ADA USER JOURNAL

Volume 41
Number 3
September 2020

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 number brings again important news for Open Source enthusiasts from the GNAT front. A survey was conducted by AdaCore to gather feedback on the idea of discontinuing the Community Editions in favor of better supporting the FSF-maintained version packaged by various Linux distributions. You can read reactions about this idea in thread [1].

The Jupyter notebooks for Ada by Maxim Reznik (first reported here in AUJ 41.2) are quickly taking shape: a series of interactive tutorials demonstrating new features of Ada 202x is already available on-line, with 10 entries at the time of this writing. Find more about these in [2], and of course visit them with your browser to witness their potential first-hand.

If you would like to go down memory lane, two threads about operating systems (supporting or implemented in Ada) contain juicy bits in the Ada and Operating Systems section. Or, if you prefer to look forward to hypothetical future Ada features, a large discussion emerged from the embers of an old thread proposing solutions to the automatic storage of indefinite types [3].

I ask for your indulgence for closing this preface with a project I started and actively develop (in collaboration, chiefly, with Fabien Chouteau from AdaCore): Alire (after Ada Library Repository), a package manager for Ada and SPARK has entered public beta, and debuts in this issue [4]. As of this writing, Alire indexes 130 libraries and executable projects that you can immediately retrieve and build with GNAT without a care in the world about having to go hunting for dependencies. (A technical paper about an early version of Alire was published in AUJ 39.3.)

Sincerely,
Alejandro R. Mosteo.

[1] "Survey on the Future of GNAT Community Edition", in Ada Practice.

[2] "Ada 2020 Jupyter Notebooks", in Ada and Education.

[3] "Proposal: Auto-allocation of Indefinite Objects", in Ada Practice.

[4] "Repositories of Open Source Software", in Ada-related Resources.

Ada and Education

Ada 2020 Jupyter Notebooks

From: Maxim Reznik <reznikmm@gmail.com>
Subject: Ada 2020 Jupyter notebooks
Date: Wed, 2 Sep 2020 06:28:07 -0700
Newsgroups: comp.lang.ada

I'm going to write a series of Jupyter notebooks about Ada 2020 support in GNAT Community Edition 2020.

First two are there:

- Ada 2020: 'Image attribute for any type
- Ada 2020: Redefining the 'Image attribute

https://github.com/reznikmm/ada-howto/tree/ce-2020

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Wed, 2 Sep 2020 23:49:46 -0700

Nice idea, this is a nice way to teach Ada indeed.

New(?) Intros to Ada and Spark on adacore.com

From: Paul Rubin <no.email@nospam.invalid>
Subject: new(?) intros to Ada and Spark on adacore.com
Date: Wed, 02 Sep 2020 23:50:01 -0700
Newsgroups: comp.lang.ada

I don't remember seeing these here before. They look promising: https://learn.adacore.com/courses/courses.html:

- Introduction to Ada
- Introduction to SPARK
- Ada for the C++ or Java Developer
- SPARK Ada for the MISRA C Developer
- Introduction to GNAT Toolchain

https://learn.adacore.com/courses/intro-to-ada/index.html

https://learn.adacore.com/courses/intro-to-spark/index.html

https://learn.adacore.com/courses/Ada_For_The_CPP_Java_Developer/index.html

https://learn.adacore.com/courses/SPARK_for_the_MISRA_C_Developer/index.html

https://learn.adacore.com/courses/GNAT_Toolchain_Intro/index.html

Solutions to J. McCormick Book

From: Werner Aeschbacher <aeschbaw@ieee.org>
Subject: Training Ada
Date: Wed, 23 Sep 2020 11:43:08 -0700
Newsgroups: comp.lang.ada

Does anybody have the solutions to the exercises of the book "Building Parallel, Embedded, and Real-Time Applications with Ada" from John W. McCormick et al ?

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 23 Sep 2020 12:58:45 -0700

Is there a claim that a solution set was published someplace? Your best bet might be to contact the authors.

Sometimes with textbooks (say in mathematics), there is a solutions book available only to instructors, so they can assign homework problems from the textbook and check students' answers against the solutions book.

I have the textbook you mention. It looks good but I haven't gotten around to reading much of it. If there's a particular exercise you're interested in, I might like to give it a try.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 23 Sep 2020 16:37:52 -0500

> Your best bet might be to contact the authors.

Agreed. John McCormick is still involved in Ada (he was on an ARA meeting this morning), so I'd expect he'd be able to give you some information.

Solutions to J. English Book

From: Jack Davy <jules1.davy@gmail.com>
Subject: Learning Ada
Date: Tue, 15 Sep 2020 03:36:34 -0700
Newsgroups: comp.lang.ada

I've just started learning Ada and am using the book "Ada95: The Craft of Object Oriented Programming", by John English. I know there are plenty of other resources such as the one on AdaCore, which covers Ada 2012, but I like the style and flow of this book. Anyway, I was wondering whether anyone in the group has the answers to the end of chapter exercises? The author has now retired and the link to them is dead.

Thanks in Advance!

From: Anders Wirzenius <anders.wirzenius@netikka.fi>
Date: Tue, 15 Sep 2020 17:31:57 +0300

Maybe this helps:

http://archive.adaic.com/docs/craft/craft.html

From: Jack Davy <algojack@tutanota.com>
Date: Tue, 15 Sep 2020 08:07:07 -0700

Thanks Anders, but I already found that link. The download has the code for the book, but no answers. I guess it's not important, I just thought it would be nice to see some sample solutions.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 15 Sep 2020 17:54:35 +0200

I don't have an answer to your exact question but there is no shortage of "sample solutions" in Ada on https://rosettacode.org/wiki/Category:Ada

HTH

PS. I still consider John English's book to be the best introduction to Ada.

From: Simon Wright <simon@pushface.org>
Date: Tue, 15 Sep 2020 18:01:56 +0100

Try here: https://www.dropbox.com/s/8k4xxpj5a67s752/adacraft.tar.gz?dl=0

Nothing like being a pack rat! My hard disk copy is dated 2012-8-25, but I don't know when I retrieved it, must have been several computers ago. Internal dates up to 2001-07-27. Readme says examples tested with GNAT 3.13p!

From: Simon Wright <simon@pushface.org>
Date: Tue, 15 Sep 2020 18:07:13 +0100

Actually, they are at adaic.com: http://archive.adaic.com/docs/craft/craft.html, see the third bullet point.

From: Jack Davy <algojack@tutanota.com>
Date: Tue, 15 Sep 2020 12:03:46 -0700

@ Ludovic, thanks for the link to rosettacode; very good source of examples. And good to hear that you rate the book highly. There don't seem to be many books on Ada, but there is a very recent one for beginners which I will probably get to fill in the gaps not covered by "The Craft". https://www.apress.com/gp/book/9781484254271

@ Simon, thanks, but I already have that file. It contains all the code in the book but not the answers to the end of chapter questions.

By the way, I see the author also wrote a GUI library for Ada called JEWL, the files for which I have also downloaded. Pity it's for Windows only. I'm a Linux user although I do have Win XP on VirtualBox, but I don't believe the current GNAT compiler will run on it.

From: Gautier write-only <gautier_niouzes@hotmail.com>
Date: Tue, 15 Sep 2020 12:28:20 -0700

Other sample sources:

Ada resources:

- https://sourceforge.net/directory/language:ada/
- https://www.adaic.org/ada-resources/

Small samples are embedded in the LEA editor (you can run it from Wine):

https://sourceforge.net/projects/l-e-a/

From the menu: Action / Code sample. Choose your sample. Hit F9 for running.

Some samples stem from Rosetta Code BTW :-)

From: Jerry Petrey <gpetrey@cox.net>
Date: Tue, 15 Sep 2020 16:00:08 -0700

> By the way, I see the author also wrote a GUI library for Ada called JEWL [...]

Yes, his JEWL package is great. I used it many times to create Windows GUI apps and still use it some. I talked to John a number of times - he was very helpful. His book is one of the best!

From: Paul Rubin <no.email@nospam.invalid>
Date: Tue, 15 Sep 2020 18:23:18 -0700

> [...] there is a very recent one for beginners which I will probably get to fill in the gaps not covered by "The Craft".https://www.apress.com/gp/book/9781484254271

I haven't examined that book directly but based on the preview and blurb, it does seem to be beginner oriented, thus likely to have gaps of its own. If you're trying to fill gaps, you probably want something more complete and advanced.

I semi-recently got another book that looks very good, though it's still sitting around without my having read much of it: Analysable Real-Time Systems: Programmed in Ada, by Andy Wellings and Alan Burns. It is basically an updated reprint of an older book by the same authors, self-published in paperback, so it is a good value.

From: Jack Davy <algojack@tutanota.com>
Date: Wed, 16 Sep 2020 00:13:22 -0700

@ Gautier, thanks for the links. When I get Windows 7 on VirtualBox I'll give the LEA editor a try, I'm not so keen on using Wine, it's a bit hit & miss. Also since I learned Vim a few years ago no other editors really do it for me, unless they have Vim bindings ;).

@ Paul, I was thinking that the beginner's Apress book would fill in the gaps regarding Ada 2012 specifically, which as I understand it has changed from previous versions mainly in regard to OOP; I'm assuming I won't need to unlearn anything if I learn the basics from an Ada 95 book. The real-time stuff would be over my head at this point I think, and not really something I had in mind when considering Ada, although I do have a background in electronics, and see that there is Ada compiler for AVR on AdaCore.

The more I look at this language the more I wonder why it isn't more popular. Maybe people just don't like the pascalish syntax, but that never put me off because I learned Turbo Pascal at Uni (25 years ago) and more recently Free Pascal/Lazarus. Never was much of a fan of the curly bracket languages.

From: Jack Davy <algojack@tutanota.com>
Date: Wed, 16 Sep 2020 00:32:32 -0700

I found an impressive list of 'Things to like about Ada' posted by a C/C++ career programmer on the AVR freaks forum (in reply #13) :

https://www.avrfreaks.net/forum/i-didnt-know-you-could-get-ada-avr

My main reason for wanting to learn Ada is the last on his list: "Promotes a professional, anti-hacker mentality. By being unforgiving the language promotes the valuable discipline of specifying and writing code more exactly, without the temptations of slipping into bit-twiddling or other programming habits that subvert (and often break) the data or code models. When proper programming discipline is not enforced by the language then it must be voluntary, and in those cases discipline can and inevitably will slip, but when the language enforces much of that discipline then there are no easy ways to avoid it, and the resulting code is higher in quality and faster to develop."

Maybe that's why Ada isn't more popular - being disciplined isn't easy, and hacking is more fun. But I've learned the hard way that it's actually much more satisfying when your programs are bug-free and work properly the first time you run them. Any language which enforces more thinking and less trial-and-error coding is a winner in my book.

From: Gautier write-only <gautier_niouzes@hotmail.com>
Date: Wed, 16 Sep 2020 02:13:54 -0700

> @ Gautier, thanks for the links. When I get Windows 7 on VirtualBox I'll give the LEA editor a try, I'm not so keen on using Wine, it's a bit hit & miss.

No worries, you can access the same samples (and the same compiler) without LEA, built on your preferred operating system.

>- https://hacadacompiler.sourceforge.io/ (source code here: https://sourceforge.net/p/hacadacompiler/code/HEAD/tree/, mirrored here: https://github.com/zertovitch/hac)

Mutatis mutandis, you get there the "tpc.exe" equivalent, whereas LEA is the "turbo.exe" :-)

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 16 Sep 2020 12:55:58 +0200

> The more I look at this language the more I wonder why it isn't more popular. [...]

I wasn't there when it happened but I read that early Ada 83 compilers were buggy, slow and outrageously expensive because marketed only at one captive customer, the US DoD. (In their defence, Ada is a particularly difficult language to implement well, orders of magnitude more so than Pascal or C). The vendors never really tried to sell Ada development tools outside the military, despite hype that Ada was the language of the future. At around the same time, C++ used the opposite strategy of selling cheap compilers, with the additional advantage of backward compatibility with C, so they won market share. Turbo Pascal was a contender back then but only on DOS and Windows, so it ultimately lost to C++, possibly in no small part because of Borland's refusal to abide by any portable standard. And then Sun marketed Java aggressively with a zero-cost compiler and promises of ultimate portability, and stole the show.

The Ada landscape changed dramatically when the first Free Software Ada 95 compiler, GNAT, arrived, but the damage to the reputation of Ada was very hard to overcome. An entire generation of military and corporate programmers, frustrated by the early compilers, became managers and dismissed Ada out of hand for decades. They and their prejudices have started to retire in the past few years and I think this is one factor in the current renaissance of Ada.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 16 Sep 2020 13:09:54 +0200

> I wasn't there when it happened but [...]

I mostly agree with your analysis, except the last part. The problem is that the culture of programming and overall education became so low that it is no more a race against C++. C++ itself is in defense and losing against languages and practices so overwhelmingly bad that even C looks as a shining beacon. Winter is coming.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Wed, 16 Sep 2020 17:01:41 +0200

> The more I look at this language the more I wonder why it isn't more popular.

Ada is a language for engineering software. Since 98% of developers are unable to do that, Ada will never be popular as long as such people are allowed to develop software.

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 16 Sep 2020 14:29:56 -0700

> @ Paul, I was thinking that the beginner's Apress book would fill in the gaps regarding Ada 2012 specifically, which as I understand it has changed from previous versions mainly in regard to OOP

I think Ada 95 OOP is not really used very much, and the changes in Ada 2012 are things like contracts, and built-in SPARK syntax. You could also look at the online book "Ada Distilled" which is about Ada 95. I found it an ok way to get started, though I never really progressed beyond that.

> I do have a background in electronics, and see that there is an Ada compiler for AVR on AdaCore.

I don't know the current state of that, but some years ago it was rather hard to use or parts were missing or whatever. These days, the AVR is in decline since it is so limited. Everyone uses ARM or maybe soon RISC-V processors even for tiny embedded stuff.

> The more I look at this language the more I wonder why it isn't more popular. Maybe people just don't like the pascalish syntax

Tooling, libraries, language verbosity, etc. As pure language, though, it is still mysterious to me what Rust offers that Ada doesn't.

Today, for most programming, "systems languages" including Ada, C, C++, and Rust are all imho somewhat niche. Unless you are dealing with specialized problems (such as embedded or OS's), computers have almost unbounded resources. So it's easier to get your work done using languages with automatic memory management, unbounded arithmetic, etc.

The main cost is consuming more machine resources and losing some timing determinism, but most of the time you can live with both of those. Ada is best for more demanding applications which usually involve realtime or high reliability constraints.

From: Mart van de Wege <mvdwege@gmail.com>
Date: Fri, 18 Sep 2020 08:53:20 +0200

> Ada is a language for engineering software. [...]

I use it for hobby stuff, for quick solutions (like generating RPG characters). Does not feel like engineering to me.

But what I do like is the elegance of the language, and the ability to describe my problem domain using distinct types.

The 'verbosity' does not bother me. I'm a fluent touch typist, Using the shift key to type braces slows me more than typing out statements to delineate blocks.

The only real nit I have with Ada is that it does not have closures.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Fri, 18 Sep 2020 12:00:47 +0200

> I use it for hobby stuff, for quick solutions (like generating RPG characters). Does not feel like engineering to me.

I do similar things, too, but I always have a design in mind, and usually start with pkg, task, & PO specs and subprogram declarations, so I suspect that after doing this for so long I can engineer simple problems in my head. Presumably others with similar experience or who are better than I do the same.

Ada-related Resources

[Delta counts are from Apr 6th to Jul 20th. --arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Mon, 02 Nov 2020 18:41:21 +0100
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 3_025 (+75) members [1]
- Reddit: 4_720 (+634) members [2]
- Stack Overflow: 1_924 (+60) questions [3]
- Freenode: 90 (+2) users [4]
- Gitter: 64 (+8) people [5]
- Telegram: 90 (+11) users [6]
- Twitter: 67 (+14) tweeters [7]
  92 (+27) unique tweets [7]

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

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

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

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

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

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

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: Mon, 02 Nov 2020 18:41:21 +0100
To: Ada User Journal readership

[This issue sees a newcomer, the Alire package manager project, debuting with 130 Ada projects ready to use. --arm]

Rosetta Code: 747 (=) examples [1]
37 (=) developers [2]
GitHub: 729 (+77) developers [3]
Sourceforge: 276 (+1) projects [4]
Open Hub: 212 (=) projects [5]
Alire:130 (new!) crates [6]
Bitbucket: 88 (-2) repositories [7]
Codelabs: 52 (+1) repositories [8]
AdaForge: 8 (=) repositories [9]

[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: Mon, 20 Jul 2020 09:38:21 +0100
To: Ada User Journal readership

[From this number on, positive ranking changes mean to go up in the ranking. This issue sees the addition of the PYPL ranking, which is computed by analyzing how often language tutorials are searched on Google. The IEEE ranking has seen no updates through 2020, and will be likely dropped soon if this situation persists. --arm]

- TIOBE Index: 39 (+4) 0.35% (+0.07%) [1]
- PYPL Index: 19 (new!) 0.62% (+0.3%) [2]
- IEEE Spectrum (general): 43 (=) Score: 24.8 [3]
- IEEE Spectrum (embedded): 13 (=) Score: 24.8 [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-2019

Ada Reference Manual 2020.1

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Ada Reference Manual info format 2020.1 released.
Date: Fri, 17 Jul 2020 10:23:15 -0700
Newsgroups: comp.lang.ada

ada-ref-man 2020.1 is now available in GNU ELPA.

