ADA USER JOURNAL

Volume 42
Number 2
June 2021

Contents

Quarterly News Digest

Alejandro R. Mosteo

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

Contents

[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,

This period saw the celebration of the Ada-Europe conference, after a year of hiatus, and only in virtual form. This is cause for celebration and a signal of hope for Adaists to meet again in the future in this close-knit event where one can mingle with newcomers, old faces, and the "guiding lights" of the language alike. Announcements about the event, for the record, are found in this number [1].

Another yearly moment of excitement for the Ada open source community is the release of the GNAT Community Edition, which we witnessed at the end of May [2]. And, speaking of open source communities, I will mention the mass exodus from the Freenode chat network due to a change in ownership and policies. The dwellers of #ada have chosen, on the 20th anniversary of the channel, the Libera Chat network as a new home. The thread announcing the news [3] is also a reminiscence about other venerable technologies that, like IRC and Usenet, still are going around.

In another curious development, a mostly off-topic thread that had seen its last post in 2014 was somehow revived, and I cannot resist reading about the computing anecdotes of times long past, in this case framed in a "Pascal vs C" context [4].

The polemic topic about Ada itself in this number was Unicode, or Ada lackluster support thereof, according to some. Opinions, hopes for a brighter future, and insights on how it came to be in its present form are discussed in [5].

Sincerely,
Alejandro R. Mosteo.

[1] "Ada-Europe Int. Conf. on Reliable Software Technologies, AEiC 2021", in Ada-related Events.

[2] "GNAT CE 2021", in Ada-related Tools.

[3] "Ada IRC Channel Migrates to Libera Chat", in Ada Practice.

[4] "Pascal vs C Language Families", in Ada and Other Languages.

[5] "Ada and Unicode", in Ada Practice.

Ada-related Events

Ada-Europe Int. Conf. on Reliable Software Technologies, AEiC 2021

[Past event, for the record. —arm]

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Int.Conf. Reliable Software Technologies, AEiC 2021
Date: Tue, 27 Apr 2021 17:46:30 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

---

Call for Participation

*** PROGRAM SUMMARY ***

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021)

7-10 June 2021, Virtual Event

www.ada-europe.org/conference2021

Organized by University of Cantabria and Ada-Europe in cooperation with ACM SIGAda, SIGPLAN, SIGBED and the Ada Resource Association (ARA)

#AEiC2021 #AdaEurope #AdaProgramming

---

General Information

The 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021), initially scheduled to take place in Santander, Spain, will be held online from the 7th to the 10th of June 2021, using the underline.io conference platform.

The conference program includes parallel tutorials on Monday 7th, and a technical program and vendor exhibition from Tuesday to Thursday. The conference also includes breaks and virtual social events that will allow networking among the participants.

Overview of the Week

Monday 7th

- Welcome Social Event
- 5 Parallel Tutorials
- Ice-Breaking Social Event

Tuesday 8th

- Ice-Breaking Social Event and Opening
- Techn. Session 1: Scheduling and mixed-criticality systems
- Keynote 1
- Techn. Session 2: Software modeling
- Social Event

Wednesday 9th

- Welcome Social Event
- Techn. Session 3: Autonomous systems
- Work-in-Progress Session
- Keynote 2
- Techn. Session 4: Ada issues and Ravenscar
- Social Event

Thursday 10th

- Welcome Social Event
- Techn. Session 5: Validation and verification tools
- Techn. Session 6: Emerging applications with reliability requirements
- Keynote 3
- Techn. Session 7: Safety challenges
- Best Presentation Award, Closing Session and Party

The program runs between 12:30 and 18:30 CEST, to allow participation from different time zones. For full details and up-to-date information, see the conference web page: http://www.ada-europe.org/ conference2021

Keynote Talks

In each of the three main conference days, a keynote will be delivered to address hot topics of relevance in the conference scope, with ample time for questions and answers. The keynotes will be:

- Ángel Conde, Data Analytics and Artificial Intelligence team leader at IKERLAN (Spain), who will present his work on "Software reliability in the Big Data era with an industry-minded focus".
- Alfons Crespo, who is with the Institute of Automation and Industrial Informatics of the Universitat Politècnica de València (Spain), will give an answer to the question "Why hypervisor-based approach is the best alternative for mixed-criticality systems".
- Tucker Taft, who is Director of Language Research at AdaCore (USA), will talk on "A sampling of Ada 2022".

Technical Sessions

Given the current sanitary situation and the need to resort to a virtual format for the conference, we will all experience the advantages and benefits of exploring new formats. The technical sessions are designed with the flipped-conference concept, where the audience can access the pre-recorded presentation materials in advance and the live sessions are devoted to short presentations of the highlights of each contribution, allowing ample time for questions and answers with the presenter. The recorded materials will also be available for some time after their sessions. The technical sessions include papers submitted to the journal track that are heading towards final acceptance and open-access publication, together with industrial, invited and vendor presentations.

Work-in-Progress Session

The Work-in-Progress session contains contributions of evolving and early-stage ideas, or new research directions. They are presented in a special session consisting of a round of very short presentations of the highlights of each contribution, followed by a poster session in the same virtual space where the breaks are held.

Exhibition

From Tuesday to Thursday the conference platform will provide access to virtual booths where participants will be able to find information on the conference exhibitors and chat with them or request meetings. The virtual break lounge where the breaks and social events will take place will also have a space for meeting with the exhibitors.

Tutorials

Five four-hour parallel tutorials are offered on Monday 7th:

- TU-1: Programming mobile robots with ROS2 and the RCLAda Ada client library, by Alejandro R. Mosteo
- TU-2: Introduction to the development of safety critical software, by Jean-Pierre Rosen
- TU-3: Parallel programming with Ada and OpenMP, by Sara Royuela, S. Tucker Taft, Luis Miguel Pinho
- TU-4: Timing verification from UML & MARTE design models: techniques & tools, by Laurent Rioux, Julio Medina and Shuai Li
- TU-5: Programming shared memory computers, by Jan Verschelde

Social Program

The virtual conference platform will offer a space under the gather.town environment to allow informal and lively gathering of the participants. This space may have different areas, such as rooms, tables, and corners where a participant can approach to talk through videoconferencing with participants in the same virtual area. This facility will be used for the breaks, poster session, exhibition and social events. Particular themes for some of the social events will be announced in the conference platform and in the web page.

Further Information

Participation for the full event, including tutorials, is free for Ada-Europe members and only 60 EUR for all others. Registration is required for all. The conference web page will shortly give full and up-to-date details on the program, the registration process and the virtual platform: http://www.ada-europe.org/conference2021

AEiC 2021 Sponsors

- AdaCore: https://www.adacore.com/
- Ellidiss: https://www.ellidiss.com/
- PTC: http://www.ptc.com/ developer-tools
- Universidad de Cantabria: https://web.unican.es/en/
- Vector: https://www.vector.com/at/en/

The conference is supported and sponsored by

- Ada-Europe: http://www.ada-europe.org/

and organized in cooperation with

- ACM SIGAda: http://www.sigada.org/
- ACM SIGBED: https://sigbed.org/
- ACM SIGPLAN: http://www.sigplan.org/
- ARA: https://www.adaic.org/community/

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2021 Publicity Chair (aka Ada-Europe 2021)

Dirk.Craeynest@cs.kuleuven.be

(V4.1)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Fri, 30 Apr 2021 07:59:16 -0000

Registration site for AEiC2021 is online: registration.ada-europe.org.

Register now for the 25th Ada-Europe Int'l Conference on Reliable Software Technologies!

Press Release - AEiC 2021, Ada-Europe Reliable Softw. Technol.

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Press Release - AEiC 2021, Ada-Europe Reliable Softw. Technol.
Date: Sun, 30 May 2021 16:22:28 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

FINAL Call for Participation

*** UPDATED Program Summary ***

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021)

7-10 June 2021, Virtual Event

www.ada-europe.org/conference2021

*** Check out tutorials! ***

www.ada-europe.org/conference2021/ tutorials.html

*** Don't miss the thematic social events on Tuesday and Wednesday ***

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

*** Register now! ***

#AEiC2021 #AdaEurope #AdaProgramming

Press release:

25th Ada-Europe Int'l Conference on Reliable Software Technologies International experts meet in virtual conference hosted by Underline Santander, Spain (31 May 2021) - Ada-

Europe together with the University of Cantabria, Spain organize from 7 to 10 June 2021 the 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021). The conference was initially scheduled to take place in Santander, Spain. According to the safety and sanitary measures under the COVID-19 pandemic, this year the conference will be a virtual event, hosted by Underline (https://underline.io). The event is in cooperation with the Ada Resource Association (ARA), and with ACM's Special Interest Groups on Ada (SIGAda), on Embedded Systems (SIGBED) and on Programming Languages (SIGPLAN).

The Ada-Europe series of conferences has over the years become a leading international forum for providers, practitioners and researchers in reliable software technologies. These events highlight the increased relevance of Ada in general and in safety- and security-critical systems in particular, and provide a unique opportunity for interaction and collaboration between academics and industrial practitioners.

This year's conference offers 5 tutorials, 3 keynotes, a technical program of 7 sessions with refereed papers, invited and industrial presentations, a work-in-progress session, an industrial exhibition and vendor presentations, and a social program.

Five parallel tutorials are scheduled on Monday, targeting different audiences:

- "Programming mobile robots with ROS2 and the RCLAda Ada client library", by Alejandro R. Mosteo;
- "Introduction to the development of safety critical software", by Jean-Pierre Rosen;
- "Parallel programming with Ada and OpenMP", by Sara Royuela, S. Tucker Taft, Luis Miguel Pinho;
- "Timing verification from UML & MARTE design models: techniques & tools", by Laurent Rioux, Julio Medina and Shuai Li;
- "Programming shared memory computers", by Jan Verschelde.

Tutorial registration is complementary for conference participants.

The industrial exhibition opens Tuesday under the Expo area in the virtual platform and also in the Lounge, which is the networking area. It runs until the end of Thursday afternoon. Exhibitors include AdaCore, PTC Developer Tools, and Ada-Europe. All conference participants are invited to the exhibition as well as to the virtual social events.

Three eminent speakers have been invited to deliver a keynote at each of the core conference days:

- Ángel Conde, Data Analytics and Artificial Intelligence team leader at IKERLAN (Spain), who will present his work on "Software reliability in the Big Data era with an industry-minded focus";
- Alfons Crespo, who is with the Institute of Automation and Industrial Informatics of the Universitat Politècnica de València (Spain), will give an answer to the question "Why hypervisor-based approach is the best alternative for mixed-criticality systems";
- Tucker Taft, who is Director of Language Research at AdaCore (USA), will talk on "A sampling of Ada 2022".

The technical program from Tuesday to Thursday presents 13 refereed technical papers and 5 invited, 6 industrial and 4 vendor presentations in sessions on:

- Scheduling and mixed-criticality systems,
- Software modeling,
- Autonomous systems,
- Ada issues and Ravenscar,
- Validation and verification tools,
- Emerging applications with reliability requirements,
- Safety challenges.

In addition, there is a work-in-progress session including 8 presentations and associated posters.

Peer-reviewed papers have been submitted to a special issue of the Journal of Systems Architecture and are heading towards final acceptance as open-access publications. Industrial and work-in-progress presentations, together with tutorial abstracts, will be offered publication in the Ada User Journal, the quarterly magazine of Ada-Europe.

The social program is hosted in a space under the gather.town environment that allows informal and lively gathering of the participants. This space has different areas, such as rooms, tables, and corners where a participant can approach to talk through videoconferencing with participants in the same virtual area. This facility will be used for the breaks, poster session, exhibition and social events. Don't miss the thematic social events at the end of each core conference day.

The Best Presentation Award will be offered during the Closing session.

The full program is available on the conference web site. Online registration is still possible.

Latest updates:

The "Final Program" is available at www.ada-europe.org/conference2021 /final-program.pdf.

Check out the tutorials in the PDF program, or in the schedule at

www.ada-europe.org/conference2021/ tutorials.html.

Registration fees are lower than ever and the registration process is done on-line. Don't delay for all details, select "Registration" at www.ada-europe.org/ conference2021 or go directly to https://registration.ada-europe.org.

The technical sessions are designed with the flipped-conference concept, where the audience can access pre-recorded presentation materials in advance. The live sessions are devoted to short presentations of the highlights of each contribution, allowing ample time for questions and answers with the presenter. The recorded materials will also be available for some time after their sessions.

The program runs between 12:30 and 18:30 CEST, to allow participation from different time zones. For more info and latest updates see the conference web site at www.ada-europe.org/conference2021.

AEiC 2021 is sponsored by AdaCore (www.adacore.com), Ellidiss (www.ellidiss.com), PTC Developer Tools (www.ptc.com/developer-tools), Universidad de Cantabria (web.unican.es/en), and Vector (www.vector.com/at/en).

Help promote the conference by advertising it.

Recommended Twitter hashtags: #AdaEurope and/or #AEiC2021.

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2021 Publicity Chair (aka Ada-Europe 2021),

Dirk.Craeynest@cs.kuleuven.be

* 25th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2021)

* June 7-10, 2021 * online event * www.ada-europe.org/conference2021 **

(V6.1)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Sun, 6 Jun 2021 10:11:09 -0000

If you plan to attend one of the #AEiC2021 #tutorials on Mon 7 Jun, don't forget to check the prerequisites: you may have to download material preferably before the tutorial starts.

See you at #AdaEurope's #OnlineConference soon!

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Thu, 10 Jun 2021 07:43:05 -0000

#AEiC2021 #AdaEurope #OnlineConference #AdaProgramming

Don't miss today's keynote!

Tucker Taft will present "A sampling of Ada 2022"

Abstract:

The forthcoming Ada 2022 revision of the Ada standard includes significant new features, which together make the language more expressive and productive in a multicore context, while enhancing its safety and support for more complete abstractions with formal contracts.

This talk will introduce these key new features with a series of examples:

- parallel loops and blocks, coupled with static detection of data races and potential blocking
- iterator syntax for incorporating filters and user-defined iterator procedures-libraries for atomic operations, including fetch-and-add and compare-and-swap
- aggregates, literals, images, and map-reduce for user-defined types
- libraries for arbitrary precision integer and rational arithmetic
- more expressive contracts using delta aggregates and declare expressions
- the Jorvik profile as the next step up from the Ravenscar profile

Call for Paper in CodeLand

From: Mockturtle <framefritti@gmail.com>
Subject: Call for Paper in CodeLand
Date: Sat, 26 Jun 2021 09:04:37 -0700
Newsgroups: comp.lang.ada

I just "tripped over" a CFP for the event CodeLand (link at the end). They accept proposals for 15-minutes pre-recorded video talk; deadline 20th of July.

CodeLand's primary audience is early-career programmers and their mentors. Among the themes I see "Technical Deep Dives" and "Path to Programmer" that could maybe be suitable for something Ada-related.

Is maybe someone interested in proposing something? I am going to think about it...

Summary of the most important info

* When: September 23-24, 2021

* How: virtual only, pre-recorded video

* Deadline: July 20, 2021 at 11:59pm UTC.

* Acceptance/reject date: August 17, 2021.

* Themes

- Code for Good
- Early-career confidence
- Open source strong
- Path to programmer
- Technical deep dives

* More info: https://cfp.codelandconf.com/events/codeland-2021

Accepted speakers will be asked to participate in a panel session (suggested but not required). They should be prepared to answer moderated attendee questions about their talk.

New Competition: Ada/SPARK Crate of the Year Award

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Subject: New competition: Ada/SPARK Crate Of The Year Award
Date: Mon, 28 Jun 2021 03:11:38 -0700
Newsgroups: comp.lang.ada

I am happy to announce AdaCore's new programming competition: The Ada/SPARK Crate Of The Year Award!

The announcement is here: https://blog.adacore.com/announcing-the-first-ada-spark-crate-of-the-year-award

And you can register here: https://github.com/AdaCore/Ada-SPARK-Crate-Of-The-Year

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Wed, 30 Jun 2021 04:44:18 -0700

A candidate project must be on github? (fair enough)

Any Alire crate project must be on github?

Thanks.

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Wed, 30 Jun 2021 05:47:49 -0700

For the Alire community index, your crate can either be in tarball format and hosted anywhere you want, or it can be a commit in a git repo in which case we ask you to host it on GitHub, SourceForge, GitLab or Bitbucket.

Ada Semantic Interface Specification (ASIS)

ASIS and libadalang

From: J-P. Rosen <rosen@adalog.fr>
Subject: ASIS for Gnat (was: Any chance of programming a web frontend in Ada 2012?
Date: Wed, 9 Jun 2021 07:02:40 +0200
Newsgroups: comp.lang.ada

> I did some progress in this direction, but ASIS4GNAT is abandoned and my project is suspended.

ASIS4GNAT is not abandoned, it is just not part of the CE edition. Pro users have access to it.

Please drop me a note if you have developed an ASIS tool, or are using an ASIS-based tool. With enough protests, we may convince AdaCore to make ASIS4GNAT available to the community.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 11 Jun 2021 11:47:12 -0700

> ASIS4GNAT is not abandoned, it is just not part of the CE edition. Pro users have access to it.

On the other hand, if you are starting a new project, libadalang is a better choice.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 11 Jun 2021 22:31:25 +0200

> On the other hand, if you are starting a new project, libadalang is a better choice.

What makes you think so?

By all means, compare the specifications of an ASIS package (Asis.Statements, Asis.Declarations) to Libadalang.Analysis and see which one is more usable...

From: Rod Kay <rodakay5@gmail.com>
Date: Sat, 12 Jun 2021 20:47:29 +1000

> Please drop me a note if you have developed an ASIS tool [...]

I switched from ASIS to libadalang for an Ada IDE project also, since I thought ASIS was abandoned.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 12 Jun 2021 16:04:19 +0200

If you want to analyze code while it is being typed, as is common in an IDE, Libadalang is certainly the way to go.

If you want to make sophisticated analysis tools, it's another story. Hopefully, my paper at AE will soon be available...

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 14 Jun 2021 05:37:57 -0700

Oh, I am looking forward to reading it.

BTW, I really liked your "Memory Management in Ada 2012" video; I've used it as a reference several times to explain to Rust-people that Ada is safer than expected because pointers aren't required for a lot of things, and so you don't have to worry about null-exclusion.

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 15 Jun 2021 14:40:41 -0700

Is JGNAT reliable, updated, available?

Thanks.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 16 Jun 2021 10:41:31 +0200

> Is JGNAT

>reliable,

I didn' try it enough to answer this

> updated,

No. The latest version is 2013. Another one of the useful stuff abandoned by AdaCore.

> available?

Yes, from Adacore's community download page.

