission and error correction on top of UDP;
- Buffering and sorting out incoming datagrams;
- Maintaining sequence numbers;
- Splitting messages that do not fit into a single datagram and reassembling them on the receiver side;
- Buffering on the sender side to service resend requests.

This is extremely difficult and a huge load for a microcontroller.

> Getting between Stream_Element_Array and a byte array is a pain and I wound up just looping over arrays, copying one byte at a time. If somebody has a better idea, let me know.

Use "in situ" conversion if you are concerned about copying. E.g.

```
pragma Import (Ada, Y);
for Y'Address use X'Address;
```

From: Simon Wright <simon@pushface.org>
Date: Sat, 31 Dec 2022 23:41:11 +0000

> My advice would be not to do this. [...]

It has to depend on the design criteria.

If you need something now, and it's not performance critical, and you have control over both ends of the channel, why not go for a low-brain-power solution?

On the other hand, when faced with e.g. SNTP, why not use Ada's facilities (e.g. [1]) to describe the network packet and use unchecked conversion to convert to/from the corresponding stream element array to be sent/received?

I'd have thought that building messages directly in a stream element array would be the least desirable way to do it.

[1] https://sourceforge.net/p/coldframe/adasntp/code/ci/default/tree/SNTP.impl/sntp_support.ads

Real_Arrays on Heap with Clean Syntax

From: Jim Paloander <dhmos.altiotis@gmail.com>
Subject: Real_Arrays on heap with overloaded operators and clean syntax
Date: Sun, 22 Jan 2023 13:34:18 -0800
Newsgroups: comp.lang.ada

Dear Ada lovers,

with stack allocation of Real_Vector (1 .. N) when N >= 100,000 I get STACK_OVERFLOW ERROR while trying to check how fast operator overloading is working for an expression

X := A + B + C + C + A + B, where A,B,C,X are all Real_Vector (1 .. N).

So my only option was to allocate on the heap using new. But then I lost the clean syntax

X := A + B + C + C + A + B

and I had to write instead:

X.all := A.all + B.all + C.all + C.all + A.all + B.all.

This is really ugly and annoying because when you are using Real_Arrays for implementing some linear algebra method who relies heavily on matrix vector products and vector updates, you do need to allocate on the heap (sizes are determined in runtime) and you do need a clean syntax. So, is there any way to simplify my life without using the .all or even without declaring A,B,C,X as access Real_Vector?

Thanks for your time!

From: Joakim Strandberg <joakimds@kth.se>
Date: Sun, 22 Jan 2023 13:56:27 -0800

Easiest solution is probably to declare a new task and specify the stack size using the Storage_Size aspect. Allocate as much stack space as you need to be able to do the calculations and do all the allocations on the declared task, not on the environment task. You will avoid the unnecessary heap allocations and have nice clean syntax.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 22 Jan 2023 23:13:14 +0100

You can define "+" on the access type, which should probably be an arena pointer for performance reasons:

```
Arena : Mark_And_Release_Pool;
type Real_Vector_Ptr is access
```

```
    Real_Vector;
   for Real_Vector_Ptr'Storage_Pool use
   Arena;
   function "+" (Left, Right :
   Real_Vector_Ptr)
      return Real_Vector_Ptr is
   begin
      if Left'Length /= Right'Length then
         raise Constraint_Error;
      end if;
      return Result : Real_Vector_Ptr :=
      new Real_Vector (Left'Range) do
         for I in Result'Range loop
            Result (I) :=
               Left (I) + Right (I - Left'First +
               Right'First);
         end loop;
      end return;
   end "+";
```

You can overload that with

```
   function "+" (Left : Real_Vector_Ptr;
   Right : Real_Vector)
      return Real_Vector_Ptr is
   begin
      if Left'Length /= Right'Length then
         raise Constraint_Error;
      end if;
      return Result : Real_Vector_Ptr :=
      new Real_Vector (Left'Range) do
         for I in Result'Range loop
            Result (I) :=
               Left (I) + Right (I - Left'First +
               Right'First);
         end loop;
      end return;
   end "+";
```

and with

```
   function "+" (Left : Real_Vector;
    Right : Real_Vector_Ptr)
      return Real_Vector_Ptr;
```

Then you will be able to write:

```
   X := A + B + C + C + A + B;
   -- Use X
   Free (X); -- Pop all arena garbage
```

But of course, the optimal way to work large linear algebra problems is by using in-place operations, e.g.

```
procedure Add (Left : in out Real_Vector;
Right : Real_Vector);
```

etc.

Regards,
Dmitry A. Kazakov

http://www.dmitry-kazakov.de

From: Jim Paloander
<dhmos.altiotis@gmail.com>
Date: Sun, 22 Jan 2023 14:49:09 -0800

> It is my impression that in the Ada community the preferred way of working is in general stack only. [...]

With great depression I realized that the preferred way is of stack only. This is very restrictive excluding all scientific modelling involving solvers for partial differential equations, linear algebra kernels, etc. It is insane. Completely insane. 3D simulations of physical phenomena may involve billions of grid-cells and at each grid-cell several unknowns are defined (velocity, pressure, temperature, energy, density, etc). That is why they are using Fortran or C++, but Ada has really cool stuff for so many things, why not vectors and matrices and heap allocation? Would you please give me an example, I googled and I cannot find a single example demonstrating how to use a task with the declaration of stack size. Why is there so little information online about so important things such as allocation?

From: Gautier Write-Only Address
<gautier_niouzes@hotmail.com>
Date: Sun, 22 Jan 2023 15:14:03 -0800

Note that Real_Arrays does not specify where things are allocated (heap or stack).

Only when you define "x : Real_Vector (1 .. n)", it is on stack. You can always write something like the snippet below.

