ADA USER JOURNAL

Volume 45
Number 3
September 2024

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,

Today, I want to highlight two conversations in the Digest that interested me particularly. Firstly, documentation about the 'Red' language has been found online, and it seems that this completes the availability of all contestants from which Ada emerged victorious [1]. Curiously, I found the style of the code not that unfamiliar for an Ada programmer.

Secondly, an animated discussion emerged around the idea of "what Jean Ichbiah would want to find in Ada 2022" [2]. Therein you can also find the reservations he had about preliminary versions of Ada 95 [3], which is in itself worth a read if you have not read them before (as I had not).

Sincerely,
Alejandro R. Mosteo.

[1] "'Red' and the DoD Language Competition", in Ada and Other Languages.

[2] "Ichbiah 2022 Compiler Mode", in Ada Practice

[3] https://web.elastic.org/~fche/mirrors/old-usenet/ada-with-null

Ada-related Events

[AEiC 2024] Ada Developers Workshop Videos and Slides

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: [AEiC 2024] Ada Developers Workshop videos and slides are public
Date: Thu, 8 Aug 2024 20:49:18 +0200
Newsgroups: comp.lang.ada

Dear Ada community,

the recordings of the talks that were held in the Ada Developers Workshop have been made available in the AEiC 2024 website [1]. The slides for each presentation can also be found there. The links can be found just under the title for each entry.

Also, huge thanks to Dirk, Nam, Fabien, the organisers of the conference; Ada-Europe and AdaCore for their sponsorship and their funding to get the technology ready to record the Workshop.

[1] https://www.ada-europe.org/conference2024/adadev.html

Best regards,
Fer & the Ada Developers Workshop team

P.S: any kind of feedback is more than welcome!

Ada Monthly Meetup, September 2024

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: Ada Monthly Meetup, September 2024
Date: Sun, 11 Aug 2024 17:21:09 +0200
Newsgroups: comp.lang.ada

I would like to announce the September (2024) Ada Monthly Meetup which will be taking place on the 7th of September at 13:00 UTC time (15:00 CEST). As always the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to Youtube.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no proposals. Nonetheless, I would like to talk about the AEiC 2024 Ada Developers Workshop, remind people about the 2024 Crate of the Year Award and maybe talk a bit about the Ada Users Society :)

Here are the connection details from previous posts: The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is Jitsi Meet The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules". I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

Ada Monthly Meetup, 5th October 2024

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: Ada Monthly Meetup, 5th October 2024
Date: Mon, 16 Sep 2024 22:43:14 +0200
Newsgroups: comp.lang.ada

I would like to announce the October (2024) Ada Monthly Meetup which will be taking place on the **5th of October at 13:00 UTC time (15:00 CEST).** As always the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to Youtube.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no proposals. Nonetheless, I would like to bring some topics that were left off during September's Meetup.

Here are the connection details from previous posts: The meetup will take place over at Jitsi [1], a conferencing software that runs on any modern browser. The link is Jitsi Meet The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules". I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

Best regards and see you soon!
Fer

[1] https://meet.jit.si/AdaMonthlyMeetup

P.S: you can see the September summary in https://forum.ada-lang.io/t/ada-monthly-meeting-september-2024/1073/6 or in YouTube (with audio issues) https://www.youtube.com/live/i_bVoiDlw5E

Ada-related Resources

[Delta counts are from July 11th to November 13th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 13 Nov 2024 19:35 CET[b]
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 8_868 (+127) members [1]
- LinkedIn: 3_549 (+28) members [2]
- Stack Overflow: 2_426 (+15) questions [3]
- Ada-lang.io: 287 (+46) users [4]
- Gitter: 271 (+13) people [5]
- Telegram: 208 (+3) users [6]
- Libera.Chat: 69 (-4) concurrent users [7]

[1] https://old.reddit.com/r/ada/

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

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

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

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

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

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 13 Nov 2024 19:39 CET[c]
To: Ada User Journal readership

GitHub: >1_000* (+260) developers [1]
Rosetta Code: 1_005 (+26) examples [2]
42 (=) developers [3]
Alire: 483 (+71) crates [4]
1_268 (+200) releases [5]
Sourceforge: 251 (-1) projects [6]
Open Hub: 214 (=) projects [7]
Codelabs: 60 (+3) repositories [8]
Bitbucket: 37 (=) repositories [9]

*This number is a lower bound due to GitHub search limitations.

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

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

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

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

[5] `alr search --list --full`

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: 13 Nov 2024 19:48 CET
To: Ada User Journal readership

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

- TIOBE Index: 25 (-1) 0.71% (-0.08%) [1]
- PYPL Index: 15 (+2) 1.03% (+0.07%) [2]
- Languish Trends: 153 (+39) 0.01% (+0.01)% [3]
- Stack Overflow Survey: 40 (+2) 0.9% (+0.13%) [4]
- IEEE Spectrum (general): 50 (-14) Score: 0.0014 0107 (-0.093) [5]
- IEEE Spectrum (jobs): 55 (-26) Score: 0.0 (-0.0173) [5]
- IEEE Spectrum (trending): 46 (-16) Score: 0.0022 (0.01) [5]

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

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

[3] https://tjpalmer.github.io/languish/

[4] https://survey.stackoverflow.co/2024/

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

Re: Ada-Lang and Its Forum

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada-Lang and it's (more active than CLA) forum
Date: Tue, 2 Jul 2024 02:55:49 -0500
Newsgroups: comp.lang.ada

[Cont'd from AUJ 45-2, April 2024 —arm]

> Ada-Lang is a community maintained and supported webpage whose intent is to give a nice "landing page" to anybody wanting to learn Ada and become a hub for all Ada users.

I was adding this site to AdaIC's "Learn" pages (I think it disappeared some years ago, it is good to see it back), and noted that nowhere does it identify itself as "Ada-Lang" or any other short name on the site itself. It just calls itself "Ada Programming Language", which is a bit grandiose (there are a number of sites that can lay claim to part of that title, but surely none that can lay claim to all of it). Within the Ada Community in particular, it helps to identify the site more precisely. And I don't think that many people really look at the links that they click on, I doubt many people using AdaIC do, so just using the domain name and assuming people know what it is without any identification elsewhere is not ideal.

My two cents worth. (Humm, given prices these days, I don't think you can actually buy anything with two cents. That's probably one cliche that needs updating. ;-)

AWS-friendly Web Hosting

From: Marius Alves <marius2023pt@gmail.com>
Subject: Ada/GNAT/AWS-friendly web hosting
Date: Thu, 12 Sep 2024 15:25:41 +0100
Newsgroups: comp.lang.ada

Researching how to build an HTTP server (serving a website) on a local machine (MacOS) using AWS (Ada Web Server) and deploy it on a web hosting provider (e.g. 1dollar-webhosting.com).

Anyone done that? I've searched but could not find [anything].

Thanks.

Some specific questions on my mind follow.

Is a macOS host required (e.g. Ultahost 15 euros/month; I'd rather stay with 1dollar)?

If the host runs on Linux then cross-building (from macOS to Linux) is required, right? GNAT does that, right?

Or, must the program be built in the host? (Thus requiring GNAT to be there.)

The host is already running an HTTP server program (probably Apache). Must it be turned off? How?

In general, can the executable be launched on a VPS (Virtual Private Server)? Which port?

Will dynamic linking work? I'm guessing not, so, static; but then, will GNAT integrate the right libraries for Linux in the executable?

Will "Community GNAT" do? (Instead of GNAT Pro.)

Are those the right questions?

Thanks, thanks, thanks, thanks, thanks, thanks and thanks.

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 12 Sep 2024 16:48:40 +0200

Adalog's site (https://www.adalog.fr/) is a standalone program written in Ada with AWS.

So are the sites for the various Ada-Europe conferences (see https://www.ada-europe.org/conference2024/ for example).

And many others...

> Is a macOS host required

No

> If the host runs on Linux then cross-building (from macOS to Linux) required, right?

Never tried, but no reason it shouldn't be possible

> Or, must the program be built in the host?

That's what I do

> The host is already running an HTTP server program (probably Apache). Must it be turned off? How?