Ada and Education

Exercism.io Needs Ada

From: Bruce Axtens <bruce.axtens@gmail.com>
Subject: Exercism.io needs Ada
Date: Sat, 19 Jun 2021 13:30:37 +0800
Newsgroups: comp.lang.ada

In case anyone is looking to encourage people to learn and use Ada, Exercism.io is a good place to learn. Lots of languages already but Ada isn't one of them. Become a maintainer or a mentor.

Maintainer: https://exercism.io/become-a-maintainer

Mentor: https://exercism.io/become-a-mentor#more-info

Bruce Axtens, vbnet maintainer and students of about 26 languages none of which are, sadly, Ada.

From: Andreas Zeurcher <zuercher_andreas@outlook.com>
Date: Mon, 28 Jun 2021 13:42:03 -0700

Since you are a maintainer of another language (Visual Basic .Net), what is the precise sequence of steps that one would need to perform to become the 1st maintainer of a new presence of Ada in Exercism.io? I think that the 1st maintainer is specifically the person who bootstraps up a new language's presence on exercism.io, correct? Apparently, the sequence of steps might begin:

1) Practice being a newbie student of some arbitrary existing(-on-exercism.io) language's exercises to get the feel for how exercism.io is supposed to operate.

then

2) Contact another language's maintainer to be the 1st maintainer of Ada's fresh presence on exercism.io.

Are there more steps than that? For example, must the 1st maintainer recruit a separate 1st mentor or must the 1st maintainer act also as 1st mentor? Are those 2 steps perfectly stated or are they botched somehow? Does some sort of vetting occur for the quality of a maintainer at time of volunteering that could cause the 1st maintainer to be rejected? Does some sort of vetting for the desirability of a programming language occur up at exercism.io “corporate” that could cause Ada itself (independent of the 1st maintainer) to be rejected? It seems that Ada and (AdaSubset-with-) SPARK would be 2 separate languages on exercism.io (otherwise it gets confusing), correct? Why doesn't exercism.io have an overt webpage that answers these questions for how to bootstrap up a language's new presence on exercism.io (as this seems to be a separate & distinct topic to becoming the 2nd-or-subsequent-mentor that does have its own webpage explanation already)?

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 29 Jun 2021 06:06:56 -0700

Should not the first step be an evaluation of this site by a programming master? Maybe this has been done; if so, please inform.

Ada-related Resources

[Delta counts are from Apr 26th to Jul 22th. —arm]

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 3_161 (+42) members	[1]
- Reddit: 7_104 (+678[1]) members	[2]
- Stack Overflow: 2_087 (+39) questions	[3]
- Libera.Chat: 76 (new[2]) users	[4]
- Freenode: 15 (-79[2]) users	[5]
- Gitter: 86 (+11) people	[6]
- Telegram: 128 (+7) users	[7]
- Twitter: 75 (+32) tweeters	[8]
74 (=) unique tweets	[8]

[1] Probably caused in part by confusion with the ADA cryptocurrency.

[2] Freenode has suffered a mass exodus due to a change in ownership. Most channels have migrated to Libera.Chat.

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

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

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

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

[5] https://netsplit.de/channels/details.php?room=%23ada&net=freenode

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

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

[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: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

Rosetta Code: 827 (+16) examples	[1]
38 (=) developers	[2]
GitHub: 763[1] (=) developers	[3]
Sourceforge: 275 (+2) projects	[4]
Open Hub: 214 (=) projects	[5]
Alire: 171 (+15) crates	[6]
Bitbucket: 89 (=) repositories	[7]
Codelabs: 52 (=) repositories	[8]
AdaForge: 8 (=) repositories	[9]

[1] 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://sourceforge.net/directory/language:ada/

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

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. The IEEE ranking deltas are in regard to the 2019 edition, as it is updated annually. —arm]

- TIOBE Index: 28 (+2) 0.48% (-0.01%)	[1]

- PYPL Index: 18 (-1) 0.75% (-0.06%) [2]

- IEEE Spectrum (general): 39 (+4) Score: 32.8 (+8.0) [3]

- IEEE Spectrum (embedded): 12 (+1) Score: 32.8 (+8.0) [3]

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

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

[3] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020

Ada Domain Names

From: Heziode <heziode@protonmail.com>
Subject: Ada domain names: who holds what, and for which purpose?
Date: Thu, 24 Jun 2021 20:03:39 +0200
Newsgroups: comp.lang.ada

By curiosity, I have checked if there is some domain name related to Ada language. I mean, domains that can be officially used to represent the language (even though according to Wikipedia, it is adaic.org. Keep in mind, I am doing this out of curiosity).

I was being surprised to discover that several domains, that could be used as an "official" website (like *lang.org, *-lang.org, *.codes, where "*" is the language name), is bought but not used, and not in sales.

Here is a table of my few research:

Domain: Owner: Registered: Country: State

ada-lang.com : AdaCore : 2016/09/20 : FR*

adalang.com: ? : 2014/03/30 : US : Florida

ada-lang.org : ? : 2020/02/11 : US : WA

adalang.org : ? : 2020/02/12 : US : WA

ada.codes : Steve Arnold : 2014/04/23 : US : CA

* Point on "thinkx.net", so DNS issue with wrong IP?

We can see that "ada-lang.org" and "adalang.org" seem to be bought by the same person/entity, but these domains are not used, and not in sales. Looks like domain retention.

Does anyone know who is behind these domains?

From: Luke A. Guest <laguest@archeia.com>
Date: Thu, 24 Jun 2021 19:27:45 +0100

Steve is nerdboy on irc/github.

The Ada Lang one seems to be her name as there is a photo on the contact page.

There is learn/getadanow.com which is David Botton's.

Ada-related Tools

SweetAda 0.3 through 0.8

[Six consecutive announcements have been merged in a single thread. —arm]

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Subject: SweetAda 0.3 released
Date: Tue, 6 Apr 2021 09:09:52 -0700
Newsgroups: comp.lang.ada

I've just released SweetAda 0.3.

SweetAda is a lightweight development framework to create Ada systems on a wide range of machines. Please refer to https://www.sweetada.org.

First of all, please re-download toolchain packages; although timestamps do not change, they contain updated versions of GCC/GNAT wrappers which are essentials for a properly working build system which should have reached a stabilization point, and it seems rather efficient and free from major issues.

Release notes

- due to changes in RTSes, switch -gnatp (pragma Suppress (All_Checks)) is again commented out (Makefile.tc.in), to bring in exception processing; re-enable it if the final object is too big for your setup
- initial implementation of a secondary stack in the SFP RTS (not fully operational yet)
- strict compiler conformance, -gnatE and -gnato are defaults in Makefile.tc.in
- suppress No_Elaboration_Code in gnat.adc
- console.ali was not dragged in under GPRbuild mode and is missing in some configurations, which could lead to undefined references
- build.gpr/configure.gpr now correctly process implicit .ali units
- configure.ads (auto-generated from configuration.in) is pragma Pure
- a kernel link phase is performed if linker script changed
- Makefile now has two more targets: "session-start" and "session-end"; like "run" and "debug" targets, they are associated with shell commands that you can define in the platform configuration.in and can be useful for starting and ending JTAG servers, remote communication, and so on; these targets have nothing special, the names are only placeholders and their purposes are completely defined by the shell commands carried on; see an example in the new platform FRDM-KL46Z, where the commands respectively define an OpenOCD server startup and shutdown action
- Makefile.tc.in could specify -gsplit-dwarf; thus you can find *.dwo DWARF files in the object directory
- still more Makefiles tweaks: now there should be no problem building in GPRbuild mode; furthermore, "make createkernelcfg" forces a distclean
- menu-dialog.sh is standardized and behaves like menu.[bat|sh], so there are now separate items "createkernelcfg" and "configure" (previously there was a single "configure" item which executed them sequentially)
- elftool has a new command switch to find a symbol value: elftool -c findsymbol= which returns the symbol value; it is used, e.g., in the FRDM-KL46Z platform to automatically find the _start address in the executable image and instruct OpenOCD to run the target with a properly resume address; see an example as outlined in .../platforms/FRDM-KL46Z/runopenocd.tcl
- mbr can read other partitions beyond the first
- mbr partition read could cause a misaligned access with some CPUs, so an assignment is replaced with a memory copy
- Dreamcast code runs on a real device, not just in the GXemul emulator; this requires:
- a HIT-0400 "BroadBand Adapter" Ethernet module
- a CDI CD-ROM burned with "Dreamcast CDI Burner" https://alex-free.github.io/dcdib/
- a dc-tool-ip utility http://napalm-x.thegypsy.com/adk/dc/dcload-ip/index.html

  (the dc-tool-ip utility will be soon replaced by a Tcl script)
- MicroBlaze has now udivsi3 and umodsi3 LibGCC assembler routines
- MemecFX12 and Spartan3E platforms now have Tcl scripts to build, download and execute SweetAda kernel
- new target: FRDM-KL46Z Freescale/NXP ARM-CortexM0 board (a.k.a. "Freedom"), only able to blink a LED (needs OpenOCD to communicate with the target from inside SweetAda)
- FS-UAE platform renamed as Amiga-FS-UAE
- typos, cosmetics and minor adjustments

Quick notes

As usual, download the three packages core, RTS and LibGGC (since many changes are system-wide), and please save your work before overwriting the filesystem.

Subject: SweetAda 0.4 released
Date: Tue, 27 Apr 2021 03:53:57 -0700

I've just released SweetAda 0.4. [...]

Release notes

- SweetAda has a new toolchain, armeb-sweetada-eabi, to handle big-endian ARMs (previously this was not necessary since ZFP does not link against libraries); affected target is DigiConnectME — and eventually your own experimental target; ARM BE targets now not need to specify big-endian switches anymore, but they should explicitly specify armeb-sweetada-eabi as the toolchain name in the platform configuration.in, i.e.: TOOLCHAIN_NAME := $(TOOLCHAIN_NAME_ARMeb) whilst ARM LE takes the default toolchain
- the non-optimized versions of divsi3/modsi3 for MicroBlaze were not selected; this is now corrected
- the download script for Dreamcast — bba.tcl — is now written in Tcl (note: requires Tcl UDP extension) and thus does not require the dc-tool-ip utility, quick'n'dirty, no error checking yet; if it is difficult to install a Tcl extensions, then stuck yourself with dc-tool-ip by just uncommenting its exec line, and do an exit
- remove useless return statements in various Tcl scripts
- use Bits.BigEndian/LittleEndian booleans in llutils unit
- ATmega328P has more MCU definitions (not complete yet)
- ArduinoUno:
- XTAL clock frequency is specified in configure.ads
- ZFP profile is forced in configuration.in to avoid problems with a small foot-print memory, thus overriding the settings in the top-level configuration.in
- BSP does nothing; tests moved in application/test-ArduinoUno
- FRDM-KL46Z has more definitions; ZFP profile is forced in configuration.in to avoid problems with a small foot-print memory, thus overriding the settings in the top-level configuration.in
- bits.ads: some Bits_XX_Mask corrected; added Bits_XX_RMask
- now RTSes have, in every CPU target, two more files: 1) Makefile.srcs.in, the list of source files used (not of particular use so far, just a reference);

2) Makefile.rts.in, contains CPU-wide switches (i.e., not dependent on the multilib selected) used during the RTS build; those switches, which normally are empty, are automatically imported in the master Makefile and added to the target platform CPU, thus assuring that the compiler agrees on both RTS code and SweetAda/user code; as a consequence of this, there is no more need to specify, e,g, "-fno-leading-underscore" in SuperH/SH4 targets, and MIPS targets inherit automatically a -G0 switch (they are the only switches which are actually used in the RTS for those targets)

- QEMU-MIPS was based on "mips" machine; this machine does not exist anymore in current QEMU and so the platform is now based on "mipssim", what changes is just the UART16450 base I/O address in monitor.S
- Nios II Terasic DE10-Lite now has a Tcl front-end (programmer.tcl) which automatically downloads the SOF bitstream and executes the SweetAda code by means of Quartus command-line utilities and jtagd daemon
- Nios II Terasic DE10-Lite exposes a configuration setup that explains practically how to override core units, i.e., it invalidates last_chance_handler in the core directory and redefines the same subprogram package in its own directory, so to avoid dragging in the entire console package (which is used by the standard implementation of last_chance_handler)
- all Tcl scripts that handle the download of code on a physical target board are possibly renamed to a standardized "programmer.tcl"
- menu-dialog.sh now always shows warnings (previously it used to show warnings only if the build failed due to hard errors)
- typos, cosmetics and minor adjustments

Subject: SweetAda 0.5 released
Date: Tue, 4 May 2021 04:09:43 -0700

[...]

Release notes

- The SFP RTS now gets Ada.Tags installed, and so it should be possible to use Ada tagged types
- there are no more multiple Makefile.rts.in scattered in every multilib directory, only a single file is stored in the RTS root path of the toolchain target
- Master Makefile does not export FPU_MODEL, corrected
- new target: Synergy-S5D9 ARM-CortexM4 board, only able to blink a LED (needs OpenOCD to communicate with the target from inside SweetAda)
- LibGCC now has adddi3/subdi3/negdi2/mulsi3/muldi3 implemented in pure Ada (although a bit superfluous, since in most cases these subprograms will be overridden by CPU's own LibGCC assembly routines)
- The MVME162-510A platform has now a little Tcl script to download a SweetAda S-record image by means of 162-Bug on-board monitor communication; very simple script (and at 19200 also very slow for big images, but good enough for testing)
- the hard disk images for some platforms (Amiga-FS-UAE, Malta, PC-x86, etc) got accidentally deleted, they are now re-integrated for testing purposes
- removed superfluous conversion in Address_Displacement
- drivers/PC: PIC_Init has now Vector_Offset_Master/Slave input parameters and can be used also from non-x86 targets
- Malta MIPS: use PIC code from PC unit rather than an ad-hoc piece of code
- drivers/PC: PIT_Counter0_Init has an input Count parameter
- drivers/PC: unit does not depend on configure.ads anymore, and so the entire drivers branch should be CPU-independent
- typos, cosmetics and minor adjustments

Subject: ANN: SweetAda 0.6 released
Date: Wed, 19 May 2021 09:56:18 -0700

[...]

Release notes

- spurious entry core/last_chance_directory was not removed in the configuration.in for the core complex, and this causes a build failure in GPRbuild mode, corrected
- Makefile.tc.in: new ADAC_SWITCHES_WARNING switches: -gnatw.q - (Activate warnings on questionable layout of record types) -gnatw_r - (Activate warnings for out-of-order record representation clauses) (unused)
- Makefile.tc.in: added DISABLE_STACK_USAGE flag (some targets do not support stack usage computation, can be set from platform-level configuration.in)
- menu-dialog.sh remains in menu until you exit explicitly (e.g., by pressing double-ESC), so you can perform various actions sequentially; if instead you specify an action as an argument in the command line then the behaviour is unchanged, exiting at once after execution
- qemu-ifup.sh/qemu-ifdown.sh are now a single common copy in libutils directory; Dreamcast makeip.tcl/scramble.tcl are now merged in makecdrom.tcl; pc-x86-bootX.tcl moved as a single copy in share directory
- package Definitions is now placed in modules directory

- more error checking in various Tcl scripts
- initial cleanup of cpus branch file layout, removed duplicate files
- new target: SiFive HiFive1 Rev B, only able to blink the on-board RGB LEDs (needs OpenOCD to download the executable)
- Synergy-S5D9: bsp.ads got accidentally deleted, corrected
- Synergy-S5D9: added SCI definitions so that it can output something on SCI (UART mode, very primitive)
- partial rewriting of the NE2000 driver, more register definitions
- removed all ugly, unpleasant, ill-designed temporary code from exceptions.adb in PC-x86 interrupt handling (which now processes, e.g., raw TCP/IP traffic from applications.adb); the same in Amiga-FS-UAE
- some changes in Ethernet FIFO queue to make it more efficient
- ATmega328P (ArduinoUno): more register definitions, timers and general purpose registers; added some low-level templates; deleted unuseful subprogram in proprietary core unit and its dependency on console
- drivers/pc:
  - revised 8254 PIT; PIT_Counter0_Init now uses MODE 2 (rate generator) instead of MODE 3 (square wave generator) as a system timer
  - simple stub for RTC handling
  - IrqX renamed to PIC_IrqX
  - Irq0 aliased to PIT_Interrupt
  - Irq8 aliased to RTC_Interrupt
- added -mno-red-zone to GCC switches in x86-64
- use rounding instead of floor integer division when computing timing counts, where appropriate

There is also a new release of QEMU emulator — 20210517 — providing QEMU 6.0.0 for Linux and Windows platforms, and QEMU 5.2.0 for OS X. The OS X version should work on El Capitan (tested on a VM, someone reported problems on later versions).

Subject: ANN: SweetAda 0.7 released
Date: Tue, 1 Jun 2021 13:18:02 -0700

[...]

Release notes

- updated targets in master Makefile ("all" was tagged default instead of "help"); the targets "kernel_info" and "kernel_libinfo" are now exposed (kernel_libinfo produces listings of library objects even if the kernel build is not successful)
- added implicit dependencies for console unit
- elftool will emit spaces instead of TABs when performing an ELF section dump, this will be noted in the next toolchain release
- the linker script filename can now be declared in the platform configuration.in by specifying "LD_SCRIPT:= <linker_script_filename>", otherwise it takes a default "linker.lds"
- the C library now implements Ada stubs for malloc/free/calloc/realloc, so C code can call these Ada subprograms via stdlib wrappers; this has also the benefit of resolve references to malloc() when secondary stack tries to return heavy (i.e., unconstrained) objects, but be sure to add "USE_LIBGCC := Y" and "USE_CLIBRARY := Y" to the configuration.in file, either the generic one in the top-level directory, or the platform-dependent one
- SFP RTS: a-except: Raise_Exception calls Last_Chance_Handler
- SFP RTS: added Ada.Assertions (for pragma Assert you need to turn on -gnata in the "Ada Run-Time Checks switches" section of Makefile.tc.in)
- core/bits: added BITZERO/BITONE/BITL/BITH/BITOFF/BITON declarations
- core/console: Print (Boolean), emits "T" or "F"
- core/llutils: HexDigit_To_U8 uses a case instead of longer ifs
- modules/definitions: added a few definitions
- added various Volatile_Full_Access aspects here and there
- corrected some section wildcards in linker scripts for ARM platforms
- x86_64 lacks some low-level CPU subprograms (but they are empty anyway) and so the build could fail with unresolved objects, added
- new libutils/libopenocd.tcl file, useful for small OpenOCD function helpers
- Digi Connect ME (NET+ARM NS7520): some more register definitions; adopted a Tcl script as front-end to OpenOCD
- Synergy S5D9: OpenOCD cfg file renamed to standard "openocd.cfg"
- Synergy S5D9: more register definitions, SCI almost completely parameterized
- platform Spartan3E renamed as Spartan3E-SK
- new target: Avnet Xilinx Spartan-3A Evaluation Kit (Spartan3A-EK, MicroBlaze v7.00.b), only able to blink a LED; the programmer.tcl front-end will download the bitstream by directly interfacing with the on-board Cypress PSoC via USB protocol (no external tools needed in a Linux environment)
- targets involving OpenOCD (DigiConnectME, FRDM-KL46Z, HiFive1, MSP432P401R, STM32F769I, Synergy-S5D9) now should specify in the platform configuration.in the OpenOCD prefix (in Windows is the installation directory, i.e., that which is the parent of bin/, etc); the default is the *nix path "/usr/local"
- in the top-level directory there are the two files .cproject and .project for Eclipse CDT; no big deal since you have absolutely no Ada support, but if you import the project and configure the *.adb and *.ads files as textual source files, you could do a make build cycle, with error signalling (clicking on the error shown in console should redirect you to the offending source line)
- typos, cosmetics and minor adjustments

Subject: ANN: SweetAda 0.8 released
Date: Mon, 14 Jun 2021 07:46:15 -0700

[...]