This includes Ada 202x draft 25, as well as Ada 2012. GNAT Community 2020 has some support for some of the new language features in Ada 202x.

There is also now a searchable info index, containing the entries in the ARM Index.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sat, 18 Jul 2020 22:40:16 -0500

Sounds good, but keep in mind this is a moving target. Draft 26 should be available next week. :-)

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 31 Jul 2020 17:55:47 -0500

> Sounds good, but keep in mind this is a moving target. Draft 26 should be available next week. :-)

My primary computer died (now fixed, knock on wood), and we've since had an ARG meeting, so this new draft will be delayed a couple of weeks. Shouldn't be too far in the future, though.

Ada-related Tools

SweetAda 0.1C-0.1F

From: gabriele.galeotti.xyz@gmail.com
Subject: SweetAda 0.1c released
Date: Tue, 7 Jul 2020 14:30:36 -0700
Newsgroups: comp.lang.ada

I've just released SweetAda 0.1c.

Windows toolchains now have libstdc++ included.

The RISCV32 and RISCV64 toolchains are now deprecated, because they end up the same. So there is now a generic RISCV toolchain. It behaves like the other two, you have just to specify the correct CPU. GCC switches that activate the 64-bit mode are "-march=rv64imafdc" and "-mabi=lp64d". Obviously there is the correspondent RTS target.

The RISCV support is a little bit usable, if you pick the QEMU-RISC-V-32 platform, it runs Ada code and does some primitive I/O in the IOEMU window, stimulating a LED and an 8-bit port.

I've tested Insight and it works very well, breakpoints and other things seem ok.

Other minor adjustments here and there.

I saw in the log that many users still try to download from a non-existent directory, i.e., sweetada.org/software/.... Please update your links, the correct directory is sweetada.org/packages/....

Thanks for your patience, I am also working on documentation.

From: gabriele.galeotti.xyz@gmail.com
Subject: SweetAda 0.1e released
Date: Wed, 22 Jul 2020 11:03:04 -0700
Newsgroups: comp.lang.ada

Hi all. I've just released SweetAda 01.e.

Go to http://www.sweetada.org and download the archive.

RTS and LibGCC packages are still valid @ 0.1c.

- general cleanup and cosmetics
- general infrastructure improvements
- QEMU-RISC-V-32 target can do serial output in a terminal
- IntegratorCP target uses LCD VGA
- Malta MIPS target uses a VGA PCI board
- handling of directories in the cpus hierarchy, which allows selective unit overriding
- Insight can be called as a toolchain component
- IOEMU configuration files are now fully consistent

Next days I will concentrate on generic low-level CPU support, documentation, and restructuring of some redundant units. Let me know, feedback is highly appreciated.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Subject: SweetAda 0.1f released
Date: Wed, 19 Aug 2020 15:40:58 -0700
Newsgroups: comp.lang.ada

Hi all, I've just released the 0.1f version of SweetAda.

- general cleanup and cosmetics
- general infrastructure improvements
- the VGA text driver is now unified across platforms; it is actually used by PC-x86, PC-x86-64 and MIPS Malta
- the ugly handling of network packets (Amiga/FS-UAE and PC-x86) is re-routed to a PBUF FIFO handler (the management is still far from ideal, but is not tied to the ISR like before)
- various I/O have now correct aspect specifiers; in particular some hardware registers with specific sizes are now correctly handled without premature optimizations
- AVR is now part of SweetAda and so 2 platforms exist: ArduinoUno and a QEMU emulator (both ATmega328P);

  the AVR support is primitive and incomplete, but, with an ArduinoUno board, is sufficient to start up the Ada infrastructure and is able to pulse the onboard LED;

  note that programming is performed by means of the AVRDUDE tool, so you should use a version suitable for your environment;

  otherwise you could use the IHEX .hex output file with your preferred tool;

  the QEMU-AVR platform can be used with GDB or Insight to trace the execution of code;
- runsweetada and IOEMU library now correctly show in argv dumps the launched executable instead of a "NULL" tag
- the parser inside the IOEMU library now expose in the .cfg file a variable (LASTPID) that carries the PID of the last launched executable (see QEMU-AVR/qemu.cfg)

There is also a new release of all QEMU emulators, at version 5.1.0.

Please note that the Linux version is linked with the SDL2 library instead of the previous GTK+3.

You can find everything at http://www.sweetada.org

GWindows 31-Jul-2020

From: gautier_niouzes@hotmail.com
Subject: Ann: GWindows release, 31-Jul-2020
Date: Fri, 31 Jul 2020 11:01:11 -0700
Newsgroups: comp.lang.ada

GWindows is a full Microsoft Windows Rapid Application Development framework for programming GUIs (Graphical User Interfaces) with Ada.

GWindows works with the GNAT development system (could be made pure Ada with some effort).

Changes to the framework are detailed in gwindows/changes.txt or in the News forum on the project site.

In a nutshell (since last announcement here):

- a few features from the extensions GWindows.Common_Controls. Ex_List_View and GWindows.Common_Controls. Ex_TV_Generic have been moved to parent package and respective parent types for broader use
- fix: a few records for binding with the Windows API were erroneously 32-bit only

GWindows Project site:
https://sf.net/projects/gnavi/

GWindows GitHub clone:
https://github.com/zertovitch/gwindows

TASH Sources

From: mockturtle <framefritti@gmail.com>
Subject: TASH sources?
Date: Mon, 3 Aug 2020 02:16:15 -0700
Newsgroups: comp.lang.ada

I wanted to try to use the Ada Tcl/Tk binding TASH [1], but the download page has links to www.adatcl.com that sends me to some chinese-written site. I guess www.adatcl.com expired?

Does someone know where I can find the sources of TASH?

Please note that, for reasons too long to be explained here, I am not interested in alternatives to TASH, unless they are Tcl/Tk bindings.

Thank you in advance.

[1] http://tcladashell.sourceforge.net/index.htm

From: mockturtle <framefritti@gmail.com>
Date: Mon, 3 Aug 2020 02:28:20 -0700

I am replying to my own post... Deep down in the Google results I found a github version of TASH

https://github.com/simonjwright/tcladashell

Despite the different name it seems like the original sourceforge TASH (or a fork?) revived on github

From: Simon Wright <simon@pushface.org>
Date: Mon, 03 Aug 2020 14:38:58 +0100

> Do someone know where I can find the sources of TASH?

I altered the project page on SF to point to the new Github site, but forgot about the project web pages. Sorry.

I've been moving my projects to Github; it's a far more pleasant and performant environment, I find.

> [1] http://tcladashell.sourceforge.net/index.htm

This page now points you to https://github.com/simonjwright/tcladashell

From: Simon Wright <simon@pushface.org>
Date: Mon, 03 Aug 2020 14:40:27 +0100

> Despite the different name it seems like the original sourceforge TASH (or a fork?) revived on github

I thought it was the same name?

Anyway, the project has moved to Github, under the same management :-)

From: gautier_niouzes@hotmail.com
Date: Mon, 3 Aug 2020 06:53:18 -0700

There is also on GitHub: https://github.com/thindil/tashy ("TASHY is short from Tcl Ada SHell Younger").

SI Units Checked and Unchecked

From: AdaMagica <christ-usch.grein@t-online.de>
Subject: SI Units Checked and Unchecked - Completela overhauled version
Date: Thu, 13 Aug 2020 05:24:06 -0700
Newsgroups: comp.lang.ada

Simplified design now available:

http://archive.adaic.com/tools/CKWG/Dimension/SI.html

The choice of using dimension checking or not is now made via a generic signature package. The user interface is unchanged.

Resource to Source

From: <s@srin.me>
Subject: Ann: Resource to source
Date: Thu, 27 Aug 2020 12:09:17 -0700
Newsgroups: comp.lang.ada

Tool "resource" is available - (MIT License) at:
https://gitlab.com/cpp8/bindata

This tool can take resource files, graphics, audio etc. and convert them into Ada (or C) source code and it can be compiled and included in the binary.

Developed for pedagogic reasons (https://github.com/RajaSrinivasan/assignments/blob/master/resource.pdf) but hoping it will be useful to the community.

Simple Components 4.51

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components for Ada 4.51 IEEE 754-2008 Decimal
Date: Mon, 31 Aug 2020 15:28:14 +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 (1 September 2020) to the version 4.50:

- The HTTP client behavior changed not to close connection when keep alive flag is set unless the server explicitly requests closing it;
- Non-standard request headers added to the HTTP implementation: X-Requested-By, X-Requested-With, X-XSRF-TOKEN, X-CSRF-TOKEN;
- The package IEEE_754.Decimal32 was added. The package implements IEEE 754-2008 decimal32 format;
- The package IEEE_754.Decimal64 was added. The package implements IEEE 754-2008 decimal64 format;
- The package IEEE_754.Decimal128 was added. The package implements IEEE 754-2008 decimal128 format;
- An implementation of 128-bit integers was added to the package IEEE_754;
- The package IEEE_754.Edit was added;
- The package provides strings formatting facilities for 128-bit integers;
- Fallback time zone names changes in the package GNAT.Sockets. Connection_State_Machine. ELV_MAX_Cube_Client.Time_Zones.

Image_Random

From: PragmAda Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Ann] Image_Random
Date: Thu, 3 Sep 2020 17:17:03 +0200
Newsgroups: comp.lang.ada

Image_Random: True random numbers from a digital camera (under Linux with the GNAT compiler) is now available in case anyone finds it useful.

https://github.com/jrcarter/Image_Random

MP Music Player

From: PragmAda Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Ann] MP
Date: Tue, 15 Sep 2020 22:11:30 +0200
Newsgroups: comp.lang.ada

MP, a Music Player based on the Gnoga audio widget, is available at https://github.com/jrcarter/MP

Ada-related Products

PTC ObjectAda V10.2 for Windows

[This PTC announcement and the following companion were already published in the previous AUJ number, although by date they properly belong in this number, so here they are again. --arm]

From: Shawn Fanning <sfanning@ptc.com>
Subject: Product Release Announcement – PTC ObjectAda V10.2 for Windows
Date: Mon, 27 Jul 2020 16:39:05 -0700
Newsgroups: comp.lang.ada

On July 22, 2020, PTC announced the availability of version 10.2 of our ObjectAda for Windows and ObjectAda64 for Windows products. This new product release provides full support for Ada 2012 language features and represents the completion of the phased implementation strategy PTC adopted for Ada 2012 language feature support within the ObjectAda technology. With ObjectAda for Windows version 10.2, the ObjectAda compiler conforms to the Ada Conformity Assessment Test Suite (ACATS) version 4.1Q and adds several new features not present in the previous release (ObjectAda version 10.1 released in May 2019) including support for storage subpools and the Default_Storage_Pool pragma, execution time enforcement of type invariants, and complete support for new Ada expression forms.

The new installation approach introduced with ObjectAda for Windows v10.x allows ObjectAda to be used with the latest releases of Microsoft's Visual Studio tools and the Windows 10 SDK. ObjectAda version 10.2 includes version 4.0.0 of the ObjectAda Ada Development Toolkit (ADT) Eclipse interface which supports Eclipse 2020-03 (4.15) or later. All of these upgrades combined make ObjectAda for Windows version 10.2 a solid, modern, and effective toolset for development of mission-critical application code in the Ada language. ObjectAda version 10.2 supports Ada 95, Ada 2005, and Ada 2012 compiler operation modes to provide compatibility with previous versions.

Additional information about ObjectAda version 10.2 is available within the Product Release Announcement which can be downloaded from https://www.ptc.com/products/developer-tools/objectada.

Customers with active subscription licenses for ObjectAda for Windows v10.x or ObjectAda64 for Windows v10.x are entitled to a no-charge upgrade to v10.2.

If you are not currently using ObjectAda and wish to learn more or if you are using an earlier release of ObjectAda and wish to upgrade, register your request at https://www.ptc.com/en/products/developer-tools/objectada/contact-sales.

PTC ApexAda V5.2 Embedded for Linux/ARMv8 64-bit

From: Shawn Fanning <sfanning@ptc.com>
Subject: Product Release Announcement – PTC ApexAda v5.2 Embedded for Linux/Armv8 64-bit
Date: Mon, 27 Jul 2020 16:42:37 -0700
Newsgroups: comp.lang.ada

On May 19, 2020 PTC announced the release of the PTC ApexAda v5.2 Embedded for Linux/Armv8 64-bit product. This product is the initial product offering based on a new 64-bit code generator for ApexAda for the Armv8 64-bit (aarch64) architecture and is our latest

release supporting 64-bit embedded application development.

The host operating system for this product is Intel x64 Red Hat Enterprise Linux v7.x/v8.x (or CentOS equivalent) distribution. Using the Linaro GNU cross-development toolchain for 64-bit Armv8 Cortex-A processors on the Linux/Intel64 host, PTC ApexAda supports the generation of Ada 95 / Ada 2005 application images that execute on ARMv8-A 64-bit (aarch64) processors (for example Arm Cortex A53, A57, A72) running 64-bit embedded Linux distributions. Examples of embedded Linux distributions which can be supported are openSUSE Leap v15.1, SUSE Linux Enterprise Server for Arm v15.1, Ubuntu Server 20.04, Wind River Linux and other Yocto-derived Linux distributions with a 64-bit kernel. Reference hardware used for the development and test of ApexAda was the Raspberry Pi 3 Model B/B+. (Raspberry Pi 4 Model B with its larger 4GB RAM configuration and other boards such as the VPX-1703 from Curtiss-Wright Defense Solutions can also be supported by ApexAda.)

Included with the 64-bit embedded compiler is the PTC® ApexAda v5.2 64-bit compiler for Linux native application development. Also included is the integrated ApexAda 64-bit C/C++ compiler which facilitates seamless development of mixed-language applications written in Ada, C, and C++. ApexAda V5.2 Embedded compilers provide a complete cross-development toolchain hosted from Linux distributions including RedHat Enterprise Edition, CentOS, and SUSE. A complete description of PTC ApexAda v5.2 Embedded for Linux/Armv8 64-bit is available within the Product Release Announcement which can be downloaded from https://www.ptc.com/products/developer-tools/apexada .

The addition of the new code generation capability for 64-bit Armv8 processors to ApexAda opens up a whole new landscape for embedded application development using ApexAda. PowerPC processors have for a long time been a design choice for our aerospace and defense customers due to their balance of performance, cost, and power characteristics. Intel processors have offered many of our customers increased performance at a cost of additional complexity and power requirements. Driven by the mobile consumer market, Arm processors provide high performance and low power advantages over Intel processors. We think these advantages combined with the flexibility provided by embedded Linux distributions and the availability of low-cost and high-performance consumer-grade development boards as well as ruggedized 64-bit Arm boards will provide substantial benefits to our customers looking to modernize existing deployed applications while mitigating risks through continued use of the same time-proven and industrial-strength ApexAda compiler technology. The 64-bit Armv8 (aarch64) processors are now well-known and proven processors with a long lifecycle and there are multiple 64-bit Linux distributions available which run on these processors. Follow-on products leveraging the new ApexAda 64-bit Armv8 (aarch64) code generation capability for other real-time operating systems are under development with prioritization based on customer interest and requirements.

If you would like to receive additional information about the new PTC ApexAda v5.2 Embedded for Linux/Intel64 to Linux/Armv8 64-bit product or wish to be contacted by a PTC Developer Tools sales representative regarding evaluations, upgrades and associated pricing, register your request at https://www.ptc.com/en/products/developer-tools/objectada/contact-sales.

Ada and Operating Systems

UNIX OS Written in Ada

From: gdotone@gmail.com
Subject: is there a version of unix written in Ada
Date: Fri, 24 Jul 2020 15:11:47 -0700
Newsgroups: comp.lang.ada

Is there a UNIX-like OS written completely in Ada?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 25 Jul 2020 11:47:35 +0300

The short answer is "no".

There have certainly been operating systems written in Ada -- the OS for the Nokia MPS-10 minicomputer is an example.

There are several real-time kernels and similar low-level SW components written in Ada, but probably they do not qualify as "Unix-like", depending on what you mean by that term.

Why do you ask?

From: Stéphane Rivière <stef@genesix.fr>
Date: Sat, 25 Jul 2020 11:36:57 +0200

See OS section of https://github.com/ohenley/awesome-ada

> There have certainly been operating systems written in Ada -- the OS for the Nokia MPS-10 minicomputer is an example.