Of course, you cannot have two programs listening on the same port, so if you want to listen to 80 or 8080, you'd better stop Apache (or any other program) to do that. As for me, I don't run Apache at all.

> In general, can the executable be launched on a VPS (Virtual Private Server)? Which port?

The port is given by the initial data of AWS

> Will dynamic linking work?

You just compile your program like any other Ada program

> Will "Community GNAT" do? (Instead of GNAT Pro.)

Yes, that's what I do

> Are those the right questions?

All questions are right....

> Thanks, thanks, thanks, thanks, thanks, thanks and thanks.

You're welcome

From: Drpi <314@drpi.fr>
Date: Thu, 12 Sep 2024 16:54:45 +0200

> The host is already running an HTTP server program (probably Apache). Must it be turned off? How?

The usual way is to use Apache (or nginx or another one) as a front end. Your application uses port 1080 (or something else) and the front end relays this port to the external 80 port.

This way, the security stuff is managed by the front end, not your application. You can also run multiple applications, each being redirected to its domain name/path.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 12 Sep 2024 18:22:28 +0200

> Researching how to build an HTTP server (serving a website) on a local machine (MacOS) using AWS (Ada Web Server) and deploy it on a web hosting provider (e.g. 1dollar-webhosting.com).

In my experience, this would be easier done with Gnoga (https://sourceforge.net/projects/gnoga/) than AWS. On a web-based system using AWS quite a while ago, we had to have a number of JS files. Although we had a lot more Ada than JS, we spent a lot more effort correcting JS errors than Ada errors.

Gautier de Montmollin has made Gnoga programs publicly available, such as his Pasta! game (http://pasta.phyrama.com/), so might be able to help with your hosting questions.

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 12 Sep 2024 19:06:08 +0200

> This way, the security stuff is managed by the front end

But security breaches mainly use known bugs in Apache... If you write your own server with AWS, the attacker knows nothing about the software that answers! And as for buffer overflow attacks... Well, it's Ada. You'll see some handled Constraint_Error in the log file, end of story!

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Thu, 12 Sep 2024 17:16:29 -0000

> But security breaches mainly use known bugs in Apache… [...]

AWS uses OpenSSL or a fair bit better LibreSSL for TLS, written in C and quite often found vulnerable. You could isolate the nginx proxy to another machine though.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 12 Sep 2024 20:48:29 +0200

> Researching how to build an HTTP server (serving a website) on a local machine (MacOS) using AWS (Ada Web Server) and deploy it on a web hosting provider (e.g. 1dollar-webhosting.com).

That depends on what the provider would allow you to upload to the host. Likely nothing executable... (:-))

> If the host runs on Linux then cross-building (from macOS to Linux) is required, right?

It is possible, but far simpler would be a virtual machine running Linux. E.g. I compile for Linux targets on virtual machines. Only for ARM I am using physical machines. You must know what kind of Linux your provider has in order to choose the right version of the libc etc. [...]

> Will dynamic linking work?

If you ship the libraries together with the server. Then if the host runs Apache it must have some TLS library installed. You must learn the version and link against it. In any case you need either OpenSSL or else GNUTLS. The HTTP server from Simple Components can use both. I believe that either can be built as a static library. I see no reason why AWS could not be linked statically. BTW you must maintain certificates on the server.

> Will "Community GNAT" do?

I am not sure if all-static builds were possible, e.g. libc, libgnat.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 12 Sep 2024 22:29:36 -0000

> we spent a lot more effort correcting JS errors than Ada errors.

Did you “use strict”?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Fri, 13 Sep 2024 11:03:03 +0200

> Did you “use strict”?

I don't know. It was quite a while ago and I didn't work on the JS. But the point is that when you use Gnoga, you don't need any to create any JS.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 12 Sep 2024 22:35:20 -0000

> But security breaches mainly use known bugs in Apache…

That’s called “security through obscurity”. Not recommended.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 12 Sep 2024 22:40:35 -0000

> The usual way is to use Apache (or nginx or another one) as a front end.

Yup, I do things this way for my Python+ASGI code, too. This is called a “reverse proxy”, though I don’t know why -- I think “server-side proxy” would be more accurate.

Make sure your back-end server is listening only on a loopback address: 127.0.0.0/8 (IPv4) or ::1 (IPv6). That way the only access to it from outside the machine is through the public web-server front end.

(Question to ponder: why does Ipv4 offer over 16 million different loopback addresses, while IPv6, with its much larger address space, has to make do with only one?)

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 13 Sep 2024 08:46:33 +0200

> That's called "security through obscurity". Not recommended.

No, AWS is public and there is nothing hidden. Just that, since there are wayyyyy more users of Apache than of AWS, attackers will not bother to try to break in

From: Stéphane Rivière <stef@genesix.org>
Date: Fri, 13 Sep 2024 15:15:03 +0200

As a professional web hoster, I strongly advise you to forget Apache and use only Nginx, both as a proxy (in your case) and as a web server (generic case). Not only does Apache have security problems, but its performance is pitiful compared to Nginx.

If you have several sites, the ideal solution is to enter everything in https/port 443 on the nginx proxy (which will be able to manage X509/TLS https certificates) and exit on as many ports 8080, 8081, 8082, etc. as you have websites.

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Fri, 13 Sep 2024 16:33:15 +0200

> Researching how to build an HTTP server (serving a website) on a local machine (MacOS) using AWS (Ada Web Server) and deploy it on a web hosting provider (e.g. 1dollar-webhosting.com).

I don't know about 1dollar, but a typical web hosting provider will only let you upload static files (HTML, pictures et cetera), limited snippets of web server configuration, and certain kinds of programs that run under their web server's control. PHP is common. Some might run Perl programs with mod_perl, or Python programs using WSGI.

Maybe some web hosts support CGI or FastCGI. Those interfaces can be implemented in Ada. I think you'll have limited use for AWS in that case, as the HTTP parsing is handled by the web server.

I think it would be hard to find a web host that lets you run arbitrary network-facing daemons. To run your own web server you want a VPS (or a physical server in a colocation facility, but if your security needs don't rule out a web host, then a VPS is also fine).

> The host is already running an HTTP server program (probably Apache). Must it be turned off? How?

A typical web host won't let you turn off their web server. They serve many customers' content from the same Apache instance, so turning that off would break all those websites.

> In general, can the executable be launched on a VPS (Virtual Private Server)?

Sure. In a VPS you have the whole operating system to yourself (maybe except for the kernel if the VPS provider uses OpenVZ). You install and run whatever programs you want, just like on your own physical computer. Maybe you'll be able to get a VPS with macOS, if that's your preference.

In a VPS it's also your responsibility to install updates regularly, and upgrade to a new major OS version from time to time. If you fail to keep up, then criminals will take over your VPS and use it as a relay when attacking others. Make sure that you'll be notified automatically when there are updates to install.

> If the host runs on Linux then cross-building (from macOS to Linux) is required, right?

GCC – and thus GNAT – can be built as a cross-compiler. Perhaps you can find one that someone has built and packaged for MacOS. Otherwise you'll need to build your own from the GCC source code, configuring it to be a cross-compiler. (That's theoretical knowledge. I have no practical experience with cross-compilation).

> Or, must the program be built in the host? (Thus requiring GNAT to be there.)

No, but in my opinion it's much easier that way. Either build on the computer you'll run on, or on another computer of the same processor architecture, running the same version of the same operating system. That way you don't need to worry about getting the wrong version of some library or build tool.

> Will dynamic linking work?

Cross-compilation should be able to work with shared libraries. Regardless of whether the libraries are shared or static, libraries for the target machine must be available on the build host. I guess you would either install packaged libraries on the target machine, and copy those to the build host, or else cross-compile the libraries too. You need to configure search paths carefully so that both the compiler and the linker find the cross-libraries instead of the native ones. This is one of the complications you avoid by building natively.

> Which port?

Normally port 443, because of course you'll use HTTPS, won't you? Optionally you can also have an HTTP server on port 80 that responds to every request with a redirection to HTTPS.

If you choose to put AWS behind a reverse proxy like DrPi suggested, then the reverse proxy listens on port 443 on your public IP address, and you tell AWS to listen on some other port and only on the localhost address, ::1 or 127.0.0.1.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 14 Sep 2024 01:38:16 -0500

> That's called "security through obscurity". Not recommended.