Release notes @ https://www.sweetada.org/release_notes.html.

Downloads available @ https://sourceforge.net/projects/sweetada.

Release notes

- ADA_MODE defaults to ADA20 (-gnat2020)
- now the RTS is not a single common archive, instead every target CPU has its own — i.e. you have to download sweetada-rts-<target>-0.8.tar.gz; this way you avoid to waste bandwidth downloading a large RTS for, e.g., the AVR, when you don't need it
- adjusted Lock_Type definitions according to Ada 2020
- Tcl scripts: use "eq"/"ne" in place of "=="/"!="
- corrected a misinterpretation in the libopenocd.tcl proc version_numeric
- various programmer.tcl front-ends do not inherited OPENOCD_PREFIX, corrected
- change "adapter_khz" to "adapter speed" in OpenOCD configurations
- NiosII RTS has -mgpopt=none switch, so it is inherited in those kind of platforms
- NiosII lacks interrupt subprograms declaration (body still TBD), declared
- adjusted linker scripts in NiosII platforms (Altera10M50GHRD, DE10-Lite)
- DigiConnectME has ARM vectors template in llkernel.s
- adjusted some values in VGA low-level register programming; the package now

can setup graphic mode 12h (640x480x16)

- burned in an EPROM, SweetAda correctly startups in a DECstation 5000/133 and output messages on the SCC8530 serial port, blinking also the rear LEDs
- added/removed some Volatile_Full_Access aspects and cleaned up record layouts that don't need it (they are placed in the object definition instead)
- every CPUs has eventually an empty LibGCC package spec (which overrides thecore one); this enforces a catch of package (mis-)using, which cause the compiler to flag an error (should you try to use it when the CPU really does not need it)
- corrected a Makefile target dependency (output listings could be out-of-synch if "make postbuild" is not explicitly called)
- typos, cosmetics and minor adjustments

Quick notes

As usual, download the three packages core, RTS and LibGCC (since many changes are system-wide), and please save your work before overwriting the filesystem.

GNAT LLVM, ACATS

From: Simon Wright <simon@pushface.org>
Subject: GNAT LLVM, ACATS
Date: Wed, 07 Apr 2021 11:58:11 +0100
Newsgroups: comp.lang.ada

I recently had success building GNAT_LLVM on macOS: see notes here [1].

Running ACATS 4.1 U via the ACATS Grading tools as patched for llvm-gnat [2], I get impressively successful results: out of 4092 tests, GCC 11.0.1 of 2021-03-31 has

Result: Overall, B-Tests, C-Tests, L-Tests, Other Tests

[...]

Total Failed: 44, 30, 14, 0, 0

Total Not-Applicable: 35, 0, 35, 0, 0

Total Special: 182, 141, 21, 10, 10

Total Passed: 3831, 1272, 2420, 61, 78

(L-tests "check that all library unit dependencies within a program are satisfied before the program can be bound and executed, that circularity among units is detected, or that pragmas that apply to an entire partition are correctly processed". 'Special' means human inspection needed.)

whereas llvm-gnat, built from the same GCC sources, has

Result: Overall, B-Tests, C-Tests, L-Tests, Other Tests

[...]

Total Failed: 48, 31, 17, 0, 0

Total Not-Applicable: 35, 0, 35, 0, 0

Total Special: 184, 141, 23, 10, 10

Total Passed: 3825, 1271, 2415, 61, 78

[1] https://github.com/AdaCore/gnatllvm/issues/20#issuecomment-809400426

[2] https://github.com/simonjwright/ACATS-grading/tree/llvm

HAC V.0.095

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Subject: Ann: HAC v.0.095
Date: Wed, 7 Apr 2021 12:23:53 -0700
Newsgroups: comp.lang.ada

HAC (HAC Ada Compiler) is a small, quick, open-source Ada compiler, covering a subset of the Ada language. HAC is itself fully programmed in Ada.

Web site: http://hacadacompiler.sf.net/

Source repositories:

#1 svn: https://sf.net/p/hacadacompiler/code/HEAD/tree/trunk/

#2 git: https://github.com/zertovitch/hac

* Improvements since v.0.085:

Modularity: HAC recursively compiles all units needed to build a main program. Currently only procedures and functions bodies are supported - no packages yet, no separate specifications.

An example can be found in exm/unit_a.adb

Enjoy!

From: Stéphane Rivière <stef@genesix.fr>
Date: Thu, 8 Apr 2021 10:26:00 +0200

Well done Gautier!

In our company, we now use HAC here to replace many big Bash scripts in our servers cluster spread in 3 european DC. Big gain of productivity and maintainability.

Incidentally, HAC is up to 7 times faster than Bash for a much lower resource consumption. Surprising when you see the poverty of the syntax and semantics of Bash.

UXStrings 20210405

From: Blady <p.p11@orange.fr>
Subject: [ANN] UXStrings package available (UXS_20210405).
Date: Sun, 11 Apr 2021 10:45:53 +0200
Newsgroups: comp.lang.ada

A second POC implementation for UXStrings is provided. The source code files end with the number 2 as for instance "uxstrings2.ads". https://github.com/Blady-Com/UXStrings/blob/master/src/uxstrings2.ads

A GNAT project file "uxstrings2.gpr" is provided with some naming conventions for both packages UXString and UXStrings.Text_IO.

Some API have been added to support ASCII 7 bits encoding for both version UXStrings 1 and 2. ASCII is a subset of UTF-8 thus no change with the internal UTF-8 representation.

However, in addition to UXStrings 1 implementation, the API is now aware if content is full ASCII. On one hand, this permits to access directly to the position of one character without iterating on UTF-8 characters. Thus this is a time improvement when content is full ASCII. On the other hand, when content is changing the API checks if the new content is full ASCII. Thus this is a time penalty when changes are not full ASCII.

English contents as programming text files are composed of lines in majority full ASCII but they may have some lines with characters out of the ASCII set. UXStrings is dealing with both.

Available on GitHub (https://github.com/Blady-Com/UXStrings) and also on Alire (https://alire.ada.dev/crates/uxstrings.html).

Feedback is welcome on the actual time improvement on your real use cases.

RAPID New Maintainer

From: Thomas <fantome.forums.tdecontes@free.fr.invalid>
Subject: RAPID
Date: Mon, 12 Apr 2021 18:56:52 +0200
Newsgroups: comp.lang.ada

Hi :-)

courtesy Oliver Kellogg, I'm officially the new RAPID maintainer :-)

I would like to know if there still exist some RAPID users :-)

I also would like to know if there are some users of other platforms than Unix or Windows who would like to use RAPID, even if it doesn't work until now.

I see that GtkAda supports at least Solaris/SPARC platform, in addition to Unix and Windows, but if no one is interested I won't waste time on a specific portability. Tell me :-)

RAPID maintainer

http://savannah.nongnu.org/projects/rapid/

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 12 Apr 2021 10:58:22 -0700

I'm on Windows and Solaris and Linux here, we might get Macintosh from long-term visitors.

*From: Thomas <fantome.forums.tdecontes@free.fr.invalid>
Date: Mon, 12 Apr 2021 20:04:51 +0200*

ok :-)

What's your relation with RAPID? (Are you a user? Are you interested? ...)

*From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 12 Apr 2021 13:01:00 -0700*

Not a user, currently.

But interested, and having a nice cross-platform common-UI would make things a lot nicer for some prospective software-upgrades at work.

One such possible nicety would be a universal administration tool, another would be a data-management/-analysis tool for visiting scientists, another possibility would be decoupling several control-programs (codebases in everything from C to VB to C#) used to operate the instrumentation here from their host-systems and increase portability.

SparForte 2.4.1

*From: Ken Burtch <koburtch@gmail.com>
Subject: ANN: SparForte 2.4.1 Released
Date: Sat, 1 May 2021 05:13:35 -0700
Newsgroups: comp.lang.ada*

SparForte 2.4.1 has been released. This mainly contains fixes for the new tab completion system.

SparForte is my Ada-based shell, scripting language and web template engine.

SparForte can be downloaded at https://www.sparforte.com

The change log is located at https://www.sparforte.com/news/2021/news_may2021.html

SparForte is a hobby and relies on contributions from volunteers.

Simple Components v4.56

*From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.56
Date: Sun, 2 May 2021 14:55:01 +0200
Newsgroups: comp.lang.ada*

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, multiple connections server/client designing tools and protocols implementations. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

Changes to the version 4.55:

- The packages Persistent.Streams and Persistent.Streams.Dump were added to implement streams backed by a fail-safe file;
- ELV/e-Q3 MAX! cube client was changed to work around cube firmware problems.

GCC 11.1.0 for MacOS

*From: Simon Wright <simon@pushface.org>
Subject: ANN: GCC 11.1.0 for macOS
Date: Sun, 02 May 2021 17:28:09 +0100
Newsgroups: comp.lang.ada*

GCC 11.1.0 x86_64-apple-darwin for macOS is available at: https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/11.1.0/native

The release no longer supports ASIS.

Libadalang and tools (gnatmetric, gnatpp, gnatstub, gnattest) are included.

Please note:

* This release is made as an installer package. Because I don't have a signing ID, you can't double-click on it; instead, right-button, Open, and ignore the warnings. Sorry.
* In the past, I've included modified versions of the compiler specs files. This time, such a modified specs file wouldn't run on El Capitan, so I've supplied the compiler as-built; if you're on Mojave or later, you need to set SDKROOT to pick up the place where Apple provides them. This means that the compiler won't automatically look in /usr/local/include (affects C, C++) or /usr/local/lib (affects all languages).

See the release notes at Sourceforge.

*From: Simon Wright <simon@pushface.org>
Date: Thu, 10 Jun 2021 17:21:57 +0100*

This release contains versions of gnatstub, gnattest, gnatpp and gnatmetric which fail to load:

$ /opt/gcc-11.1.0/bin/gnattest —help

dyld: Library not loaded: @rpath/libgnarl-11.dylib

Referenced from: /opt/gcc-11.1.0/bin/gnattest

Reason: image not found

Abort trap: 6

Workaround: export DYLD_FALLBACK_LIBRARY_PATH =/opt/gcc-11.1.0/lib/gcc/x86_64-apple-darwin15/11.1.0/adalib

You may not be aware that gprbuild now lets you specify building standalone static libraries "for Library_Interface use (list-of-units);"

- this doesn't work on macOS, and in fact cannot work, because it uses features of binutils object binaries that aren't available in Mach-O. The effect of this is that a static link against such a library will fail if the library involves any tasking. If you try to fix this by using the relocatable version, and then move the executable, it won't find the GNAT runtime dylibs.

I wonder why the GNAT runtime dylibs are all the way down there without a symlink in $prefix/lib?

*From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 2 May 2021 18:54:25 +0200*

Thanks Simon!

As a side note. The version 11 brings new incompatibilities breaking old code. In some cases X'Access is no longer accepted and needs to be replaced by X'Unchecked_Access.

I am too lazy to analyze whether that is a bug or feature, just be aware.

I dare say that every Ada style guideline should require 'Unchecked_Access everywhere. The issue became a permanent maintenance nightmare.

*From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 3 May 2021 10:29:38 +0200*

> The release no longer supports ASIS.

That's unfortunate. Actually, GNAT FSF could be a good opportunity to continue support for ASIS.

> Libadalang and tools (gnatmetric, gnatpp, gnatstub, gnattest) are included.

But I guess not gnatcheck, since it needs ASIS.

*From: Simon Wright <simon@pushface.org>
Date: Mon, 03 May 2021 12:14:11 +0100*

> But I guess not gnatcheck, since it needs ASIS.

gnatcheck isn't in any of the recent releases of CE. [1], see the [Tools] section at the end, says it's not.

The [Tools] section also says ASIS is available as an add-on. But it says that about GNATtest.

[1] https://www.adacore.com/gnatpro/comparison

From: Luke A. Guest
<laguest@archeia.com>
Date: Mon, 3 May 2021 11:46:51 +0100

> That's unfortunate. Actually, GNAT FSF could be a good opportunity to continue support for ASIS.

Why? ASIS is dead, even the WG don't bother anymore.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 3 May 2021 13:50:38 +0200

> Why? ASIS is dead, even the WG don't bother anymore.

Many tools depend on ASIS, and there might well be an update of the standard. It is still supported by GnatPro (and PTC).

From: Bill Findlay
<findlaybill@blueyonder.co.uk>
Date: Mon, 03 May 2021 16:16:50 +0100

> GCC 11.1.0 x86_64-apple-darwin for macOS is available at:

Thanks for that Simon.

My KDF9 emulator (~25KSLOC of Ada 2012) compiles and runs correctly, but the (stripped) object code is about 10% bigger and runs about 10% slower than a version compiled with GNAT CE 2020.

Is an Apple Silicon compiler a reasonable thing to hope for in the not too distant future? (Hint 8-)

From: Simon Wright
<simon@pushface.org>
Date: Mon, 03 May 2021 16:44:36 +0100

Sorry to hear that.

> Is an Apple Silicon compiler a reasonable thing to hope for in the not too distant future? (Hint 8-)

Work is in progress, but not so far by me since I don't own any Apple Silicon :-(

From: Stephen Leake
<stephen_leake@stephe-leake.org>
Date: Tue, 04 May 2021 10:47:53 -0700

> I am too lazy to analyze whether that is a bug or feature, just be aware.

I had a similar issue upgrading from GNAT Community 2020 to GNAT Pro 21. The GNAT compiler has gotten smarter about enforcing accessibility rules.

Since those rules are there to prevent dangling references, they should be respected; I fixed my code to compile with 'Access.

It is a pain that GNAT didn't get this totally correct the first time around, but that's life.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Tue, 4 May 2021 22:12:42 +0200

> I had a similar issue upgrading from GNAT Community 2020 to GNAT Pro 21.

Alas.

> Since those rules are there to prevent dangling references, they should be respected; I fixed my code to compile with 'Access.

For example:

```
declare
   Location : Abstract_Layer'Class
renames
   Abstract_Layer'Class (Under.all);
begin
   ...
   if Location'Access =
      Location.Widget.Bottom then
```

This does not compile anymore. Clearly there cannot be any dangling references here.

Recent changes broke a lot of code involving comparisons of access types, especially if an anonymous access type is involved. Among them are cases when even 'Unchecked_Access does not help. So, one should resort to awful 'Address instead.

The situation is quite dire. I would even suggest introducing a built-in operation to compare an object with an access, e.g.

```
P'Refers (X) -- True if P points to X
```

since comparison of access types became too volatile.

> It is a pain that GNAT didn't get this totally correct the first time around, but that's life.

I am not a language lawyer to judge. My impression that in practice accessibility rules significantly reduce safety and code quality rather than add it.

AdaStudio-2021 Release 07/05/2021 Free Edition

From: Leonid Dulman
<leonid.dulman@gmail.com>
Subject: Announce : AdaStudio-2021 release 07/05/2021 free edition
Date: Fri, 7 May 2021 00:05:38 -0700
Newsgroups: comp.lang.ada

I'm pleased to announce AdaStudio-2021 new release, based on Qt-6.1.0-everywhere extended with modules: qtconnectivity qtgraphicaleffect qtlocatio qtmultimedia qtsensors qtserialbus qtserialport qtwebchannel qtgamepad (qtscript and qtwebengine - work in progress).

Qt 6 is a new long-term project and I hope to solve these problems in next releases.

Qt6ada version 6.1.0 open source and qt6base.dll, qt6ext.dll (win64), libqt6base.so, libqt6txt.so(x86-64) built with Microsoft Visual Studio 2019 x64bin Windows, gcc x86-64 in Linux.

Package tested with GNAT gpl 2020 Ada compiler in Windows 64bit, Linux x86-64 Debian 10.4

Qt-6.1.0 everywhere opensource prebuilt binaries for win64 and x86-64 are included into AdaStudio-2021.

In new AdaStudio release was added new module qt6opencvada based on OpenCV 4.5.2 for camera capture, recording, transmit and receive.

qt6opencvada supports face detection and recognition. OpenCV-4.5.2 binaries prebuilt for win64 and x86-64 and included to AdaStudio.

AdaStudio-2021 includes the following modules: qt6ada,vtkada,qt6avada, qt6cvada and voice recognizer.

Qt6Ada is built under aGNU GPLv3alicense https://www.gnu.org/licenses/lgpl-3.0.html.

Qt6Ada modules for Windows, Linux (Unix) is available from Google drive https://drive.google.com/folderview?id=0B2QuZLoe-yiPbmNQRl83M1dTRVE&usp=sharing
(It can be mounted as virtual drive with ExpandDrive (https://www.expandrive.com/download-expandrive/)), go to Adastudio directory and load index.html to browser.

[Removed list of all file contents. —arm]

The full list of released classes is in "Qt6 classes to Qt6Ada packages relation table.docx"

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

Leonid(leonid.dulman@gmail.com)

Gnoga 1.6a and 2.1-beta.

From: Blady <p.p11@orange.fr>
Subject: [ANN] Gnoga version 1.6a and 2.1-beta.
Date: Sat, 22 May 2021 10:07:46 +0200
Newsgroups: comp.lang.ada

Gnoga (https://sourceforge.net/projects/gnoga) version 1.6a has been released on SF GIT: https://sourceforge.net/p/gnoga/code/ci/dev_1.6/tree

Zipped source code is also available on: https://sourceforge.net/projects/gnoga/files

See HISTORY for details: https://sourceforge.net/p/gnoga/code/ci/dev_1.6/tree/HISTORY

Gnoga version V2.1-beta has been released on SF GIT branch dev_2.1: https://sourceforge.net/p/gnoga/code/ci/dev_2.1/tree

This version 2.1 is at the same functionality level as 1.6 with in addition the support of dynamic Unicode strings via the UXStrings library (https://github.com/Blady-Com/UXStrings).