Anyway, after a certain size, you may have to find compromises, like avoiding operators (they do too many allocations & deallocations in the background, even assuming elegant heap-allocated objects) and also give up plain matrices, against sparse matrices or band-stored matrices, typically for solving Partial Differential Equations.

```
with Ada.Numerics.Generic_Real_Arrays;
procedure Test_Large is
  type Float_15 is digits 15;
  package F15_R_A is new
    Ada.Numerics.Generic_Real_Arrays
    (Float_15);
  use F15_R_A;
  procedure Solve_it
   (x : in     Real_Vector;
    y : out    Real_Vector;
    A : in     Real_Matrix) is
  begin
   null;  --  Here, the big number-crunching
  end;
   n : constant := 10_000;
  type Vector_Access is access
  Real_Vector;
  type Matrix_Access is access Real_Matrix;
  x, y : Vector_Access := new Real_Vector
   (1 .. n);
  A    : Matrix_Access := new Real_Matrix
  (1 .. n, 1 .. n);

begin
  Solve_it (x.all, y.all, A.all);
  --  !! Deallocation here
end;
```

From: Leo Brewin
<leo.brewin@monash.edu>
Date: Mon, 23 Jan 2023 12:14:47 +1100

Here is a slight variation on the solution suggested by Gautier. It uses Ada's "rename" syntax so that you can avoid all the .all stuff. I use this construction extensively in my large scale scientific computations.

```
with Ada.Numerics.Generic_Real_Arrays;
with Ada.Unchecked_Deallocation;
procedure Test_Large is
   type Float_15 is digits 15;
   package F15_R_A is new
     Ada.Numerics.Generic_Real_Arrays
     (Float_15);
   use F15_R_A;
   procedure Solve_it
     (x : in     Real_Vector;
      y : out   Real_Vector;
      A : in     Real_Matrix) is
   begin
     null;  --  Here, the big number-crunching
   end;
   n : constant := 10_000;
   type Vector_Access is access
   Real_Vector;
   type Matrix_Access is access
   Real_Matrix;
   x_ptr, y_ptr : Vector_Access := new
   Real_Vector (1 .. n);
   A_ptr        : Matrix_Access := new
   Real_Matrix (1 .. n, 1 .. n);
   x : Real_Vector renames x_ptr.all;
   y : Real_Vector renames y_ptr.all;
   A : Real_Matrix renames A_ptr.all;
   procedure FreeVector is new
      Ada.Unchecked_Deallocation
      (Real_Vector,Vector_Access);
   procedure FreeMatrix is new
      Ada.Unchecked_Deallocation
     (Real_Matrix,Matrix_Access);
begin
   Solve_it (x, y, A);
   -- Deallocation here
   FreeVector (x_ptr);
   FreeVector (y_ptr);
   FreeMatrix (A_ptr);
end;
```

From: Jim Paloander
<dhmos.altiotis@gmail.com>
Date: Sun, 22 Jan 2023 22:01:58 -0800

Thank you very much, would a for Real_Vector_Access'Storage_Pool use localPool; save you from the need to free the vectors and matrix yourself?

On the other hand, is there any way of avoiding temporaries? Possibly a modern version of the Real_Array using expression generic syntax or something similar? Since you are using scientific computations extensively, you must be aware of Fortran. Have you compared Fortran's complex numbers with Ada's for inner products or similar computations to see who is faster? You see, I like a lot of things about Ada, but the syntax is really difficult to follow. Sometimes it gives me the impression that it is more difficult than really needed to be. For example there should be a way for Real_Arrays to allocate memory internally and not on the stack directly like containers. And for containers to provide an indexer Vector(i) and overloaded operators similarly to Real_Vectors. But the fact that they do not give me the impression that this Language, although being designed by the army for mission critical applications, never realized that modern mission critical need to simplify mathematical calculations providing an easy syntax. I am surprised that after so many years and so many updates to the Standard the designers of the Language did not realize that such mathematical syntax should be simplified to attract more people from scientific computing, who are tired with Fortran 10000 ways of declaring something a variable.

From: Egil H H <ehh.public@gmail.com>
Date: Sun, 22 Jan 2023 23:50:11 -0800

> wanted to find the video where Jean Pierre Rosen talks about how memory is handled in the Ada language from FOSDEM perhaps 2018-2019.

It was in 2016: https://archive.fosdem.org/2016/schedule/event/ada_memory/

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 23 Jan 2023 09:51:55 +0100

>> Something came up and I had to send my previous reply/e-mail as is. I wanted to find the video where Jean Pierre Rosen talks about how memory is handled in the Ada language from FOSDEM perhaps 2018-2019. Unfortunately I have been unable to find it.

>>

> It was in 2016:

> https://archive.fosdem.org/2016/schedule/event/ada_memory/

Thanks Egil, you were faster than me...

I also have a full tutorial at several Ada-Europe conferences. No video, but I can send the slides to those interested.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 23 Jan 2023 09:28:46 +0100

> I was not sure whether or not it can be avoided with Implicit_Dereference,

> type Accessor (Data: not null access Element) is limited private with Implicit_Dereference => Data;

If you create a new wrapper type, anyway, then it is easier to define operations directly on that new type.

> Otherwise what you described for operator+ one has to do for every operator overloaded inside Real_Arrays package.

You should not use the standard library anyway. It is not intended for large problems, which require specific approaches and methods, like sparse matrices, concurrent processing and so on.

> The optimal way to work large linear algebra problem is what you describe because unfortunately Ada does not allow what Fortran does since 30 years ago or more.

I am not sure what you mean. It is quite possible to design a wrapper datatype allocating vectors/matrices in the pool. E.g. Ada's Unbounded_String is such a thing. Real_Arrays were not designed this way because see above.

> But in C++ you can reproduce the same functionality as Fortran using Expression Templates and Template Metaprogramming.

Nothing prevents you from wrapping Real_Array in a generic way:

```
generic
   with package Real_Arrays is new
Numerics.Generic_Real_Arrays (<>);
package Generic_Pool_Real_Arrays is
   ...
end Generic_Pool_Real_Arrays;
```

> Perhaps Ada should allow something like that. Because for maintainability reasons the best would be to write the mathematical expressions as close as possible to the mathematical formulas.

There is no problem with that as you can define operations on pointers.

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Mon, 23 Jan 2023 09:39:39 +0100

> Are you aware of any libraries similar to Real_Arrays, but who allocated memory internally using heap?

The most natural way to work with an array of FPT numbers is for the programmer to declare an array indexed by some index type. Done. If GNAT gets in the way there, it might be worth a note sent to its maintainers. Whenever a programmer is tasked with considering memory allocation, then depending on one's propensity towards working on memory allocation it is inconvenient and distracting. Math programs don't make you do this, I think.