That's the wrong way to look at it. An Ada program is better thought of as "security by simplicity and correctness", because you are running an Ada that only does a few things (and which can be thoroughly tested, checked with static analysis, and so on) rather than a general program that does a zillion things (with many combinations that can't be tested).

The only place "obscurity" comes into it is that no one else is running the exact same program as you. So attacks that depend on any sort of knowledge of the program cannot succeed.

In any case, there is no such thing as "secure", there are only levels, and for the sorts of non-critical stuff that we're doing, an Ada program is certainly secure enough. I wouldn't try to run a storefront on it (although that would be more because you'd have a hard time convincing your bank that it is OK than any real problems), or anything that needs high-level security.

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Sat, 14 Sep 2024 12:02:05 -0000

> work with Gnoga (https://v22.soweb.io).

Runs on Android/IOS. Does that require an internet web server?

From: Stéphane Rivière <stef@genesix.org>
Date: Sat, 14 Sep 2024 15:00:00 +0200

> Runs on Android/IOS.

Yes, v22.Gui/Gnoga is responsive. Tested with 5" smartphones as old as Nexus 5 (with a browser more recent than the stock one to handle websockets). Also tested on 43" 4K ;)

On some iOS devices, the menu bar is slightly offset. I didn't look too hard. It's a Safari problem. It works fine with Firefox and Chrome.

> Does that require an internet web server?

Not necessarily. v22.Gui/Gnoga supports itself X509 TLS https certificates (tested). However, for various reasons (such as the possibility of having several web applications on the same instance and on

the same 80/443 input port), in production, I've always chosen to have a Nginx proxy on the front end, which is also more flexible and handles automatic switching from http/80 to https/443.

adaic.org; Is There a Problem?

From: John Mccabe <john@nospam.mccabe.org.uk>
Subject: adaic.org; is there a problem?
Date: Tue, 17 Sep 2024 16:19:47 -0000
Newsgroups: comp.lang.ada

Sorry to ask here; I wasn't sure where else to go, but is www.adaic.org ok for everyone? I'm just seeing a mostly white screen with a blackish bar at the top on Firefox, Chrome and Edge (Chrome on both Windows and Android). It might just be me, but I thought I'd ask in case anyone else sees it like that and can prod the right people to fix it or, alternatively, just let me know that it's a problem at my end!

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 17 Sep 2024 18:35:16 +0200

FOOBAR.

It looks as though a significant part of the HTML is missing.

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Tue, 17 Sep 2024 16:42:37 -0000

Thanks for that Bill; at least I'm not going mad then :-)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Tue, 17 Sep 2024 17:59:37 -0000

I noticed the problem with adaic.org as well, and have informed Randy, its webmaster, yesterday already. Stay tuned until he kicks the server back into action... ;-)

From: Blady <p.p11@orange.fr>
Date: Mon, 23 Sep 2024 20:31:42 +0200

I noticed a similar problem with www.ada-auth.org?

From: Luke A. Guest <laguest@archeia.com>
Date: Tue, 24 Sep 2024 13:56:47 +0100

Yup both down, just tried them.

Ada-related Tools

GNAT Studio 25.0 for macOS Ventura.

From: Blady <p.p11@orange.fr>
Subject: [ANN] GNAT Studio 25.0 for macOS Ventura.
Date: Fri, 26 Jul 2024 12:02:15 +0200
Newsgroups: comp.lang.ada

Here is a very preliminary version of GNAT Studio 25.0wa as a standalone app for macOS:
https://sourceforge.net/projects/gnuada/files/GNAT_GPL Mac OS X/2024-ventura

NEW:

The GNATStudio launcher looks for a gnatstudio_launcher.rc file in the .gnatstudio folder from either $HOME or $GNATSTUDIO_HOME locations. If it exists, we can define some environment variables with the standard syntax VAR=VALUE. If the VAR exists then VALUE is appended to it. If not, VAR is created with VALUE. Thus, it permits to set extra PATH to GNAT compiler and builder folders or GPR_PROJECT_PATH. If a line begins with '#' then it is not considered. An example file of gnatstudio_launcher.rc is provided in the archive. Modify the content and put it in your .gnatstudio folder.

See readme for details.

Limitation: Ada Language Server has some latences and doesn't respond when parsing source code with more than 1000 lines. It may be due to some compilation options I missed.

There could be some other limitations that you might meet.

Feel free to report them here.

Any help will be really appreciated to fix these limitations.

KDF9 Pascal, Thanks to Ada

From: Moi <findlaybill@blueyonder.co.uk>
Subject: KDF9 Pascal, thanks to Ada
Date: Tue, 2 Jul 2024 01:27:43 +0100
Newsgroups: comp.lang.ada

Some time ago it occurred to me that the best way to illustrate the remarkable architecture of the EE KDF9 would be to write a cross-compiler that generates idiomatic KDF9 Usercode (assembly language) and displays it in association with the source code.

I chose Pascal as the source language, having compiler texts available for retargeting.

PASKAL, which implements a large subset of Pascal, is now available.

The only parts of Pascal not implemented are file types and packed types, including the 'text' type, which means that there is no Standard Pascal I/O. However, I provide some basic KDF9-oriented output routines as a stopgap. They are more than adequate to show the correct execution of, for example, the Whetstone Benchmark, and many other classic codes, such as Quicksort.

PASKAL is written in Pascal, using the fpc compiler, and in Ada 2012, and is included with V11.2c of ee9, my KDF9 emulator (also in Ada 2012).

Included with it are the following documents:

* PASKAL: Users' Guide
* PASKAL: Object Program Structure.
* PASKAL: Implementation Overview

Compiled binaries are available for:

* Apple Silicon Macs
* Intel Macs
* 64-bit Intel (Debian Bookworm) Linux
* 64-bit Raspberry Pi (Debian Bookworm) OS

The Intel Linux binary should run under WSL on MS Windows 10 or 11.

Get your copy here:
http://www.findlayw.plus.com/KDF9/#PSK

There is a direct link there to the Users' Guide. It includes an example of a complete Pascal program and the corresponding KDF9 Usercode, should that be the extent of your interest.

Gnoga's 10th Anniversary - 2.2 Released.

From: Blady <p.p11@orange.fr>
Subject: Gnoga's 10th anniversary - V2.2 released.
Date: Sun, 8 Sep 2024 18:30:49 +0200
Newsgroups: comp.lang.ada

Gnoga was born on SourceForge [1] on September 8, 2014.

Gnoga (GNU Omnificent Gui for Ada) is the multi-platform graphics library created natively in Ada. I immediately liked Gnoga for the coherence and simplicity of these APIs naturally fitting together. The programmer can rely on Ada for his business code and on the multitude of Javascript libraries for the graphical interface.

For 10 years Gnoga has evolved in maturity to fulfill its founding principles:

- providing a framework and associated tools for developing GUI applications using the Ada language, leveraging web technologies for application developers
- developing native applications for desktop and mobile just as easy to create, all using the same code base
- providing better tools means better application quality
- offering the application developer a powerful toolset for secure cloud based computing, mobile apps, desktop apps and web apps the combination not found in any other set of tools in any other language

Gnoga statistics:

- 1031 commits
- 2200 downloads
- 2196 posts on the mailing list
- 56 tickets

You'll find a special Gnoga's wiki anniversary page [2] with some materials and my testimony.

Feel free to post your testimony, your own story with Gnoga.

On this occasion, Gnoga V2.2a has been released [3] and [4], with main changes:

- Added key field to keyboard event
- If present command line options gnoga-host, gnoga-port, gnoga-boot and gnoga-verbose will override host, port, boot file and verbosity programmed in source code (see TIPS).
- Improve logging implementation in a separate package in order to allow user defined logging handlers.
- Add a backslash compatibility mode on the behavior of Escape_String for SQLite with the one for MySQL.
- Change MYSQL_Real_Connect profile to better match with documentation

This version has been tested on macOS 13.6 and GNAT 14.1. Please provide feedback of other environments.