Wasn't aware, thanks! Find that... Very few refs on the net...

https://dl.acm.org/doi/abs/10.1145/989798.989799

From: Jesper Quorning <jesper.quorning@gmail.com>
Date: Sat, 25 Jul 2020 07:43:15 -0700

> is there a unix like OS written completely in Ada?

Do not know if it is unix-like, but this [1] looks active. Maybe he needs help..

My own dream was to port GNU/Hurd to Ada while renaming it to something not hurding so much.

[1] https://github.com/ajxs/cxos

From: Andreas Zuercher <ZUERCHER_Andreas@outlook.com>
Date: Sat, 25 Jul 2020 12:20:25 -0700

> is there a unix like OS written completely in Ada?

In 1981, there in fact was one that had 2 public releases with work in progress on Version 3: iMAX-432, depending on how puritanical one wishes to be about what is or is not Unix-like. (iMAX-432 was far more Unix-like than, say, MVS-like or CP/M-like.)

If anyone has an inside negotiating track at Intel (or the contracting firm that Intel hired to develop it), perhaps they would be willing to open-source the old iMAX432 operating system that was released for the iAPX432 processor that was designed from the ground up to have an Ada-centric instruction set. Although it was more Multics-esque than Unix-esque* and although it was written specifically for the iAPX432 (and thus had much iAPX432-only assembly language), it should be relatively easily transliterable into other ISAs because the iAPX432 ISA more closely resembles Java bytecode, LLVM bitcode, and C# CIL/MSIL than other rudimentary machine codes of that era, due to being object-based/OO-lite in the hardware's machine code (which is what doomed the iAPX432 in the early 1980s: it was so complex that it required 3 separate IC dies in 3 separate ceramic packages, and it ran relatively hot).

* Conversely, both Multics & our modern Unix are nowadays birds of the same feather despite the multi-decade dislocation in time from each other, due to both having:

1) multiple threads per address space;

2) multiple DLLs per address-space;

3) multiple memory-mapped files (i.e., mmap(2) in Unixes versus snapping segment-files in Multics);

4) IPC based on multiple threads or multiple processes pending on a single message-queue;

5) soft real-time thread scheduling priorities in addition to time-sharing scheduling priorities;

and

6) a GNU-esque long-form whole-words and short-form abbreviated-letters of each hyphenated command-line flag

are birds of much the same father, as opposed to 1970s-era spartan Unix that abhorred all of these multiplicities, hence AT&T's uni-based name in AT&T's

1970-divorce-from-MIT's/GE's/AT&T's/Honeywell's-Project-MAC in defiance of Project MAC's multi-based name, because the tongue-in-cheek humor of Unix's name as eunuchs is Multics castrated. Eschewing singleton this and singleton that, Unix nowadays is no longer a castrated eunuch, due to reintroducing a cousin-like variant of nearly every multiplicity feature of Multics other than the multiple rings (unless one counts VM hypervisors nowadays as reintroducing a cousin of that one too).

https://en.wikipedia.org/wiki/IMAX_432

From: Stéphane Rivière <stef@genesix.fr>
Date: Sun, 26 Jul 2020 21:45:50 +0200

> I remember someone was writing an OS in Ada, but I do not remember who was, nor the name of the project, nor if it was unix-ish.

In the very old archive https://stef.genesix.org/aide/aide-src-1.04.zip you will find:

- The last RTEMS 3.2.1 Ada sources (yes... old RTEMS releases are offered in two flavors: Ada and C) comes with docs & manuals.

- the Ada sos-os Ada series (based from edu-os in C)

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Mon, 27 Jul 2020 00:15:52 +0200

> - The last RTEMS 3.2.1 Ada sources (yes... old RTEMS releases are offered in two flavors : Ada and C) comes with docs & manuals.

Marte OS implements Minimal Real-Time POSIX.13 in Ada, so it should be Unix-like.

https://marte.unican.es/

The same group recently announced M2OS, which is also in Ada, but not Unix-like.

https://m2os.unican.es/

From: Stéphane Rivière <stef@genesix.fr>
Date: Mon, 27 Jul 2020 09:40:05 +0200

> In 1981, there in fact was one that had 2 public releases with work in progress on Version 3: iMAX-432, depending on how puritanical one wishes to be about what is or is not Unix-like. (iMAX-432 was far more Unix-like than, say, MVS-like or CP/M-like.)

Very interesting Andreas, thanks for this part of Ada and CPU history...

From: Stéphane Rivière <stef@genesix.fr>
Date: Mon, 27 Jul 2020 09:40:04 +0200

> The same group recently announced M2OS, which is also in Ada, but not Unix-like.

> https://m2os.unican.es/

Not aware of that, Thanks Jeffrey

I will test that, the Toolchain is Linux based and includes GDB...

From: nobody in particular <nobody@devnull.org>
Date: Mon, 27 Jul 2020 14:58:36 +0000

> If anyone has an inside negotiating track at Intel (or the contracting firm that Intel hired to develop it), perhaps they would be willing open-source the old iMAX432 operating system that was released for the iAPX432 processor that was designed from the ground up to have an Ada-centric instruction set.

I guess it could be worthwhile contacting Steve Lionel who recently retired from Intel after working for DEC, COMPAQ, HP, on Fortran compilers. He has a blog site, I'll not post the details here so as not to encourage automated spam. Doctor Fortran is his nickname.

Ada on OpenVMS Retake

From: gérard Calliet <gerard.calliet@pia-sofer.fr>
Subject: Ada on OpenVMS, where to have a new beginning
Date: Mon, 17 Aug 2020 19:14:17 +0200
Newsgroups: comp.lang.ada

I participated in a GNAT Ada build for Itanium OpenVMS (https://github.com/AdaLabs/gnat-vms) a few years ago. It is based on a GCC 4.7.3 .

I'm coming back to this work to maintain it and make it evolve, in a general approach of making Ada available in OpenVMS environments (VAX, Alpha, Itanium, and soon x86). (http://www.vmsadaall.org/index.php/en/)

For VAX and Alpha we have at least DEC Ada and Alsys Ada. On Itanium I have to maintain GNAT Ada on GCC. For x86 I have to base on the GNAT Ada front end for LLVM, since VSI ports VMS to x86 (https://vmssoftware.com/updates/state-of-the-port/) basing the compilers on LLVM.

I know that AdaCore dropped commercial support for GNAT Ada on OpenVMS in 2015. It's not the commercial reasons that interest me.

In approaching this project again, I would like to know as much as possible about how far AdaCore's people or helpers have come in their developments for OpenVMS, what problems they have dealt with in the GCC upgrades they have resolved, only considered, and those they have seen as too difficult and blocking. The question arises as well for the upgrades (with for example around this time the transition of the GCC build to C++) as for the evolution of the debug management.

If the answers raise confidentiality issues, I don't want to put anyone in trouble, but I'm looking for indications on who to negotiate with.

It's not impossible that AdaCore's people were among the last to develop GCC for OpenVMS Itanium. They may also be able to inform me about the build of the C and C++ part for GCC OpenVMS. I think indeed to associate to my efforts for Ada the exploration of the availability of a C++ GCC for Itanium OpenVMS.

This resumption of [this] project is quite at its beginning. My goal is to open as much as possible the work and its results to a collaborative work, in Open Source standards. One of my first tasks will be to update the current repository to allow opened development.

From: Andreas Zeurcher <ZUERCHER_Andreas@outlook.com>
Date: Mon, 17 Aug 2020 11:56:56 -0700

For those interested, a hobbyist license of OpenVMS is available from VMS Software, Inc., which is the new owners of VMS instead of HPE. There is also a free Alpha emulator for Windows 10 as well.

https://training.vmssoftware.com/hobbyist

From: nobody in particular <nobody@devnull.org>
Date: Tue, 18 Aug 2020 18:49:13 +0000

It is unlikely yet perhaps Steve Lionel will have some info on this. Although he was not involved with Ada (to my knowledge) he was a fixture in the compiler community for Fortran and probably more, at DEC, COMPAQ, and HP over a long period and might be able to identify likely suspects to contact.

This year the VMS port to Intel X86 was finally completed

https://vmssoftware.com/updates/state-of-the-port/

https://sciinc.com/remotevms/vms_techinfo/vms_news/OpenVMSOnX86-64.asp

I remember a lengthy discussion in the VMS newsgroup many years ago regarding the future of Ada on VMS. I believe the guys at the above companies were involved. I think the conclusion was

they would not or could not handle it in-house and I believe the Ada they had on VMS was only 95. There were murmurings that they would try to find somebody to do it but I did not hear that AdaCore ever released anything.

Thank you, I'll follow this thread with interest.

Ada and Other Languages

CLU and Alphard Grammars

From: Oliver Kellogg <olivermkellogg@gmail.com>
Subject: CLU and Alphard grammars available in HTML
Date: Thu, 23 Jul 2020 15:02:09 -0700
Newsgroups: comp.lang.ada

The research languages CLU and Alphard had some influence on the design of Ada [1].

The available grammar documents [2], [3] are in Postscript or PDF format, in the case of Alphard in a somewhat hard to read typeface due to being scanned from the original document.

For an HTML version of the grammars, see

http://okellogg.de/proglang/CLU-syntax.html

http://okellogg.de/proglang/alphard-collected-syntax.html

[1] Ada 83 LRM section 1.3

See e.g. http://archive.adaic.com/standards/83lrm/html/lrm-01-03.html

[2] CLU Reference Manual Appendix A

See e.g. http://okellogg.de/proglang/CLU-syntax.pdf

[3] An informal definition of Alphard

See e.g. http://okellogg.de/proglang/An_informal_definition_of_Alphard.pdf

From: "oliverm...@gmail.com" <olivermkellogg@gmail.com>
Date: Wed, 19 Aug 2020 13:27:33 -0700

Update:

Translation of the full "Informal Definition of Alphard" document to HTML is in progress, see

http://okellogg.de/proglang/an-informal-definition-of-alphard.html

100% completion ETA is within the next few weeks.

Ada Practice

Ada on Apple's New Processors Licensing Concerns

[The thread started with compiler backend concerns, but most of it evolved towards licensing issues in view of the optimizations that the Apple Store may perform to intermediate code. As is often the case with licensing arguments, no entirely satisfactory consensus was reached on the actual situation, and any conclusions in any case should be vetted by qualified experts. One possible takeaway, as Fabien Chateau summarizes in one of his posts, is that the GNAT-LLVM frontend opens many possibilities that did not exist before, which is a net positive in any case.

The complete thread can be found at https://groups.google.com/g/comp.lang.ada/c/lHQQfRATKno --arm]

From: Jerry <list_email@icloud.com>
Subject: Ada on Apple's new processors
Date: Mon, 22 Jun 2020 15:53:00 -0700
Newsgroups: comp.lang.ada

Apple is beginning its third nightmare transition to a new processor family. What does this mean for Ada on macOS?

Can we hope for a native compiler anytime soon? We will have Rosetta 2 until we don't. (Original Rosetta lasted for two OS generations and then it was taken away.) I could tell you the story of needing to run a small PowerPC program to set up a slightly old Apple WiFi device a couple years ago. Buy Parallels. Call Apple and send $30 to get Snow Leopard Server--that's 10.6. Virtualize Snow Leopard Server on Parallels to run the WiFi set-up program in Rosetta.)

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 23 Jun 2020 03:42:46 -0700

> Apple is beginning its third nightmare transition to a new processor family. What does this mean for Ada on macOS?

> Can we hope for a native compiler anytime soon?

I suppose native toolchain will be based on LLVM, thus it will allow to use GNAT LLVM on new processors.

[A large discussion is omitted at this point on the implications of GCC code generation in regard to the Runtime Library Exception (RLE) clause of GPLv3. However, as later was pointed out, GNAT LLVM does not have any relation to GCC.

Arnaud Charlet from AdaCore eventually jumped in to clarify the status of GNAT LLVM licensing, which re-sparked a somewhat more focused discussion in relation to the original topic, which follows. --arm]

From: charlet@adacore.com
Date: Thu, 25 Jun 2020 00:21:01 -0700

> The compiler links to GNAT-LLVM, the runtime doesn't.

> Pretty sure that the AdaCore people said it won't fall under GPL.

That's correct, there is no issue here. The GNAT LLVM compiler is a tool and is licensed under GPLv3, which is just fine and the proper license for a tool. The runtime which is linked with your executable comes from the gcc.gnu.org repository and contains the GCC RunTime exception license.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Thu, 25 Jun 2020 10:55:39 +0100

> That's correct, there is no issue here. [...]

Can you confirm that using FSF GNAT with GNAT-LLVM (GPLv3) does or does not enable the IR clause in the GPLv3?

From: charlet@adacore.com
Date: Thu, 25 Jun 2020 03:14:31 -0700

> Can you confirm that using FSF GNAT with GNAT-LLVM (GPLv3) does or does not enable the IR clause in the GPLv3?

It does not and in any case, invoking this clause is a red herring since as explained in the license, the concern and what's not allowed is using an intermediate representation and feed it to a proprietary (non-GPL-compatible) software to e.g. optimize it or further process it. LLVM is a GPL-compatible Software, so this is irrelevant.

From: Simon Wright <simon@pushface.org>
Date: Thu, 25 Jun 2020 12:03:14 +0100

> It does not and in any case, invoking this clause is a red herring since as explained in the license, the concern and what's not allowed is using an intermediate representation and feed it to a proprietary (non-GPL-compatible) software to e.g. optimize it or further process it.

Optikos has (at last) made clear his concerns about this: if it is indeed the case that Apple requires App Store developers to deliver bitcode for further proprietary optimizations then there might be an issue.

Depends on whether LLVM IR (which I understand is logically equivalent to bitcode) can count as target code? I've seen it described as LLVM assembler ...

From: "Luke A. Guest" <laguest@archeia.com>
Date: Thu, 25 Jun 2020 12:25:59 +0100

> [...] further proprietary optimizations [...] might be an issue.

> Depends on whether LLVM IR [...] can count as target code?

Indeed. The only thing I've found so far is this:

https://thenextweb.com/apple/2015/06/17/apples-biggest-developer-news-at-wwdc-that-nobodys-talking-about-bitcode/

Quote from near the top:

'This means that apps can automatically “take advantage of new processor capabilities we might be adding in the future, without you re-submitting to the store.”'

From the apple docs it links to at the top:

"Bitcode is an intermediate representation of a compiled program. Apps you upload to App Store Connect that contain bitcode will be compiled and linked on the App Store. Including bitcode will allow Apple to re-optimize your app binary in the future without the need to submit a new version of your app to the App Store. "

So, it looks like he [Andreas Zuercher, aka Optikos --arm] is right.

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Tue, 30 Jun 2020 04:16:38 -0700

> this later closed-source processing of the app by Apple for nonjailbroken ARM-based Macs and iDevices to distribute via the App Store seems to violate terms of at least the RLE [Runtime Library Exception] if not GPLv3 too.

The Apple app store is incompatible with the GPL since long ago: https://www.fsf.org/blogs/licensing/more-about-the-app-store-gpl-enforcement

I don't see anything new here.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Tue, 30 Jun 2020 05:28:45 -0700

>

> The Apple app store is incompatible with the GPL since long ago: https://www.fsf.org/blogs/licensing/more-about-the-app-store-gpl-enforcement

Yes, when the developer's app is GPLed, the App Store's terms and the GPL's terms are mutually incompatible. Historically, GPLing an app would have been by developer choice (unless somehow violating the RLE which was rare in practice because using garden-variety unmodified IR-unadorned GNAT, GCC, and so forth resulted in an Eligible Compilation Process in RLE).

> I don't see anything new here.

What is new here is that there appear to be well-reasoned ways (e.g., the Wide legal theory along this thread [that LLVM IR is a kind of IR code according to the RLE --arm]) that GNAT-LLVM could •force• a developer's app to [be] GPLed against the developer's will by easy-to-enact-in-GNAT-LLVM violations of the RLE's terms that cause the Compilation Process to not achieve the stricter Eligible Compilation Process definition, due to Apple's closed-source manipulations of LLVM IR bitcode.

Perhaps the work-around is that GNAT-LLVM-based developers of apps should •never• submit LLVM IR bitcode to Apple's App Store's app-intake procedure. In the past as far back as 2015, submitting bitcode instead of machine code was optional. It is unclear with the new ARM-based Macs, whether that optionality will continue in the future, or whether that optionality has already been curtailed.

(Conversely, under the Narrow legal theory along this thread [that LLVM IR is equivalent to assembly code and not an actual IR for RLE purposes --arm], your claim is correct, nothing has changed: if an app-developer doesn't want to suffer the mutual incompatibility of the GPL and Apple App Store, then don't choose GPL as the license for the app, because despite its name LLVM IR bitcode is merely assembly language which is unregulated by RLE.)