Also, std::vector and its relatives shield the programmer from the absurdly clever, yet unreadable memory allocation that needs to be stuffed behind the scenes. More importantly, though, C++ introduced std::move semantics after a few decades of its existence, to address copying when using chains of +. It might be interesting to see Ada's in-situ construction of return values in comparison.

> Similarly to the Containers.Vector. But Vector has such an awful syntax. There should be something like an indexer [i] similarly to the C++ std::vector to make things simpler

.at() does some of what Ada does. Is

```
v.at(k) = 4;
```

less awful than

```
v(k) := 4;
```

?

Another thing: Mathematical notation has ellipsis, thus

```
A + B + ... + Y + Z;
```

Most general purpose languages don't have ellipsis for this kind of expression. However, even mathematical formulas use what programmers can usually achieve, too. The usual

```
\sum_k A_k.
```

No "+" at all, and an array of vectors, not single ones. Going further, some like to write

```
reduce("+", A);
```

In Ada, you could have a generic function for this, or use a function pointer.

The .all thing vanishes automatically whenever you want to refer to a particular component of the pointed-at object, as opposed to all of them. So, A.all(K) is the same as A(K). Likewise, .all can be dropped if want to invoke the pointed-at subprogram if it has parameters.

Broadcast / Iterate to All Connection Objects via Simple Components?

From: A.J. <ianozia@gmail.com>
Subject: Broadcast / iterate to all Connection objects via Simple Components?
Date: Tue, 7 Feb 2023 12:29:39 -0800
Newsgroups: comp.lang.ada

Hello everyone,

In an effort to better learn network programming in Ada, I've been working through the Protohacker Challenges (https://protohackers.com/), and the current challenge (number 3) is to create a chat server.

I am using a TCP Connections Server with Simple Components, specifically a Connection_State_Machine, but I've run into a problem. I'm trying to send a message received via "procedure Process_Packet (Client : in out Server_Connection)" to all connected Clients.

My (potentially incorrect) thought on how to accomplish this is to iterate through all of the clients currently connected, and use Send to send the message received to those clients. I've been struggling with how to actually do this though, since I couldn't use "function Get_Clients_Count (Listener : Connections_Server) return Natural" from within Process_Packets.

Another thought I had could be to just place every message received in a central queue, and then once all of the packets have been received, to then process that queue and send the results to every connected client.

I tried overriding "procedure On_Worker_Start (Listener : in out Connections_Server)", thinking that I could use it to read such a queue, but it never seemed to be called from within my program and I'm still unsure how to iterate through the Connection objects anyway.

Am I approaching this the right way, or am I missing something very obvious? I've read the test files that came with Simple Components, including the data server but couldn't see a way to get each client to interact with each other. If I didn't explain this well enough, please let me know, I'll be happy to clarify.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Wed, 8 Feb 2023 10:55:11 +0100

For an example of this, see the Chattanooga demo that comes with Gnoga (https://sourceforge.net/projects/gnoga/). A screenshot and intermittently working (not right now) on-line version are available at https://sourceforge.net/p/gnoga/wiki/Gnoga-Gallery/.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Sun, 12 Feb 2023 23:28:26 -0800

I am not sure how familiar you are with Network programming in general (not just as it would be done in Ada). Using a blocking Send could actually kill your performance. You mentioned you would be sending a message to one client after another. Imagine one of the clients has small socket buffers, and is busy doing something else at the moment so not reading your message immediately. If you are sending a large message, your server would only be able to send part of the message, then it would block until the client has read enough that there is space again in the socket buffers to send the rest of the message. That could take ... days. In the meantime, your server is not doing anything else, and no other client gets sent anything...

Instead, you need to use non-blocking sockets. When Send returns, it has sent whatever it could for the moment. You then need to monitor the socket (and all other similar ones) using something like select (which is limited to sockets < 1024, so pretty useless for an actual server in practice) poll (better version of select) or epoll (the best in my opinion). I have written a similar server that has 25000 concurrent clients, and serves them all with 10 worker tasks. That would never fly with blocking sockets.

A similar approach when receiving messages from clients, by the way. The message might have sent only part of its message, so you need to give up temporarily, and come back to it when poll tells you there is something new to read.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 09:30:22 +0100

> Using a blocking Send could actually kill your performance. [...] A similar approach when receiving messages from clients, by the way.

Yes. All networking in Simple components is built on non-blocking sockets (socket select).

P.S. This poses difficulties for users, who see all communication turned upside down being driven by arbitrary socket events rather than by the protocol logic. This was a reason I argued for introducing co-routines with task interfaces in Ada.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Mon, 13 Feb 2023 00:44:01 -0800

> sockets (socket select).

Have you taken a look at epoll(), on Linux it is so much more natural to use, and so much more efficient in practice. The example I mentioned above (a server with 25_000 concurrent connections) cannot work with select (which only accepts file descriptors up to 1024), and is slow with poll (since the result of the latter is the number of events, and we need to iterate over all registered sockets every time).

> This was a reason I argued for introducing co-routines with task interface in Ada.

In my own code, I basically provide an epoll-based generic framework. One of the formal parameters is a `Job_Type` with one primitive operation `Execute`. The latter is initially called when a new connection is established, and is expected to do as much non-blocking work as it can (Execute is run in one of the worker tasks). When it cannot make progress, it returns a tuple (file_descriptor, type_of_event_to_wait_for) to indicate when it needs to be called again in the future, for instance some data became available to read on the specified file_descriptor. The intent is that the `Job_Type` is implemented as a state machine internally.

Of course, a state machine is one of the two ways I know (along with a task) to implement the equivalent of a co-routine in Ada. So I 100% agree with you that co-routines would be very useful in simplifying user code, in particular in the scenario we are discussing here!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 11:55:07 +0100

> Have you taken a look at epoll(), on Linux ?

The implementation is on top of GNAT.Sockets, so no.

> It is so much more natural to use, and so much more efficient in practice.

Well, if there is Linux kernel level support why it is not used in socket select as it is in epoll? I would expect them do that at some point or drop epoll... (:-))

> [...] The intent is that the `Job_Type` is implemented as a state machine internally.

Yes, state machine is what I want to avoid. With complex layered protocols it imposes incredible difficulties requiring auxiliary stacks and buffers with errors almost intractable either by testing or by formal proofs.

> So I 100% agree with you that co-routines would be very useful in simplifying user code, in particular in the scenario we are discussing here!