See HISTORY for details: https://sourceforge.net/p/gnoga/code/ci/dev_2.1/tree/HISTORY

V2.1 has been tested (demos, tests, tutorials) with GNAT Community 2020 on macOS 11.2 with Safari 14.

I propose that new features will be added only on this version.

Bug fixes will still be added on version 1.6.

Volunteers are welcome to test it further on their own configuration.

Some testing on Windows and Linux configuration will be appreciated.

Contributors are welcome.

Feel free to report detailed issues on Gnoga list or create tickets on SF:

https://sourceforge.net/p/gnoga/mailman/

https://sourceforge.net/p/gnoga/tickets/

Regards, Pascal.

https://blady.pagesperso-orange.fr

GNAT CE 2021

From: Adamagica <christ-usch.grein@t-online.de>
Subject: GNAT CE 2021 is out
Date: Thu, 27 May 2021 10:37:58 -0700
Newsgroups: comp.lang.ada

Just downloaded, no problems so far. Heureka.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Sat, 29 May 2021 10:17:12 -0700

There is a problem in the implementation of the new Reduce attribute. The Roman_Number example in RM 4.2.1(15/5ff) does not work. GNAT uses the wrong subtype for the Accum subtype. It's been reported.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Wed, 2 Jun 2021 13:40:03 -0700

Regarding the Windows version: each time I call gprbuild on a project, everything is recompiled, in place of an incremental compilation (normally, only the Ada files that were modified since last build are recompiled). Did anyone else notice that?

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 08 Jun 2021 10:54:05 -0700

I see something similar but different (since 2019); no sources are recompiled, but everything is linked again, which in my builds is slow.

Assuming you are seeing the same thing, if you keep repeating the same build, it eventually finishes. If you look in *.bexch, after each build you will see one more gpr hash added; I gather they should all be added the first time. So the number of builds needed is the number of *.gpr you 'with', transitively. After that, each source code change only requires one build; changes to *.gpr (and some other things?) reset all the gpr hashes.

I have not reported this to AdaCore; I don't have access to a support contract for Windows. It would make sense to report it to the community channel.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 11:59:24 -0700

> Regarding the Windows version: each time I call gprbuild on a project, everything is recompiled [...]

I'm jumping at the end of your battle....

I've observed some small differences in the way compilation options are handled by gprbuild. In particular the Builder.Default_Switches and Compiler.Default_Switches were the source of a problem as far as I'm concerned. Not systematic but this resulted in your observed behavior.

From time to time I'm having the problem you mention and in most cases it was related to some incorrect definition in one of my GNAT projects. When this happens I use one of -vl or -vm options. And then... I lose a lot of time to spot the issue :-)

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Wed, 16 Jun 2021 09:24:37 -0700

> When this happens I use one of -vl or -vm options.

Actually my issue is very similar to one that appeared with GNAT GPL 2017 (see "GNAT GPL 2017 incremental compilation"). It is related to configuration pragma files. At the time the solution was to have in the .gpr project file compiler options expressed like

"-gnatec=" & project'Project_Dir & "debug.pra"

because gprbuild doesn't run in the same directory as GNAT since the GPL 2017 version. Now (four GPL/CE versions later) it's a bit trickier since gprbuild doesn't find the configuration pragma file at its correct place, even though GNAT does.

The verbosity switch (-vm) was helpful to confirm that (gprbuild shows reasons for recompilation). Thanks for the reminder!

PragmAda Reusable Components

From: Pragmada Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Reminder] The PragmAda Reusable Components
Date: Tue, 1 Jun 2021 09:39:29 +0200
Newsgroups: comp.lang.ada

The PragmARCs are a library of (mostly) useful Ada reusable components provided as source code under the GMGPL or BSD 3-Clause license at https://github.com/jrcarter/PragmARC.

This reminder will be posted about every six months so that newcomers become aware of the PragmARCs. I presume that those who want notification when the PragmARCs are updated have used Github's notification mechanism to receive them, so I no longer post update announcements. Anyone who wants to receive notifications without using Github's mechanism should contact me directly.

Archlinux 'gcc-ada-debug' AUR Package.

From: Rod Kay <rodakay5@gmail.com>
Subject: ANN: Archlinux 'gcc-ada-debug' AUR package.
Date: Fri, 11 Jun 2021 11:47:01 +1000
Newsgroups: comp.lang.ada

A gcc-ada-debug package has been added to the Archlinux AUR. It is identical to the official gcc-ada package except that debug symbols have not been stripped.

This allows for setting breakpoints on the standard exceptions of the Ada runtime library in GDB, as opposed to seeing a message such as 'Your Ada runtime appears to be missing debug information".

Ada Resource Embedder for C, Ada and Go

From: Stephane Carrez <stephane.carrez@gmail.com>
Subject: ANN: Ada Resource Embedder for C, Ada and Go
Date: Fri, 11 Jun 2021 06:30:42 -0700
Newsgroups: comp.lang.ada

I created a new tool to allow embedding any file in an Ada, C or Go binary. While embedding files, you can apply some transformations such as running a Javascript minifier (closure), compressing the file, encrypting it, ... The tool generates Ada, C or Go files that you compile with your program. In its simplest form, you can access the embedded content as a:

```
type Content_Access is access constant
Ada.Streams.Stream_Element_Array;
```

So the generated code only depends on Ada.Streams.

There are many modes that are explained in the documentation. For an overview, have a look at:

https://blog.vacs.fr/vacs/blogs/post.html?post=2021/06/11/Advanced-Resource-Embedder

And don't hesitate to fork, hack, and submit pull requests to:

https://github.com/stcarrez/resource-embedder

Well, for me it was a fun project :-)

Stephane

Ps: Go has a `go:embed` but It was fun to write the Go generator :-)

From: Stéphane Rivière <stef@genesix.org>
Date: Fri, 11 Jun 2021 15:51:52 +0200

Hi Stephane,

This is typically what I needed to improve my current AIDE v2 project (Ada Instant Development Environment - source GNAT CE 2019 2020 2021 - target Debian / Ubuntu with subtarget station (with GNATStudio, HAC, libs, debug aware RTS, and goodies) or server (bare minimal).

Many thanks!!!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 18:19:15 +0200

I considered embedding into relocatable libraries similar to Windows' resource, e.g. for versioning plugins etc., but then decided to use an easier and more universal method.

I simply put an Ada constructing function into the library. The function is exported. The address returned by GetProcAddress or dlsym goes to Unchecked_Conversion to an access to subprogram, and here you are.

The obvious advantage of this method over embedding is that the object can be tagged of any derived type, which no embedding can do. And you can add whatever further initialization or checks you might need.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 10:32:56 -0700

Can you elaborate a little? I don't see what you put in your Ada constructing function. I do see how you use it but not how and where you put the original content or file.

Let's suppose you have some documentation file 'config/example.conf'. How would you integrate it in a binary with your solution?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 20:25:55 +0200

> I don't see what you put in your Ada constructing function. I do see how you use it but not how and where you put the original content or file.

In my case I do not deal with files, it is always objects. In the simplest case it could be a record type like:

```
type Library_Data is record
   Do_This  : not null access procedure;
   Get_That : not null access function
                 return That'Class;
end record;
```

That the library provides.

> Let's suppose you have some documentation file 'config/example.conf'.

The documentation would be an object ready for rendering. Say the documentation renderer is a GTK text view widget. Then that would require a text buffer:

```
type Library_Data is record
   Do_This : not null access procedure;
   Get_That : not null access function
                 return That'Class;
   Documentation :  not null
                    Gtk_Text_Buffer;
end record;
```

The caller will drop the text buffer Documentation into a Gtk_Text_View widget to show the documentation.

I usually use programmatically generated content. E.g. HTML documentation would be a set of subprograms with parameters that put a portion of HTML into a stream/string. I then decorate their output as necessary, e.g. <tr>...</tr> if that must be a table cell etc.

The point is that documentation is almost never a static text, but has all sorts of parameters the provider does not know in advance.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 11:44:29 -0700

Thanks Dimitry for the clarification.

Different requirements lead to different solutions.

With ARE, I want to embed a Javascript file (such as jQuery), minify it with closure, compress it with gzip and make it available as raw content so that the web server can service it without reading any file. The content being accessible through either an Ada generated variable or through a function, it is mapped in memory (we avoid an open, read, close system call plus the gzip stuff) and the server only has to return the buffer content.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 21:53:07 +0200

Same here. In the case of an integrated HTTP server I simply store the HTTP pages as a set of string constants and functions generating dynamic portions of. The HTTP server uses no external files. Most pages never exist in any moment of time, because the server generates portions of them and sends away chunks as soon as possible. For this reason I do not compress any pages. It would waste too much resources on a small embedded system and does not really matter for a large PC.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 23:15:30 -0700

> Same here. [...]

Not the same. ARE takes a static content and converts it in an Ada source that you compile.

Compression can help even on embedded systems because it reduces the size of data transfer. A jquery-3.4.1.js file is around 273K. Minified by closure it is reduced to 89K. If you also compress it, it reduces to 31K.

On slow networks these size reductions make a difference.

There is no waste of resources because the compression is made during the build process not at runtime.

I would say the opposite: what you get is smaller in size.

From: Stéphane Rivière <stef@genesix.org>
Date: Sat, 12 Jun 2021 14:03:41 +0200

> Your projects are very interesting, can I have the link for AIDE?

Github is coming - watch https://github.com/sowebio in july

Alpha is here https://stef.genesix.org/pub/ada/aide

Don't read v2.14 but v0.14 :) The v1 was a completely different beast. I just keep the numbering...

aide - binary

aide-2.14.zip - sources

v20-0.3.zip is the GP library used for AIDE

All this stuff is KISS¹ but very usable :)

sow - AIDE for Debian & Ubuntu Manual v34.pdf - read issues and to do list at the end. You should *wait for v2.15* release to test it, even the 'big view' already works. I have a few bugs to fix for a 'full flown experience' :)

sow - v20 Ada Library User Manual v28.pdf - API ref (a must read to understand AIDE code) and more (methodology101 for kids and new coders). HAC Ada Compiler User Manual v82.pdf manual for HAC (HAC is included in AIDE. It's a very capable Ada subset interpreter, replacing Bash on all our servers cluster) HAC is a Gautier de Montmollin project (refs in doc). HAC is seven times faster than Bash, more productive and strongly typed!

[1] According to the "Keep It Simple Stupid" theory

GNAT CE 2021 for Intel MacOS

From: Simon Wright <simon@pushface.org>
Subject: ANN: GNAT CE 2021 for Intel macOS
Date: Thu, 17 Jun 2021 17:25:45 +0100
Newsgroups: comp.lang.ada

GNAT CE 2021, built for macOS El Capitan .. Big Sur, at Sourceforge: https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2021-x86_64-darwin-bin/

Github (scroll down to the Assets section): https://github.com/simonjwright/distributing-gcc/releases/tag/gnat-ce-2021

Status of AdaControl (Cont.)

[See “Status of AdaControl” in AUJ 42-1, March 2021. —arm]

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Subject: Re: Status of AdaControl
Date: Sun, 9 May 2021 13:41:19 -0700
Newsgroups: comp.lang.ada

In the following blog post, you can find installation notes for using GNAT CE 2019 - just for the purpose of running AdaControl free edition!

https://gautiersblog.blogspot.com/2021/04/cleaning-up-hac-sources-with-adacontrol.html

A bit tedious, but doable. Hopefully the community will see again in the future, from time to time, updated snapshots with GNAT, ASIS and AdaControl "synchronized"...

The post also shows a demonstration of AdaControl "in action". Amazing tool!

Ada and Operating Systems

Ada Is Back on VMS

From: Vms Ada Alliance <contact@vmsadaall.org>
Subject: Ada is back on VMS
Date: Mon, 3 May 2021 17:42:26 +0200
Newsgroups: comp.lang.ada

Hello,

For whom the information is unknown, VMS itself is back since 2014. The company VSI (VMS Software Inc. Vmssoftware.com) negotiated with HP to give it support and future. Since this decision, VSI gives quality support to VMS on Alpha and Itanium hardware, and now the port to x86 and virtualization is about to be completed. Restricted Early Adopter Kits can be obtained, which run on Oracle Virtual Boxes or vmware. A full field test will begin in June, and the production release will be here at the end of this year.

For us, more important, the information is that Ada is back on VMS.

It had always been available on Alpha (and VAX) as DEC Ada, and is still there. It had been on Itanium until the end of 2014, as a GNAT Ada GCC implementation, supported by Adacore. AdaCore ended its support at the end of 2014, and we organized a rebuild from FSF sources, with the help of David Sauvage, AdaLabs. This build is presented here (https://github.com/AdaLabs/gnat-vms). On our side we can provide the binaries for users who can guarantee they have a professionnel or hobbyist licence for VMS (*) (the build uses VMS headers). (have a look here: http://www.vmsadaall.org/index.php/en/). The pia-sofer company (Play It Again SOFtwarE Renew, piasofer.fr) offers support for the package.

The novelty is that we can now think of a future of Ada on VMS, on x86. VSI organized its set of compilers using LLVM as the back end, version 10. And there is on github a prototype of a GNAT Ada front end for LLVM. So, we have just to test how to make them interact. It’s not a trivial project, but truly exciting.

We are just at the beginning of this work, and we’ll inform on this group how we are progressing.

For sure everyone is invited to participate. It’s a little bit difficult to get an Itanium, where you could test our build. Some itaniums are available via users club initiatives (not yet in France, but the users club vmsgenerations has a project on that). It’s easier to get a free Alpha emulator (for example here: http://www.migrationspecialties.com/FreeAXP.html). And you can get a hobbyist licence by VSI for both (https://vmssoftware.com/community/community-license). Important, we do think, to remember all the VMSisms before getting in the project. To test the GNAT Ada front end on Linux is straightforward, and it is also a preliminary work before porting on VMS (https://github.com/AdaCore/gnat-llvm). We don’t know when VSI will offer hobbyist licenses on VMS x86. Everyone who is thrilled to test some VMS x86 (and on Ada compiler) can contact us, and we’ll organize vpn accesses (contact@vmsadaall.org).

Our particular effort is about universality. We think it’s very important to address the cultural continuity of Ada on VMS from VAX to x86. There are a lot of good things written for Ada on VMS we have to reuse (for example for debug). And because the more modern solutions are around AdaCore solutions, it is important to understand similarities and differences between GNAT Ada on GCC and GNAT Ada on LLVM.

It’s a community effort. We hope VSI and/or AdaCore will get involved in the return of Ada on VMS. And we know there has always been a good collaboration between AdaCore and communities. But companies have their constraints, while individuals or Open Source structures can be immediately reactive. Perhaps also it is a sort of guarantee for users when a community is there.

Don’t hesitate to contact us: contact@vmsadaall.org

(*) It is a fullfledged 4.3.7 version, with all its GNAT tools. And we give in the same package the gcc C compiler and the gcc C++ compiler (of that version). We are working on the libstdc++-v3 (also here, every help welcomed, including just testing).

Ada and Other Languages

Pascal vs C Language Families

[Out of the blue a thread last seen in 2014 (!) was resurrected. It is an interesting yet mostly off-topic talk full of anecdotes about early compilers. —arm]

From: Paul Rubin <no.email@nospam.invalid>
Subject: Re: why the pascal family of languages (Pascal, Ada, Modula-2,2,Oberon, Delphi, Algol,...) failed compared to the C family?
Date: Thu, 27 May 2021 09:00:16 -0700
Newsgroups: comp.lang.ada

\>>Algol 60 did not have a defined I/O.

> <?> Just curious -- do you mean the I/O was all by linked in

> function/subroutines rather than being keywords in the language?

Yeah, something like that. But there were successful Algol 60 implementations, including on Burroughs and Univac mainframes. C. A. R. Hoare supposedly called Algol 60 "a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors.

>>I/O in Pascal was flawed.

> Well... It probably worked quite well in the original OS...

It wasn't just the I/O:

http://doc.cat-v.org/bell_labs/why_pascal/

Borland Turbo Pascal was very popular and apparently practical, though. I never used it but I have the impression that it (like most deployed Pascal implementations) somehow supplied workarounds to the limitations described in the paper above.

These were interesting:

* Things Turbo Pascal is Smaller Than:

https://prog21.dadgum.com/116.html

* Personal History of compilation speed part 2 (scroll down for the part about Turbo Pascal):

https://prog21.dadgum.com/47.html

The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released. That is disappointing based on how cool the above articles make it sound.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Thu, 27 May 2021 13:53:29 -0400

> It wasn't just the I/O: http://doc.cat-v.org/bell_labs/why_pascal/

I believe the original goal for Pascal was to be a teaching language for algorithm development, and wasn't meant to be a real application programming language. Heck -- the earlier VMS Pascal required one to link into the FORTRAN libraries if one needed things like sin()/cos() functions.

> it (like most deployed Pascal implementations) somehow supplied workarounds to the limitations described in the paper above.

Which made it a "lock-in" language -- it wasn't Pascal as defined by Wirth and UCSD. One had to use compatible hardware (Windows, as I recall). Not much use when writing a satellite control (ground station) system on a VAX/Alpha machine (and yes, one such WAS written in VMS Pascal*... A few years later the new version was on HPUX [or whatever they called it] written in C -- they went from PDP-11 assembly to VAX Pascal to HP C)

>The binary of Turbo Pascal was eventually released for no cost download

I believe Turbo Pascal evolved into Borland's Delphi, which added OOP features. Now Embarcadero... And available in a "community edition" https://www.embarcadero.com/products/delphi/starter (interesting: they allow either Delphi OR C++Builder community editions, but not both on a computer [time for virtual machine images <G>]) Community license needs to be renewed annually (though is a free download as I read the site). OUCH -- Professional level is $1600 to start, and $400/year renewal (the initial $1600 is "perpetual", the $400/year is a subscription for updates/upgrades)

>disappointing based on how cool the above articles make it sound.

It feels bloated to me, but there is FreePascal with the Lazarus IDE.