[1] https://sourceforge.net/p/gnoga/code/ci/45c76779e7af7b869deacc698478eb3ef25cfe91

[2] https://sourceforge.net/p/gnoga/wiki/Gnoga-Anniversary

[3] https://sourceforge.net/projects/gnoga/files

[4] https://sourceforge.net/p/gnoga/code/ci/dev_2.2/tree

Ada Inside

Canal+ Crash

From: Nicolas Paul Colin De Glocester <master_fontaine_is_dishonest @strand_in_london.gov.uk>
Subject: Canal+ crash
Date: Fri, 19 Jul 2024 23:41:44 +0200
Newsgroups: fr.comp.lang.ada, comp.lang.ada

Canal+ uses Ada but one is alleging that Canal+ suffered a crash today with Windows. Cf.

https://www.UniversFreeBox.com/article/568957/orange-canal-et-bouygues-telecom-annoncent-a-leurs-abonnes-etre-touches-par-la-panne-informatique-mondiale

Cf. a complaint by Mister Brukardt that Ada cannot control non-Ada software on a shared system.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 20 Jul 2024 09:23:11 +0200

It is not about Ada. It is about the fundamental principle that security cannot be added on top of an insecure system. The lesson never learned is that security levels impose safety problems not solving security issues. Modern security architectures are nothing but a huge scam.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sat, 20 Jul 2024 07:43:18 -0000

> It is about the fundamental principle that security cannot be added on top of an insecure system.

Actually, it can. Notice how the Internet itself is horribly insecure, yet we are capable of running secure applications and protocols on top of it.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 20 Jul 2024 11:08:47 +0200

> we are capable of running secure applications and protocols on top of it.

Of course we can. That is the whole idea of the scam. Why on earth do we need security updates? Do you update your screwdriver each week?

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 21 Jul 2024 01:04:44 -0000

> Why on earth do we need security updates?

Because computer systems are complex, and new bugs keep being discovered all the time.

> Do you update your screwdriver each week?

I don't depend on my screwdriver to keep my bank account secure.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 21 Jul 2024 09:22:06 +0200

> Because computer systems are complex, and new bugs keep being discovered all the time.

This does not make sense. You can create a very complex system out of screwdrivers and still each screwdriver would require no update.

Systems consist of computers and computers of software modules. There is nothing inherently complex about making a module safe and bug free. Security interactions are primitive and 100% functional. There are no difficult issues with non-functional stuff like real-time problems. It is purely algorithmic while all mathematical complexity of cryptography is NOT what gets updated. It is complex only because it was designed as a Wood Block Tumbling Game.

> I don't depend on my screwdriver to keep my bank account secure.

I don't need a bank account to fasten the screws. Application area is irrelevant.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 21 Jul 2024 11:00:36 +0300

> Security interactions are primitive and 100% functional. There is no difficult issues with non-functional stuff like real-time problems.

Well, several recent attacks use variations in execution timing as a side-channel to exfiltrate secrets such as crypto keys. The crypto code can be functionally perfect and bug-free, but it may still be open to attack by such methods.

But certainly, most attacks on SW have used functional bugs such as buffer overflows.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 21 Jul 2024 11:10:06 +0200

> But certainly, most attacks on SW have used functional bugs such as buffer overflows.

A problem that has been solved since 1983, and even before (Pascal had bounds checking). Sigh…

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 21 Jul 2024 11:19:30 +0200

> Well, several recent attacks use variations in execution timing as a side-channel to exfiltrate secrets such as crypto keys.

It is always a tradeoff between the value of the information and costs of breaking the protection. I doubt that timing attack are much more feasible in that respect than brute force.

> But certainly, most attacks on SW have used functional bugs such as buffer overflows.

Exactly. Non-functional attacks are hypothetical at best. They rely on internal knowledge which is another problem. An insider work is the most common case of all breaches. So, maybe, it is better to update the staff? (:-))

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 21 Jul 2024 11:34:14 +0200

> A problem that has been solved since 1983, and even before (Pascal had bounds checking). Sigh...

Yup, however some crackpot could always suggest an attack on bounds checking, e.g. exception vs. not, index to bounds comparison dependent on the actual values etc., and then produce a lengthy paper on a constructed absolutely unrealistic scenario... (:-))

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 21 Jul 2024 14:31:27 +0300

> I doubt that timing attack are much more feasible in that respect than brute force.

Security researchers and crypto implementers seem to take timing attacks quite seriously, putting a lot of effort into making the crucial crypto steps run in constant time.

> Non-functional attacks are hypothetical at best. They rely on internal knowledge which is another problem.

As I understand it, the "internal knowledge" needed for timing attacks is mostly what is easily discoverable from the open source-code of the SW that is attacked.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 21 Jul 2024 18:49:27 +0200

> Security researchers and crypto implementers seem to take timing attacks quite seriously

Cynically: they certainly know how to butter their bread...

> the "internal knowledge" needed for timing attacks is mostly what is easily discoverable from the open source-code

Considering many many layers of software to predict timing from code in uncontrolled environment would be a challenge.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 21 Jul 2024 21:52:58 -0000

> You can create a very complex system out of screwdrivers and still each screwdriver would require no update.

There is an old engineering adage, that the complexity of a system arises, not so much from the number of individual components, as from the number of potential interactions between them.

If you have a box full of screwdrivers, then all you have is a box full of screwdrivers.

If you have a computer system made up of a bunch of modules interacting with each other, then you could have, potentially, quite a complex system indeed.

Look up the term “combinatorial explosion” to learn more.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 21 Jul 2024 21:53:46 -0000

> A problem that has been solved since 1983, and even before (Pascal had bounds checking). Sigh...

Pascal had no checking for memory leaks or double-frees.

Rust certainly seems to be a next-generation solution to these sorts of memory problems.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 21 Jul 2024 21:55:10 -0000

> Considering many many layers of software to predict timing from code in uncontrolled environment would be a challenge.

And yet it has been successfully done on the hardware itself, right down under all those layers of software (cf Spectre/Meltdown).

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 22 Jul 2024 08:36:08 +0200

> Pascal had no checking for memory leaks or double-frees.

> Rust certainly seems to be a next-generation solution to these sorts of memory problems.

We were talking about bounds checking, that Pascal had. Nowadays, you should not use pointers directly, but containers. Pointers are necessary only for writing containers, thanks to Ada's features not found in other languages, like allocating dynamically sized arrays on the stack.

Note that in Rust, containers are written using unsafe Rust, therefore Rust is not better than Ada on that aspect, it is a complicated solution to a problem that Ada doesn't have.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 22 Jul 2024 09:16:09 +0200

> If you have a computer system made up of a bunch of modules interacting with each other, then you could have, potentially, quite a complex system indeed.

Tight coupling = bad design. No difference to screwdrivers. However you can take integer arithmetic if you dislike screwdrivers. However complex system you build, there is no need to update integers.

> Look up the term “combinatorial explosion” to learn more.

Bad design leads to explosion of non-trivial unanticipated system states making it unpredictable. This is what happens when you add security on top. You patch holes drilling new ones to fix the patches.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Tue, 23 Jul 2024 01:48:12 -0000

> We were talking about bounds checking, that Pascal had.

Which is only one potential pitfall for bugs with security implications.

Ada and Other Languages

“Red” and the DoD Language Competition

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Subject: “Red” And The DoD Language Competition
Date: Fri, 6 Sep 2024 01:55:00 -0000
Newsgroups: comp.lang.ada

While browsing around for Ada-related docs some years ago, I came across this site:
https://iment.com/maida/computer/redref/index.htm
which collects info on the DoD’s “Strawman”, “Woodenman”, “Tinman”, “Ironman” and “Steelman” series of RFPs, and the specs for the “Red” language that didn’t become Ada.