I'd like to have special Ada "tasks" acting as co-routines on top of proper tasks yielding when the socket buffer is empty or full.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Mon, 13 Feb 2023 03:07:04 -0800

> Well, if there is Linux kernel level support why it is not used in socket select as it is in epoll?

Because in practice the Linux developers don't get to modify such APIs, which are mandated by Posix, or Unix, or some RFC. So the API for select and poll will *never* change.

epoll is definitely the modern approach on Linux, until of course someone finds something even better. epoll is fully thread safe too, which is very nice when used from Ada. Using select() is totally outdated at this point, and means you can never handle more than 1000 simultaneous clients, and that only if you do not have other file descriptors open (database, files,...)

The person who developed GNAT.Sockets has left AdaCore a while ago, so "they" (which I assume is what your message was referring to) are actually unlikely to update that. They also have strong concerns about platform-agnostic support, and epoll is linux-specific at this point (likely also BSD). There exist multiple libraries out there that provide an API common to multiple platforms, and that use epoll on linux. Maybe that's what would make sense, but nowadays with Alire, I would expect someone to build a crate there rather than AdaCore modify GNAT.Sockets.

> Yes, state machine is what I want to avoid. With complex layered protocols it imposes incredible difficulties requiring auxiliary stacks and buffers

with errors almost intractable either by testing or by formal proofs.

Tell me about auxiliary stacks :- In practice, in my experience, you can have a single incoming buffer which is used by one state, and then another when the first state is no longer active,... so we do not need to have too many buffers, but that definitely is not trivial. Currently, I have a stack of iterators reading from a socket, buffering on top of that, then decompressing LZ4 data, then decoding our binary encoding to Ada values.

> I'd like to have special Ada "tasks" acting as co-routines on top of proper tasks yielding when the socket buffer is empty or full.

This is an approach we had discussed at AdaCore before I left. There are multiple drawbacks here: the limited stack size for tasks by default (2MB), the fact that entries cannot return indefinite types, the fact that currently those tasks are assigned to OS threads (so too many of them does impact resource usage),...

A colleague had found an external library that would provide several stacks and thus let people implement actual co-routines. We did not do much more work on that, but it was a nice proof of concept, and efficient. I think things are mostly blocked now, as the ARG has been discussing these topics for quite a few years now.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 12:57:19 +0100

> [...] This is an approach we had discussed at AdaCore before I left. [...]

My idea is to have these as pseudo-tasks scheduled by the Ada run-time and not mapped onto any OS threads. A proper thread would pick up such a task and run it until it yields. The crucial point is to use the stack of the pseudo-task in place of the thread's stack or backing it up and cleaning the portion of the stack at the point of yielding, whatever.

> [...] the ARG has been discussing these topics for quite a few years now.

I have an impression that ARG's view on co-routines totally ignores the use case of communication stacks and other cases state machines show their ugly faces...

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 13 Feb 2023 15:22:19 +0200

[snip discussion of network programming details, retain discussion about co-routines]

So your co-routines would (1) have their own stack and (2) be independently schedulable, which implies (3) having their own execution context (register values, instruction pointer, etc.) How is that different from the Ada concept of a "task"? How could the ARG separate between a "task" and a "co-routine" in the Ada RM?

There exist Ada compilers and run-times where the tasking concept is implemented entirely in the run-time system, without involving the underlying OS (if there even is one). That approach was mostly abandoned in favour of mapping tasks to OS threads, which makes it easier to use potentially blocking OS services from tasks without blocking the entire Ada program.

So is your problem only that using OS threads is less "efficient" than switching and scheduling threads of control in the run-time system? If so, that seems to be a quality-of-implementation issue that could be solved in a compiler-specific way, and not an issue with the Ada language itself.

The point (from Emmanuel) that task entries cannot return indefinite types is certainly a language limitation, but seems to have little to do with the possible differences between tasks and co-routines, and could be addressed on its own if Ada users so desire.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 16:10:15 +0100

> So your co-routines would (1) have their own stack and (2) be independently schedulable, which implies (3) having their own execution context (register values, instruction pointer, etc.)

Sure. You should be able to implement communication logic in a natural way:

1. Read n bytes [block until finished]

2. Do things

3. Write m bytes [block until finished]

4. Repeat

> How is that different from the Ada concept of a "task"?

It is no different, that the whole point of deploying high level abstraction: task instead of low level one: state machine.

> How could the ARG separate between a "task" and a "co-routine" in the Ada RM?

Syntax sugar does not bother me. I trust ARG to introduce a couple of reserved words in the most annoying way... (:-))

> So is your problem only that using OS threads is less "efficient" than switching and scheduling threads of control in the run-time system?

This too. However the main purpose is control inversion caused by callback architectures. A huge number of libraries are built on that pattern. This is OK for the library provider because it is the most natural and efficient way. For the user implementing his own logic, be it communication protocol, GUI etc. it is a huge architectural problem as it distorts the problem space logic. So the goal is to convert a callback/event driven architecture into plain control flow.

> If so, that seems to be a quality-of-implementation issue that could be solved in a compiler-specific way, and not an issue with the Ada language itself.

In Ada 83 there was no way to pass a procedure as a parameter. We used a

task instead... (:-))

But sure, a possibility to delegate a callback to an entry call without intermediates is certainly welcome.

[...]

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 13 Feb 2023 16:43:31 +0100

> that task entries cannot return indefinite types is certainly a language limitation

That's what Holders are intended for... (changing indefinite types into a definite one)

From: Jeremy Grosser <jeremy@synack.me>
Date: Mon, 13 Feb 2023 08:40:05 -0800

> epoll is definitely the modern approach on Linux, until of course someone finds something even better.

For high performance networking, io_uring [1] is the new kid on the block, but the API involves a scary amount of pointer manipulation, so I'm not convinced that it's safe to use yet.

While epoll is thread safe, there are some subtleties. If you register a listening socket with epoll, then call epoll_wait from multiple threads, more than one thread may be woken up when the socket has a waiting incoming connection to be accepted. Only one thread will get a successful return from accept(), the others will return EAGAIN. This wastes cycles if your server handles lots of incoming connections. The recently added (kernel >=4.5) EPOLLEXCLUSIVE flag enables a mutex that ensures the event is only delivered to a single thread.