My last Python exposure was on my TRS-80 model 4; Alcor Pascal (under a RatShack license name -- Model 3 was a pure Alcor release). It had the odd feature of allowing one to manually edit the "object" files. Once one learned the structure (they were ASCII) one could cut&paste functions/subroutines).

http://www.trs-80.org/alcor-pascal/

I seem to recall having Blaise II (editor) configured to work like VMS EDT (a bit of a trick, as the numeric pad only had 3 PF keys, not the 4 found on a VT100)

The realtime group did a survey of languages when they upgraded from the PDPs -- choices were VMS assembly, FORTRAN77, Pascal, and C. We had something like 30 people in the realtime group, and 70 people in the rest of the program skilled in F77. They tossed out C as error-prone, F77 as "old", assembly as "why change processor, then", and argued that many graduates at the time were learning Turbo Pascal in college.

When I saw the evaluation email, I sent back one that pointed out that Turbo Pascal had a lot of add-ons that would not be meaningful in a Wirth level Pascal (VMS did allow separate compilation, and linking to libraries written in other languages -- DEC had a common set of built-ins to define passing arguments as value/reference/descriptor so one could match the convention of the library language). I also pointed out that, having ignored the expertise of the 80+ F77 programmers, and gone the mile to choose Pascal, they might have fallen onto their faces and picked DEC VMS Ada -- a language designed for realtime processing...

A few years later, the department manager confessed that Pascal had been a mistake (I believe the lead realtime programmer had threatened to leave if Pascal was not picked -- manager caved in).

From: Wilson <winslowleon171@gmail.com>
Date: Thu, 27 May 2021 19:47:12 -0400

> Can we all blame this success of the C family of languages on Dennis Ritchie and Brian Kernighan brilliance and it being used for Unix? Another reason that C became so widespread was cost. In the 1970s many organizations (including universities) used the PDP 11 series of computers because they were cheap hardware.

Their OS for the PDP 11 on the other hand cost thousands of dollars per computer whereas C and UNIX were free. This was a bargain most owners found irresistible. Many schools made UNIX and C their standard educational language. This in turn generated graduates who only knew C putting pressure on employers to use C. Alas, it quickly became a self-feeding runaway success.

In short, a free lunch is hard to turn down no matter what comes with it.

From: John Perry <john.perry@usm.edu>
Date: Thu, 27 May 2021 17:34:07 -0700

> Borland Turbo Pascal was very popular and apparently practical, though.

I used Turbo Pascal in college 40 years ago, and yes! it did supply workarounds. Later I realized they looked a lot like features of Modula-2 (also by Wirth) and of Ada. Wikipedia tells me (And Therefore It Is True (TM) ;-)) that some of them come from UCSD Pascal.

This next paragraph is from memory, which may be corrupted, and I may have misunderstood it first, so don't take it too seriously, but: people who paid attention to what Wirth said and wrote about compiler design were able to produce small and fast compilers. Somewhere you can find a report written by one of Wirth's students about how they tried to modify one of their compilers to use a tree instead of a fixed-size array with linear search for the symbol table. Everyone except Wirth was sure that the tree would be both better and more useful, and everyone except Wirth turned out to be wrong. As I say, if it interests anyone I'm sure an online search will find it (but it might not be trivial, which is why I'm not doing it now myself).

> The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released. That is disappointing based on how cool the above articles make it sound.

FreePascal is an open-source reimplementation of Turbo Pascal. It boasts many of the speed advantages that Turbo Pascal has. I've never used it beyond occasionally downloading & playing with it, then forgetting about it.

From: Shark8 <onewingedshark@gmail.com>
Date: Fri, 28 May 2021 05:37:48 -0700

That story comes from the paper "Oberon: The Overlooked Jewel" — https://www.semanticscholar.org/paper/Oberon-The-Overlooked-Jewel-Franz/d48becdaf5c3d962e2778f804e8c64d292de408b—

> In order to find the optimal cost/benefit ratio, Wirth used a highly intuitive metric, the origin of which is unknown to me but that may very well be Wirth's own invention. He used the compiler's self-compilation speed as a measure of the compiler's quality. Considering that Wirth's compilers were written in the languages they compiled, and that compilers are substantial and non-trivial pieces of software in their own right, this introduced a highly practical benchmark that directly contested a compiler's complexity against its performance. Under the self-compilation speed benchmark, only those optimizations were allowed to be incorporated into a compiler that accelerated it by so much that the intrinsic cost of the new code addition was fully compensated.

> And true to his quest for simplicity, Wirth continuously kept improving his compilers according to this metric, even if this meant throwing away a perfectly workable, albeit more complex solution. I still vividly remember the day that Wirth decided to replace the elegant data structure used in the compiler's symbol table handler by a mundane linear list. In the original compiler, the objects in the symbol table had been sorted in a tree data structure (in identifier lexical order) for fast access, with a separate linear list representing their declaration order. One day Wirth decided that there really weren't enough objects in a typical scope to make the sorted tree cost-effective. All of us Ph.D. students were horrified: it had taken time to implement the sorted tree, the solution was elegant, and it worked well – so why would one want to throw it away and replace it by something simpler, and even worse, something as prosaic as a linear list? But of course, Wirth was right, and the simplified compiler was both smaller and faster than its predecessor.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 28 May 2021 15:28:48 +0200

I remember that Turbo Pascal had only one error message, something like "Syntax error in expression" with, God forbid, no column number.

Otherwise yes, it was pretty fast, much faster than MS-DOS C/C++ compilers of the time, Borland's own C++ including.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Fri, 28 May 2021 07:49:44 -0700

> The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released.

Perhaps source code was not *officially* released but you find it easily on the Web (for TP 6.0) :-).

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Fri, 28 May 2021 08:01:10 -0700

Interestingly, the HAC Ada Compiler (https://hacadacompiler.sourceforge.io/, https://github.com/zertovitch/hac) is a distant descendent of Pascal-S by Wirth and identifiers are searched linearly via a linked list.

And yes, the compiler is very fast :-).

From: Paul Rubin <no.email@nospam.invalid>
Date: Fri, 28 May 2021 12:22:10 -0700

> Perhaps source code was not *officially* released but you find it easily on the Web (for TP 6.0) :-).

Very interesting I did find what I think you are referring to, though it is much larger than the old versions. It's weird that it looks like an official Borland release of some kind (complete with postal address for tech support). I didn't realize they had ever let that out. Anyway, thanks for the tip. The compiler proper seems to be about 25KLOC of almost entirely uncommented assembly code, plus Pascal headers. I will see if I can understand any of it. I wonder if it might be a disassembly, or otherwise obfuscated by Borland (by stripping comments) for the purpose of the release.

Sometime back I looked rather hard for the source code of TP (any version) and I found something claiming to be TP source code, but was actually something like a Windows wrapper. But this is different.

I think someone might have also published a disassembly of a released TP 2.0 binary, but I'm more interested in the original source as a historical artifact, rather than something to actually use and run in this day and age.

The comparable and amazing for the era BDS C was released as source code here: https://www.bdsoft.com/resources/bdsc.html

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 30 May 2021 17:12:50 +0100

> Otherwise yes, it was pretty fast, much faster than MS-DOS C/C++

Apparently, they were fast because the Turbo compilers didn't do optimisations due to the limitations of the machines.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Sun, 30 May 2021 20:00:00 +0100

They were fast only by comparison with very slow compilers. I remember, around 1987, someone telling me in astonishment that Turbo ran at 2KSLOC/minute. I was unimpressed, as I had worked on a compiler that ran at 20KSLOC/min a decade earlier.

The 20KSLOC compiler ran on a 1.5MIPS machine.

From: Paul Rubin <no.email@nospam.invalid>
Date: Mon, 31 May 2021 20:34:18 -0700

Yes, but 1) 20KSLOC per what unit of time, 2) what language did it compile 1.5 mips is probably faster than a PC-XT (8088) but slower than a PC-AT (80286). I remember being impressed with the speed of Turbo C on the 386 that I used for a while. I never used Turbo Pascal. CP/M and the original PC were somewhat before my time.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 01 Jun 2021 12:23:30 +0100

> Yes, but 1) 20KSLOC per what unit of time,

Per minute, as I said originally.

> 2) what language did it compile?

Pascal.

From: Paul Rubin <no.email@nospam.invalid>
Date: Tue, 01 Jun 2021 09:46:06 -0700

Ok, but that's maybe 5x slower than Turbo Pascal, which compiled 1000s of LOC per second on machines of that class.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 2 Jun 2021 16:38:59 +0200

> May I ask what is meant by "comparable machines"?

Around the end of 80's we used the dhrystone benchmark to compare machines.

> Here's why I ask: it can't have been a machine based on the Intel 8088, because that wasn't available until 1979. An elderly embedded engineer I know says that the CPU used in the PC series, at least the early PC's (8088 & 286) was a terrible CPU. He likes to joke how his <1MHz 6809-based "Trash 80 Color Computer" at $500 could run circles around the >4MHz 8088-based IBM at $1500.

That is my recollection too. I remember that an elderly 1MHz PDP-11 outperformed 12MHz 286. But the instruction set of PDP-11 was eons ahead of the 286's mess.

> So I'm curious if you know the basis of the claim that it was comparable hardware: clock speed, RAM, etc.It is simply that C compilers were garbage

that time. C is a difficult language to compile compared to Turbo Pascal, especially using the methods that were popular then.

The only decent C compiler I remember from that time was DEC VAX C.

Even advanced machines like Motorola 68k, HP had horrific C compilers. (Sun's SPARC C was somewhat better.) The first thing to do was to bootstrap an early GCC on these machines, because the standard compilers were insufferable. I carried the sources with me on a tape (maybe even on a reel, I do not remember). That was fun.

From: Simon Wright <simon@pushface.org>
Date: Wed, 02 Jun 2021 18:26:10 +0100

> Even advanced machines like Motorola 68k, HP had horrific C compilers.

HP salesman: "Our compiler supports ANSI C (except for function prototypes)"

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Wed, 02 Jun 2021 19:13:06 +0100

> May I ask what is meant by "comparable machines"?

Machines that ran other programs at about the same speed, I specifically had in mind the Whetstone and the Ackermann function benchmarks.

> Here's why I ask: it can't have been a machine based on the Intel 8088, because that wasn't available until 1979.

The 20KSLOC/min compiler ran on an ICL 1906S, which had a Whetstone rating of ~800 KWIPS.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 12:17:07 -0700

> But the Pascal family of languages (including Ada) have clearly failed to become popular, at least compared to the C-family (C, C++, C#, ...)

>

> The question is why did this happen?

By chance.

Most of the time a concept/idea is simply too ahead-of-time, or misunderstood.

In the 80s, there was P-code on the Apple II. It was beautiful and fast. But it was too early.

25 years later, Java bytecode came out, exactly the same thing, a cheeky clone.

It was a success (at least, commercially).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 2 Jun 2021 18:11:10 -0500

P-code existed on a lot of machines; there was even a hardware CPU version of it. At least one of the early Ada compilers was built for as well.

> 25 years later, Java bytecode came out, exactly the same thing, a cheeky clone. It was a success (at least, commercially).

Given all of the above, p-code was a relative success as well. It just died out for whatever reason before Java came around doing approximately the same thing. (Quite possibly the possibility of practical just-in-time compilation made Java stick around longer than p-code.)

But the reality of it is that the hype machine got behind Java for whatever reason, but never really did behind Ada or p-code.

(Note that the intermediate code used by Janus/Ada was based on the ideas of p-code [with Ada-specific stuff like exception handling and tasking]; code at the level has a number of advantages. We never built an interpreter for it although it would be possible.)

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 02 Jun 2021 16:40:52 -0700

In the 80s, there was P-code on the Apple II... 25 years later, Java bytecode came out, exactly the same thing, a cheeky clone.

The JVM isn't really comparable to P-code. It is a lot fancier, including features for multithreading and for garbage collection.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 18:04:44 -0700

Well, obviously they put into the thing those mandatory features, plus 25 years of technological advantages (the P-code had to run on a 64kB, 1-MHz 6502), but the base concept is nearly the same, a stack-based VM.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 18:29:09 -0700

> Given all of the above, p-code was a relative success as well.

I understand. No doubt that also the pcode was successful, but it is a pity that it couldn't make the point in the 90s, at least on microcomputer-class machines.

Still talking about the Apple II, I think one of the reasons is that it was way too complicated for the average user, with no less than 5 thick manuals and an underlying OS like the UCSD that was a radical departure in those times.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Thu, 03 Jun 2021 12:51:19 -0400

> it is a pity that [p-code] couldn't make the point in the 90s, at least on microcomputer-class machines.

Faster processors, and the "standardization" on IBM's architecture, may have contributed... Since there was only "one" machine the portability of P-code (same object files, just provide a new interpreter layer for different architecture) wasn't a factor anymore.

> an underlying OS like the UCSD that was a radical departure in those times.

While the UCSD compiler was reasonable, the UCSD OS that went with it was a bit of a pain. All files had to be contiguous (no jumping to the next free sector), so one ended up periodically compressing the disk so files were at the front and all free-space consolidated. That also meant only one open output file per floppy drive, as output files opened in the largest contiguous free-space.

Ada Practice

Unreachable Branches in Case Statements

From: Reinert <reinkor@gmail.com>
Subject: Did I find a bug here?
Date: Thu, 1 Apr 2021 23:15:22 -0700
Newsgroups: comp.lang.ada

Assume this simple program:

```
procedure test0 is
  type A_Type is (A,B,C);
  subtype A_sub_Type is A_Type with
  Static_Predicate => A_sub_Type in A | B;
  X : A_type  := A;
  Y : A_sub_Type  := A;
begin
  case A_sub_Type(X) is
    when A => null;
    when B => null;
    when others => null;  -- ???? Should the
                          -- compiler complain here?
  end case;
end test0;
```

Should the compiler complain about "when others => null"? My compiler does not (running debian 10 updated, gnat-8).

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 2 Apr 2021 09:30:34 +0200

A case statement is allowed to have alternatives that cover no value. A friendly compiler can warn you that "this branch covers no value", but what you wrote is not illegal (and sometimes useful, if you have variants of your software that use slightly different definitions of the type).

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Fri, 2 Apr 2021 11:33:11 +0300

Recent discussion in ISO SC22 WG9, about the Ada part of the ISO "programming language vulnerabilities" document, brought out that if the selecting expression (here AB_Type(X)) in a case statement or case expression has an invalid representation (for example, is an uninitialized variable with an out-of-range value), an Ada compiler is required to raise Constraint_Error if there is no "others" alternative, but if there is an "others" alternative the compiler can instead let execution proceed to that alternative without raising Constraint_Error.

In effect, "others" can cover all values, even those that are outside the nominal subtype of the selecting expression. See RM 5.4(12) and 5.4(13).

So if the programmer is worried about such cases (invalid representations from uninitialized variables or other causes such as Unchecked_Conversion), they can add an apparently unnecessary "others" alternative even if the other alternatives already cover all valid values. However, note that the compiler may choose to raise Constraint_Error even if there is an "others" alternative; RM 5.4 (10.d). To avoid that uncertainty, the program can perform an explicit 'Valid check before the case statement.

From: Reinert <reinkor@gmail.com>
Date: Fri, 2 Apr 2021 22:46:02 -0700

.....snip...

> values. However, note that the compiler may choose to raise

Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

I try as much as possible to avoid "others" to make the compiler point out or to remind (in my large programs) where to add (or check for) possible alternatives in case I extend the value range of a variable. Then it may happen I need to put in for example something like "when A | B | C => null;" instead of "others".

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 3 Apr 2021 08:41:32 +0200

> Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

Yes, but I recommend "case AB_Type'(X) is", i.e. a qualification rather than a conversion.

For the record:

A conversion carries the message: Take a value of type A and get the corresponding value from type B.

A qualification carries the message: I assume that the value belongs to (sub)type A.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 3 Apr 2021 11:18:47 +0300

> Le 03/04/2021 à 07:46, reinert a écrit :

>> Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

> Yes,

I believe not. The use of X as an argument to a type conversion is an "evaluation" of X, by RM 4.6(28), which can be a bounded error by RM 13.9.1(9) if X'Valid is False.

That bounded error can lead to an exception or simply to continued execution with the invalid value.

> but I recommend "case AB_Type'(X) is", i.e. a qualification rather than a conversion.

That also requires an evaluation of X, by RM 4.7(4), and again can be a bounded error if X'Valid is False.

The use of X in X'Valid is explicitly defined to mean that X is "read" but is not "evaluated", RM 13.9.2(12)(13). So it seems to be the only safe way to check for validity.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 3 Apr 2021 14:37:08 +0200

> I believe not. The use of X as an argument to a type conversion is an "evaluation" of X [...]

Hmm, yes. I was thinking about eliminating "when others" because the intended range was covered. If you want to check for invalid values, 'Valid is the only way to go (that's why it was added to the language!)

Ada IRC Channel Migrates to Libera Chat

From: Rod Kay <rodakay5@gmail.com>
Subject: [ANN] Ada IRC channel on Freenode
Date: Wed, 7 Apr 2021 18:44:44 +1000
Newsgroups: comp.lang.ada

About twice a year we try to advertise the #ada channel on the Freenode IRC network. Celebrating its twentieth birthday this year, the channel continues to be active and friendly. These days it averages about 80 users at a time, large enough to support lively and informative discussions but small enough so it's not a madhouse.

Topics range all over the map, from building the latest GNAT to writing an OS in Ada to daily Ada programming issues to how to use PolyORB to use the Distributed Systems Annex. The stated topic is discussing Ada in the context of free and open-source software, but commercial users are equally welcome.

So fire up your favorite IRC client and come join us. The network is homed at irc.freenode.net, but has servers all over the world. Visit www.freenode.net on the web for details. Hope to see you soon!

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Wed, 7 Apr 2021 08:54:08 -0000

> About twice a year we try to advertise the #ada channel on the Freenode IRC network.

Wow! It's like a blast from the past; I've just started (re)using usenet and IRC's still around! Does anyone still use Gopher and Veronica? :-)

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Wed, 07 Apr 2021 11:39:01 -0400

> Does anyone still use Gopher and Veronica? :-)

I believe the Mother Gopher was killed off some years ago. However, there seem to be some 300 servers still out there based on links from Wikipedia.

From: Luke A. Guest <laguest@archeia.com>
Date: Wed, 7 Apr 2021 18:43:10 +0100

There are still gopher's out there, I think one might even be Ada related. Never heard of Veronica.

From: Jeffrey R. Carter
Date: Wed, 7 Apr 2021 20:24:11 +0200

Veronica was a successor to Jughead, which was a successor to Archie, a tool for indexing and searching FTP archives.

From: Shark8 <onewingedshark@gmail.com>
Date: Wed, 7 Apr 2021 12:01:16 -0700

> Does anyone still use Gopher and Veronica? :-)

I have a friend who does retro-computing and, IIRC, he's got a Gopher server running.