From: Luke A. Guest <laguest@archeia.com>
Date: Sat, 7 Sep 2024 17:43:26 +0100

We have all the colours now:
https://www.reddit.com/r/ada/comments/165f5zg/common_hol_phase_1_reports/

Ada Practice

Accessing the Command Line

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Subject: Accessing The Command Line
Date: Thu, 4 Jul 2024 00:08:56 -0000
Newsgroups: comp.lang.ada

```
with Ada.Command_Line;
with Ada.Text_IO;
procedure Echo is
package cli renames
            Ada.Command_Line;
   package tio renames Ada.Text_IO;
   package int_io is new tio.Integer_IO
         (Num => Integer);
begin
   tio.put("my name: ");
   tio.put(cli.Command_name);
   tio.Put_Line("");
   tio.Put("nr args: ");
   int_io.Put(cli.Argument_Count, width => 1);
   tio.Put_Line("");
   for i in 1 .. cli.Argument_Count loop
      tio.put("[");
      int_io.put(i, width => 1);
      tio.put("]: ");
      tio.put(cli.argument(i));
      tio.put_line("");
   end loop;
end Echo;
```

Comments:

Ada, like Python, offers the convenience of being able to specify local “nicknames”

for imported packages, to save some typing.

Having become used to the convenience of printf-style formatting in C and other languages that have adopted it (including Lisp and Python), I don't miss the tedium of having to format and output one item at a time. Though I recognize that there is no way to do printf style in a type-safe fashion, short of going to a fully-dynamic language.

Being able to access the POSIX command line via some globally-defined entity instead of arguments to a "mainline" procedure is something that just about every decent language offers. C is notably absent from this list.

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 03 Jul 2024 18:16:01 -0700

> Though I recognize that there is no way to do printf style in a type-safe fashion, short of going to a fully-dynamic language.

C++ does that polymorphism stuff in iostream so you'd write std::cout << "my name: " << command_line << ...

Haskell does something sort of similar with additional machinery.

printf for some people is one of the motivations for dependent types.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 4 Jul 2024 01:50:59 -0000

> C++ does that polymorphism stuff in iostream so you'd write std::cout << "my name: " << command_line << ...

I know. The disadvantage of the C++ scheme is you cannot easily reorder items as necessary to fit the grammar of localized messages. That, I think, is why lots of other languages (including Python and Lisp) have copied the printf scheme, yet none have seen fit to copy the C++ scheme.

> printf for some people is one of the motivations for dependent types.

I wonder how you would express such a thing in an Ada-friendly fashion?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 4 Jul 2024 13:27:05 +0200

Remember that you can concatenate strings:

```
>      tio.put("my name: ");
>      tio.put(cli.Command_name);
>      tio.Put_Line("");
```

```
Tio.Put_Line (Item => "my name: " &
Cli.Command_Name);
```

Image functions thus allow similar simplifications. 'Image is one such function, if you can accept the initial space for non-negative values:

```
>      tio.Put("nr args: ");
>      int_io.Put(cli.Argument_Count,
   width => 1);
>      tio.Put_Line("");
```

```
Tio.Put_Line (Item => "nr args:" &
Cli.Argument_Count'Image);
```

For simple cases you can roll your own:

```
function Image (Value : in Integer) return
String is
   Raw : constant String := Value'Image;
begin -- Image
   return Raw ( (if Value < 0 then 1 else 2) ..
   Raw'Last);
end Image;
```

```
>         tio.put("[");
>         int_io.put(i, width => 1);
>         tio.put("]: ");
>         tio.put(cli.argument(i));
>         tio.put_line("");
```

```
Tio.Put_Line (Item => '[' & Image (I) & "]: " &
Image (Cli.Argument (I) ) );
```

For more complex uses, you can use something like PragmARC.Images[.Image]

(https://github.com/jrcarter/PragmARC).

You probably should review the definition of Ada.Text_IO

(http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-A-10.html), especially

```
for procedure New_Line.
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 4 Jul 2024 15:01:05 +0200

> with Ada.Command_Line;

> with Ada.Text_IO;

[...]

A general advice processing strings, any strings: messages, commands, payload etc.

Always read a complete string into a fixed size buffer (safety). Never use streams. Process the whole string consequently. Never tokenize. Never copy anything. Ada has slices.

The same is true for the output. Build a complete substring in a buffer. Consequently. Flush the complete substring to the output.

Do not use Unbounded_String.

From: Rod Kay <rodakay5@gmail.com>
Date: Fri, 5 Jul 2024 01:13:36 +1000

> I wonder how you would express such a thing in an Ada-friendly fashion?

There is the 'GNAT.Formatted_String' package, which provides 'printf' functionality.

Unfortunately, its formatting is somewhat buggy and has been so for many years. Usage is quite simple and reasonably elegant but the occasional incorrect formatting is a major problem, essentially rendering the package useless.

There is also the new 2022 f"X = {An_X_Variable} notation for embedding Variable'Image into strings, which is very nice. However, it does not allow for formatting, so not useful for your needs. Just thought I'd mention it, as it is now available in GCC 14.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 4 Jul 2024 18:15:54 +0200

>> printf for some people is one of the motivations for dependent types.

> I wonder how you would express such a thing in an Ada-friendly fashion?

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

From: Ben Bacarisse <ben@bsb.me.uk>
Date: Thu, 04 Jul 2024 20:42:00 +0100

> ... Though I recognize that there is no way to do printf style in a type-safe fashion, short of going to a fully-dynamic language.

Not so. Haskell has Text.Printf.

From: Paul Rubin <no.email@nospam.invalid>
Date: Thu, 04 Jul 2024 15:06:00 -0700

> Not so. Haskell has Text.Printf.

Text.Printf is not fully type safe. printf "%d\n" "foo" throws a "bad formatting character" exception, really amounting to a runtime type exception.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 4 Jul 2024 23:54:49 -0000

> Remember that you can concatenate strings:

> Tio.Put_Line (Item => "my name: " & Cli.Command_Name);

I'm sure I can, but I'm not sure what the point is. Let Ada collect the pieces in its own buffers. That saves copying steps.

> PragmARC.Images[.Image] (https://github.com/jrcarter/PragmARC).

I don't really feel the need to resort to third-party libraries just to do simple I/O.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 5 Jul 2024 10:58:00 +0200

> I'm sure I can, but I'm not sure what the point is. Let Ada collect the pieces in its own buffers. That saves copying steps.

Agreed. I don't understand why people dislike printing piece by piece. In the old FORTRAN, you could write only line by line, but this time is long gone…

With the various Put procedures, you have individual formatting options that you don't have otherwise. Moreover, there is a nice property that few people noticed: if you have an algorithm writing data to a file, with loops and so on, you can keep the exact same structure replacing every Put with the matching Get, and you will read your data correctly. This feature goes away as soon as you have a 'Image.

Reduction Expressions

From: Simon Wright <simon@pushface.org>
Subject: Reduction expressions
Date: Tue, 13 Aug 2024 13:36:54 +0100
Newsgroups: comp.lang.ada

Are the Accum_Type & Value_Type (ARM 4.5.10(9/5)) of a reduction attribute reference required to be definite?

ARM 4.5.10(24/5) & (25.5) seem to imply so, which explains why GNAT doesn't support e.g. String.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 19 Aug 2024 22:59:04 -0500

Accum_Subtype (we changed the name since it is a subtype, not a type; various clarifications were made to the wording as well in AI22-0011-1, AI22-0047-1, and AI22-0069-1) most likely has to be definite since the accumulator is of that type, and the bounds/constraints of the accumulator are thus defined by the initial value. In most uses, the first call on Reduce would then raise Constraint_Error (because the bounds/constraints are incorrect). I don't think there is any reason that the Value_Subtype has to be definite for a sequential reduce (a parallel reduce requires the two subtypes to statically match).

Note that if someone has a clever way to use an indefinite result, it is allowed. For instance, I could see a class-wide result making sense in some limited circumstances. But I don't think String would do anything useful, since the bounds are determined by the initial value.

BTW, this answer is essentially topic #1 of AI22-0011-1.

From: Simon Wright <simon@pushface.org>
Date: Tue, 20 Aug 2024 22:23:27 +0100

> Accum_Subtype (we changed the name since it is a subtype, not a type;

Amazing how a person (I) can have used Ada for ~40 years and still be hard put to it to describe the difference, at least in a case like this one, where the ARG members clearly see meanings that leave me lukewarm if not cold. Maybe "the heart of twilight"?

> But I don't think String would do anything useful

String was just the simplest indefinite type for an example.

> BTW, this answer is essentially topic #1 of AI22-0011-1.

Thanks for the pointer.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Tue, 20 Aug 2024 23:30:54 -0000

> Amazing how a person (I) can have used Ada for ~40 years and still be hard put to it to describe the difference

I thought the difference was obvious. “subtype” is the C equivalent of “typedef”, just giving a new name to an existing type. So