> They also have strong concerns about platform-agnostic support, and epoll is linux-specific at this point (likely also BSD). [...]

On BSD, the kqueue [2] API provides similar functionality to epoll. I believe kqueue is a better design, but you use what your platform supports.

libev [3] is the library I see used most commonly for cross-platform evented I/O. It will use the best available polling syscalls on whatever platform it's compiled for. Unfortunately, it's composed mostly of C preprocessor macros.

I've already written an epoll binding [5] that's in the Alire index. GNAT.Sockets provides the types and bindings for the portable syscalls.

For the Protohackers puzzles, I've written a small evented I/O server using those bindings [6]. Note that this server does not use events for the send() calls yet, which may block, though in practice it isn't an issue with the size of the payloads used in this application. I do plan to refactor this to buffer data to be sent when the Writable (EPOLLOUT) event is ready.

So far, I've found tasks and coroutines to be unnecessary for these servers, though coroutines would make it possible to implement Ada.Streams compatible Read and Write procedures, providing a cleaner interface that doesn't expose callbacks to the user.

[1] https://lwn.net/Articles/776703/

[2] https://people.freebsd.org/~jlemon/papers/kqueue.pdf

[3] https://linux.die.net/man/3/ev

[4] https://github.com/JeremyGrosser/epoll-ada

[5] https://github.com/JeremyGrosser/protohackers/blob/master/src/mini.adb

From: philip...@gmail.com <philip.munts@gmail.com>
Date: Mon, 13 Feb 2023 17:55:52 -0800

> In an effort to better learn network programming in Ada, I've been working through the Protohacker Challenges (https://protohackers.com/), and the current challenge (number 3) is to create a chat server.

I know it probably defeats the purpose of what you are trying to learn, but you are going to wind up just reinventing AMQP (broker based, meaning there is a intermediary computer running something like RabbitMQ to manage message queues) or ZeroMQ (brokerless), both implementations of so-called enterprise messaging protocols. Both seem to scale pretty well to thousands of clients.

It is pretty easy to do an Ada thin binding for the ZeroMQ C library libzmq.

From: A.J. <ianozia@gmail.com>
Date: Sat, 18 Feb 2023 17:27:02 -0800

Thank you for all of the responses and discussion, it pointed me in the right direction! The "chat server"[1] (if you could call it that) does work, and my friends and I were able to telnet into it and chat. One of my friends even tried throwing things at the server to break it, but it didn't crash!

Dmitry, maintaining a list of clients was the vital part I was missing. I played around with using synchronized queues and tasks, but ended up defaulting to an ordered map with a UID as the key and wrapped it in a protected type. I couldn't get Send() to send more data than Available_To_Send (after calling it, Available_To_Send ended up returning 0, and continued to do so despite wrapping Send() in a loop), but increasing the send buffer to 8kb per connection worked fine. I would simply loop through that ordered map each time I needed to send something to all of the clients.

I really like simple components, and it would be neat if the GNAT maintainers implement epoll in the backend for Linux systems, kqueue for BSD and MacOS. Any server I write will be for Linux though anyway. I'm also interested in trying to benchmark Simple Component's connections server (both pooled and standard) against epoll to see how it fares. Perhaps the clever tasking that the Connections Server utilizes can keep up with epoll despite what GNAT.Sockets utilizes!

Regarding coroutines vs tasks, I think at a high level it's hard to differentiate, but at a lower level, when I think of tasks vs what a coroutine would be, I think of Go, and their "goroutines."[2] Creating a task in Ada, at least on Linux, ends up creating a pthread, and you get all of the overhead that comes with threading (it's initialized in the kernel). coroutines are managed by the go runtime (I believe in user space) and have much less overhead to create or manage, since it's not creating a specific thread.

Ada 202x supports the "parallel" block[3] though I understand no runtime has utilized it yet-- would that end up being a coroutine or is it meant for something else?

[1] https://github.com/AJ-Ianozi/protohackers/tree/main/budget_chat/src

[2] https://www.geeksforgeeks.org/golang-goroutine-vs-thread/

[3] http://www.ada-auth.org/standards/22rm/html/RM-5-6-1.html

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 19 Feb 2023 16:37:51 +0200

> Creating a task in Ada, at least on Linux, ends up creating a pthread

With the current GNAT compiler, yes. But not necessarily with all Ada compilers, even on Linux.

> coroutines are managed by the go runtime (I believe in user space) and have much less overhead to create or manage

Some Ada compilers may have run-times that implement Ada tasks within the run-time, with minimal or no kernel/OS interaction.

> Ada 202x supports the "parallel" block [...]-- would that end up being a coroutine or is it meant for something else?

As I understand it, the parallel execution constructs (parallel blocks and parallel loops) in Ada 2022 are meant to parallelize computations using multiple cores -- that is, real parallelism, not just concurrency.

The Ada2022 RM describes each parallel computation in such a parallel construct as its own thread of control, but all operating within the same task, and all meant to be /independent/ of each other. For example, a computation on a vector that divides the vector into non-overlapping chunks and allocates one core to each chunk.

Within a parallel construct (in any of the parallel threads) it is a bounded error to invoke an operation that is potentially blocking. So the independent computations are not expected to suspend themselves, thus they are not co-routines.

The parallelism in parallel blocks and parallel loops is a "fork-join" parallelism. In other words, when the block or loop is entered all the parallel threads are created, and all those threads are destroyed when the block or loop is exited.

So they are independent threads running "in" the same task, as Dmitry wants, but they are not scheduled by that task in any sense. The task "splits" into these separate threads, and only these, until the end of the parallel construct.

Moreover, there are rules and checks on data-flow between the independent computations, meant to exclude data races. So it is not intended that the parallel computations (within the same parallel construct) should form pipes or have other inter-computation data flows.

Ada Array Contiguity

From: Rod Kay <rodakay5@gmail.com>
Subject: Ada array contiguity.
Date: Mon, 20 Feb 2023 00:34:55 +1100
Newsgroups: comp.lang.ada

I've been told that Ada array elements are not guaranteed to be contiguous unless the 'Convention C' aspect is applied.

Is this correct?

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 19 Feb 2023 15:28:23 +0100

The strength of Ada is that it protects you from all implementation details, thus allowing compilers to choose the most efficient implementation. Therefore, the answer is yes.