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Thu, 8 Apr 2021 14:28:34 -0000

> I have a friend who does retro-computing and, IIRC, he's got a Gopher server running.

Weirdo ;-)

From: Rod Kay <rodakay5@gmail.com>
Date: Thu, 27 May 2021 08:21:48 +1000

Due to a hostile, commercial takeover of Freenode, the #ada IRC channel has moved to the 'irc.libera.chat' server.

Different Public and Private Type Views

From: Drpi <314@drpi.fr>
Subject: GtkAda : Trying to derive a widget
Date: Thu, 8 Apr 2021 21:27:51 +0200
Newsgroups: comp.lang.ada

[The thread initially dealt with an ambiguity issue, but I have trimmed it down to the part about type views which I find more relevant. —arm]

I'm trying to create a GtkAda widget derived from a standard widget.

```
-- debug_panel.ads
with Gtk.Scrolled_Window;
use Gtk.Scrolled_Window;
with Gtk.Text_View;
use Gtk.Text_View;
package Debug_Panel is
   type Debug_Panel_Record is new
   Gtk_Scrolled_Window_Record
   with private;
   -- [...]
private
   type Debug_Panel_Record is new
   Gtk_Scrolled_Window_Record
   with record
      Text : Gtk_Text_View;
   end record;
end Debug_Panel;
```

[...]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2021 00:27:26 +0200

[...]

P.S. When you create new widget it is better to use a more general ancestor hiding insufficient details, e.g.

```
   type Debug_Panel_Record is
      new Gtk.Widget.Gtk_Widget_Record
      with private;
   ...
private
   type Debug_Panel_Record is
      new Gtk.Scrolled_Window.
      Gtk_Scrolled_Window_Record with
   record
      ...
   end record;
```

This makes the code less fragile if you later decide to choose another container widget for the base.

From: Drpi <314@drpi.fr>
Date: Fri, 9 Apr 2021 07:28:46 +0200

> P.S. When you create new widget it is better to use a more general ancestor hiding insufficient details

I'm surprised it is possible to write such a thing in Ada. What does the compiler do with this?

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 9 Apr 2021 08:12:19 +0200

> I'm surprised it is possible to write such a thing in Ada.

There is no problem, since Gtk.Scrolled_Window. Gtk_Scrolled_Window_Record is a descendant of Gtk.Widget.Gtk_Widget_Record. There is no lie: a Debug_Panel_Record IS A Gtk_Widget_Record. The private view has more information: it actually IS A Gtk_Scrolled_Window_Record, but the extra properties are not accessible outside from the package body.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2021 08:18:28 +0200

> I'm surprised it is possible to write such a thing in Ada.

The public view is Gtk_Widget_Record [with no record members], the full view is Gtk_Scrolled_Window_Record [with record members you specified].

From: Drpi <314@drpi.fr>
Date: Fri, 9 Apr 2021 13:32:03 +0200

>There is no lie: a Debug_Panel_Record IS A Gtk_Widget_Record. The private view has more information: it actually IS A Gtk_Scrolled_Window_Record, but the extra properties are not accessible outside from the package body.

Ok. That's interesting. One more thing learned today :)

Importing Types at Run Time

From: Daniel Norte Moraes <danielcheagle@gmail.com>
Subject:How get/use a 'type' from a Ada shared library loaded at run time (plugin)?
Date: Thu, 15 Apr 2021 01:48:17 -0700
Newsgroups: comp.lang.ada

Hi All!

How to use Ada 'type(s)' from a library loaded at run time (dlopen &Cia)? Are they possible?

My main need is to declare variables from these types or and make dispatching calls based on these types.

Any help is welcome.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 15 Apr 2021 11:28:24 +0200

> How to use Ada 'type(s)' from a library loaded at run time (dlopen &Cia)? Are they possible?

Like any other type.

You should not forget to initialize the library though. The library project should normally have

```
for Library_Auto_Init use "false";
```

because otherwise it would likely deadlock under Windows [if you wish to make your project portable]. Under Linux I am not sure if it deadlocks, never tested for that.

> My main need is to declare variables from these types or and make dispatching calls based on these types.

You cannot declare variables of types declared in a dynamically loaded library for the obvious reason that you cannot dynamically refer to a package from the library.

But you can dispatch on a class-wide object which specific type is declared in a dynamically loaded library.

For example you can call a dynamically loaded constructing function that returns T'Class where T is declared outside the library and then dispatch to an overridden primitive operation of S derived from T inside the library.

To my understanding library initialization expands dispatching tables with all tagged types declared in the library.

P.S. What happens on finalization is an intriguing question. I would rather never attempt to unload an Ada library...

Official Ada Logo/Icon

From: Heziode <heziode@protonmail.com>
Subject: Does Ada have an official logo/icon?
Date: Thu, 15 Apr 2021 14:43:20 +0200
Newsgroups: comp.lang.ada

By reading an article, I visited the official website of JWT (JavaScript Web Token): https://jwt.io

On this website, they list implementations in different languages, particularly Ada. However, I have never seen this logo for Ada language.

So, the question is: Does Ada have an official logo/icon?

After some research, there does not seem to be a "really official" logo, but just recommendation (see http://getadanow.com/#mascot)

Unchecked_Deallocation Usefulness

[Although the original post addressed another issue, I have trimmed the thread to a particular side topic about the usefulness of Unchecked_Deallocation. —arm]

From: Drpi <314@drpi.fr>
Subject: Unchecked_Deallocation with tagged types
Date: Sat, 17 Apr 2021 23:45:27 +0200
Newsgroups: comp.lang.ada

I have the following types :

```
type t_Element_Record is tagged null
record;
type t_Str_Record (Str_Length : Natural)
is new t_Element_Record with private;
```

Do I have to create a Unchecked_Deallocation procedure for each tagged type or only one for the root tagged type (and the compiler manages the effective tagged type)?

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Sun, 18 Apr 2021 01:46:07 -0700

Side note: did anyone already suggest a new keyword: unchecked_free and a special statement:

```
unchecked_free Some_Pointer;
```

?

From: Jeffrey R. Carter
Date: Sun, 18 Apr 2021 11:09:59 +0200

> unchecked_free Some_Pointer;

For every access variable P, there could exist the attribute procedure

```
P'Free;
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Apr 2021 12:13:25 +0200

> For every access variable P, there could exist the attribute procedure
>
> P'Free;

I like the idea of attaching it to a variable rather than to type.

I remember the claim that originally making it a generic procedure with an indigestible name was meant as a barrier for lazy programmers. Plus some considerations regarding garbage collection lurked in the subconscious.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 12:20:56 +0200

> P'Free;

Which would defeat the goal of Unchecked_Deallocation.

Ada (or more precisely Ichbiah) chose to deallocate through an instantiation of a generic for good reason. Deallocation is a place where many problems can come from, it is important to be able to trace where they are. The current solution forces you to put "with Unchecked_Deallocation" on top of every module that deallocates, therefore telling the reader that there is some danger in it, and making it easier to find where all the deallocations happen.

Now, I know that in those days, ease of writing is considered more important than ease of reading and long term maintenance...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Apr 2021 12:34:39 +0200

> [...]The current solution forces you to put "with Unchecked_Deallocation" on top of every module that deallocates, [...]

No, that does not work. If we are supposed to search for all calls to deallocate, then attribute 'Free is much easier to look after than first looking for "with Unchecked_Deallocation", then for an instantiation of it with the types in question and then for the name of the instance.

> Now, I know that in those days, ease of writing is considered more important than ease of reading and long term maintenance...

Yes, but dangling pointers is a more complex problem, that effortlessly passes brief code inspections. I agree with you in general, but disagree in this case. *IF* pointers are used *THEN* Unchecked_Deallocation only obfuscates things rather than helps.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 17:14:12 +0200

> [...] much easier to look after than first looking for "with Unchecked_Deallocation"

meant it the other way round: if the module has no "with unchecked_deallocation", you know it does not deallocate anything, without looking at the entire body.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Sun, 18 Apr 2021 08:23:40 -0700

Not at all: the instantiation can be defined in another package - and it is often the case - with any name (Free, Dispose, ...). So actually with the present way it is difficult to track where unchecked deallocation is used, plus it is tedious for the programmers. The P'Free attribute or the "unchecked_free P;" statement would be straightforward to track.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 17:53:11 +0200

Well, P'Free can also be in another package... Of course, we are talking here only about the direct, actual deallocation.

If you want to precisely know where deallocation is used, use AdaControl (for any solution). If you want to be confident that there is no direct deallocation in a module, the generic wins.

And after all, the attribute only saves you one line of code... (OK, two if you count the "with" ;-))

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 20 Apr 2021 21:35:59 +0200

> OTOH, an Ada follow-on would most likely have access types with automatic deallocation as proposed by Tucker in one of the many AIs on ownership. So using any form of explicit deallocation would be discouraged (as would the use of raw pointer types).

I do not understand how that could work, it sounds like a halting problem to me, but anyway, where is a problem? Add a whole new hierarchy of access types independent of the existing one.

From: Jeffrey R. Carter
Date: Tue, 20 Apr 2021 22:32:16 +0200

> 'Free makes more sense in a new language (an Ada follow-on).

Right. I don't think it would be a good idea to add it to Ada.

But I think a new language should not have pointers at all.

No more radical than not having arrays.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 21 Apr 2021 00:10:40 +0300

> But I think a new language should not have pointers at all. No more radical than not having arrays.

It seems to me that a language without arrays and pointers would be very difficult to use in an embedded, real-time, close-to-HW context. So we would lose the nice wide-spectrum nature of Ada.

From: Jeffrey R. Carter
Date: Wed, 21 Apr 2021 10:35:45 +0200

I don't see that pointers are needed for such S/W.

Brukardt has recently been discussing the idea that a high-level language such as Ada should not have arrays, which is why I referenced it. Such a language might not be convenient for such systems.

But the idea is that arrays are a low-level implementation feature that are usually used to implement higher-level abstractions, such as sequences and maps. A language without arrays would have direct support for such abstractions. My experience is that most uses of arrays in embedded, real-time S/W are also for such abstractions, so it would probably not be too great a problem.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 21 Apr 2021 12:11:07 +0200

> I don't see that pointers are needed for such S/W.

Try to load and bind a relocatable library without pointers.

> A language without arrays would have direct support for such abstractions.

That is not enough, even if providing such abstractions were viable. Which is not, because they would be far more complex than array abstraction and resolve none of the problems array abstraction has. E.g. container subtypes constrained to subtypes of elements and/or subtypes of keys.

Array is a simplest case of container. If you cannot handle arrays, how do you hope to handle maps?

Then see above, and explain how an opaque map will deal with a shared memory mapped into the process address space? Or what would be the primitive operation Write of Root_Stream_Type?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 Apr 2021 19:49:24 -0500

> It seems to me that a language without arrays and pointers would be very difficult to use in an embedded, real-time, close-to-HW context.

It's important that a new language have a way to interface to existing hardware and software. So there has to be something that maps to C arrays and pointers (and the equivalent for hardware). But that doesn't necessarily have to be something that is used outside of interfacing. An Ada example is Unchecked_Unions -- they exist for interfacing but shouldn't be used otherwise. A fixed vector type and a raw general access type would do the trick, but those could be something that are almost never used outside of interfacing packages.

Ada and Unicode

From: Drpi <314@drpi.fr>
Subject: Ada and Unicode
Date: Sun, 18 Apr 2021 00:03:11 +0200
Newsgroups: comp.lang.ada

Hi,

I have a good knowledge of Unicode: code points, encoding... What I don't understand is how to manage Unicode strings with Ada. I've read part of ARM and did some tests without success.

I managed to be partly successful with source code encoded in Latin-1. Any other encoding failed. Any way to use source code encoded in UTF-8? In some languages, it is possible to set a tag at the beginning of the source file to direct the compiler which encoding to use. I wasn't successful using -gnatW8 switch. But maybe I made too many tests and my brain was scrambled.

Even with source code encoded in Latin-1, I've not been able to manage Unicode strings correctly.

What's the way to manage Unicode correctly?

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 18 Apr 2021 01:02:06 +0100

It's a mess imo. I've complained about it before. The official stance is that the standard defines that a compiler should accept the ISO equivalent of Unicode and that a compiler should implement a flawed system, especially UTF-8 types, http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-A-4-11.html

Unicode is a bit painful, I've messed about with it to some degree here: https://github.com/Lucretia/uca.

There are other attempts:

1. http://www.dmitry-kazakov.de/ada/strings_edit.htm
2. https://github.com/reznikmm/matreshka (very heavy, many layers)
3. https://github.com/Blady-Com/UXStrings

I remember getting an exception converting from my unicode_string to a wide_wide string for some reason ages ago.

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 19 Apr 2021 01:29:35 -0700

> Any way to use source code encoded in UTF-8?

Yes, with GNAT just use "-gnatW8" for compiler flag (in command line or your project file):

```
-- main.adb:

with Ada.Wide_Wide_Text_IO;
procedure Main is
  Привет : constant Wide_Wide_String :=
            "Привет";
begin
  Ada.Wide_Wide_Text_IO.Put_Line
  (Привет);
end Main;

$ gprbuild -gnatW8 main.adb
$ ./main
Привет
```

> In some languages, it is possible to set a tag at the beginning of the source file to direct the compiler which encoding to use.

You can do this by putting the Wide_Character_Encoding pragma (This is a GNAT specific pragma) at the top of the file. Take a look:

```
-- main.adb:

pragma Wide_Character_Encoding (UTF8);

with Ada.Wide_Wide_Text_IO;
procedure Main is
  Привет : constant Wide_Wide_String :=
            "Привет";
begin
  Ada.Wide_Wide_Text_IO.Put_Line
  (Привет);
end Main;

$ gprbuild main.adb
$ ./main
Привет
```

> What's the way to manage Unicode correctly?

You can use Wide_Wide_String and Unbounded_Wide_Wide_String type to process Unicode strings. But this is not very handy. I use the Matreshka library for Unicode strings. It has a lot of features (regexp, string vectors, XML, JSON, databases, Web Servlets, template engine, etc.).

URL: https://forge.ada-ru.org/matreshka

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Mon, 19 Apr 2021 02:08:35 -0700

> Any way to use source code encoded in UTF-8 ?

```
for Switches ("non_ascii.ads")
use ("-gnatiw", "-gnatW8");
```

from the GNAT user guide, 4.3.1 Alphabetical List of All Switches:

`-gnati`c"

Identifier character set (`c' = 1/2/3/4/8/9/p/f/n/w). For details of the possible selections for `c', see *note Character Set Control: 4e.

This applies to identifiers in the source code

`-gnatW`e"

Wide character encoding method (`e'=n/h/u/s/e/8).

This applies to string and character literals.

> What's the way to manage Unicode correctly?

There are two issues: Unicode in source code, that the compiler must understand, and Unicode in strings, that your program must understand.

(I've never written a program that dealt with utf strings other than file names).

-gnati8 tells the compiler that the source code uses utf-8 encoding.

-gnatW8 tells the compiler that string literals use utf-8 encoding.

package Ada.Strings.UTF_Encoding provides some facilities for dealing with utf. It does _not_ provide walking a string by code point, which would seem necessary.

We could be more helpful if you show what you are trying to do, you've tried, and what errors you got.

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 11:28:34 +0200

> pragma Wide_Character_Encoding (UTF8);

Wide and Wide_Wide characters and UTF-8 are two distinct things. Wide and Wide_Wide characters are supposed to contain Unicode code points (Unicode characters). UTF-8 is a stream of bytes, the encoding of Wide or Wide_Wide characters. What's the purpose of "pragma Wide_Character_Encoding (UTF8);"?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 11:34:26 +0200

> -gnati8 tells the compiler that the source code uses utf-8 encoding.

>

> -gnatW8 tells the compiler that string literals use utf-8 encoding.

Both are recipes for disaster, especially the second. IMO the source must be strictly ASCII 7-bit. It is less dangerous to have UTF-8 or Latin-1 identifiers, they could be at least checked, except when used for external names. But string literals would be a ticking bomb.

If you need a wider set than ASCII, use named constants and integer literals. E.g.