```
subtype A is B;
```

(where A and B are simple identifiers) is valid, whereas

```
type A is B;
```

is not: a “type” declaration always creates a new type: you have to write at least

```
type A is new B;
```

and now you have two types with different names that are structurally the same, but not compatible.

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Tue, 20 Aug 2024 16:41:55 -0700

> I thought the difference was obvious. “subtype” is the C equivalent of “typedef” [...]

A subtype with no added constraint is similar to a C typedef, but given

```
subtype Digit is Integer range 0..9;
```

Digit is distinct from Integer (though they're both the same type).

C doesn't have anything directly corresponding to Ada subtypes.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Wed, 21 Aug 2024 01:37:22 -0000

> Digit is distinct from Integer (though they're both the same type).

“Integer range 0..9” is a subtype of Integer, and is valid for example as a return type where Integer is expected. The “subtype” declaration doesn’t actually create the subtype: “Digit” is just a shorthand name for that, just like a C typedef.

From: Simon Wright <simon@pushface.org>
Date: Wed, 21 Aug 2024 08:47:49 +0100

> I thought the difference was obvious. [...]

Yes, I've understood that for a long time but ... ARM22 4.5.10(8,9)[1] say

(8) The expected type for a reduction_attribute_reference shall be a single nonlimited type.

(9) In the remainder of this subclause, we will refer to nonlimited subtypes Value_Type and Accum_Type of a reduction_attribute_reference. ...

and in AI 22-0011-1 [2] starting at 22-Oct-2021 5:25 PM,

* SB: raises a series of observations,

* STT: "... You really need to think of Accum_Type as a particular *subtype*"

* SB: "Ok, I was confused - Accum_Type is a subtype, not a type. So a lot of my message was noise."

If SB can be confused, so can I!

[1] http://www.ada-auth.org/standards/22rm/html/RM-4-5-10.html#p8

[2] http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ai22s/ai22-0011-1.txt?rev=1.2

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 Aug 2024 23:27:48 -0500

> If SB can be confused, so can I!

Which is why we changed the name - if SB can be confused, it is a good bet that there is something wrong with the wording. That's why I usually recommend bleeding edge users use the bleeding edge RM - no point in rediscovering all of the bugs that we already know about. Unfortunately, in this case, I'm the only one that has the bleeding edge RM because I haven't finished adding all of the approved AIs to it. This group is some that I've done, which is why the answer to your question was relatively easy to find.

Ichbiah 2022 Compiler Mode

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Subject: Ichbiah 2022 compiler mode
Date: Thu, 5 Sep 2024 11:52:37 -0000
Newsgroups: comp.lang.ada

I guess this is a very subjective question.

A number of Ada users have expressed that they would rather Ada was simpler whilst others desire more features.

I appreciate Ada 83 portability but also like a lot of modern Ada features.

Out of interest. Could anyone help me with what a GNAT or other compiler Ichbiah_2022_Mode might look like. Perhaps it might be possible to use pragmas to get an estimated mode of what features he might keep or drop.

I can continue research but currently I do not have the details of his objections to Ada 95 and how those may have continued through to today is perhaps a nuanced question.

What do you think Ichbiah would jettison from Ada 2022? All comments welcome.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 5 Sep 2024 15:40:35 +0200

> I do not have the details of his objections to Ada 95

Ichbiah's objections to Ada 95 are in https://web.elastic.org/~fche/mirrors/old-usenet/ada-with-null

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Thu, 5 Sep 2024 16:08:01 -0000

What does this mean?

"elimination of accuracy constraints in subtypes"

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 5 Sep 2024 21:24:05 +0200

> "elimination of accuracy constraints in subtypes"

See ARM-95 J.3 (https://www.adaic.org/resources/add_content/standards/95lrm/ARM_HTML/RM-J-3.html),

Reduced Accuracy Subtypes.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 5 Sep 2024 19:03:22 -0500

> What do you think Ichbiah would jettison from Ada 2022?

My recollection is that he wanted a more complex "class" feature, which IMHO would have made Ada more complex, not simpler.

In any case, I can't guess what Ichbiah would have suggested after 40 years of experience. (He probably would have moved on to some other language anyway, you have to be somewhat resistant to change to stick with a single language for your entire career. I seem to resemble that remark... ;-)

What I can do is suggest what an RLB_2022 mode would look like, as I did the exercise when we all were cooped up during the early days of the pandemic. My philosophy is that Ada has a lot of combinations of features that cause a lot of implementation trouble, but which are not very useful. So I want to reduce the combinations that cause trouble. I note that every feature is useful for something (else it wouldn't be in Ada in the first place). But some things are not useful enough for the trouble that they cause. Also note that I am not worrying about compatibility with Ada, which is always a problem when updating Ada itself.

Here's some highlights off the top of my head:

(1) Simplify the resolution model; essentially everything resolves like a subprogram. For instance, objects resolve similarly to enumeration literals. This substantially reduces the danger of use clauses (having matching profiles and names is less likely than just matching names), and eliminates the subtle differences between a constant and a function (they should really act the same).

(2) Operator functions have to be primitive for at least one of the types in the profile. (Operators in a generic formal part have a pseudo-primitive requirement.) That includes renamings. In exchange for that, operators have the same visibility as the type (which means they are always directly visible when any object of the type is visible). One then can eliminate "use type" (since it would literally do nothing).

(3) A number of syntax options are eliminated. Matching identifiers are required at the end of subprograms and packages. Initializers are always required (<> can be used if default initialization is needed). Keyword "variable" is needed to declare variables (we do not want the worst option to be the easiest to write, as it is in Ada).

(4) Anonymous types of all sorts are eliminated. For access types, we would use aspects to declare properties (static vs. dynamic accessibility, "closure" types, etc.). For arrays, see next item.

(5) The array model would be greatly simplified. New Ada users (and old ones as well) have a hard time dealing with the fact that the lower bound is not fixed in Ada. Additionally, the existing Ada model is very complex when private types are involved, with operators appearing long after a type is declared. The more complex the model, the more complex the compiler, and that means the more likely that errors occur in the compiler. There also is runtime overhead with these features. The basic idea would be to provide the features of an Ada.Containers.Vector, and no more. Very little is built-in. That means that arrays can only be indexed by integers, but that is a good thing: an array indexed by an enumeration type is really a map, and should use a map interface. So I would add a Discrete_Map to the Ada.Containers packages. Bounded_Arrays are a native type (most of the uses of arrays that I have are really bounded arrays built by hand).

A side-effect of this model change is to greatly simplify what can be written as discriminant-dependent components. Discriminant-dependent arrays as we know them are gone, replaced by a parameterized array object that has only one part that can change. Much of the nonsense associated with discriminant-dependent components disappears with this model.

(6) Static items have to be declared as such (with a "static" keyword rather than "constant"). Named numbers are replaced by explicit static constants. (I would allow writing Universal_Integer and Universal_Real, so one could declare static objects and operations of those types.)

(7) Types and packages have to be declared at library-level. This means that most generic instances also have to be declared at library-level. Subtypes, objects, and subprograms still can be declared at any nesting level. I make this restriction for the following reasons:

(A) Accessibility checks associated with access types are simplified to yes/no questions of library-level or not. The only cases where accessibility checks do any real good is when library-level data structures are constructed out of aliased objects. These would still be allowed, but almost all of the complication would be gone. Even if the check needs to be done dynamically, it is very cheap.

(B) Tagged types declared in nested scopes necessarily require complex dynamic accessibility checks to avoid use of dangling types (that is, an object which exists of a type that does not exist).

(C) Reusability pretty much requires ODTs to be declared in library-level packages. Mandating that won't change much for most programs, and you'll be happier in the long run if you declare the types in library packages in the first place.

(D) There are a lot of semantic complications that occur from allowing packages in subprograms, but this is rarely a useful construct.

(8) Protected types become protected records (that is, a regular record type with the keyword "protected"). Primitive operations of a protected record type are those that are protected actions. (Entries can be declared and renamed as such, they would no longer match procedures, which leads to all kinds of nonsense.) This would eliminate the problems declaring helper types and especially *hiding* helper types for protected types. (See the problems we had defining the queues in the Ada.Containers to see the problem.) The protected operations would allow the keyword "protected" in order to make the subprograms involved explicit.

(9) Strings are not arrays! Strings would be provided by dedicated packages, supporting a variety of representations. There would be a Root_String'Class that encompasses all string types. (So many

operations could be defined on Root_String'Class).

(10) Variable-returning functions are introduced. They're pretty similar the semantics of anonymous access returns (or the aliased function returns suggested by Tucker). This means that a variable can easily be treated as a function (and indeed, a variable declaration is just syntactic sugar for such a function).

(11) Various obsolete features like representation_clauses, representation pragmas, and the ability to use 'Class on untagged private types are eliminated or restricted.

There were a couple of areas that I never made up my mind on:

(A) Do we need tasks at all? Parallel and task are very much overlapping capabilities. But the parallel model would need substantial changes if we were to allow suspension of parallel threads (Ada 2022 does not allow this). Suspension seems necessary to support intermittent inputs of any type (including interrupts) without wasting resources running busy-wait loops.

(B) Should type conversions be operators or remain as the type name as in Ada? A type conversion operator, automatically constructed, would allow user-defined types to have the same sort of conversions to numeric and string types that the predefined do. But an operator would make conversions easier, which is probably the wrong direction for a strongly typed language.

(C) I wanted to simplify the assignment model, but my initial attempt did not work semantically. I'm not sure that simplification is possible with the Ada feature set (I'm sure Bob and Tuck tried to do that when creating Ada 95, but they failed). The main issue is that one would like to be able to replace discriminant checks on user-defined assignment. (Imagine the capacity checks on a bounded vector; Ada requires these to match, but that's way too strong; the only problem is if the target capacity cannot hold the actual length of the source object. A user-defined replacement would be helpful.)

My $20 worth (this was a lot more work than $0.02!!). I probably forgot a number of items; my actual document is about 20 pages long.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Fri, 6 Sep 2024 00:58:05 -0000

> Keyword "variable" is needed to declare variables

One language idea I toyed with years ago was that