(BTW: try to find a definition of "contiguous". At byte level? At word level? What if the element does not fill a byte?)

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 19 Feb 2023 16:59:42 +0200

> Therefore, the answer is yes.

I tried to find a rule on "contiguity" in the Ada 2022 RM, but failed. Can you point to one? Perhaps this rule is a consequence of C standard rules for arrays (pointer arithmetic), and the general idea that Ada should allow Convention C for a type only if that type is really compatible with the C compiler (in question).

For a constrained array type I would choose to specify the size of the component type, and the size of the array type to be the length of the array times the component size. That should (also) ensure that the elements are stored contiguously (if the Ada compiler accepts this size specification).

It seems (RM B.3(62.4/3)) that Ada compilers are not required to support Convention C for unconstrained array types. RM B.3 (Interfacing with C/C++) declares such types with the Pack aspect, but that may or may not (AIUI) give a contiguous representation.

> (BTW: try to find a definition of "contiguous". At byte level? At word level? What if the element does not fill a byte?)

Indeed. But it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 19 Feb 2023 16:08:09 +0100

> it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

Rather: the bit offset of an element is a linear function of its position.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 19 Feb 2023 18:10:44 +0100

> it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

Certainly not if Comp'Size is not an integer number of bytes.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 19 Feb 2023 19:54:13 +0200

> Certainly not if Comp'Size is not an integer number of bytes.

I'm not so certain. By choosing various roundings-up of the component size, you can choose between "bit-contiguous", "byte-contiguous", etc.

For example, bit-contiguous with 2-bit components:

```
type Comp is (A, B, C, D) with Size => 2;
type Arr is array (1 .. 10) of Comp
  with Pack, Size => 10 * Comp'Size;
```

Nybble-contiguous with Comp'Size => 4, byte- (octet-) contiguous with Comp'Size => 8, etc.

(However, I haven't checked that eg. GNAT does the "right thing" with such Size clauses, just that it accepts them. It does require the Pack aspect for the array type when Comp'Size is not a multiple of 8.)

> Rather: the bit offset of an element is a linear function of its position.

That is ordering by index, but not contiguity: there may still be gaps between elements. However, I assume you meant that the slope of the linear function equals the component size, and then it includes contiguity.

The relationship of index order to memory-location order is certainly an aspect that should be considered when interfacing to C or HW.

Pet peeve: on more than one occasion I have been disappointed that Ada representation clauses do not let me specify the index-order of packed array elements in a word, relative to the bit-numbering order, and I have had to fall back to using several scalar-type record components, c1 .. c7 say, instead of one array-type component, c(1..7).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 19 Feb 2023 20:05:28 +0100

> That is ordering by index, but not contiguity: there may still be gaps between elements. [...]

No gaps = packed = the most dense representation.

Contiguity is rather that the gaps are regular and can be considered a part of each element. E.g. a video buffer with strides is not contiguous.

> The relationship of index order to memory-location order is certainly an aspect that should be considered when interfacing to C or HW.

An definition of contiguous array equivalent to linearity is that the array body representation is isomorphic to slicing.

> Pet peeve [...]

This is as blasphemous as asking for n-D slices... (:-))

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 19 Feb 2023 23:02:36 +0100

> I've been told that Ada array elements are not guaranteed to be contiguous unless the 'Convention C' aspect is applied.

The ARM says little about how the compiler represents objects in the absence of representation clauses. However, ARM 13.7(12) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-13-7-1.html#I5653) says, "Storage_Array represents a contiguous sequence of storage elements."

ARM 13.9(17/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-13-9.html#I5679) says that a compiler that supports Unchecked_Conversion should use a contiguous representation for certain constrained array subtypes.

Using convention Fortran should also ensure a contiguous representation, add can apply (unlike convention C) to multidimensional arrays.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 20 Feb 2023 08:12:41 +0100

> type Comp is (A, B, C, D) with Size => 2;

> type Arr is array (1 .. 10) of Comp

> with Pack, Size => 10 * Comp'Size;

> Nybble-contiguous with Comp'Size => 4, byte- (octet-) contiguous with Comp'Size => 8, etc.

Of course, if you add representation clauses, the compiler will obey them. But the OP's question was whether it was /guaranteed/ to have contiguous representation, and the answer is no - for good reasons.

From: Rod Kay <rodakay5@gmail.com>
Date: Thu, 2 Mar 2023 00:22:25 +1100

Thank you all for the replies.

To summarise then, contiguity is not guaranteed unless the array is of convention C, convention Fortran or representation clauses are applied.

Ada.Containers.Vectors Capacity

From: Rod Kay <rodakay5@gmail.com>
Subject: Is this a compiler bug?
Date: Sun, 19 Mar 2023 17:17:20 +1100
Newsgroups: comp.lang.ada

Came across this during a port of the Box2D physics engine.

It's a generic Stack package using 'ada.Containers.Vectors' to implement the stack.

One generic parameter is the 'initial_Capacity' of the stack, used in the 'to_Stack' construction function, via the Vectors 'reserve_Capacity' procedure.

In the 'to_Stack' function, the Capacity is reserved correctly but in the test program when the stack is created and assigned to a variable, the capacity is 0.

Here is the (very small) source code ...

https://gist.github.com/charlie5/7b4d863227a510f834c2bfd781dd50ba

The output I get with GCC 12.2.0 is ...

```
[rod@orth bug]$ ./stack_bug

to_Stack ~ Initial Capacity: 256

to_Stack ~ Before reserve:  0

to_Stack ~ After reserve:  256

stack_Bug ~ Actual Capacity: 0
```

Regards.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 19 Mar 2023 11:33:50 +0100

I think this is acceptable behavior. See ARM A.18.2 (147.19/3, 147.20/3, & 147.b/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-A-18-2.html). The first two sections define the behavior of procedure Assign, while the last states "Assign(A, B) and A := B behave identically".

Assign (A, B) only changes the capacity of A if A.Capacity < B.Length.

So if the compiler does not use build-in-place for the initialization of the variable, then the assignment of the function result should not change the capacity of the variable from its (apparent) default of zero (there is, of course, no requirement for the capacity of a default-initialized vector).

The discussion of capacities for vectors is only meaningful for a subset of possible implementations, so messing with capacities may have no meaningful effect at all.