From: charlet@adacore.com
Date: Tue, 30 Jun 2020 07:35:03 -0700

> We need clarification on whether the translation from GCC's IR to LLVM's IR invokes this clause. I'm not sure if GNAT final IR before the GNAT-LLVM backend is GENERIC or GIMPLE.

GNAT LLVM doesn't use nor depend on GCC at all: it goes directly from the GNAT tree to LLVM bitcode, there is never any GENERIC nor GIMPLE in sight by design and never can be (unlike with the old DraggonEgg FWIW).

> I've had a quick look in GNAT-LLVM and I cannot see any flags enabling the output of GCC's IR, only LLVM's IR.

See above.

By the way the reason I haven't answered other messages is mainly because I am not familiar with Apple's specific constraints here, so I'd rather not make any statement about them rather than making wrong statements and you shouldn't draw any conclusion from the fact that I haven't replied to some of the messages in this thread.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Tue, 30 Jun 2020 15:46:43 +0100

> GNAT LLVM doesn't use nor depend on GCC at all: it goes directly from the GNAT

Ok, makes sense.

> tree to LLVM bitcode, there is never any GENERIC nor GIMPLE in sight by design and never can be (unlike with the old DraggonEgg FWIW).

But, GNAT is 1 part of GCC and the GPLv3 mentions IR, what constitutes the IR? Surely it covers the Ada AST IR?

Does the GPL infect across the different IR boundaries?

[...]

From: Simon Wright <simon@pushface.org>
Date: Tue, 30 Jun 2020 21:01:59 +0100

> GNAT LLVM doesn't use nor depend on GCC at all: it goes directly from the GNAT tree to LLVM bitcode, there is never any GENERIC nor GIMPLE in sight by design and never can be (unlike with the old DraggonEgg FWIW).

It seems to me that there's a lot of argument about things which can't or won't be changed.

AdaCore have produced GNAT-LLVM as a proof of concept, aimed really at targets not supported by GCC but of interest to AdaCore's customers.

GNAT-LLVM code itself is (C) AdaCore, and is GPLv3. The gcc/ada code is (C) FSF, and is GPLv3.

No change there.

The current build takes the RTS from FSF GCC, though clearly it could take it from elsewhere (e.g. some bare metal RTS).

That RTS is (C) FSF, GPLv3 + runtime exception.

Some here have thought, Aha! LLVM, RTS with runtime exception, people could produce apps for iOS!!!!

Then, cold reality strikes: it looks as though there's a conflict between the actual terms of the runtime exception and Apple's requirements for code to be submitted to the App Store (it needs to be in LLVM IR or equivalent); the code would very likely lose the protection of the runtime license umbrella.

Now, guys, given that there's Apple on one side standing on a mountain of money and a prickly attitude to what they'll accept for their app store, and on the other side a very much smaller developer community, who's going to risk going to court to put a GNAT app on to the App Store?

Whether you could make such an app and run it on iPhones privately, without going

through Apple & the App Store, I don't know. I'm sure the NSA can.

[...]

From: "Luke A. Guest" <laguest@archeia.com>
Date: Tue, 30 Jun 2020 21:20:27 +0100

>> By that point, there should be a strong track record of technical knowledge regarding Apple's bitcode submission policies to the App Store to relay to the attorneys so that they can simply turn the legal crank to make a decision/adjustments of whether/how GNAT-LLVM is to transition out of experimental status.

> I'd have thought that AdaCore's response to this idea would be to ask where you got the idea that iOS & the App Store would feature as a candidate target.

I'm developing SDLAda, there are mobile targets. I don't see why Ada shouldn't. Jesus, even COBOL can compile to mobile according to an article I read a while ago. If AdaCore and Ada users want people not thinking that Ada is an ancient language, then it needs to wake up, smell the coffee and get on mobile.

From: Wesley Pan <wesley.y.pan@gmail.com>
Date: Tue, 30 Jun 2020 15:07:05 -0700

> I'm developing SDLAda, there are mobile targets. I don't see why Ada shouldn't. [...]

I COMPLETELY agree with Luke! We need Ada to expand to things like mobile, gaming, and other "more exciting" markets to help attract the new generations of software engineers and to stay relevant in the public's eyes. The gaming industry alone rivals that of Hollywood. By the end of 2019, GTA 5 sold more than 100 million copies worldwide, earning its publisher more than $6 billion on a $265 million development budget. That's not chump change. How can members of the Ada community ever really jump into such industries if the same issues like the license keep coming up as roadblocks?!

I'm not in any way suggesting the Ada community give up its focus on the safety and reliability angle. Those are very important too. But, if you were to have affordable/free Ada tools for mobile/gaming on one side, and expensive tools for the next-gen Mars rover on the other side, which do you think would attract more end users?

BTW, even the new "cool" Rust language is being used to develop apps for mobile. Ada apps....?

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Tue, 30 Jun 2020 21:46:47 -0700

> I'd have thought that AdaCore's response to this idea would be to ask where you got the idea that iOS & the App Store would feature as a candidate target.

Gee, either

a) we all concurrently pulled it out of thin air via overactive imagination as you imply,

or

b) the following extant events & facts transpired:

At least an engineer at AdaCore (if not AdaCore speaking as an organization) wrote the following on the GNAT-LLVM repository's README.md:

“[GNAT-LLVM] is a work-in-progress research project that's not meant for and shouldn't be used for industrial purposes. It's meant to show the feasibility of generating LLVM ••bitcode•• for Ada.” (emphasis added)

LLVM.org did not organically produce bitcode out of their own volition. Bitcode was Apple's idea, Apple's design, contributed by Apple to benefit primarily Apple as a strategic technology to facilitate Apple's OS-optimization & processor-switcheroo goals without Apple begging all app developers to resubmit a plethora of minor-variation apps every time Apple has a bright idea or Big New Thing. So when GNAT-LLVM's README.md is explicitly calling out bitcode emission as the A#1 top-priority reason for GNAT-LLVM to exist, by using that very term bitcode, it is quite clear that the intended reading of README.md is referring to the Apple-Apple-Appleness of bitcode since bitcode was announced at Apple's Worldwide Developer Conference in June 2015 as a key technology related to App Store submission and downstream proprietary processing by Apple post-submission:

https://TheNextWeb.com/apple/2015/06/17/apples-biggest-developer-news-at-wwdc-that-nobodys-talking-about-bitcode

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Wed, 1 Jul 2020 02:23:29 -0700

> Gee, either

> a) we all concurrently pulled it out of thin air via overactive imagination as you imply,

> or

> b) the following extant events & facts transpired:

The answer is a).

> it is quite clear that the intended reading of README.md is referring to the Apple-Apple-Appleness of bitcode since bitcode was announced at Apple's Worldwide Developer Conference in June 2015 as a key technology related to App Store submission and downstream proprietary processing by Apple post-submission:

It seems like you focus too much on details of a simple README. LLVM bitcode is sometimes used to talk about the general LLVM IR.

The example use cases mentioned by the README are bringing more tooling to the Ada ecosystem, for instance with KLEE, or "connecting the GNAT front-end to the LLVM code generator".

It took time and effort to publish GNAT-LLVM on GitHub, and AdaCore had absolutely no obligation to do so. To be honest, I am personally a bit disappointed to see such a long discussion on what is allegedly not possible to do with GNAT-LLVM (and was absolutely not possible before anyway), rather than all the possibilities that GNAT-LLVM opens.

From: Simon Wright <simon@pushface.org>
Date: Wed, 01 Jul 2020 12:03:10 +0100

> It took time and effort to publish GNAT-LLVM on GitHub, and AdaCore had absolutely no obligation to do so. To be honest, I am personally a bit disappointed to see such a long discussion on what is allegedly not possible to do with GNAT-LLVM (and was absolutely not possible before anyway), rather than all the possibilities that GNAT-LLVM opens.

Personally, I thank AdaCore for making such an interesting project available.

A couple of postings down Jeffrey Carter quoted this, which seems apt in the current context:

"Propose to an Englishman any principle, or any instrument, however admirable, and you will observe that the whole effort of the English mind is directed to find a difficulty, a defect, or an impossibility in it. If you speak to him of a machine for peeling a potato, he will pronounce it impossible: if you peel a potato with it before his eyes, he will declare it useless, because it will not slice a pineapple."

Charles Babbage

From: Wesley Pan <wesley.y.pan@gmail.com>
Date: Thu, 2 Jul 2020 17:51:36 -0700

> It took time and effort to publish GNAT-LLVM on GitHub [...]

Hi Fabien,

That's a fair point. As with any compiler related development (and software in general), I'm sure the amount of time and effort it took to create GNAT-LLVM was significant. Aside from the licensing issue/debate, it is a really great contribution to the Ada community and I

hope it becomes production quality in the very near future.

AdaCore is the main (if not only) company that continues to make innovative and very helpful tools related to Ada (e.g. libadalang and LearnAda). As you pointed out, AdaCore was not obligated to make such contributions. GNAT-LLVM could very well have been kept in closed doors to only further AdaCore's internal development.

When news about GNAT-LLVM first came out, I for one thought it would finally allow people to create IOS apps in Ada and to further the adoption of the language. Sadly, not sure that will ever happen now...

From: gautier_niouzes@hotmail.com
Date: Fri, 3 Jul 2020 04:08:00 -0700

> It took time and effort to publish GNAT-LLVM on GitHub [...]

There is a bias here: the people discussing on comp.lang.ada tend to be busy... discussing on comp.lang.ada - and less busy doing actual programming. Chatting and programming are incompatible activities IMHO. At least you cannot do both at exactly the same time...

From: Simon Wright <simon@pushface.org>
Date: Thu, 02 Jul 2020 10:54:27 +0100

> "The LLVM code representation is designed to be used in three different forms: as an in-memory compiler IR, as an on-disk bitcode representation (suitable for fast loading by a Just-In-Time compiler), and as a human readable assembly language representation", which to me precisely matches "data in any format that is used as a compiler intermediate representation, or used for producing a compiler intermediate representation".

On thinking about this further, I can't help wondering whether this is deliberate.

From: antispam@math.uni.wroc.pl
Date: Fri, 3 Jul 2020 17:18:40 +0000

> On thinking about this further, can't help wondering whether this is deliberate.

Why doubt? FSF clearly did not want what Apple is doing now. Apple understood this well, left GCC development and started promoting LLVM. FSF lawyers formulated appropriate licencing language. So the remaining question is if they did a good job. Basically folks here are searching for a loophole. Loopholes happen, but FSF was careful, so do not bet on this.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Fri, 3 Jul 2020 11:31:33 -0700

> Why doubt? [...]

These have been my exact concurring conclusions as well for over 2 years now, when I back then ceased coding up my own variant resembling what is now known as GNAT-LLVM. Some of my design/coding work was hinted at in multiple of my postings here on c.l.a. back then. I figured out these ••chilling effects•• on my own over 2 years ago.

Question about Best Practices with Numerical Functions

From: mockturtle <framefritti@gmail.com>
Subject: Question about best practices with numerical functions
Date: Fri, 3 Jul 2020 22:30:52 -0700
Newsgroups: comp.lang.ada

I have a question about the best way to manage a potential loss of precision in a numerical function. This is a doubt that came to my mind while writing a piece of software; now I solved the specific problem, but the curiosity remains.

Let me explain.

Recently I needed to write an implementation of the Lambert W function (is the function that given y finds x such that x*exp(x)=y). This function cannot be expressed with elementary functions and the algorithm I found basically solves the equation in an iterative way. Of course, if you fix the maximum number of iterations, it can happen that the convergence is not fast enough and you obtain a result that is potentially less precise than what you would expect.

I was wondering how to manage such a non convergence case. Please note that I am supposing that I am writing a "general" function that could be used in many different programs. If the function was specific for a single program, then I would choose the line of action (e.g., ignore, log a warning or raise an exception) depending on the needs of the specific program.

(Incidentally, it turned out that the implementation converges nicely for any value of interest; nevertheless, the curiosity remains...)

I can see few line of actions that would make sense

[1] Raise an exception.

Maybe this is a bit too drastic since there are cases where a moderate loss of precision does not matter (this was my case, i just needed one or two decimal digits)

[2] Let the function have an optional "precision" parameter and raise an exception if the precision goes below that

[3] Let the function return a record with a field Value with the actual result and a field Error with the estimated precision.

This would make the code a bit heavier since instead of calling

```
X := Lambert(Y);
```

you would say

```
X := Lambert(Y).Value;
```

Not really a huge deal, however...

[4] Print a warning message to standard error or some logging system and go on.

This sounds like the worst option to me. The message could be overlooked and, moreover, it supposes there is some logging facilities or that the standard error is available for logging... Remember that the function should be general, to be used in any program.

[5] ???

Any suggestions?

Thank you in advance

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sat, 4 Jul 2020 09:50:57 +0200

> [5] ???

[5] Interval computations is the best way to handle rounding errors:

https://en.wikipedia.org/wiki/Interval_arithmetic

An Ada implementation is here:

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

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Sat, 4 Jul 2020 05:45:57 -0500

Most Fortran Lapack use INFO code.

"All documented routines have a diagnostic argument INFO that indicates the success or failure of the computation, as follows:

INFO = 0: successful termination

INFO < 0: illegal value of one or more arguments -- no computation performed

INFO > 0: failure in the course of computation"

https://www.netlib.org/lapack/lug/node138.html

So you could follow that.

[...]

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 6 Jul 2020 10:02:39 -0700

> [5] ???

> Any suggestions?

One way to do this would be to use fixed-point types:

(1) Convert your input to the fixed-point that has a "good enough" delta for the precision you want.

(2) Run the algorithm.

(3) Convert back to your normal value-type.

This assumes you're using floating-point or integers, but one nice thing about fixed-point is that it has a bounded error when dealing with operations, unlike floating-point. -- I remember some years ago seeing a bug report dealing with floating-point, where the particular error simply couldn't have happened with fixed-point.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 7 Jul 2020 00:23:06 +0200

> One way to do this would be to use fixed-point types:

Rounding error is bounded in both cases. Fixed-point has same error regardless of the values involved in the operations. Floating-point has error depending on the values.

I would say that floating-point error would be roughly the same for addition, subtraction and multiplication, provided fixed-point does not overflow. It will be hugely better for division.

Using fixed-point arithmetic has only sense for a few marginal cases of rounding.

Furthermore converting many algorithms to fixed-point might turn quite non-trivial as you will have to ensure absence of overflows and underflows. Where floating-point computation just would lose some precision, fixed-point will catastrophically fail.

General Circular Buffer

From: Daniel <danielnorberto@gmail.com>
Subject: General circular buffer example not tied to any specific type
Date: Sat, 4 Jul 2020 10:00:26 -0700
Newsgroups: comp.lang.ada

Hello, any theoretical example of buffer I can find is always tied to a specific type.

I'm looking for any example of Ravenscar buffer able to use any type of data at the same time.

I suppose it will need to serialize all data and manipulate it as a group of bytes.

Does anybody know any example of this written in Ada?

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sat, 4 Jul 2020 19:25:22 +0200

Ring buffer of indefinite elements would be OK. As an element you can use this:

```
type Item (Size : Stream_Element_Count)
is record
   Data : Stream_Element_Array (1..Size);
end record;
```

Instantiate the generic buffer with this type. Use stream attributes to serialize/deserialize.

Alternatively you can do it with Storage_Element in the above and use a fake storage pool to store/restore objects. Or a combination "for X'Address use Y" with pragma Import (Ada, X);

If the type set is somewhat statically known you can use a variant record as an element too.

In some cases you can have a ring buffer of type tags and a set of ring buffers. For each type tag you would keep values in a separate ring buffer.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Thu, 24 Sep 2020 06:39:43 +0200

> Does anybody knows any example of this written in Ada?

Hmmm, you know, Ada is a strongly typed language, therefore what you put in a buffer must have a well defined type.

There are two possibilities:

1) If you can accept several buffers, one for each type, make it generic and instantiate it as many times as you need

2) Make a buffer of Stream_Elements, and use the streaming attributes ('Read, 'Write) to turn any type into stream elements.

Ada.Streams.Stream_IO can also be handy in some cases.

From: Simon Wright <simon@pushface.org>
Date: Fri, 25 Sep 2020 15:32:01 +0100

As J-P has said, you could use 'Write and 'Read (or better, 'Output and 'Input) to write to a stream.

The beginnings of an alternative, which I last worked on a while ago, is at [1]; it's an Ada implementation of part of MessagePack[2] (boolean, integer, float, string). Still a way to go!

Writing arbitrary data to a stream using 'Write/'Output suffers from the disadvantage that the reading side won't know what to expect unless you have some protocol in place. This Message_Pack doesn't eliminate this at all.

For a while, I supported a scheme where all the data to be transmitted had to be instances of a tagged type e.g. Base; as far as I can remember, you output the data using Base'Class'Output and read it in using Base'Class'Input.