```
«name» : «type»;
```

declared a variable, while

```
«name» : «type» := «value»;
```

declared a constant. So, no initialization of variables at declaration time allowed.

> (10) Variable-returning functions are introduced.

Is this like updater functions in POP-11, or “setf” in Lisp? So you have a procedure

```
set_var(«var», «new value»)
```

which is declared to be attached to «var» in some way, such that when you write

```
«var» := «new_value»
```

this automatically invokes set_var?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Fri, 6 Sep 2024 13:07:27 +0200

> Out of interest. Could anyone help me with what a GNAT or other compiler Ichbiah_2022_Mode might look like.

I have no idea what he would have done. For an idea of what I think a language should have, you can look at my informal description of King (https://github.com/jrcarter/King).

From: Simon Wright <simon@pushface.org>
Date: Fri, 06 Sep 2024 22:22:08 +0100

> (A) Do we need tasks at all? Parallel and task are very much overlapping capabilities.

I don't think I've ever wanted parallel. Most embedded system tasks are one-off, aren't they?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 7 Sep 2024 20:13:03 +0300

> I don't think I've ever wanted parallel. Most embedded system tasks are one-off, aren't they?

More and more embedded systems use multi-core processors and do heavy, parallelizable computations. "Parallel" is intended to support that in a light-weight way. In a recent discussion with the European Space Agency, they expressed interest in using OpenMP for such computations on-board spacecraft with multi-core processors, which is an "embedded" context.

Regarding tasks in embedded systems, I agree that most are one-off, but I have occasionally also used tens of tasks of the same task type.

I disagree with Randy's view that tasks and "parallel" are much overlapping. Tasks are able to communicate with each other, but AIUI parallel tasklets are not meant to do that, and may not be able to do that. Tasks can have different priorities; tasklets cannot.

From: Nioclás Pól Caileán De Ghloucester <master_fontaine_is_dishonest @strand_in_london.gov.uk>
Date: Sat, 7 Sep 2024 22:34:52 +0200

"[...] they expressed interest in using OpenMP for such computations [...]"

Hei!

Most of the languages which are referred to by WWW.OpenMP.org/resources/openmp-compilers-tools facilitate bugs. (The Spark which is referred to thereon is not the Ada-related Spark language.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 11 Sep 2024 23:39:27 -0500

>> (10) Variable-returning functions are introduced.

> this automatically invokes set_var?

No, it is a function that returns a variable, meaning you can assign into the function result. If you have:

```
function Foo return variable Integer;
```

then you can use Foo on either side of an assignment:

```
Foo := 1;

Bar := Foo + 1;
```

Essentially, this idea treats:

```
Var : variable Integer;
```

as syntactic sugar for

```
function Var return variable Integer;
```

The worth of that is two-fold: (1) Objects and functions now resolve the same; (2) one can write a function that acts exactly like an object, and thus can replace it in all uses.

Note that Ada currently has generalized reference objects and functions that return anonymous access types, and both of these act similarly to a variable returning function. But neither is quite a perfect match.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Thu, 12 Sep 2024 22:24:29 -0000

> No, it is a function that returns a variable, meaning you can assign into the function result.

I think an updater function would be more generally useful. Because some things you want to update might not (depending on the implementation) live independently in an explicit variable. And it seems good not to constrain implementations unnecessarily.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 14 Sep 2024 01:18:25 -0500

> I think an updater function would be more generally useful.

Unfortunately, "updater" functions don't work with the Ada model of components, because you can't tell what to do when a component appears or disappears in an assignment. (That's why Ada doesn't allow overloading ":=".) And composition is very important to Ada -- stand-alone objects are pretty rare outside of those for scalar types. I don't think something that only worked with stand-alone objects would be very useful (can't use those with ODTs, for instance).

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sat, 14 Sep 2024 07:18:29 -0000

> Unfortunately, "updater" functions don't work with the Ada model of components [...]

But it's just syntactic sugar, nothing more. Instead of

```
a := obj.get_prop()

obj.set_prop(a)
```

(both of which have valid Ada equivalents), you can unify them into