For an unbounded stack based on a linked list (with no concept of capacity) you could use PragmARC.Data_Structures.Stacks.Unbounded.Unprotected (https://github.com/jrcarter/PragmARC/blob/Ada-12/pragmarc-data_structures-stacks-unbounded-unprotected.ads).

From: Rod Kay <rodakay5@gmail.com>
Date: Mon, 20 Mar 2023 13:24:40 +1100

Thank you, Jeffrey, for the detailed reply.

I'm now using a limited record with an extended return for 'build-in-place' initialisation and am getting the behavior I desired.

Why Don't All Initialising Assignments Use 'build-in-place'?

From: Rod Kay <rodakay5@gmail.com>
Subject: Why don't all initialising assignments use 'build-in-place' ?
Date: Tue, 21 Mar 2023 23:06:03 +1100
Newsgroups: comp.lang.ada

I'm sure there must be a good reason. All I can think of is that it may somehow break backwards compatibility wrt controlled types (a vague stab in the dark).

Any thoughts?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 25 Mar 2023 03:39:14 -0500

(1) Didn't want to make work for implementers.

(2) You shouldn't be able to tell (since it is required for all cases involved finalization). Finalization is the only way to inject user-defined code into the initialization process.

(3) True build-in-place can be expensive and complex (especially for array types).

(4) Build-in-place requires functions compiled to support it (must pass in the place to initialize into). That might not be the case (especially if a foreign convention is involved). Also see (3) - an implementation might have a cheaper way to return some types that doesn't support build-in-place.

There's probably more, those are off the top of my head. If it is cheap, it would be silly for an implementation to do anything else. (Don't ask what Janus/Ada does. ;-) Otherwise, most people want the fastest possible code.

From: Rod Kay <rodakay5@gmail.com>
Date: Sun, 26 Mar 2023 16:10:33 +1100

Thanks, Randy. I somehow imagined that build-in-place would be faster :/.

So using 'extended return' *everywhere* would decrease performance, I guess.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 26 Mar 2023 12:41:00 +0200

> So using 'extended return' *everywhere* would decrease performance, I guess.

You seem to think that using an extended return requires building in place. This is not required by the ARM.

"Built in place" is defined in ARM 7.6 (17.1/3-17.p/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-7-6.html#I4005). An initial value is required to be built in place when

1. The object (or any part of the object) being initialized is immutably limited
2. The object (or any part of the object) being initialized is controlled and the initialization expression is an aggregate

In all other cases, it is up to the compiler to decide whether or not to build in place.

This holds regardless of the the kind of return statement used if the initialization expression is a function call.

Thus the initialization of an immutably limited object is done in place even if the initialization expression is

* an aggregate

* a function call with a simple return statement

while the initialization of an integer object may be by copy even if the initialization expression is a function call with an extended return statement.

From: Rod Kay <rodakay5@gmail.com>
Date: Mon, 27 Mar 2023 15:44:33 +1100

> You seem to think that using an extended return requires building in place. This is not required by the ARM.

Yes, I did rather think that. Appreciate the correction.

Assignment Access Type with Discriminants

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Assignment access type with discriminants
Date: Wed, 22 Mar 2023 10:19:28 +0100
Newsgroups: comp.lang.ada

I stumbled on a curious fact.

The value of an object with a discriminant can be changed to a value with a different discriminant if the type's discriminants are defaulted.

Right?

Wrong! Not through an access type!

```
procedure Test is
   type F is (F1, F2, F3);
   type Foo (K : F := F1) is record
      case K is
         when F1 =>
            X1 : Integer;
         when F2 =>
            X2 : Float;
         when F3 =>
            X3 : String (1..2);
      end case;
   end record;
   type Foo_Ptr is access all Foo;
   X : aliased Foo;
   P : Foo_Ptr := X'Access;
begin
   X := (F2, 1.0);   -- OK
   P.all := (F1, 3); -- Error!
end Test;
```

Is this a compiler bug or intentional language design? Any language lawyers?

From: Björn Lundin <bnl@nowhere.com>
Date: Wed, 22 Mar 2023 10:31:58 +0100

> I stumbled on a curious fact. [...] Is this a compiler bug or intentional language design? Any language lawyers?

I get

```
Execution of ./test terminated by
unhandled exception

raised CONSTRAINT_ERROR :
test.adb:18 discriminant check failed

Call stack traceback locations:

0x402c33 0x402b27 0x7f335b5cfd8e
0x7f335b5cfe3e 0x402b63
0xfffffffffffffffe

bnl@hp-t510:/usr2$ gnatls -v

GNATLS Pro 22.2 (20220605-103)
```

Linux 64bit - ubuntu 22.04

So it is (also) present on that platform at least

From: G.B.
<bauhaus@notmyhomepage.invalid>
Date: Wed, 22 Mar 2023 15:10:44 +0100

Some experiments point at the general access type.

```
   type Foo_Ptr is access Foo; -- sans `all`
   X : Foo;
   P : Foo_Ptr := new Foo;
   type Foo1 is new Foo_Ptr (K => F1);
begin
   X := (F2, 1.0);   -- OK
   P.all := (F1, 3); -- _no_ Error!
   Foo1 (P).all := (F1, 3);
end Test;
```

(Doesn't rejection for general access types seem reasonable if assignment would otherwise require adjusting the storage layout of a variable, including all access paths to components? Just guessing.)

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Thu, 23 Mar 2023 12:51:03 +0100

> Some experiments point at the general access type.

You get no error because you do not change the discriminant. Change your code to:

```
    P.all := (F2, 1.0); -- Error!
```

> (Doesn't rejection for general access types seem reasonable if assignment would otherwise require adjusting the storage layout of a variable, including all access paths to components?

I guess that an implementation must allocate memory for any value unless you constraint the discriminants in a subtype. But I am not a language lawyer to judge.

From: Adamagica
<christ-usch.grein@t-online.de>
Date: Thu, 23 Mar 2023 09:53:23 -0700

I do hope, this answers the question:

3.10(14/3) … The first subtype of a type defined by … an access_to_object_definition is unconstrained if the designated subtype is an ... discriminated subtype; otherwise, it is constrained.

4.8(6/3) If the designated type is composite, then … the created object is constrained by its initial value (even if the designated subtype is unconstrained with defaults).