[1] https://sourceforge.net/u/simonjwright/msgpack-ada/code/ci/master/tree/

[2] https://en.wikipedia.org/wiki/MessagePack

Fixed vs Float Precision and Conversions

From: Björn Lundin <b.f.lundin@gmail.com>
Subject: Fixed vs float and precision and conversions
Date: Tue, 7 Jul 2020 23:10:20 +0200
Newsgroups: comp.lang.ada

I've for years run an interface towards external part on a raspberry pi that communicates with JSON over http (JSONRPC2)

the versions are [...]

I have found this reliable but suddenly I have got some rounding troubles. Or I perhaps just discovered it now.

I have a fixed type

```
type Fixed_Type is delta 0.001 digits 18;
```

but JSON does not support that. So I get floats instead. I use gnatcoll.json as parser by the way

[Skipped example that boils down to converting a float to a fixed point type. --arm]

The message (JSON) contains a value 5.10 (in a float), but that is converted (sometimes I think) to the fixed_type variable with value 5.099. This gets me into trouble further down in the code.

So - What should I do instead?

should I express my Fixed_Type in another way?

I need to be able to express 6.5 % (0.065) which I could not with type Fixed_Type is delta 0.01 digits 18;

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 8 Jul 2020 00:30:29 +0300

According to RM 4.6(31), conversion to a decimal fixed-point type does not round, but truncates toward zero if the operand is not a multiple of the "small" of the target type, which is usually the case here if Floats are base-two.

You should perhaps change the conversion (Target := Fixed_Type(Tmp)) to round, by doing Target := Fixed_Point'Round (Tmp).

Note, I haven't tried it.

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 8 Jul 2020 20:10:35 +0200

The main reason for asking was to see if I got the whole concept of fixed types wrong or not.

I did expect 'You should do this or that one-liner' as Niklas proposed. I did not get that to work though.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 8 Jul 2020 21:21:45 +0300

> I did expect 'You should do this or that one-liner' as Niklas proposed. I did not get that to work though

Oh. What happened when you tried? How did it fail?

From: Shark8 <onewingedshark@gmail.com>
Date: Tue, 7 Jul 2020 14:58:40 -0700

> but JSON does not support that. So I get floats instead.

This is a limitation of JSON, IIUC: all numeric are IEE754 floats -- see: https://www.json.org/json-en.html

If you have access to both sides of the serialization, you could define an intermediate serialization say a string of "Fixed_Type#value#" where 'value' is the string-representation you need. -- You can extract the value by indexing on the '#' characters, extracting the portion in between, and feeding that via Fixed_Type'Value(EXTRACTED _SUBSTRING), and produce it via "Fixed_Type#" & Fixed_Type'Image(fp_value) & '#'.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 8 Jul 2020 20:36:47 +0200

> The main reason for asking was to see if I got the whole concept of fixed types wrong or not.

Fixed-point is conceptually a scaled integer. You should deal with it accordingly. [It could be a bit surprising in Ada where conversion to integer rounds. In most languages conversion to integer truncates]

[...]

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 8 Jul 2020 21:39:20 +0200

> Oh. What happened when you tried? How did it fail?

I did a test routne like below, but realized that Float(5.10) - which was converted to Fixed_Type(5.099) is a valid fixed_type of course.

so

```
Fix1 := Fixed_Type(Flt);
```

or

```
Fix2 := Fixed_Type'Round(Flt);
```

does not really matter, since both may return 5.099 when given 5.10

[...]

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 8 Jul 2020 23:34:46 +0300

> I did a test routine like below, but realized that Float(5.10) - which was converted to Fixed_Type(5.099) is a valid fixed_type of course.

No, see below. You are confusing decimal (base-10) reals with binary (base-2) floats. [...] The point is that Float'(5.10) is not exactly 5.10, because base-2 floats cannot represent decimal fractions exactly. Since the result, as you showed in your first post, of converting (with truncation) Float'(5.10) to Fixed_Type is 5.099, the actual (binary) value of Float'(5.10) is a little less than 5.10, so the truncation gives 5.099 instead of 5.100.

But Fixed_Type'Round (Float'(5.10)) will always give 5.100.

[...]

Binary Search SPARK Proof

From: mockturtle <framefritti@gmail.com>
Subject: My new post on dev.to about SPARK
Date: Thu, 9 Jul 2020 07:16:20 -0700
Newsgroups: comp.lang.ada

first a bit of disclaimer: this is about a recent post of mine on dev.to I post this here since I think that maybe someone in this group could be interested.

Recently I wrote a small binary search procedure for a software of mine. Since I always wanted to start using SPARK, I thought that this could be a nice small problem to start playing around with SPARK. The post on dev.to is about my experience.

If you are curious

https://dev.to/pinotattari/proving-the-correctness-of-a-binary-search-procedure-with-spark-ada-34id

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Thu, 9 Jul 2020 08:27:16 -0700

I shared it on reddit: https://www.reddit.com/r/programming/comments/ho4zzp/proving_the_correctness_of_a_binary_search/

Go upvote :)

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Thu, 9 Jul 2020 18:35:39 +0200

> Recently I wrote a small binary search procedure for a software of mine.

This is good, but why did you write it from scratch? Why not start with an available, reusable binary search? Then you would have a proven, generally useful component.

This is an interesting pedagogical example, but the actual algorithm is too specialized to be of general use.

From: Simon Wright <simon@pushface.org>
Date: Thu, 09 Jul 2020 21:00:18 +0100

Interesting!

I thought to have a bit of a play with it, and I found that neither CE 2019 nor CE 2020 will prove as is, including "assertion might fail, cannot prove Bottom < Top"; but it proves just fine with

```
type Element_Type is new Integer;
```

or

```
subtype Element_Type is Integer;
```

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 10 Jul 2020 06:17:21 +0200

Hmmm... The following O(N**2) function:

```
function Is_Sorted (Table : Array_Type)
return Boolean
  is (for all L in Table'Range =>
      (for all M in Table'Range =>
        (if L > M then Table (L) > Table (M))))
  with Ghost;
```

can be changed to a O(N) function:

```
function Is_Sorted (Table : Array_Type)
return Boolean
  is (for all L in Table'First .. Table'Last -1
   => Table (L) < Table (L+1))
  with Ghost;
```

From: Paul Rubin <no.email@nospam.invalid>
Date: Thu, 09 Jul 2020 23:04:03 -0700

> Hmmm.. The following O(N**2) function: [...] can be changed to a O(N) function [...]

Should it matter? The code is never executed. It's only used as a specification for the theorem prover.

By the way, Riccardo, thanks for posting that. It was impressive to see that an executable-looking spec like that could be proved automatically with the help of just a few pragmas. I hadn't posted yet because I haven't yet had a chance to try building and playing with the program.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 10 Jul 2020 09:47:16 +0200

> Should it matter? The code is never executed.

For Spark, no (although I think that the simpler version is more understandable). But if you run it through an Ada compiler with assertions on, then it will make a difference.

One Discriminated Task per CPU

From: Olivier Henley <olivier.henley@gmail.com>
Subject: 'Number_Of_CPUs' tasks creation, with discriminants, running simultaneously.
Date: Mon, 20 Jul 2020 06:51:12 -0700
Newsgroups: comp.lang.ada

My goal is to distribute similar work across multiple tasks. Incidentally, I want those tasks to start simultaneously, each task having some 'indexed' work (discriminants), and ideally block from main until they are done with their workload.

I got this working by declaring the tasks individually. What I would like to achieve is to leverage 'System.Multiprocessors. Number_Of_CPUs' for the number of tasks created. What is the idiomatic way of achieving what I want?

I tried with an array of tasks, but the problem becomes I do not know either how to start them simultaneously with parameterization or coordinate their exit point with main.

I am lurking for the most 'clean/simple' solution possible.

You can see the actual working code fixed at 8 tasks here: https://github.com/ohenley/xph_covid19/blob/master/src/xph_covid19.adb#L280-L287

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Mon, 20 Jul 2020 19:45:54 +0200

For simple parameterization, you can use a discriminant with a default that is a function call:

```
subtype Task_ID is Integer range
   0 .. System.Multiprocessors.
   Number_Of_CPUs;
subtype Valid_Task_ID is Task_ID range
   1 .. Task_ID'Last;
Last : Task_ID := 0;
function Next_ID return Valid_Task_ID is
begin
   Last := Last + 1;
   return Last;
end Next_ID;
task type T (ID : Valid_Task_ID := Next);
type T_Set is array (Valid_Task_ID) of T;
Worker : T_Set;
```

Each Worker (i) will have its ID determined during elaboration, and they will all start at the "begin" that follows the declaration of Worker. The order of the discriminants is arbitrary; there is no guarantee that Worker (I) will have ID of I. Elaboration is sequential, so each Worker will have a unique ID.

(I think Ada 2X will allow a way to insure that the ID equals the index, but I'm not sure how it will work.)

To block a subprogram until the Worker tasks all complete, declare them in a block statement:

```
Create_Workers : declare
   Worker : T_Set;
begin
   null;
end Create_Workers;
```

From: Olivier Henley <olivier.henley@gmail.com>
Date: Mon, 20 Jul 2020 13:31:45 -0700

> The order of the discriminants is arbitrary; there is no guarantee that Worker (I) will have ID of I. Elaboration is sequential, so each Worker will have a unique ID.

As long as all the tasks get a unique ID, I am fine.

Function as a discriminant at elaboration ... should have thought about it but it looks like I am missing some wisdom points.

Thank you Jeffrey.

From: onox <denkpadje@gmail.com>
Date: Wed, 22 Jul 2020 12:05:01 -0700

Another way is to use an extended return:

spec:

```
   type Worker;

   task type Worker_Task (Data : not null
   access constant Worker);

   type Worker is limited record
      ID : Positive;
      T  : Worker_Task (Worker'Access);
   end record;

   type Worker_Array is array (Positive
   range <>) of Worker;

   function Make_Workers return
   Worker_Array;
   body:
   function Make_Workers return
   Worker_Array is
   begin
      return Result : Worker_Array
        (1 .. Positive (Count)) do
        for Index in Result'Range loop
            Result (Index).ID := Index;
        end loop;
      end return;
   end Make_Workers;

   Workers : constant Worker_Array :=
      Make_Workers;
   pragma Unreferenced (Workers);
```

Two Ada 2012 Vendors

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Subject: Two Ada-12 Vendors
Date: Thu, 23 Jul 2020 14:13:35 +0200
Newsgroups: comp.lang.ada

It appears that PTC ObjectAda 10.x is an Ada-12 compiler, making two vendors with Ada-12 compilers*. Only took 8 years.

https://www.ptc.com/-/media/Files/PDFs/Developer-Tools/PTC-ObjectAda-64-for-Windows-10_1_RB.pdf

*An Ada-12 compiler implements at least the entire core language of ISO/IEC 8652:2012. Please don't clutter this thread with posts about compilers that don't meet this definition.

From: Dirk Craynest <dirk@orka.cs.kuleuven.be>
Date: Thu, 23 Jul 2020 17:24:39 -0000

>https://www.ptc.com/-/media/Files/PDFs/Developer-Tools/PTC-ObjectAda-64-for-Windows-10_1_RB.pdf

The above PDF is an announcement from May 27, 2019, and mentioned that the "release expands the support for Ada 2012 language features to include the complete set of Ada 2012 container packages and support for the associated Ada 2012 language constructs required by those packages". Hence a partial Ada 2012 implementation.

But perhaps even more interesting, PTC announced yesterday, July 22, 2020, the release of PTC ObjectAda for Windows Version 10.2: https://developer-tools-us.ptc.com/Announcements/Products/ObjectAda/1000/10.2/RB-20200722-ObjectAda%20for%20Windows%20V10.2.pdf

And the subtitle of that announcement reads: "New native Ada compiler release provides complete Ada 2012 language support".

A small extract from the text:

<start_quote>

"ObjectAda for Windows version 10.2 represents the completion of the phased implementation strategy PTC adopted for Ada 2012 language feature support within the ObjectAda technology.", stated Shawn Fanning, Software Development Director at PTC. "With ObjectAda for Windows version 10.2, the ObjectAda compiler conforms to the Ada Conformity Assessment Test Suite (ACATS) version 4.1Q and adds several new features including support for storage subpools and the Default_Storage_Pool pragma, execution time enforcement of type invariants, and complete support for new Ada expression forms.

<end_quote>

For more information, see the PDF at the 2nd URL above.

Dirk

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

Survey on the Future of GNAT Community Edition

From: Wesley Pan <wesley.y.pan@gmail.com>
Subject: AdaCore's survey regarding the future of GNAT Community Edition
Date: Fri, 24 Jul 2020 14:42:16 -0700
Newsgroups: comp.lang.ada

I just discovered this via Ada Planet...

Link to the Google Docs survey: https://docs.google.com/forms/d/e/1FAIpQLSet9x3UNUFmfWt5v-8Jb7dW8BgKiJxyEMJ_TFm0G2UJKx5OmQ/viewform

Reddit discussion: https://www.reddit.com/r/ada/comments/hwgbwa/survey_on_the_future_of_gnat_community/

Survey summary reproduced below...

GNAT Ecosystem Community Survey

Hello Ada supporters,

We are writing this message here to present, discuss and get feedback on a plan that we at AdaCore want to put in place. Over the next couple of years, we want to experiment with an evolution of the GNAT ecosystem and would like your help.

So far, there are three grand families of GNAT releases:

- GNAT Pro: An AdaCore release with professional support and high level quality assurance. Available on many different targets (PowerPC, Leon, vxWorks, etc.).
- GNAT Community: An AdaCore release with a lower level of quality assurance, less targets, and a pure GPL license for the run-time.
- GNAT FSF: community built compiler from the FSF source tree. Available from Linux distributions or Msys2 on Windows, for instance.

Moving forward, we are looking to simplify the situation and remove GNAT Community from the picture.

The plan is to reach a point where AdaCore would not release GNAT Community compilers and instead instruct non-professional users to use GNAT FSF builds. We would still keep making GNAT Studio and SPARK releases, and libraries such as AWS and xmlada will be available in the Alire package manager (http://alire.ada.dev). With this plan we also want to invest some more time to help the maintainers of GNAT packages in Linux, BSD, or Windows (msys2) distributions, for instance, and potentially contribute when necessary. Our intention is to contribute to various communities building GNAT packages so that what can be done today with GNAT Community will be doable tomorrow from these community-led builds.

Why are we working on this plan?

We have noticed that GNAT Community's pure GPL license on the run-time is seen as a barrier to new Ada users. More specifically, understanding the consequences of the GPL licence is complex. The result is that newcomers will often be introduced to Ada/SPARK by a legal licence discussion rather than looking at the value of the technology. This will, understandably, scare people off.

On top of this, we are witnessing a widespread misunderstanding around the openness of the Ada language and the GNAT compiler, some people seem to think that Ada and GNAT are proprietary technologies. We see this phenomenon as detrimental to the growth of the Ada community. Of course this misunderstanding will not fade in a couple days, but we think that removing GNAT Community will make the situation clearer and will allow us to better communicate on the situation of the Ada compiler ecosystem.

Besides general comments and discussion around this plan, we would appreciate your feedback in this survey form. Please help us spread the word. The more feedback we get, the more we will be able to move in the right direction.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 24 Jul 2020 19:17:30 -0700

Interesting that they did _not_ post about this survey in this newsgroup. I guess we're just _so_ yesterday ...

From: Stéphane Rivière <stef@genesix.fr>
Date: Sat, 25 Jul 2020 12:35:39 +0200

Many thanks Wesley!!!

Fascinating... (C) Spock

Finally, we're back to the previous situation without the GPL barrier...

GNAT 3.15p was born again ;)

All the arguments given are exactly those I addressed to AdaCore at the highest level at the time (may be about 15 years ago?)...

Originally, GNAT was funded precisely to be available to everyone, including commercial use.

I answered their survey, in a constructive way, of course.

Very good news.

From: Stéphane Rivière <stef@genesix.fr>
Date: Sat, 25 Jul 2020 12:35:43 +0200

> Interesting that they did _not_ post about this survey in this newsgroup. I guess we're just _so_ yesterday ...

I deeply agree! NGs are so (too?) efficient.

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Mon, 27 Jul 2020 01:36:37 -0700

> Interesting that they did _not_ post about this survey in this newsgroup. I guess we're just _so_ yesterday ...

I was going to do it this week ;)

From: DrPi <314@drpi.fr>
Date: Mon, 27 Jul 2020 10:07:48 +0200

What about Ada for microcontrollers?

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Mon, 27 Jul 2020 01:38:35 -0700

> What about Ada for microcontrollers?

With this plan arm-elf and riscv-elf toolchain will be available for Linux and Windows at least.

I am doing a lot of microcontroller programming myself so don't worry about that :)

From: foo wong <crap@spellingbeewinnars.org>
Date: Wed, 29 Jul 2020 02:44:08 -0700

[...]

I just wanted to say that I am very happy to read this thread.