```
a:= obj.prop

obj.prop := a
```

What difference does writing it differently make?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 11 Sep 2024 23:46:18 -0500

> Tasks are able to communicate with each other, but AIUI parallel tasklets are not meant to do that, and may not be able to do that. Tasks can have different priorities; tasklets cannot.

I was (of course) presuming that "tasklets" would get those capabilities if they were to replace tasks. That's what I meant about "suspension", which is not currently allowed for threads in Ada (parallel code is not allowed to call potentially blocking operations). If that was changed, then all forms of existing task communication would be allowed.

I'm less certain about the value of priorities, most of the time, they don't help writing correct Ada code. (You still need all of the protections against race conditions and the like.) I do realize that they are a natural way to express constraints on a program. So I admit I don't know in this area, in particular if there are things that priorities are truly required for.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 12 Sep 2024 10:42:38 +0300

> I was (of course) presuming that "tasklets" would get those capabilities

Ok, I understand. In that case, what "parallel" adds to the current tasking feature is an easy way to create a largish and perhaps dynamically defined number of concurrent threads from a "parallel" loop, where the threads are automatically created when the loop is started and automatically "joined" and destroyed when the loop completes.

I don't mind at all if a future Ada evolution merges tasks and "parallel", although it might defeat the easier access to multi-core true parallelism that is the goal of the "parallel" extension, AIUI.

> I'm less certain about the value of priorities

Priorities (or the equivalent, such as deadlines) are absolutely required for real-time systems where there are fewer cores than concurrent/parallel activities so that the system has to schedule more than one such activity on one core.

If Ada did not have tasks with priorities, most of the Ada applications I have worked on in my life would have had to avoid Ada tasking and retreat to using some other real-time kernel, with ad-hoc mapping of the kernels's threads to Ada procedures.

Despite the transition to multi-core processors, I think that there will continue to be systems where scheduling is required, because the number of concurrent/parallel activities will increase too.

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 12 Sep 2024 11:04:58 +0200

> I was (of course) presuming that "tasklets" would get those capabilities [...]

Well, tasks are not only for speeding up code. They can be a very useful design tool (active objects, independent activities). I think the Ada model is clean and simple, I would hate to see it disappear.

> I'm less certain about the value of priorities [...]

If you had as many cores as tasks, you would not need priorities. Priorities are just optimization on how to manage cores when there are not enough of them.

I know that people use priorities to guarantee mutual exclusion, and other properties. All these algorithms were designed at the time of mono-CPU machines, but they fail on multi-cores. Nowadays, relying on priorities for anything other than optimization is bad - and dangerous- design.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 12 Sep 2024 11:07:01 +0200

> I don't mind at all if a future Ada evolution merges tasks and "parallel" [...]

To me usefulness of "parallel" is yet to be seen, while tasks proved to be immensely useful on all architectures available.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 12 Sep 2024 14:35:27 +0300

> Well, tasks are not only for speeding up code. They can be a very useful design tool (active objects, independant activities). I think the Ada model is clean and simple, I would hate to see it disappear.

I agree.

> Priorities are just optimization on how to manage cores when there are not enough of them.

In some contexts it could be optimization -- for example, to increase throughput in a soft real-time system -- but in hard real-time systems priorities (or deadlines) are needed for correctness, not just for optimization.

> I know that people use priorities to guarantee mutual exclusion, and other properties. All these algorithms were designed at the time of mono-CPU machines, but they fail on multi-cores.

In SW for multi-core systems it can be beneficial to collect tasks that frequently interact with each other or with the same single-user resources in the same core, and then the mono-core mutual-exclusion algorithms like priority ceiling inheritance can be used for that group of tasks, while using other algorithms for mutual exclusion between tasks running in different cores.

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Thu, 12 Sep 2024 12:36:19 -0000

>If Ada did not have tasks with priorities, most of the Ada applications I have worked on in my life would have had to avoid Ada tasking and retreat to using some other real-time kernel, with ad-hoc mapping of the kernels's threads to Ada procedures.

Counter intuitively it is possible that this is holding Ada back. A lot of Ada code cannot run without some fairly complex runtime support due to tasks, protected objects, finalization etc.. Runtimes have to be developed for each chip instead of each cpu. At Least I assume that that is why these features are not available to e.g. the light cortex-m33 or cortex-m4 or cortex-m0+ runtimes. This requires rewriting code which isn't required with equivalent C code such as containers and ip stacks etc.. Even support for the Ada interrupt package is missing but it looks like porting that support to chips is less work and research.

If you need advanced multi core support then using an OS seems like a more suitable situation to be in to me.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 12 Sep 2024 18:43:45 +0300

> Counter intuitively it is possible that this is holding Ada back. [...]

True, however an Ada RTS can implement many of the tasking features with moderate effort on top of non-Ada real-time kernels such as FreeRTOS, VxWorks, etc., as AdaCore have done for some kernels. At least for the Ravenscar and Jorvik profiles. AIUI, the processor-specific stuff is then mainly in the kernel, not in the RTS.

> If you need advanced multi core support then using an OS seems like a more suitable situation to be in to me.

Using a large OS like Linux would not be acceptable for many embedded systems. Fortunately the smaller real-time kernels are adding multi-core support too.

The great advantage of using the standard Ada tasking feature, special syntax and all, is that your embedded Ada program can then be executed on a PC or other non-embedded computer, for testing or other purposes, tasking and all. It can also be analysed by static-analysis tools such as AdaControl for race conditions and other tasking-sensitive issues.

From: Nioclás Pól Caileán De Ghloucester <master_fontaine_is_dishonest @strand_in_london.gov.uk>
Date: Fri, 13 Sep 2024 22:45:03 +0200

> Counter intuitively it is possible that this is holding Ada back [...]

A book by Burns and Wellings unsensibly boasts that the demanding runtime demands of Ada are an advantage because if you are with them then you are with them, whereas as Kevin Chadwick points out - they are not easy to make.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 14 Sep 2024 01:13:28 -0500

> [...] in hard real-time systems priorities (or deadlines) are needed for correctness, not just for optimization.

This I don't buy: priorities never help for correctness. At least not without extensive static analysis, but if you can do that, you almost certainly can do the correctness without depending upon priorities.

I view priorities as similar to floating point accuracy: most people use them and get the results they want, but the reason for that is that they got lucky, and not because of anything intrinsic. Unless you do a lot of detailed analysis, you don't know if priorities really are helping or not (and similarly, whether your results actually are meaningful in the case of floating point).

Anyway, I don't see any such changes coming to Ada, but rather to some separate follow-on language (which necessarily needs to be simpler), and thus some things that are sometimes useful would get dropped.

(Different message)

...

> [...] what "parallel" adds to the current tasking feature is an easy way to create a largish and perhaps dynamically defined number of concurrent threads from a "parallel" loop [...]

I think the parallel block is more useful for general tasking. The advantage of using parallel structures is that they look very similar to sequential structures, and one lets the system do the scheduling (rather than trying to figure out an organization manually).

One of the advantages of the model I'm thinking about is that it separates concerns such as parallel execution, mutual exclusion, inheritance, organization (privacy, type grouping), and so on into separate (mostly) non-overlapping constructs. Ada started this process by having tagged types a separate construct from packages; you need both to get traditional OOP, but you can also construct many structures that are quite hard in traditional "one construct" OOP. I think that ought to be done for all constructs, and thus the special task and protected constructs ought to go. We already know that protected types cause problems with privacy of implementation and with inheritance. Tasks have similar issues (admittedly less encountered), so splitting them into a set of constructs would fit the model.

In any case, this is still a thought experiment at this time, whether anything ever comes of it is unknown.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 14 Sep 2024 08:47:49 +0200

> The advantage of using parallel structures is that they look very similar to sequential structures, and one lets the system do the scheduling (rather than trying to figure out an organization manually).

Tasking is not about scheduling. It is about program logic expressed in a sequential form. It is about software decomposition. Parallel constructs simply do not do that.

> One of the advantages of the model I'm thinking about is that it separates concerns such as parallel execution, mutual exclusion, inheritance, organization (privacy, type grouping), and so on into separate (mostly) non-overlapping constructs.

To me it is exactly *one* construct: inheritance. You should be able to inherit from an abstract protected interface at any point of type hierarchy in order to add mutual exclusion:

```
type Protected_Integer is new Integer and
   Protected;
```

> Ada started this process by having tagged types a separate construct from packages;

I see modules and types as unrelated things.

> you need both to get traditional OOP, but you can also construct many structures that are quite hard in traditional "one construct" OOP. I think that ought to be done for all constructs, and thus the special task and protected constructs ought to go.

Constructs yes, they must go. It must be all inheritance. The concepts must stay.

> We already know that protected types cause problems with privacy of implementation and with inheritance. Tasks have similar issues (admittedly less encountered) [...]

The problems are of syntactic nature, IMO.

There is an issue with an incomplete inheritance model. You need not just complete overriding but also more fine mechanisms like extension in order to deal with entry point implementations. The same problem is with constructors and destructors, BTW. What should really go is Ada.Finalization mess replaced by a sane user construction hooks model for all types, class-wide ones included.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sat, 14 Sep 2024 07:19:47 -0000

> ... priorities never help for correctness.

Concurrent programming was never about correctness, it was about efficiency/performance (throughput, latency, whatever is appropriate). And priorities are just another part of this.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 14 Sep 2024 11:12:43 +0300

> This I don't buy: priorities never help for correctness. At least not without extensive static analysis, but if you can do that, you almost certainly can do the correctness without depending upon priorities.

You misunderstood me; perhaps I was too brief.

I said "hard real-time systems", which means that the program is correct only if it meets its deadlines, for which priorities or deadline-based scheduling are necessary if there are fewer cores than concurrent/parallel activities, and the application has a wide range of deadlines and activity execution times.

(To be honest, there is the alternative of using a single thread that is manually sliced into small bits, interleaving all the activities increment by increment, according to a static, cyclic schedule, but that is IMO a horribly cumbersome and

unmaintainable design, though unfortunately still required in some contexts.)

I believe we agree that priorities should be used for other things, such as controlling access to shared data, only if there is a well-defined and safe mechanism for it, such as protected objects with priority ceilings and priority inheritance on a single core.