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Thu, 23 Mar 2023 20:09:21 +0200

> I do hope, this answers the question:
>
> 3.10(14/3) … The first subtype of a type defined by … an access_to_object_definition is unconstrained if the designated subtype is an ... discriminated subtype; otherwise, it is constrained.

What do you infer from this, relating to Dmitry's original example code and the error? The "first subtype .. defined" here is the access subtype, and I don't see how that affects an assignment /via/ this access subtype to the accessed object.

(It is not clear to me how an access subtype that is constrained differs from one that is unconstrained. Can someone clarify?)

> 4.8(6/3) If the designated type is composite, then … the created object is constrained by its initial value (even if the designated subtype is unconstrained with defaults).

That rule applies to objects created by allocators, but the original example code has no allocators (some later variants do). The object in question is created by a declaration (which includes the "aliased" keyword), not by an allocator.

Also, AARM 3.10 contains the following notes on "Wording Changes from Ada 1995":

26.d/2 {AI95-00363-01} Most unconstrained aliased objects with defaulted discriminants are no longer constrained by their initial values. [...]

26.k/2 {AI95-00363-01} The rules about aliased objects being constrained

by their initial values now apply only to allocated objects, and thus have been moved to 4.8, “Allocators”.

This seems to mean that aliased objects created by declarations are /not/ constrained by the initial value, so it should be possible to change the discriminant. This seems to be a change from Ada 95 to Ada 2005. I don't see why that change could not be done via an access to the object.

I added some output to Dmitry's original code, with this result:

Date: Thu, 23 Mar 2023 20:55:48 +0200

```
   X'Constrained = FALSE
   P'Constrained = TRUE
   P.all'Constrained = TRUE
```

The first two values of 'Constrained (for X and P) are as expected by the RM rules, and the third value (for P.all) is consistent with the error, and seems valid for Ada 95, but the wording change quoted above suggests that it is wrong for Ada 2005 and later. This leads me to suspect that GNAT has not been fully updated for this RM change, so it would be a GNAT bug. Still, the addition of

```
   subtype Foo2_Ptr is Foo_Ptr (K => F2);
```

to Dmitry's original example provokes this error message:

```
   fuf.adb:16:24: access subtype of
   general access type not allowed

   fuf.adb:16:24: discriminants have
   defaults
```

which suggests that at least this part of AI95-00363 has been implemented, as noted in AARM 3.10:

14.b/2 Reason: {AI95-00363-01} [...] Constraints are not allowed on general access-to-unconstrained discriminated types if the type has defaults for its discriminants [...]

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 23 Mar 2023 18:04:36 +0100

> I stumbled on a curious fact.

An access value is always constrained by its initial value; this is necessary because of constrained access subtypes. Here is a slightly modified version of your example:

```
procedure Test is
   type F is (F1, F2, F3);

   type Foo (K : F := F1) is record
      case K is
         when F1 =>
            X1 : Integer;
         when F2 =>
            X2 : Float;
         when F3 =>
            X3 : String (1..2);
      end case;
   end record;
   type Foo_Ptr is access all Foo;
   type Foo_Ptr2 is access Foo;
   X : aliased Foo;
   P : Foo_Ptr := X'Access;
   PF2: Foo_PTR2 (F2);
begin
   X := (F2, 1.0);   -- OK
   PF2 := new Foo (F2);
   P := PF2.all'Access;
   P.all := (F1, 3); -- Error!
end Test;
```

Without this rule, PF2.all would now designate a value whose discriminant is F1!

> An access value is always constrained by its initial value; this is necessary because of constrained access subtypes.

But constrained access subtypes are not allowed for general access types like Foo_Ptr in the example.

> Here is a slightly modified version of your example:

> [...]

> Without this rule, PF2.all would now designate a value whose discriminant is F1!

This error is understandable and valid, because now P.all is PF2.all which is an allocated object and therefore constrained by its initial value with K = F2.

But why should the same apply when P designates X, which is unconstrained? Is it just an optimization (in the RM) so that a general access value does not have to carry around a flag showing whether its designated object is constrained or unconstrained?

Perhaps it would be better to make the assignment P := PF2.all'Access illegal, because it in effect converts a constrained access value (PF2) to an unconstrained access subtype (P), and so in some sense violates the prohibition of constrained subtypes of general access types.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 23 Mar 2023 20:53:02 +0100

> Perhaps it would be better to make the assignment P := PF2.all'Access illegal [...]

Yes this is a substitutability violation. Such cases never go without a punishment. In this case it is an implementation overhead.

Consider:

```
procedure Set (Destination : in out Foo;
Source : Foo) is
begin
  Destination := Source;
end Set;
```

The compiler cannot implement Set in a natural way, because Destination might be arbitrarily constrained by the caller. E.g. when the actual for Destination is P.all. So, the constraint must be passed together with the actual. Quite a burden.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 24 Mar 2023 10:41:20 +0100

> But why should the same apply when P designates X, which isunconstrained? [...]

I didn't dig in the RM in all details, but I think this comes from the fact that being constrained (always) is a property of the pointer (more precisely, its subtype), not of the pointed-at object.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 25 Mar 2023 03:51:07 -0500

The rule is question is 4.1(9/3):

If the type of the name in a dereference is some access-to-object type T, then the dereference denotes a view of an object, the nominal subtype of the view being the designated subtype of T. If the designated subtype has unconstrained discriminants, the (actual) subtype of the view is constrained by the values of the discriminants of the designated object, except when there is a partial view of the type of the designated subtype that does not have discriminants, in which case the dereference is not constrained by its discriminant values.

We have to do that so as otherwise the access value would have to carry a designation as to whether the object was allocated or not.

This rule was inherited from Ada 83.

IMHO, this rule is stupid. It's even more stupid with the hole for types that have partial views without discriminants. The *proper* solution is to get rid of the rarely used and mostly useless access constraints, and then have no extra restrictions on access values. But that's considered too incompatible.

Ada in Jest

Ada Lovelace Cosplay

From: Mockturtle <framefritti@gmail.com>
Subject: Ada Lovelace cosplay
Date: Mon, 16 Jan 2023 09:48:58 -0800
Newsgroups: comp.lang.ada

Well, yes, someone cosplayed Ada...

https://blog.adafruit.com/2013/10/24/from-scratch-ada-lovelace-costume/