I have written several disparaging posts about AdaCore and my take on the situation was:

- GNAT Pro: professional support, more targets.
- GNAT Community: Lower level of quality assurance, less targets, and a pure GPL license for the run-time for a demoware experience
- GNAT FSF: least quality assurance designed to push users to the community build or Pro ASAP

I hope I was wrong all along and either way, the future just got a little brighter as AdaCore's two offerings will now be suitable for free or non-free software.

I don't believe that there is anything illegal about re-distributing GNAT Pro so if the gap in quality was so large between Pro and FSF, I think one paying customer might take pity on us eventually and release Pro to the world and reset the gap for a while.

With dark days setting in for the avionics industry (for a while at least), maybe AdaCore will eventually reconsider and will release Pro as their FSF offering to broaden their user base.

From: Simon Wright <simon@pushface.org>
Date: Wed, 29 Jul 2020 17:44:30 +0100

> I have written several disparaging posts about AdaCore and my take on the situation was: [...]

IMO you've been wrong all along, at least so far as the quality is concerned. Assurance, yes; number of targets, yes; support, yes.

> [...] I think one paying customer might take pity on us eventually and release Pro to the world and reset the gap for a while.

I wouldn't have done this; but we were stuck on an old release for a long time, so wouldn't have helped.

> With dark days setting in for the avionics industry (for a while at least), maybe AdaCore will eventually reconsider and will release Pro as their FSF offering to broaden their user base.

The only difference between the pro compiler and FSF is a few months.

From: Kevin K <kevink4@gmail.com>
Date: Sat, 15 Aug 2020 09:38:23 -0700

I hadn't seen this before today. The main impediment to me with the free version compared to the community edition is that with the community edition AdaCore took a set of packages that build together correctly. I tried to build a set of components from the free version (gcc 8.2 and later), gprbuild, etc. At the time, the other important components didn't all build successfully. Some components were ahead of others. So I wasn't able to build, for example, gps.

From: Roger Mc <rogermcm2@gmail.com>
Date: Sat, 15 Aug 2020 19:24:06 -0700

> I hadn't seen this before today. The main impediment to me with the free version compared to the community edition is that with the community edition AdaCore took a set of packages that build together correctly. [...]

Unfortunately, the AdaCore community 2020 edition doesn't include gps so I am currently using the community 2020 tool-chain and gps from community 2019. I did try to build gps from the current AdaCore community source but was unsuccessful. The main problem being that AdaCore seems to be in the midst of doing the necessary upgrade from Python 2 to Python3. I did attempt to do Python3 modifications myself but eventually got to a stage where I could proceed no further.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Sun, 16 Aug 2020 03:42:11 +0100

> I hadn't seen this before today. The main impediment to me with the free version compared to the community edition is that with the community edition AdaCore took a set of packages that build together correctly. [...]

You'll have that issue on FSF because AdaCore don't tag for FSF releases like they should.

Having CE available is a massive mistake, one which they are realising far too late, imo.

From: Simon Wright <simon@pushface.org>
Date: Sun, 16 Aug 2020 11:08:18 +0100

> Unfortunately, the AdaCore community 2020 edition doesn't include gps

I _think_ this is on macOS? i.e. Linux, Windows include it? (presumably under its new name GNATstudio (modulo capitalisation))

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sun, 16 Aug 2020 05:08:13 -0700

> > Unfortunately, the AdaCore community 2020 edition doesn't include gps

> I _think_ this is on macOS? i.e. Linux, Windows include it? (presumably under its new name GNATstudio (modulo capitalisation))

Windows has <gnat>/bin/gnatstudio.exe

From: Roger Mc <rogermcm2@gmail.com>
Date: Sun, 16 Aug 2020 05:54:29 -0700

> I _think_ this is on macOS? i.e. Linux, Windows include it? (presumably under its new name GNATstudio (modulo capitalisation))

Yes. My system is macOS.

The source for GPS doesn't appear available from the AdaCore community version for any platform?

The source that I tried to build from was obtained from GIT.

I can only find GNATstudio under Ada core pro. Is it available elsewhere?

From: Stéphane Rivière <stef@genesix.fr>
Date: Mon, 17 Aug 2020 10:51:34 +0200

> You'll have that issue on FSF because AdaCore don't tag for FSF releases like they should.

> Having CE available is a massive mistake, one which they are realising far too late, imo.

Fully agree.

For the records, GVD (Gnu Visual Debugger) was buildable (under Windows[1] or Linux) but I _never_ succeeded to build GPS...

[1] For the now deprecated AIDE https://stef.genesix.org/aide/aide.html

From: Simon Wright <simon@pushface.org>
Date: Wed, 19 Aug 2020 15:29:00 +0100

> I did try to build gps from the current AdaCore community source but was unsuccessful. The main problem being that AdaCore seem to be in the midst of doing the necessary upgrade from Python 2 to Python3. I did attempt to do Python3 modifications myself but eventually got to a stage where I could proceed no further

I've reached the same stage. I can manage some of the 2-to-3 fixes (not the one in gobject-introspection, though), but the real problem for me is that there isn't a consistent complete set of sources, and some aren't provided on the AdaCore community site (e.g. pygobject, langkit, libadalang, libadalang-tools, ada_language_server). And, so far as I can see, langkit (20.2) isn't consistent with libadalang (20.2). And, my Python venv has got screwed.

Netflix & Twitter.

From: Andreas Zeurcher <ZUERCHER_Andreas@outlook.com>
Date: Wed, 19 Aug 2020 11:09:35 -0700

> I've reached the same stage. [...]

If multiple well-skilled people cannot build a GPL-licensed source code with the source code as provided and instructions as provided, wouldn't that be a black-&-white flagrant violation of the GPL? The natural conclusion seems to be: either the source code provided mismatched or the narrative instructions to build were omitting some secret-sauce, either of which was an unintentional or intentional preventative of success. The unintentionality versus intentionality would be able to be determined only after the fact by observing the root-cause of the preventative of successful building once that root cause is discovered/reported. This irreproducibility is both notable and highly interesting.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 19 Aug 2020 21:11:31 +0200

> If multiple well-skilled people cannot build a GPL-licensed source code with the source code as provided and instructions as provided, wouldn't that be a black-&-white flagrant violation of the GPL?

What else you expect when GTK and Python are used? GTK is practically impossible to bootstrap. Python is full of bugs and incompatibilities. On top of that for some mysterious reason AdaCore decided to use config scripts. No wonder it is a nightmare anywhere outside Linux.

If you think that commercial code delivered in sources is any better, you are wrong. Building from sources working out of the box is a rare exception. Most vendors simply check out the code from the repository and send it to you. They have no resources or desire to supply you with a working toolchain tailored for your

targets, nor have they necessary knowledge anyway.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Wed, 19 Aug 2020 21:21:20 +0100

> [...] No wonder it is a nightmare anywhere outside Linux.

I build on Linux and see my previous comment.

From: Simon Wright <simon@pushface.org>
Date: Wed, 19 Aug 2020 21:17:43 +0100

> [...] This irreproducibility is both notable and highly interesting.

Well - I have every sympathy with people who make a binary release and then move on in staggered stages, aiming for another binary release in a year's time. The sort of problem you encounter is that the the version of gobject-introspection on the CE site won't compile with Python 3.8.5, because of the removal of the DL_EXPORT macro that was deprecated with Python 2.3; while libadalang _requires_ Python 3.8.5. The latest glib uses Yet Another Build Tool (meson/ninja), and the script doesn't export a header required by gtk-3.14+ ... it's not so much DLL Hell as a version compatibility tightrope.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Wed, 19 Aug 2020 21:19:50 +0100

> If multiple well-skilled people cannot build a GPL-licensed source code with the source code as provided and instructions as provided [...]

Every component from AdaCore is an absolute fucker to build.

From: Simon Wright <simon@pushface.org>
Date: Wed, 19 Aug 2020 21:50:29 +0100

> Every component from AdaCore is an absolute fucker to build.

The "front-line" components (gnatcoll*, gprbuild, xmlada & friends) build pretty reliably for me, even taking the master branch.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Wed, 19 Aug 2020 22:35:07 +0100

> The "front-line" components (gnatcoll*, gprbuild, xmlada & friends) build pretty reliably for me, even taking the master branch.

That's only after you work out which commit to build, after many attempts at building.

From: <steve@cunningsystems.com>
Date: Wed, 19 Aug 2020 23:12:07 -0700

> I think that I also found this. Sometimes, the master worked, occasionally the stable (?) worked but I'd sometimes (often?) have to resort to an earlier build to achieve success!

Note that the libadalang/master and langkit/master repositories are often out ahead of gps/master. You'll have better luck if you build them from stable branches, then gps/master should build.

Similarly spark2014/master is often ahead of FSF gcc/master. gcc/master is usually in sync with spark2014/fsf branch.

From: Roger Mc <rogermcm2@gmail.com>
Date: Wed, 19 Aug 2020 22:42:24 -0700

> [...] the real problem for me is that there isn't a consistent complete set of sources [...]

I think I managed to do all the Python 3 conversions but couldn't get linking to work.

I recall that getting the prerequisites built was a challenge and found that if I changed anything in any of them I'd have to go back and start building them from scratch again. Worse, I seem to recall that for at least one of them, probably langkit, I'd have to delete it and reload it from its archive file.

It's comforting to find that many of the opinions expressed in this thread are similar to my own.

[Around this point, the thread veers off towards Python specifics, although with a relation to Ada features. --arm]

From: Roger Mc <rogermcm2@gmail.com>
Date: Wed, 19 Aug 2020 22:48:34 -0700

> What else you expect when GTK and Python are used? [...]

Incredibly, Python seems to be the language of choice for "teaching" the now-defunct discipline of software development, even by leading universities as far as I can discover.

Most of the rules of disciplined software development seem to have been discarded long ago. In particular, the maintainability aspect seems to have disappeared.

From: Stéphane Rivière <stef@genesix.fr>
Date: Thu, 20 Aug 2020 08:43:13 +0200

> Incredibly, Python seems to be the language of choice for "teaching" the now-defunct discipline of software development, even by leading universities as far as I can discover.

It's not incredible. It's led to 737max failure, It's Idiocracy.

RM about Idiocracy could be the movie Idiocracy. I urge you to see it. Its nickname is: the movie which has become a documentary. It's delightfully vulgar but above all incredibly relevant and funny.

https://en.wikipedia.org/wiki/Idiocracy

In any case, AdaCore's iechoice of python is miserable. It would have been wiser, if there was a need for a scripting language, to implement a subset of Ada (like Gautier de Montmollin HAL :) For the doc, they even abandoned GNU/Texinfo for Python...

Moreover, the GPS code has always been problematic. I remember the first versions where a very large portion of the code was in C because they had integrated a full version of an old version of berkeley DB...

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Mon, 31 Aug 2020 07:54:43 -0700

> In any case, AdaCore's choice of python is miserable. It would have been

> wiser, if there was a need for a scripting language, to implement a

> subset of Ada (like Gautier de Montmollin HAL :) For the doc, they even

> abandoned GNU/Texinfo for Python...

I agree.

And they also used Python for libadalang: "libadalang is using the Langkit framework as a basis, and is at the time of writing the main project developed using it. The language specification, while embedded in Python syntax, is mostly its own language, the Langkit DSL, that is used to specify the part of Ada syntax and semantics that are of interest to us."

I wonder if it would have been possible to create a library of objects directly in Ada, somehow equivalent in features to Python's Objects, but with the advantage of strong typing and a compilation to native instructions. It would require a major use of interfaces, one for each Python's built-in type class, but it would have been a foundation for many other applications.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 1 Sep 2020 11:22:04 -0700

> And they also used Python for libadalang [...]

> I wonder if it would have been possible to create a library of objects directly in Ada, somehow equivalent in features to Python's Objects, but with the advantage of strong typing [...]

Langkit uses the advanced features of Python to create a Domain Specific Language (DSL) for defining Abstract Syntax Trees. The DSL also defines much of the user API for accessing the syntax tree after parsing. You could accomplish something similar by using a grammar generator (WisiToken or Langkit :) to create a parser for the desired DSL (as WisiToken does), but then defining the API would be done separately, and the correspondence between the API and the

syntax tree maintained manually (ie error-prone). I think Python is a good choice for this application - there are probably other languages with similar features that could have been used, but Ada is not one.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 1 Sep 2020 20:59:52 +0200

> Langkit uses the advanced features of Python to create a Domain Specific Language (DSL) for defining Abstract Syntax Trees.

Which is a big mistake, as well. Each intermediate is yet another point of error.

> I think Python is a good choice for this application - there are probably other languages with similar features that could have been used, but Ada is not one.

I doubt there could exist applications for languages like Python. Anyway, GPS is demonstratively not.

I also do not believe in heavily scripted IDEs. I certainly do not want GPS becoming Emacs. Any usability GPS has, comes from not being Emacs, or, for that matter, Visual Studio with its horrific VB scripts.

Proposal: Auto-allocation of Indefinite Objects

From: Yannick Moy <moy@adacore.com>
Subject: Re: Proposal: Auto-allocation of Indefinite Objects
Date: Mon, 27 Jul 2020 00:47:30 -0700
Newsgroups: comp.lang.ada

[This thread continues from AUJ 41.2, on the topic of having a mechanism to transparently allocate indefinite objects as if they were definite, e.g., as record members. --arm]

Hi Stephen,

> My proposal is that it should (sometimes?) be possible to declare objects of indefinite types such as String and have the compiler automatically declare the space for them without the programmer having to resort to access types.

I agree with the goal.

> Benefits:

>

> 1. Easier, especially for newbies/students.

> 2. Safer due to reduced use of access types.

> 3. Remove the need to have definite and indefinite versions of generic units.

I agree with 2 only if we can combine this with safe handling of aliasing. It would be terrible to have such a feature lead to unsafe code if you somehow copy the pointer. Also, for strings that's possibly not the only change needed. What you'd like really is to be able to reassign the string to some larger/smaller string, like you do when using Unbounded_String.

On 2020-04-04, Jeffrey R. Carter wrote:

> the ARG is aware of them and has chosen to take no action. That seems unlikely to change.

On the other hand, AdaCore has launched a project to collect/discuss ideas/suggestions/problems regarding the evolution of Ada and SPARK: https://github.com/AdaCore/ada-spark-rfcs

Feel free to open an Issue there on that topic.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Mon, 27 Jul 2020 11:21:16 +0200

> I agree with the goal.

You have it already. It's called Unbounded_String.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 27 Jul 2020 11:49:28 +0200

> You have it already. It's called Unbounded_String.

Not really.

1. Unbounded_String is a compromise needed when the string length changes during its life. The great majority of cases allocate [and initialize] a string just once. [addressed to be the cases when using a discriminant does not work]

2. There is nothing for arrays that are not strings and for other indefinite types. E.g.:

```
type Node_Type is record
   Item : new Element_Type'Class;
   Prev : Node_Ptr_Type;
   Next : Node_Ptr_Type;
end record;
```

3. There is nothing for serialization and marshaling objects logically containing strings and other indefinite types.

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Mon, 27 Jul 2020 17:48:40 -0000

> My proposal is that it should (sometimes?) be possible to declare objects of indefinite types such as String and have the compiler automatically declare the space for them [...]

In one sense we already have this ... in that we can do this in a Declare block, where stack allocation is a practical implementation.

But what about cases where (for whatever reason) we want it allocated on the heap?

In another sense we have it as JP Rosen said, for the specific example Unbounded_String.

Is there any way we could generalise the (storage, access and lifetime aspects of) Unbounded_String for unconstrained arrays and discriminated records in such a way that Unbounded_String can be a simple instantiation of one of these?

But without the full flexibility (or overhead) of controlled types. So, somewhere in between, as:

1. Controlled type
+ 2. Unconstrained Array or
  + Discriminated Record
  + 3. Unbounded String (instance of 2)

2) can be implemented internally using pointers, but externally appears to be a data object, just like Unbounded_String does, with similar semantics.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 27 Jul 2020 22:02:57 +0200

> 2) can be implemented internally using pointers, but externally appears to be a data object, just like Unbounded_String does, with similar semantics.

No, the point is that Unbounded_String is exactly opposite to what is required. In no case it should appear as an object of a different type!

Compare access to string P with unbounded string U:

```
for I in P'Range loop -- This is OK
   P(J) := 'a' -- This is OK
```

Now would you do:

```
   To_String (U) (J) := 'a' -- Garbage!
```

What if the original object must be a class-wide object, task, protected object, limited object etc?

Ada's access types delegate all operations to the target object, except assignment. This is the key property that the proposal in my view must retain.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Mon, 27 Jul 2020 22:31:34 +0200

> Is there any way we could generalise the (storage, access and lifetime aspects of) Unbounded_String [...]

Ada.Strings.Unbounded can be considered a combination of Ada.Containers.Indefinite_Holders instantiated for String and Ada.Containers.Vectors instantiated with Positive and Character, with some additional operations added.