```
Celsius : constant String := Character'Val (16#C2#) & Character'Val (16#B0#) & 'C';
```

From: Simon Wright <simon@pushface.org>
Date: Mon, 19 Apr 2021 12:15:29 +0100

> воскресенье, 18 апреля 2021 г. в 01:03:14 UTC+3, DrPi:

>> Any way to use source code encoded in UTF-8?

> Yes, with GNAT just use "-gnatW8" for compiler flag

But don't use unit names containing international characters, at any rate if you're (interested in compiling on) Windows or macOS:

https://gcc.gnu.org/bugzilla/show_bug.cgi?id=81114

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 12:50:40 +0100

> But don't use unit names containing international characters,

There's no such thing as "character" any more and we need to move away from that. Unicode has the concept of a code point which is 32 bit and any "character" as we know it, or glyph, can consist of multiple code points.

In my lib, nowhere near ready (whether it will be I don't know), I define octets, Unicode_String (utf-8 string) which is an array of octets and Code_Points which an iterator produces as it iterates over those strings. I was intending to have an iterator for grapheme clusters and other units.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 12:56:34 +0100

> There are two issues: Unicode in source code, that the compiler must understand, and Unicode in strings, that your program must understand.

And this is where the Ada standard gets it wrong, in the encodings package re utf-8.

Unicode is a superset of 7-bit ASCII not Latin 1. The high bit in the leading octet indicates whether there are trailing octets. See https://github.com/Lucretia/uca/blob/master/src/uca.ads#L70 for the data layout. The first 128 "characters" in Unicode match that of 7-bit ASCII, not 8-bit ASCII, and certainly not Latin 1. Therefore this:

```
package Ada.Strings.UTF_Encoding
   ...
   subtype UTF_8_String is String;
   ...
end Ada.Strings.UTF_Encoding;
```

Was absolutely and totally wrong.

...and, before someone comes back with "but all the upper half of latin 1" are represented and have the same values." Yes, they do, in Code points which is a 32 bit number. In UTF-8 they are encoded as 2 octets!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 14:52:43 +0200

> subtype UTF_8_String is String;

> Was absolutely and totally wrong.

It is a practical solution. Ada type system cannot express differently represented/constrained string/array/vector subtypes. Ignoring Latin-1 and using String as if it were an array of octets is the best available solution.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 14:00:39 +0100

They're different types and should be incompatible, because, well, they are. What does Ada have that allows for this that other languages don't? Oh yeah! Types!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 15:10:49 +0200

They are subtypes, differently constrained, like Positive and Integer. Operations are the same, values are differently constrained. It does not make sense to consider ASCII 'a', Latin-1 'a', UTF-8 'a' different. It is the same glyph differently encoded. Encoding is a representation aspect, ergo out of the interface!

BTW, subtype is a type.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 14:15:24 +0100

> They are subtypes, differently constrained, like Positive and Integer.

No they're not. They're subtypes only and therefore compatible. The UTF string isn't constrained in any other ways.

> It is the same glyph differently encoded.

As I already said in Unicode the glyph is not part of Unicode. The single code point character concept doesn't exist anymore.

> BTW, subtype is a type.

subtype is a compatible type.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 15:31:28 +0200

> [...]

> subtype is a compatible type.

Ada subtype is both a sub- and supertype, i.e. substitutable [or so the compiler thinks] in both directions. A derived tagged type is substitutable in only one direction.

Neither is fully "compatible", because otherwise there would be no reason to have an exactly same thing.

From: Vadim Godunko <vgodunko@gmail.com>
Date: Mon, 19 Apr 2021 06:18:22 -0700

> What's the way to manage Unicode correctly?

Ada doesn't have good Unicode support. :(So, you need to find a suitable set of "workarounds".

There are few different aspects of Unicode support need to be considered:

1. Representation of string literals. If you want to use non-ASCII characters in source code, you need to use -gnatW8 switch and it will require use of Wide_Wide_String everywhere.
2. Internal representation during application execution. You are forced to use Wide_Wide_String at the previous step, so it will be UCS4/UTF32.
3. Text encoding/decoding on input/output operations. GNAT allows to use UTF-8 by providing some magic string for Form parameter of Text_IO.

It is hard to say that it is a reasonable set of features for the modern world. To fix some of the drawbacks of the current situation we are developing a new text processing library, known as VSS.

https://github.com/AdaCore/VSS

At the current stage it provides encoding independent API for text manipulation, encoders and decoders API for I/O, and JSON reader/writer; regexp support should come soon.

Encoding independent API means that applications always use Unicode characters to process text, independently from the real encoding used to store information in memory (UTF-8 is used for now, UTF-16 will be added later for interoperability with Windows API and WASM). Coders and encoders allow translation from/to different encodings when applications exchange information with the world.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 19 Apr 2021 15:24:36 +0200

> They're different types and should be incompatible, because, well, they are.

They are not so different. For example, you may read the first line of a file in a string, then discover that it starts with a BOM, and thus decide it is UTF-8.

BTW, the very first version of this AI had different types, but the ARG felt that it would just complicate the interface for the sake of abusive "purity".

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 19 Apr 2021 06:50:42 -0700

What's the purpose of "pragma Wide_Character_Encoding (UTF8);"?

This pragma specifies the character encoding to be used in program source text...

https://docs.adacore.com/gnat_rm-docs/html/gnat_rm/gnat_rm/implementation_defined_pragmas.html#pragma-wide-character-encoding

I would suggest also this article to read:

https://two-wrongs.com/unicode-strings-in-ada-2012

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 17:48:18 +0200

> Code points which is a 32 bit number. In UTF-8 they are encoded as 2 octets!

A code point has no size. Like universal integers in Ada.

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 18:07:32 +0200

> They're different types and should be incompatible, because, well, they are

I agree.

In Python2, encoded and "decoded" strings are of same type "str". Bad design.

In Python3, "decoded" strings are of type "str" and encoded strings are of type "bytes" (byte array). Both are different things and can't be assigned one to the other. Much more clear for the programmer. It should be the same in Ada. Different types.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 20 Apr 2021 14:06:26 -0500

> They're different types and should be incompatible

If they're incompatible, you need an automatic way to convert between representations, since these are all views of the same thing (an abstract string type). You really don't want 35 versions of Open each taking a different string type.

It's the fact that Ada can't do this that makes Unbounded_Strings unusable (well, barely usable). Ada 202x fixes the literal problem at least, but we'd have to completely abandon Unbounded_Strings and use a different library design in order for it to allow literals. And if you're going to do that, you might as well do something about UTF-8 as well -- but now you're going to need even more conversions. Yuck.

I think the only true solution here would be based on a proper abstract Root_String type. But that wouldn't work in Ada, since it would be incompatible with all of the existing code out there. Probably would have to wait for a follow-on language.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 20 Apr 2021 14:13:30 -0500

> BTW, the very first version of this AI had different types, but the ARG felt that it would just complicate the interface for the sake of abusive "purity".

Unfortunately, that was the first instance that showed the beginning of the end for Ada. If I remember correctly (and I may not ;-), that came from some people who were wedded to the Linux model where nothing is checked (or IMHO, typed). For them, a String is simply a bucket of octets. That prevented putting an encoding of any sort of any type on file names ("it should just work on Linux, that's what people expect"). The rest follows from that.

Those of us who care about strong typing were disgusted, the result essentially does not work on Windows or macOS (which do check the content of file names - as you can see in GNAT compiling units with non-Latin-1 characters in their names), and I don't really expect any recovery from that.

Accessibility Rules and Aliased Parameters

From: Simon Wright <simon@pushface.org>
Subject: GCC 11 bug? lawyer needed
Date: Mon, 03 May 2021 17:08:20 +0100
Newsgroups: comp.lang.ada

This code results in the error shown:

```
 1. package Aliased_Tagged_Types is
 2.
 3.    type T is tagged null record;
 4.
 5.    function P (Param : aliased T)
       return Boolean
 6.      is (False);
 7.
 8.    function F (Param : T) return
        Boolean
 9.      is (Param.P);
              |
   >>> actual for explicitly aliased formal is
too short lived
10.
11. end Aliased_Tagged_Types;
```

The compiler code that results in this error is at sem_ch4.adb:1490, and was introduced for Ada202x accessibility checking reasons.

```
--  Check whether the formal is aliased
-- and if the accessibility level of the
-- actual is deeper than the accessibility
-- level of the enclosing subprogram to
-- which the current return statement
-- applies.

  [...]

 if Is_Explicitly_Aliased (Form)
   and then Is_Entity_Name (Act)
   and then Static_Accessibility_Level
             (Act, Zero_On_Dynamic_Level)
               > Subprogram_Access_Level
                 (Current_Subprogram)
 then
    Error_Msg_N ("actual for explicitly aliased
    formal is too" & " short lived", Act);
 end if;
```

For those interested, this issue affects Alire.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 4 May 2021 22:54:43 -0500

We spent a lot of time and effort in the ARG talking about this case (see AI12-0402-1). The Ada 2012 RM does indeed say this case is illegal. The reason is that aliased parameters are designed so that one can return part of them in the return object of the function. And a normal parameter is assumed to be local (since its accessibility is unknown) - that means it is too local for an aliased parameter of a function that is used in some non-local way (including being returned from a non-local function).

However, since one cannot return a part of a parameter for a function that returns an elementary type (other than anonymous access returns, which have special rules anyway), we added an exception to the rules for that case in Ada 202x. (We tried a number of more liberal exceptions, but they were complex and had [unlikely] holes.) So the most current rule is that the call of P is legal.

That wasn't decided until the December ARG meeting, so it happened after the GNATPro 21 release (and I expect that the GNAT CE is derived from that version). And I'd guess that in Ada 2012 mode, this check would remain as it is (the change was not made retroactively - not sure why).

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 5 May 2021 03:01:06 -0700

> And a normal parameter is assumed to be local (since its accessibility is unknown) - that means it is too local for an aliased parameter of a function that is used in some non-local way

RM 3.10(9/3): Finally, a formal parameter or generic formal object of a tagged type is defined to be aliased.

RM 6.4.1(6/3): If the formal parameter is an explicitly aliased parameter, the type of the actual parameter shall be tagged or the actual parameter shall be an aliased view of an object.

Both of these conditions are fulfilled here.

There are many more places about explicitly aliased parameters in the RM. I've read them all. It left me wondering.

I do not see what aliasing a tagged parameter buys. A parameter of a tagged typed is aliased per se, or do I misread the RM.

I'm having big problems trying to understand the RM.

I will try to grock the AI.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 5 May 2021 09:10:01 -0700

> I will try to grock the AI.

Hm, I'm still confused. Can anyone please come up with some examples that explain what this is all about?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 5 May 2021 19:39:11 -0500

> Can anyone please come up with some examples that explain what this is all about?

See 6.4.1(6/3): there is an accessibility check on the actual parameter of an aliased parameter. This allows an aliased parameter to have the accessibility of the return object of a function, rather than local accessibility. There's a bunch of rules in 3.10.2 that combine to have the right effect.

You see the result in an operation like "Reference" in the containers. If you have:

```
function Foo (A : in out Container;
Idx : in Natural) return access Element;
```

then an implementation of:

```
function Foo (A : in out Container)
return access Element is
begin
   return A.Data(Idx)'Access; -- (1)
end Foo;
```

(1) is illegal, as A has local to Foo accessibility, while the anonymous access has the accessibility of the return object (the point of call), which is necessarily outside of Foo.

You can change (1) to:

```
return A.Data(Idx)'Unchecked_Access;
-- (1)
```

but now you can create a dangling pointer, for instance if Foo is assigned to a library-level access type and the actual for A is not library-level.

But you can change the parameter to "aliased", then the accessibility check is moved to the call site (where it must always succeed for the vast majority of calls). There's no accessibility check at (1) in that case (which could be at best a dynamic check, which is a correctness hazard, and also has an overhead cost). And you still have the safety of not being able to create a dangling pointer.

It is a bit weird that this property is tied to "aliased" parameters. This property came first, and we discussed the syntax to use for a long time. Eventually it was decided to call them "aliased" parameters, but of course that meant it was necessary to generalize the usages.

This special rule does have the downside of being able to fail in some safe cases, like the one noted by the OP. That doesn't happen for procedures, since aliased parameters have no special semantics for procedures. We decided to remove the special semantics for functions for which it is impossible to return a part of the parameter (that is, any elementary-returning function), as that special semantics provides no benefit in such a case (but it does have a cost).

I agree that the original author of that program should not have used "aliased" in the way that they did (they don't need the special semantics), but we realize that some people would prefer to *explicitly* mark things as aliased when they are going to take 'Access (and not worry about the type of the parameter -- after all, it could change). That is, they don't want to depend on the implicit behavior of tagged types -- or perhaps they don't even know about it. Which leads to the problem that occurs here, as "aliased" has slightly different meanings for functions (now just composite functions) and procedures.

Since this is real code that didn't work as expected, it seemed to make sense to reduce the problem with a minor language tweak.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Thu, 6 May 2021 06:07:23 -0700

Thank you, Randy, for the nice explanation. There're still some hazy places, but I begin to see the big picture.

From: Simon Wright <simon@pushface.org>
Date: Thu, 06 May 2021 21:02:54 +0100

The original code, from the Alire project, had (I've edited it slightly)

```
package Holders is new
Ada.Containers.Indefinite_Holders
(Node'Class);

type Tree is new Holders.Holder

  and ...

function Root (This : Tree) return
Node'Class is (This.Constant_Reference);
```

where that Constant_Reference is inherited (eventually) from Ada.Containers.Indefinite_Holders. Holder,

```
function Constant_Reference
  (Container : aliased Holder) return
  Constant_Reference_Type;
pragma Inline (Constant_Reference);
```

Shame it had to be there.

I've just tried splattering 'aliased' wherever the compiler told me it was needed; it's now spreading into other packages. Ugh.

The solution might just be using composition rather than inheritance.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 6 May 2021 22:51:43 +0200

> The solution might just be using composition rather than inheritance.

In my experience mixing handles with target types does not work anyway regardless of the accessibility rules mess.

I tend to use interfaces instead:

```
type Abstract_Node_Interface is interface
 ...;
```

Then both the handle and the target type implement Abstract_Node_Interface. The target type goes into hiding, the client needs not to see it.

This requires manual delegation in all primitive operations of handles: dereference + call. But in the end it pays off. Especially with trees, because in mutator operations I can check the reference count of the node and choose to clone it (and maybe the subtree) if there are multiple external handles to it.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 6 May 2021 18:59:28 -0500

> function Constant_Reference

> (Container : aliased Holder) return Constant_Reference_Type;

> pragma Inline (Constant_Reference);

Constant_Reference is the case for which these semantics were designed. Hard to avoid it there. ;-)

Note that by returning Node'Class rather than an elementary type, you don't get to use the new rule tweak. Since all tagged types are by-reference (not by copy), the "Root" routine has to return the object that it has, which ultimately is part of Tree. So you actually need "aliased" on Root, since you are (ultimately) returning a part of the formal parameter (and which could become dangling if you pass in an object which is too local).

> I've just tried splattering 'aliased' wherever the compiler told me it was needed; it's now spreading into other packages. Ugh.

I think you need to make a copy of the return object somewhere; the obvious answer is to replace function Constant_Reference with function Element. Of course, if the return object is large enough, that could be expensive. (That doesn't work if you want to write the node, but the use of Constant_Reference doesn't allow that anyway, so in this case it doesn't matter.)

> The solution might just be using composition rather than inheritance.

Yeah, or using handles more as Dmitry says. In any case, it seems like some redesign is necessary.

From: Simon Wright <simon@pushface.org>
Date: Sat, 08 May 2021 11:17:18 +0100

> I think you need to make a copy of the return object somewhere; the obvious answer is to replace function Constant_Reference with function Element.

That appears to be a fine workaround! Thanks!

Stacktraces on Raspberry Pi

From: Björn Lundin <b.f.lundin@gmail.com>
Subject: stacktrace gan on raspberry pi
Date: Mon, 10 May 2021 18:01:50 +0200
Newsgroups: comp.lang.ada

Hi!

I got a raspberry pi 4 - 8 Gb, with

Ubuntu 20.04 LTS on.

I use the GNAT provided by apt -

ubuntu@ubuntu:~/svn/wcs-std/target/message$ gnatls -v

GNATLS 9.3.0

Copyright (C) 1997-2019, Free Software Foundation, Inc.

I seem to get some info out on a crash [...] but addr2line gives

ubuntu@ubuntu:~/svn/wcs-std/target/message$ addr2line -e

./message_utility 0xaaaab8fc2b84 0xaaaab8fc5924 0xaaaab900c364 0xaaaab90024dc 0xaaaab8fbab48 0xaaaab8fbe364 0xaaaab8fb9848 0xffffa687c08c 0xaaaab8fb9898

??:0

[Repeated for every address. —arm]

??:0

Is stacktracing not implemented (which I suspect it is not) or is there another tool to use? (like atos on macos)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 11 May 2021 13:17:26 +0200

AFAIK and all necessary disclaimers...

You cannot get trace under GNAT ARM, because this is what we recently requested for our GNAT Pro ARM cross compiler. AdaCore confirmed the issue and fixed it for us.

So, in some near future it might arrive at the FSF GNAT.

Proliferation of Reserved Words

From: Jeffrey R. Carter
Subject: Proliferation of Reserved Words
Date: Mon, 31 May 2021 22:51:56 +0200
Newsgroups: comp.lang.ada

Ada 83 (in)famously had 63 reserved words, which was considered a lot at the time (languages like C and Pascal had about half that). Considering only those related to tasking, there were 7:

abort accept do entry select task terminate

Yet many of these have similar/related meanings, and perhaps some overloading would have been a good idea.

entry and accept ... do go hand in hand. One could replace accept with something like an entry body, eliminating 2 reserved words.

An entry is very like a procedure, so one could use procedure instead. It might be necessary to distinguish between a "task procedure" (declared in a task spec) and a "normal procedure" (declared anywhere else). Another reserved word eliminated.

abort/terminate are pretty much the same thing. We could eliminate abort and just use terminate. (One could argue for using end, but given how often "end Name;" appears when not terminating a task, that would be confusing.)

So we're left with select, task, and terminate, less than half as many. I haven't looked in detail at the others, but presumably some reduction is possible there.

Ada 95 added protected and requeue. Some Ada-83 compilers implemented "passive tasks" that were similar to protected objects; formalizing that would have required defining pragma Passive, leaving no need for protected.

There may be a need for requeue, but I've only used it to work around the limitations on what a protected action may do, so I'm skeptical.

ISO/IEC 8652:2007 added synchronized. I think that could have simply reused task.

Ada 12 didn't expand this set of reserved words.

Ada 2X proposes adding parallel. Again, I think reusing task ("task begin" and "task loop") would be fine.

So we will have 11 tasking-related reserved words (unless I've missed some), but we only need select, task, and terminate (and maybe requeue), nearly a factor of 4 difference.

Maybe Ada 3X will add concurrent, and then there won't be any tasking terms that aren't reserved words.

What do others think? Should Ada have made a greater effort at overloading reserved words from the beginning? Should we belatedly object to adding parallel when we have so many choices already? Or is having a large set of reserved words, many of them with similar meanings, a good thing?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 31 May 2021 23:27:49 +0200

I believe that most reserved keywords can be simply unreserved. Actually there is no syntactic necessity except for a few. The rest is kept reserved for the sake of regularity only.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 1 Jun 2021 00:54:31 -0500

At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved. The last time (and I forget precisely when that was), the ARG had a slight majority in favor of unreserved keywords as well as reserved words. However, it was resoundly rejected at the WG 9 level. At that time, WG 9 still voted by countries, and it turned out that pretty much everyone in favor of unreserved keywords were from North America. Most Europeans were appalled. Of course, that meant a WG 9 vote with 2 in favor and a large number against.

So whenever you think there are too many reserved words in Ada, be assured that it was repeatedly suggested that they not all be reserved, but certain countries would not allow it. At this point, we've given up, since it really would not help much - the majority of words that likely ever be reserved already are (it would most likely matter if a new proposal tried to reserve some commonly used term - "yield" came up some proposals for Ada 202x that didn't go anywhere).

Jeff Carter should note the 8 different uses for "with" in the syntax before he accuses anyone of not reusing reserved words in Ada. It's just the case that it's hard to write something meaningful with the existing reserved words (we almost always try).

"parallel" is an interesting case. In my world view, it is wildly different from a task, because it is *checked*, does not *block* or *synchronize* with another thread (all synchronization is via objects or completion), is automatically created (in looping constructs) and therefore requires substantial less care than writing a task. There is another world-view where essentially the checking is not worthwhile and ergo must be suppressed, that performance matters to the point at which a compiler isn't allowed to make choices, and essentially requires *more* care than a task. In that second world-view, parallel constructs are either harmful or worthless. But even there, having a keyword makes it a lot easier to avoid them than trying to figure out which libraries to block. :-)

From: Paul Rubin
<no.email@nospam.invalid>
Date: Tue, 01 Jun 2021 00:40:50 -0700

> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

I remember this as a fundamental decision of PL/I that made PL/I quite hard to parse using the automata-based methods developed not long afterwards. I don't know what consequences that had for PL/I or anything else, if any. But I think it was retrospectively considered a mistake. It's a lot easier to separate parsing and scanning if you can have reserved words.

OTOH I know that C compilers sometimes (usually?) handle typedefs by having the parser tell the scanner to treat the typedef name as keyword-like, after it sees that a typedef has been defined.

From: Jeffrey R. Carter
Date: Tue, 1 Jun 2021 11:51:44 +0200

> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

Unreserved keywords are one approach, though I'm not aware if they come with any negatives. Then the question becomes which reserved words could become unreserved keywords. (This can be restricted to reserved words related to tasking/concurrency to avoid going through all the reserved words.)

> At that time, WG 9 still voted by countries

Does that imply that the voting has since changed and such a proposal might now be accepted?

> Jeff Carter should note the 8 different uses for "with" in the syntax before he accuses anyone of not reusing reserved words in Ada.

I agree that the ARG has done a good job in reusing reserved words in many cases, "with" being the most obvious. I concentrated on tasking/concurrency reserved words since that seems to be an exception.

From: Robin Vowels
<robin.vowels@gmail.com>
Date: Thu, 3 Jun 2021 01:48:29 -0700

>> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

> I remember this as a fundamental decision of PL/I that made PL/I quite hard to parse using the automata-based methods developed not long afterwards. I don't know what consequences that had for PL/I or anything else, if any. But I think it was retrospectively considered a mistake.

It was definitely never considered a mistake in PL/I. Not having reserved words means that you do not have to steer clear of using any particular words when you design a program. It also means that a program will continue to compile even when new keywords are introduced into the language. Over the years, new keywords were introduced into PL/I, without invalidating existing programs.

Reserved words are the bane of COBOL.

GNAT in Homebrew

From: Simon Wright
<simon@pushface.org>
Subject: Homebrew, GNAT
Date: Tue, 01 Jun 2021 09:04:41 +0100
Newsgroups: comp.lang.ada

Homebrew (https://brew.sh) is a package manager for macOS.

Since releasing GCC 11.1.0 for macOS (at Sourceforge and now Github[1]), people have been saying what a good idea it would be to have it in Homebrew.

I understand that a binary (pre-built) component for Homebrew is called a "cask". If someone who knows how to build a "cask" wants to do so for GCC+Ada I would help, but for the moment that's as far as it goes.

From: Simon Wright
<simon@pushface.org>
Date: Tue, 01 Jun 2021 17:10:03 +0100

> Since releasing GCC 11.1.0 for macOS (at Sourceforge and now Github[1]),

[1] https://github.com/simonjwright/building-gcc-macos-native/releases/tag/gcc-11.1.0.1

From: Bill Findlay
<findlaybill@blueyonder.co.uk>
Date: Wed, 02 Jun 2021 00:02:36 +0100

How does that relate to GNAT CE 2021? I see that AdaCore have not released a version for macOS.

From: Simon Wright
<simon@pushface.org>
Date: Wed, 02 Jun 2021 09:51:15 +0100

No; and apparently GNAT CE 2021 is going to be the last CE release for any target.

It is possible to build CE 2021 for macOS (I and another on the GNAT-OSX mailing list are disagreeing somewhat on how to configure for this), but at some point we have to bite the bullet.

What I'm not sure of is gnatprove. If the compiler sources don't match what gnatprove expects you'll get build failures or, at best, runtime failures. And how far could you trust it even if it appeared to work?

Of course, if you need to trust it you'll be happy to pay.

From: Mark Lorenzen
<mark.lorenzen@gmail.com>
Date: Thu, 3 Jun 2021 22:55:56 -0700

> No; and apparently GNAT CE 2021 is going to be the last CE release for any target.

Why do you say that? Can you provide any sources?

From: Simon Wright
<simon@pushface.org>
Date: Fri, 04 Jun 2021 08:38:10 +0100

> Why do you say that? Can you provide any sources?

Of course, I may be macOS-biased here :-)

Remarks at [1],

"We see a majority in favor or recommending GNAT FSF.

"The result is less clear for the removal of GNAT community. The comments along the answers show that people against this are worried about the ease of use. So we are going to work on that aspect."

"We don't expect anyone to build GCC/GNAT themselves, and that is why part of our plan is to help maintainers of OS distribution make good GNAT package."

"AdaCore will continue to provide the SPARK toolset on Linux and Windows. There is no runtime coming with the toolset, so no possible license confusion, it's only an analysis tool!" [[Can we run Linux apps in Docker on a Mac? looks possible, and might I think be an OK solution for gnatprove given the above. M1 macs??]]

and [2],

"Most likely this version of the compiler will be the last in the GNAT Community Edition release chain. In the future, the compiler collected from open source GCC texts can be installed using a batch manager Alire."

[1]

https://www.reddit.com/r/ada/comments/j6oz6i/results_of_the_survey_on_the_future_of_gnat/

[2] https://www.altusintel.com/public-yy39qc/

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Fri, 4 Jun 2021 03:08:16 -0700

Thank you very much. It looks like AdaCore will provide builds of FSF GCC to distro maintainers instead of distributing GNAT as CE. That's fine - as long as I don't have to build GNAT myself from the FSF distro :-)

Ada Lovelace Pint Glass

From: Anatoly Chernyshev <achernyshev@gmail.com>
Subject: Ada Lovelace Pint Glass
Date: Fri, 4 Jun 2021 03:51:38 -0700
Newsgroups: comp.lang.ada

I'm sure many of you will like the idea of having a beer glass dedicated to lady Ada:

https://cognitive-surplus.com/collections/beer-glasses/products/ada-lovelace-pint-glass-1

Web Frontend in Ada 2012

From: Marius Amado-Alves <amado.alves@gmail.com>
Subject: Any chance of programming a web frontend in Ada 2012?
Date: Tue, 8 Jun 2021 01:56:14 -0700
Newsgroups: comp.lang.ada

It seems that currently the only languages web browsers execute reliably are JavaScript and JBC (Java Byte Code), with WASM (Web Assembly) soon to join the group.

Is there a way to program a web frontend in Ada 2012, maybe by translation to one of the above languages?

(Preferably with a binding to the DOM and the BOM.)

(Maybe via LLVM? GNAT already generates LLVM, right?)

Thanks a lot.

From: Jeffrey R. Carter
Date: Tue, 8 Jun 2021 11:21:23 +0200

Have you looked at Gnoga? https://sourceforge.net/projects/gnoga/

From: Max Reznik <reznik@adacore.com>
Date: Tue, 8 Jun 2021 02:35:26 -0700

Indeed, there is a project to run Ada in the browser using WebAssembly. It's named AdaWebPack[1].

It provides a toolchain based on GNAT LLVM and a customized runtime.

The runtime has some restrictions for now, such as no exception handling due to current state of WebAssembly. The toolchain building could be complicated, so the project provides a Docker image.

The project provides the simplest example (See online: https://www.ada-ru.org/files/wasm/index.html).

This site (in Russian) uses it to provide some construction calculations https://mycalcs.ru/

Also take a look a short blog post: https://blog.adacore.com/android-application-with-ada-and-webassembly

I think, you can reach the author on the Telegram channel https://t.me/ada_lang

[1] https://github.com/godunko/adawebpack

Grom: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 8 Jun 2021 07:45:00 -0700

> Have you looked at Gnoga? https://sourceforge.net/projects/gnoga/

Yes. Looks great and reliable. Quick read of the well written user_guide (I must be rainman cause I spotted this typo: Gnoga.Gui.Element.Canvas)

Looks too complicated for my present needs, but definitely a reference to keep. Thanks, Jeff.

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 8 Jun 2021 08:01:20 -0700

A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

From: Max Reznik <reznik@adacore.com>
Date: Tue, 8 Jun 2021 08:22:53 -0700

Speaking about "a single language" for frontend and backend. There was an idea to port Annex E (DSA) to AdaWebPack and use it as a communication channel between a web server written in Ada and WebAssembly client part.

From: Maxim Reznik <reznikmm@gmail.com>
Date: Tue, 8 Jun 2021 09:26:57 -0700

> A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

I made some progress in this direction, but ASIS4GNAT is abandoned and my project is suspended. The only user I have moved to AdaWebPack :)

The source code of the translator is part of the Matreshka project.

https://forge.ada-ru.org/matreshka/wiki/Web/A2JS

There is GitHub mirror:

https://github.com/reznikmm/matreshka

From: Shark8 <onewingedshark@gmail.com>
Date: Thu, 10 Jun 2021 06:33:55 -0700

> Speaking about "a single language" for frontend and backend. There was an idea to port Annex E (DSA) to AdaWebPack and use it as a communication channel between a web server written in Ada and WebAssembly client part.

I've been advocating this idea [well similar, I'm not a fan of WASM] for years now. Seriously: The DSA has the potential to be the Ada equivalent of being the "killer app" or "killer feature" for getting use.

From: Paul Rubin <no.email@nospam.invalid>
Date: Thu, 10 Jun 2021 17:44:18 -0700

> A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

It's unclear to me why anyone would want to do this, since it combines the disadvantages of both Javascript (interpreted, non-deterministic execution) and Ada (manual memory management etc.)

If you want to use a typed language that gets translated into Javascript, you might be better off using Purescript (purescript.org) or even something like Agda.

Ada to WASM might make more sense than Ada to Javascript, of course.

Going beyond Ada 2022

From: Stephen Davies <joviangm@gmail.com>
Subject: Adacore Blog - Going Beyond Ada 2022
Date: Tue, 8 Jun 2021 06:28:35 -0700
Newsgroups: comp.lang.ada

The following may be of interest (I was pleased to see fixed lower bounds being considered):

https://blog.adacore.com/going-beyond-ada-2022

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 9 Jun 2021 07:11:43 +0200

Of course, everyone is welcome with helping the evolution of Ada. But I remind the community that there is the Ada-Comment list (http://www.ada-auth.org/comment.html) which is open to the public.

I'm afraid that this initiative of AdaCore is another step in trying to control the language.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 9 Jun 2021 08:10:49 -0700

What troubles me is this statement:

<quote>What happens afterwards

...

Finally, a member of the AdaCore team will give a final decision about the RFC's inclusion in GNAT, and potential submission to the ARG if necessary.</quote>

Sounds like creating dialects. RFCs implemented in GNAT, but not submitted to ARG. What a mess!

Ada is not owned by AdaCore!

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 09 Jun 2021 09:33:10 -0700

It sounded like the RFCs platform is intended to experimentally test proposed new features, which sounds better than standardizing them without testing them first. And hasn't GNAT always has had extensions beyond the ARM Is this anything new?

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 9 Jun 2021 13:53:10 -0700

Yes, extensions as allowed in the RM - impl-defined pragmas, attributes...

But not extensions with new syntax.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Thu, 10 Jun 2021 04:13:26 -0700

I must admit I do not see the grudge here. As I understand it, the goal is indeed to test proposals in practice, before they make it to the standard. There has been a number of evolutions to the language that are not so convenient to use, for instance, or not flexible enough. Having a prototype implementation for people to play with is a nice idea (and what most other languages do in practice). AdaCore does not propose to control the language. As far as I can tell, these prototypes (like early implementation of what is already in Ada 2020) are generally controlled via the -gnatX switch. If you do not use that, then you do not have access to those new proposals either.

This is akin to implementing some pre-processing tool (for instance using ASIS or libadalang) to test those prototypes, except it might be easier to do directly in the compiler for AdaCore. But nothing prevents anyone from writing such a preprocess to test their own proposals.

The language remains controlled by the standard, and although there is a large number of AdaCore employees in the ARG, that's not all of it. The ada-comment list has another purpose than discussing tentative evolution to the language (and email doesn't lend itself too well to that purpose anyway).

From: Andreas Zeurcher <zuercher_andreas@outlook.com>
Date: Sun, 13 Jun 2021 13:29:44 -0700

> I must admit I do not see the grudge here.

Complaint or concern (for the welfare of Ada) should be the word there. Grudge is a loaded term of long-term hatred, which by definition is absent in this •newly•-arisen topic of 2 competing RFC-esque nonemail commentary/planning/consensus-building forums (ARG's versus AdaCore's). By having 2 competing consensus-building forums, clearly not all the wood can ever truly be behind one arrow in the consensus building, to paraphrase Scott McNealy.

> As I understand it, the goal is indeed to test proposals in practice, before they make it to the standard.

Your wording implies the fundamental danger: If AdaCore productizes a particular design & implementation prior to even submitting an AI to ARG (as "making it to the standard"s body), then the ARG is implicitly compelled to accept or veto, at the wholesale level, the •entirety• of AdaCore's work on this proposed feature

1) when a quite-different alternative might have been wiser and better for Ada or

2) when course-correction at a key point of departure where AdaCore went off-course might have been wiser.

> There has been a number of evolutions to the language that are not so convenient to use, for instance, or not flexible enough. Having a prototype implementation for people to play with is a nice idea

"Having a prototype implementation for people to play with is a nice idea" ••only as long as it remains playing nice & fairly•• at the ARG. As soon as ARG effectively/practically cannot veto & reject some prototype from AdaCore as a partially- or fully-misguided rotten-egg brain-fart, then AdaCore could utilize this technique to try to occlude & preclude all dissenting views in ARG.

> (and what most other languages do in practice).

C++ for example rarely if ever has cases where some core-language feature appeared in GCC or Clang prior to having at least one N-series proposal submitted to the ISO14889 committee (and indeed, not only submitted, but achieving some degree of partial consensus, at least factionally). For example, Clang has automated reference counting (ARC) only in Objective-C-based modes of operation (including in only certain Objective-C-centric situations of Objective-C++), not in C++ proper. Likewise with Microsoft's C++/CLI and C++/CX keeping evolution to the core language separate from the committee-draft-proposal-centric main language—a few nonstandard pragma-esque attributes here & there notwithstanding.

Comparing an ISO/IEC-standardized language's process to, say, Python's is disingenuous, because Python is a language historically with a benevolent dictator-individual and a normative reference-implementation interpreter as 1st-class citizen from which all other Python interpreters or compilers are expected to conform meticulously as 2nd-class citizens. Scala (versus Scala Native) operates much the same way as Python, in this reference-implementation-as-1st-class-citizen-all-others-2nd-class-citizens regard. Certainly what Ada community might fear is a situation where AdaCore's GNAT is the 1st-class citizen reference-implementation to which all other Ada compilers must conform downstream as mere 2nd-class citizens, where Ada would become the Python model and the Scala-ScalaNative model.

> AdaCore does not *propose* to control the language.

(emphasis added)

Not all control schemes in the history of humankind have been publicly announced a priori ahead of time, even by slip-of-the-tongue leaks, let alone full-fledged well-crafted well-publicized proposals. Indeed, some firm control schemes really are pure innocence (not even control schemes at all) at their early stages, only to occur by happenstance as time marches onward (to be realized in historical analysis in retrospect) as quite pernicious after the fact.

> As far as I can tell, these prototypes (like early implementation of what is already in Ada 2020) are generally controlled via the -gnatX switch. If you do not use that, then you do not have access to those new proposals either.
>
> This is akin to implementing some pre-processing tool (for instance using ASIS or libadalang) to test those prototypes, except it might be easier to do direcly in the compiler for AdaCore.

The problem is not implementing industrial-practice prototypes ••of ARG's proposed AIs•• in the GNAT compiler (or any other vendor's compiler). The problem is implementing •nonAIs• (other than pragmas and pragma-esque constructs) in the compiler that then are later harshly utilized as established industrial practice to standardize as close to verbatim from the GNAT implementation as possible, given that no other compiler's design of that feature is as mature. Indeed, ISO/IEC rules strongly favor homologizing •existing•

industrial practice over a standards body pontificating any fresh creativity not yet seen in industrial practice. Homologize is the actual term-of-art there, as utilized throughout ISO, IEC, EU, UN, and other let's-just-get-along international bodies. Any fresh creativity by ARG competing with AdaCore's establish industrial practice goes against the entire concept of homologizing unless ARG (or whichever let's-all-just-get-along homologizing body) can demonstrate that fresh creativity is absolutely necessary, due to insurmountable impracticalities of endorsement of (one of) the existing industrial practice(s) or blending the multiple industrial practices (of which there likely would be none from the other Ada vendors at the time of debate & standardization of the AI).

> But nothing prevents anyone from writing such a preprocess to test their own proposals.

>

> The language remains controlled by the standard, and although there is a large number of AdaCore employees in the ARG, that's not all of it. The ada-comment list has another purpose that discussing tentative evolution to the language (and email doesn't lend itself too well to that purpose anyway).

Then why doesn't AdaCore simply utilize ARG's existing forum of discussion instead of having their own competing forum? It seems that only 2 are needed: ①ARG's comment forum and ②email. It seems that 3 are not needed; it seems that 3's a crowd: 🄰AdaCore's comment forum and 🄱ARG's comment forum and 🄲email.

From: John Mccabe
<john@nospam.mccabe.org.uk>
Date: Mon, 14 Jun 2021 10:35:03 -0000

> C++ for example rarely if ever has cases where some core-language feature appeared in GCC or Clang prior to having at least one N-series proposal submitted to the ISO14889 committee

FWIW, though, the C++ committee appear to think it's acceptable to strangle the specification of certain features based on whether or not it makes it hard for a specific compiler to implement.

A few weeks ago I had reason to look into the justification for something in C++ that appeared, to me, to be stupid and illogical, and the reasoning was because "clang does something this way and, if we took the sensible approach for this feature, it would mean clang would have to massively change".

So now the C++ world is saddled with a specification that's compromised by specific implementations.

I've been trying to find the discussion I had about this with some of my colleagues at work; if I do, I'll let you know what it is!

However, this is, basically, a potential risk with the AdaCore RFC approach; if they forward the feature to the ARG and the ARG comes back with "well, nice, but it would be better if it did this", and AdaCore say "but that would be too hard for us now", then what happens?

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 14 Jun 2021 17:50:26 -0500

> AdaCore say "but that would be too hard for us now", then what happens?

This is a double-edged sword, of course; if something is hard to implement (even if better), it might never get adopted at all (look at Algol 68 for a historical example of a committee ignoring practical considerations).

And this sort of thing has occurred as far back as Ada 9x: various Ada 9x proposals were withdrawn because of opposition from implementers (especially DEC, which never actually built an Ada 95 compiler). Perhaps it would have been better if those proposals had gone forward, but that's hard to say. It's also possible that those proposals would have prevented construction of some Ada 95 compilers.

My point is that there needs to be a balance; one should not let one implementer or one group dictate everything, but one cannot ignore implementers either. (A corollary to that: the implementers should not ignore the ARG, either! That happened to some degree with Ada 202x.)