The To_String and To_Unbounded_String operations of Unbounded_String are similar to the Element and Replace_Element operations of Holder, which do not exist for Vector.

The indexed operations of Unbounded_String are similar to the indexed operations of Vector, which do not exist for Holder.

If Ada.Containers.Vectors had an additional generic formal type

```
type Fixed is array (Index_Type range
  <>) of Element_Type;
```

and 2 new operations

```
function To_Fixed (From : Vector)
  return Fixed;
function To_Vector (From : Fixed)
  return Vector;
```

then we wouldn't need Ada.Strings.Unbounded.

From: Brian Drummond
<brian@shapes.demon.co.uk>
Date: Tue, 28 Jul 2020 14:28:53 -0000

> No, the point is that Unbounded_String is exactly opposite to what is required. In no case it should appear as an object of a different type! [...]

```
>     To_String (U) (J) := 'a' -- Garbage!
```

That wasn't the aspect of Unbounded I was getting at. I agree ... garbage.

What I meant was that Unbounded doesn't load New, dereferencing, deallocation etc onto the programmer, but hides the access details, and our indefinite type should do the same (the compiler can probably to a better job than the programmer anyway).

I'm suggesting something more like the C++ reference, signalling (perhaps by adding a reserved word "indefinite") that fixed size allocation won't work; and implementation is more in line with a controlled type but with system-provided Initialise,Adjust,Finalize providing the required operations (no need for the programmer to provide them).

```
A : String := "hello" -- a definite string
P : access String := new String'("hello");
Q : indefinite String := "hello";
...
  begin
    for I in P'Range loop -- This is OK
      P(J) := 'a'; -- This is OK
      Q(J) := 'a'; -- also OK. But index out of
           --range would raiseConstraint Error

...

    Q := "hello_world"; -- deallocates,
    -- allocates with new bounds

...

  end;    -- deallocate Q here.
```

It follows that "indefinite" cannot also be "aliased" unless we want to implement smart pointers. For simplicity I'd suggest disallowing "aliased indefinite" on the grounds that "access" can (should) be used instead.

Records (including tagged, class wide, discriminated) should work the same, but probably with shallow copy on assignment if they contain access types.

If there is no re-allocation (no different size assignment) the compiler is free to substitute direct (stack) storage instead of heap allocation and implicit access types. So for example instead of

```
    A : constant String := "done";
 ...
  loop
    declare
      P : String := Get_Line;
    begin
        exit when P = A;
    end;
  end loop;
```

```
  A : constant String := "done";
  Q : indefinite String;
 ...
  loop
    Q := Get_Line;
    exit when Q = A;
  end loop;
```

the implementation can be either an implicit declare block or an implicit access type. However, where Q has several reassignments within a block, and the compiler can't determine the size, an implicit access type must be used. (If it can, it can warn that "indefinite " is unnecessary).

> What if the original object must be a class-wide object, task, protected object, limited object etc?

> Ada's access types delegate all operations to the target object, except assignment. This is the key property that the proposal in my view must retain.

Indefinite can also be applied to records (discriminated, class wide, etc) here the size is indeterminate and may vary on reassignment. Assignment would always be shallow copy (where the record contained access types).

From: "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de>
Date: Tue, 28 Jul 2020 16:59:09 +0200

> I'm suggesting something more like the C++ reference, signalling (perhaps by adding a reserved word "indefinite") that fixed size allocation won't work;

Equivalent of C++ reference in Ada is renaming.

```
> Q : indefinite String := "hello";
```

I think the keyword is misleading. Maybe this:

```
  Q : new String := "hello";
```

And I don't like initialization. It was a mistake to have limited return. The syntax must stress that all initialization is strictly in-place. No copies involved because the pool is fixed.> ...

```
>     begin
>         Q := "hello_world"; --
    deallocates, allocates with new bounds
> ...
>     end;    -- deallocate Q here.
```

The rule could be "same pool" as of the container. In the case of a block, the pool is the stack. In the case of a record member, the pool is the pool of where the record itself is allocated. So that you could allocate all [the full] object in the same pool.

> It follows that "indefinite" cannot also be "aliased" unless we want to

> implement smart pointers. For simplicity I'd suggest disallowing "aliased

> indefinite" on the grounds that "access" can (should) be used instead.

It makes sense, but there are use cases for having it aliased:

```
X : indefinite T;
Y : indefinite S (X'Access);
     -- Access discriminant
```

[...]

> Assignment would always be shallow copy (where the record contained access types).

That would be inconsistent. IMO, it should be a deep copy, provided such a component would not make the type limited, of which I am not sure.

From: Brian Drummond
<brian@shapes.demon.co.uk>
Date: Wed, 29 Jul 2020 15:33:33 -0000

> Equivalent of C++ reference in Ada is renaming.

OK. Not quite sure how complete the correspondence between reference and renaming is, but I can see similarities.

[...]

> The rule could be "same pool" as of the container. In the case of a block, the pool is the stack. In the case of a record member, the pool is the pool of where the record itself is allocated. So that you could allocate all object in the same pool.

Looks like a good rule. Saves the compiler having to plant deallocations if the whole pool is to be de-allocated.

[...]

>> Assignment would always be shallow copy (where the record contained access types).

> That would be inconsistent. IMO, it should be a deep copy, provided such a component would not make the type limited, of which I am not sure.

Honest question: Inconsistent with what? I suggested shallow copy just for simplicity, and for no (ahh) deeper reason.

But again, I'm probably missing something.

Thank you for your thoughts. I don't know if this is worth developing into an AI.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 29 Jul 2020 18:20:24 +0200

[...]

In general, there are two close but not equivalent objectives: one is handling indefinite components of records; another is a transparent holder object integrated into the language (without generic mess).

Your use case is about the latter. My is rather the former.

I doubt it is possible to unite both objectives in a single AI.

On 29/07/2020 17:33, Brian Drummond wrote:

> I suggested shallow copy just for simplicity, and for no (ahh) deeper reason. But again, I'm probably missing something.

If you make a shallow copy of

```
type Node_Type is record
   Item : new Element_Type;
   Prev : Node_Ptr_Type;
   Next : Node_Ptr_Type;
end record;
```

you create a dangling pointer should the original node disappear. A deep copy would create a new target for new Item.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 30 Jul 2020 20:28:32 +0200

[...]

> The compiler should be able to determine if [...] the use of Q (the indefinite type) is equivalent to a Declare block (i.e. can be on the stack; new stack frame in each iteration; no relocation ever required) or not.

I don't want the compiler deciding where Q is allocated, especially because this could break things:

1. Large object moved to the stack
2. Lock-free code starting using heap lock when moved from the stack.

The mechanism should be transparent. I do not like Unbounded_String for many reasons. Fiddling with the heap is one of them. I do not know which heuristic it uses to reduce reallocation and how much extra memory it takes under which circumstances.

[...]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sun, 9 Aug 2020 19:31:06 -0500

> I don't like compiler relocating objects. If the pool is a stack (or heap organized as a stack) it might be unable to do this.

This is not that hard to deal with. Janus/Ada handles discriminant-dependent components of mutable objects this way: they are allocated on the stack, but if they have to be reallocated they move to the heap.

I note that the original idea already exists for discriminant-dependent components -- that's a bit more painful to use but hardly difficult. The main issue is that most compilers fail to support these components properly, using some sort of max-size implementation unconditionally rather than switching to a pool-based implementation when the max size is too large. I've never understood why Ada compilers were allowed to make such a limitation (it becomes a major limitation when working on non-embedded programs), while similar limitations on case statements and aggregates are not allowed.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sun, 9 Aug 2020 19:39:31 -0500

> I don't want the compiler deciding where Q is allocated, especially because this could break things:

> 1. Large object moved to the stack

The compiler is buggy IMHO if this breaks something. Any compiler has to be able to deal with objects that exceed the maximum stack frame, and move those to somewhere that they will fit (or reject completely).

Yes, most compilers are buggy this way (including mine in a few cases). So what?

> 2. Lock-free code starting using heap lock when moved from the stack.

Expecting a compiler not to use the heap is silly in any case (outside of the No_Heap restriction - use that in Janus/Ada and the compiler refuses to do anything outside of elementary types). The compiler is supposed to be making the programmer's life easier, not adding new hurdles.

> I do not know which heuristic it uses to reduce reallocation and how much extra memory it takes under which circumstances.

That's the idea of such mechanisms. If you really need control, you do not use these abstractions and instead write the stuff yourself explicitly using access types and the like.

Otherwise, you use containers and unbounded strings, and they do what they do. There's no free lunch. But the need to be explicit should be very rare - the main problem is programmers with insufficient trust that a compiler will do the right thing.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 10 Aug 2020 10:57:44 +0200

> That's the idea of such mechanisms. If you really need control, you do not use these abstractions and instead write the stuff yourself explicitly using access types and the like.

Right, that is my take on the proposal. If I am ready to compromise on #1 and #2, I can use an abstraction hiding pool access. Otherwise I want a language construct being more safe than raw access types.

> Otherwise, you use containers and unbounded strings, and they do what they do.

No, from the abstraction point of view they do not. They indeed abstract the memory allocation aspect, but they do that at the cost of *everything* else. Unbounded_String is no string anymore. Container is neither array nor record type. Unbounded_String must be converted forth and back. For containers I must use ugly hacks like iterators to make them resemble arrays and records introducing whole levels of complexity to fight through every time the compiler or I miss something.

In most cases I prefer to keep a clear array or record interface at the expense of manual memory management.

> There's no free lunch.

I think with a better type system there could be a whole banquet. (:-))

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 10 Aug 2020 10:58:20 +0200

> Janus/Ada handles discriminant-dependent components of mutable objects this way: they are allocated on the stack, but if they have to be reallocated they move to the heap.

What do you do if such an object is allocated via pool-specific access type?

[...]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 19 Aug 2020 19:10:27 -0500

> I think with a better type system there could be a whole banquet. (:-))

Maybe. but IMHO a better type system would get rid of arrays and strings altogether and only have containers/records of various sorts. The complexity of having both solving the same problems (not very well in the case of arrays/strings) doesn't buy much. I suspect that a user-defined "." as you've proposed elsewhere would eliminate most of the rest of the problems (and unify everything even further).

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 19 Aug 2020 19:13:15 -0500

> What you do if such an object is allocated via pool-specific access type?

The whole object goes in that pool. The entire mechanism in Janus/Ada is built around pools - the stack is represented by a pool object as well as various other pools to support the mechanism.

[...]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 20 Aug 2020 19:49:34 +0200

> The whole object goes in that pool. [...]

OK, but then you are back to the problem that you do not know how that pool works. The user pool might require a certain order of objects inside it and your interference with relocation will break it.

[...]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 20 Aug 2020 19:49:44 +0200

> [...] IMHO a better type system would get rid of arrays and strings altogether and only have containers/records [...]

But records and arrays are needed as building blocks of containers. How would you get rid of them?

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Thu, 20 Aug 2020 16:19:52 -0400

>But records and arrays are needed as building blocks of containers.

And likely needed for any embedded or low-level work where they are mapped to things like (GP) I/O ports or such...

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 20 Aug 2020 18:25:46 -0500

> [...] The user pool might require a certain order of objects inside it and your interference with relocation will break it.

Such a pool does not implement the interface as defined in 13.11. It's OK of course to write a pool that depends on implementation-specific properties (I've done it many times), but such a pool is not usable with portable Ada code. If the pool allows any sort of allocation at any time, then it will work just fine with the Janus/Ada implementation.

[...]

Note that this is the reason that Ada doesn't support specifying the pool used by a container. It would not be reasonable to restrict the allocations in any way, so implementation-dependent pool designs would not work.

[...]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 20 Aug 2020 18:30:07 -0500

> But records and arrays are needed as building blocks of containers. How would you get rid of them?

There's no reason that a compiler couldn't "build-in" a simple bounded vector container as the basic building block. We already do that for things like Ada.Exceptions, Unchecked_Conversion, and Unchecked_Deallocation, so it's no harder to do that for a vector. (Probably would need some sort of fixed vector for interfacing purposes as well, to deal with other language's and/or system's memory layout.)

One could do something similar for records, although I would probably leave them as in Ada and just allow user-definition of "." (via a getter/setter pair).

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 20 Aug 2020 18:33:40 -0500

> And likely needed for any embedded or low-level work where they are mapped to things like (GP) I/O ports or such...

Yes, a fixed vector container would be needed for interfacing (probably wouldn't use it for anything else). But there's no reason that can't be provided as a container, so long as representation guarantees (esp. Component_Size) are included. Remember that containers (in Ada 202x) have indexing, aggregates, and all of the useful basic operations. The stuff that's missing is the same stuff that adds a vast amount of complexity to Ada (and possibilities for bugs) - hardly anyone would miss it.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 21 Aug 2020 08:46:07 +0200

> There's no reason that a compiler couldn't "build-in" a simple bounded vector container as the basic building block.

That simply replaces the word "array" with four words "simple bounded vector container." The construct is still there and it is still built-in. The syntax and usability are drastically worse, though.

> One could do something similar for records, although I would probably leave them as in Ada and just allow user-definition of "." (via a getter/setter pair).

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 21 Aug 2020 09:08:46 +0200>

>> I meant that if you used a pool behind the scenes for local objects you could do that task-specific eliminating interlocking.

> Whether that would be worthwhile would depend on how expensive the locking is.

It could be very expensive on a multi-core architecture. I also think about scenarios when the object is used inside a protected action. I would not like to see any pool interaction in an interrupt handler!

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sat, 22 Aug 2020 23:48:13 -0500

> That simply replaces the word "array" with four words "simple bounded vector container." The construct is still there and it is still built-in. The syntax and usability are drastically worse, though.

??? The syntax of use is the same (as it is in Ada 2012). Declaration would be an instance, about the same length and wordiness as an array declaration. Yes, junk like slices, settable/retrievable bounds, and built-in operations that are rarely used would be gone, but so would the rather substantial overhead that those things entail. There'd be a lot more flexibility in implementation, which would allow better implementations.

Virtually every array that I write has a fixed size (capacity really) and a usage high-water mark (a "length"). Having that generated automatically would be usually better than having to reinvent it literally every time I program something. (And as you've noticed repeatedly, Ada's type abstraction isn't good enough to make it practical to build anything reusable to do that.)

>> I would probably leave them as in Ada and just allow user-definition of "."

???

The basic idea would be to eliminate the huge number of special cases that exist in Ada resolution and essentially make *everything* a subprogram call at its heart. Ada did that for enumeration literals and that model makes sense for pretty much everything: object usage, indexing, selection, etc. It would be much easier to prove that resolution is doing the right thing (I don't think that would be practically possible for Ada).

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sat, 22 Aug 2020 23:52:30 -0500

> Really? I would miss array conversions, slices, equivalence of same length index ranges, constrained array subtypes etc.

Those things are mostly useful for making work for programmers. Note that I'm assuming that Strings are a completely separate abstraction - a UTF-8 string is not an array and shouldn't be treated as one. (Indexing of individual characters being very expensive.) Fixed constrained

arrays would be available for interfacing (they're not really useful for much else). Note that a bounded vector is allocated statically, so there's no extra cost to using it (unlike an unbounded vector or string).

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sun, 23 Aug 2020 00:03:01 -0500

> [...] I also think about scenarios when the object is used inside a protected action. I would not like to see any pool interaction in an interrupt handler!

Interrupt handlers shouldn't be doing anything other than unblocking tasks. I think it is a mistake to allow anything else (as there are always problems with race conditions if you do so). So no heap possibilities as very little is going on.

Available Ada Compilers

From: gdotone@gmail.com
Subject: Is there another ada compiler
Date: Sun, 2 Aug 2020 19:22:10 -0700
Newsgroups: comp.lang.ada

Is there another Ada compiler other than AdaCore?

From: gautier_niouzes@hotmail.com
Date: Mon, 3 Aug 2020 00:54:26 -0700

Check here: http://unzip-ada.sf.net/#adacomp

or here https://www.adaic.org/ada-resources/pro-tools-services/

for instance.

If you are looking for another *open-source* compiler, but rather incomplete:

https://hacadacompiler.sourceforge.io/

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 3 Aug 2020 06:51:54 -0700

RR Software has Janus/Ada.

PTC has ObjectAda and ApexAda.

Green Hills has an Ada compiler.

DDC-I has a compiler.

IBM used to have a compiler. (I'm not sure they do any more.)

There's also work being done on some open source compilers like HAC or my own Byron.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Mon, 3 Aug 2020 15:29:34 +0100

> RR Software has Janus/Ada.

> PTC has ObjectAda and ApexAda.

> Green Hills has an Ada compiler.

> DDC-I has a compiler.

> IBM used to have a compiler. (I'm not sure they do any more.)

They sold it.

All the above are commercial and cost £££££'s

> There's also work being done on some open source compilers like HAC or my own Byron.

HAC is not going to be a full compiler, so it's really worth mentioning.

Byron's not anywhere near close to generating assembly.

In answer to the OP's question, no, there isn't another open source compiler.

From: nobody in particular <nobody@devnull.org>
Date: Mon, 3 Aug 2020 15:18:36 +0000

> RR Software has Janus/Ada.

Honest company run by good guy Randy Brukhard, who is a long time participant on the newsgroup. Unfortunately, not available on the platform I wanted it for. Hobbyist-friendly.

> PTC has ObjectAda and ApexAda.

There was a recent announcement here in the newsgroup, unfortunately without any pricing. Pricing is also not found on the PTC website. In the past, Aonix did have a hobbyist compiler but I haven't seen it for years.

> Green Hills has an Ada compiler.

Huge money and the salesman I spoke with displayed significant disdain when I turned out to be an individual rather than a company. Did not disclose pricing. However, in speaking with another participant off-list, I was given some sense of the pricing.

> DDC-I has a compiler.

Not sure if anything past '83 is supported. But do check if you're interested. I believe JOVIAL is also available from DDC-I.

> IBM used to have a compiler. (I'm not sure they do any more.)

It was sold to a company in Washington, D.C. which I believe still sells the Ada 95 compiler. I don't believe they support any additional standards after 95. I'm sorry, I can't remember the name.

I attempted to get a hobbyist distribution to run on the Hercules z/Architecture emulator (which also supports MVS, MVS/ESA, and OS/390) but was not successful. Appeared to be a reasonable guy and the product was well integrated in MVS/ESA but probably not generally useful to most people in this newsgroup. If it is, would be worth identifying the company and starting a dialog.

Lastly, we should mention gcc-ada which was still out there for Linux and Solaris last I looked, and even for some unusual platforms like Solaris SPARC. The SPARC platform maintainer was very helpful and I got a copy at some point, I can't remember but I think around gcc5.

There used to be GNAT 3.15p (last non-GPL) release but it was cruelly excised from all servers and download sites when AdaCore happened.

We should note, GNAT / AdaCore were created on the backs of American taxpayers via a grant to New York University. Unfortunately, the taxpayers got the shaft and a profitable business was born to continue the fun.

From: Micronian Coder <micronian2@gmail.com>
Date: Mon, 3 Aug 2020 10:37:05 -0700

Just because a compiler is not free does not mean it is not relevant to someone. In addition, the OP did _not_ specifically ask for an open source compiler. They asked if there are other compilers. Hobbyists generally want a free compiler, so by default GNAT is the one that is used. For companies who want commercial support and are fine with paying money, then the other options are perfectly fine.

Of the commercial ones listed, Janus/Ada is the more affordable one for an individual willing to spend money (see http://www.rrsoftware.com/html/companyinf/prices.htm), especially if they are a student (http://www.rrsoftware.com/html/companyinf/educ.htm). While it's not as up to date as GNAT in terms of Ada2012 support, it's still enough to develop software with (note: Windows only which is fine for many people and can probably run on Wine for Linux users). Judging by Randy's posts in this group, one can expect good support from RRSoftware.

I should point out that PTC is known to provide free access to their compilers if it is for developing *open source* Ada software. Gautier has confirmed this on Reddit (https://www.reddit.com/r/ada/comments/hw33kr/ptc_objectada_for_windows_version_102_outprovides/fyzgpss?utm_source=share&utm_medium=web2x). So there is potential

From: gautier_niouzes@hotmail.com
Date: Mon, 3 Aug 2020 14:44:43 -0700

> We should note, GNAT / AdaCore were created on the backs of American taxpayers via a grant to New York University. [...]

It's called public-private partnership ;-)

See Tesla or SpaceX for other examples. Is it so bad?

BTW, weren't most early Ada vendors essentially financed by the US DoD?

As a consolation, consider that the American taxpayers have become (at least for a while) minority contributors to the US budget...

From: Andreas Zuercher <ZUERCHER_Andreas@outlook.com>
Date: Mon, 3 Aug 2020 17:23:59 -0700

> We should note, GNAT / AdaCore were created on the backs of American taxpayers via a grant to New York University. [...]

Well, I am not usually in the habit of saying nice things about GNAT, but let us compare FSF's GCC GNAT with FSF's GCC CHILL. Ada and CHILL are fierce competitor languages: one from NATO military and the other from ITU-T telecom, where Ada trended a little more toward Wirth family of languages as inspiration whereas CHILL trended a little more toward PL/I as inspiration. Both languages had a 2-decade mandate to be utilized in their respective industrial sectors, but each's mandate had evaporated by the latter half of the 1990s.

Ada had AdaCore arise through several mergers as the for-profit support company for open-source software, analogous to Cygnus Solutions during the 1990s, and its acquirer RedHat until this day. CHILL had a different business model entirely. CHILL compilers were produced by the telecom companies that were self-mandated to use CHILL. If Ada had that business model, Raytheon would have authored its own compiler, Lockheed-Martin would have authored its own compiler, Boeing would have authored its own compiler, Airbus would have authored its own compiler, and so forth. Eventually the telecom companies in Europe fatigued of the effort needed to write a compiler for an evolving language standard (ITU-T Z.200 and ISO 9496), so 2 of them (Alcatel or Siemens, IIRC) outsourced their internal compiler development to Per Bothner, who eventually landed at Cygnus Solutions, after University of Wisconsin at Madison (years after Randy). Eventually, Cygnus Solutions convinced FSF to allow their CHILL compiler into GCC.

Shortly after FSF GCC admitted CHILL into its compiler suite, RedHat bought Cygnus Solutions and nearly all of the European telecom companies were finalizing the financially painful governmental reform where PTTs (postal-telephone-telegraph agencies of governments) were divesting their relationship with the equipment manufacturers—much like AT&T divested WesternElectric/Lucent and Bell Canada no longer had Northern Telecom as favorite-son supplier during much the same 1990s time period. Long story short, when FSF pleaded for someone anyone to update GCC CHILL to GCC 3.X internals, no one stepped forward to fund the effort with money, and most especially no one donated source code as in-kind support. GCC CHILL as donorware ended as of GCC 2.95.

Whatever or however one might critique FSF GNAT versus AdaCore GNAT Pro differences or delays or never achieving perfect congruence among any pairwise matching of any of their releases, GNAT's viability to continue maintenance & evolution is far better that CHILL's donorware-based approach that failed miserably under the same FSF GCC umbrella during the same time period. So matters could be far far worse than they are.

PolyORB and the DSA Annex

From: tonyg <tonythegair@gmail.com>
Subject: Polyorb abd the DSA Annex
Date: Mon, 3 Aug 2020 04:04:27 -0700
Newsgroups: comp.lang.ada

I just tried to build the git cloned copy of PolyORB (failed on the configure!) with the 2020 community version of GNAT. It said I had no GNAT Ada compiler. It was on the path and it pointed to the gcc compiler on the path. Are they compatible? Is there still a PolyORB "enthusiast" list ?

From: "Luke A. Guest" <laguest@archeia.com>
Date: Mon, 3 Aug 2020 12:40:32 +0100

Distributed annex is being removed from GNAT due to "lack of customer interest."

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 3 Aug 2020 06:47:12 -0700

This is pretty sad, and IMO, stupid; the ability to [relatively] easily make distributed applications via DSA is a killer feature and, in conjunction with Ada2020 'parallel' blocks/loops would make for a very attractive system.

IOW, the "lack of customer interest" is an excuse to shoot themselves in the foot.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 3 Aug 2020 16:16:13 +0200

The reality is a bit more complex.

Distributed Annex is based on RPC.

Ada is largely used in the field applications, embedded, real-time. RPC are pretty much useless there, as well as in massively parallel applications.

For service-oriented sluggish applications RPC might be OK, but CORBA is a blocker there, because static topology/configuration is too rigid for such applications. (Static topology is less and less tolerated in the former as well)

P.S. I have an almost ready distributed Annex implementation based on inter process communication (no network, same box), but I have no information how to dock it into GNAT.

Dynamic Variable Creation a la PHP

From: Ian Douglas <ian@vionia.com>
Subject: Newbie question # 2
Date: Thu, 6 Aug 2020 11:40:35 -0700
Newsgroups: comp.lang.ada

I did try Google search and assorted books but could not find an answer.

In PHP, let's say we have a variable $fruit which contains the string "banana".

In PHP, if I do $$fruit, then it creates a variable $banana, which I can then do things with.

Does Ada support any such concept of taking the contents of one variable and using THAT as a variable?

I'm reading in a file which has the name of an object followed by some properties so I want to use the name as a variable ...

File is something I created, so it's not some random stuff, and the variables will be existing already.

From: Simon Wright <simon@pushface.org>
Date: Thu, 06 Aug 2020 19:56:23 +0100

I'd think of a record type to contain the properties, and then a map from object name to properties:

```
type Properties is record
   Length : Positive;
   Width  : Positive;
end record;
package Object_Maps is new
Ada.Containers.Indefinite_Ordered_Maps
   (Key_Type     => String,
    Element_Type => Properties);
Objects : Object_Maps.Map;
```

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 6 Aug 2020 22:20:11 +0300

> In PHP, let's say we have a variable $fruit which contains the string "banana". If I do $$fruit, then it creates a variable $banana, which I can then do things with.

Fortran has a similar feature, NAMELIST, for reading values into variables also named in the input. It can also be used for output.

Ada does not have such a feature.

[...]

From: Ian Douglas <ian@vionia.com>
Date: Thu, 6 Aug 2020 12:41:53 -0700

> I'd think of a record type to contain the properties, and then a map from object name to properties:

Yes, the variables are actually records.

> package Object_Maps is new Ada.Containers.Indefinite_Ordered_Maps

Okay that's a new construct I haven't come across yet. Let me see what I can dig up on that.

From: Ian Douglas <ian@vionia.com>
Date: Thu, 6 Aug 2020 12:45:04 -0700

> Ada does not have such a feature.

I figured as much, probably "unsafe programming practice" at the end of the day.

[...]

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 6 Aug 2020 23:08:00 +0300

> I figured as much, probably "unsafe programming practice" at the end of the day.

I wouldn't say so. I think "namelist" input is a perfectly reasonable function to have in some programs, and is not particularly unsafe in any way -- if the programmer can limit the set of variables that can be named and changed by such input, which is the case in Fortran (and also in our various suggestions for implementing it in Ada).

PHP is (I believe) an interpreted language, so the symbol table is around at run-time, which makes it easy for PHP to support variables that refer to any other variable by its symbolic name. This can make "namelist" input in PHP unsafe, since the input can change any variable -- including variables that the programmer did not intend to be changeable in this way.

Ada is usually compiled, and the symbol table is not present when the compiled program runs, so it would be harder to implement a "namelist" input/output feature. But not impossible, as Fortran shows.

Ada on Beaglebone Black

From: Ricardo Brandão <rbrandao.br@gmail.com>
Subject: Running ADA on Beaglebone Black
Date: Sun, 9 Aug 2020 07:39:58 -0700
Newsgroups: comp.lang.ada

I've just acquired a Beaglebone black and I'm trying to run a simple program in Ada.

I tried to install GNAT but apt-get install GNAT or anything similar doesn't work.

I didn't find any place with the repository, nor any tutorial.

How is the best way to run Ada in the BBB?

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sun, 9 Aug 2020 17:58:54 +0200

What distribution? APT suggests Debian or Ubuntu. In any case I would do a full upgrade to the latest version of either.

Under Debian buster you should do

```
apt install gnat-8
```

Under Ubuntu it would be

```
apt install gnat-10
```

And also:

```
apt install gprbuild
```

If these do not work check /etc/apt/sources.list

From: Philip Munts <philip.munts@gmail.com>
Date: Sun, 9 Aug 2020 10:17:12 -0700

Depending on what you want to accomplish (i.e. whether it needs the full Debian OS), and whether you are willing and able to do cross-compilation, I think that MuntsOS (https://github.com/pmunts/muntsos) is the easiest way to develop and run Ada programs on a BeagleBone or Raspberry Pi.

I have found that installing and maintaining and especially configuring I/O with Debian on a BeagleBone to be a pain in the nether regions. In contrast, I have attempted to make MuntsOS as easy to use as possible. Its only real downside is that it requires cross-toolchains running on a Linux (preferably Debian or Ubuntu) development host. Windows Subsystem for Linux (WSL1 or WSL2) works perfectly well for the development host, though.

From: Ricardo Brandão <rbrandao.br@gmail.com>
Date: Sun, 9 Aug 2020 12:56:35 -0700

> APT suggests Debian or Ubuntu. In any case I would do a full upgrade to the latest version of either.

Yes, I ran sudo apg-get update and GNAT appear in apt-cache search gnat

> Under Debian buster you should do

> apt install gnat-8

> And also:

> apt install gprbuild

I ran these commands and worked fine.

Thank you so much

MITRE's Top-25 List of 2020 Software-bug Categories

From: Andreas Zuercher <ZUERCHER_Andreas@outlook.com>
Subject: MITRE's top-25 list of 2020 software-bug categories
Date: Sat, 22 Aug 2020 09:31:13 -0700
Newsgroups: comp.lang.ada

https://www.bleepingcomputer.com/news/security/mitre-shares-this-years-top-25-most-dangerous-software-bugs/

Proper intended usage of Ada-specific features mitigates 9 of them, including a few that hit interpreted scripting languages hard. Others of the 25 are design-level almost independent of programming language. Still others of the 25 are cavalier/insufficient WWW-oriented string-processing or SQL string-processing or director-filename string-processing that could be conceivably done just as badly in Ada.

Conversely, if HOLWG were still pursuing their language-design goals today, certainly this MITRE* report would shape the design of an evolving GreenGreenerGreenest language today, instead of Ada solving primarily yesteryear's programming/software-engineering challenges well.

* defense contractor

From: Shark8 <onewingedshark@gmail.com>
Date: Tue, 25 Aug 2020 12:09:47 -0700

The interesting portion, in tabular form.

Rank - ID - Name - Score

1 CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') 46.82

2 CWE-787 Out-of-bounds Write 46.17

3 CWE-20 Improper Input Validation 33.47

4 CWE-125 Out-of-bounds Read 26.50

5 CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer 23.73

6 CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') 20.69

7 CWE-200 Exposure of Sensitive Information to an Unauthorized Actor 19.16

8 CWE-416 Use After Free 18.87

9 CWE-352 Cross-Site Request Forgery (CSRF) 17.29

10 CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') 16.44

11 CWE-190 Integer Overflow or Wraparound 15.81

12 CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') 13.67

13 CWE-476 NULL Pointer Dereference 8.35

14 CWE-287 Improper Authentication 8.17

15 CWE-434 Unrestricted Upload of File with Dangerous Type 7.38

16 CWE-732 Incorrect Permission Assignment for Critical Resource 6.95

17 CWE-94 Improper Control of Generation of Code ('Code Injection') 6.53

18 CWE-522 Insufficiently Protected Credentials 5.49

19 CWE-611 Improper Restriction of XML External Entity Reference 5.33

20 CWE-798 Use of Hard-coded Credentials 5.19

21 CWE-502 Deserialization of Untrusted Data 4.93

22 CWE-269 Improper Privilege Management 4.87

23 CWE-400 Uncontrolled Resource Consumption 4.14

24 CWE-306 Missing Authentication for Critical Function 3.85

25 CWE-862 Missing Authorization 3.77

From: Andreas Zuercher <ZUERCHER_Andreas@outlook.com>
Date: Tue, 25 Aug 2020 12:43:13 -0700

> Would 've been nice if you'd have also given the examples and how Ada solved them.

I am not going to write an entire textbook here on c.l.a, but here are the nine of the top twenty-five subcategories that I consider Ada diligently trying to mitigate or eliminate when properly utilized:

• 2nd-most frequent: CWE-787 Out-of-bounds Write

• 3rd-most frequent: CWE-20 Improper Input Validation

• 4th-most frequent: CWE-125 Out-of-bounds Read

• 5th-most frequent: CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

• 8th-most frequent: CWE-416 Use After Free

• 11th-most frequent: CWE-190 Integer Overflow or Wraparound

• 13th-most frequent: CWE-476 NULL Pointer Dereference

• 17th-most frequent: CWE-94 Improper Control of Generation of Code ('Code Injection')

• 23rd-most frequent: CWE-400 Uncontrolled Resource Consumption

There are 1,248 Common Weakness Enumerations (CWEs) that MITRE lobs against software development (instead of against hardware development), so you can peruse the 26th through 1,248th if you so desire. Query 699 is the one for looking at the full inventory of subcategories of software defects. These 1,248 subcategories (and the aforementioned top-25 subcategories) fall into 40 more-macroscopic broader categories.

https://cwe.mitre.org/data/definitions/699.html

I claim that next-gen Ada (AdaNG, pronounced “a dang” as in do we give a dang or not) would use these 1,248 categories as measuring stick of expressibility of software-engineering correctness, just as HOLWG's Green and Ada used Steelman as measuring stick of the ability to express software-engineering correctness.