# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editorial Policy for <em>Ada User Journal</em></td>
<td>74</td>
</tr>
<tr>
<td>Editorial</td>
<td>75</td>
</tr>
<tr>
<td>Quarterly News Digest</td>
<td>77</td>
</tr>
<tr>
<td>Conference Calendar</td>
<td>91</td>
</tr>
<tr>
<td>Forthcoming Events</td>
<td>97</td>
</tr>
<tr>
<td>Ada-Europe 2019 Industrial Presentations</td>
<td></td>
</tr>
<tr>
<td>M. Martignano</td>
<td></td>
</tr>
<tr>
<td>“A “New” C Static Analyzer: the Compiler”</td>
<td>99</td>
</tr>
<tr>
<td>T. A. Beyene, C. Herrera, V. Nigam</td>
<td></td>
</tr>
<tr>
<td>“Verification of Ada Programs with AdaHorn”</td>
<td>103</td>
</tr>
<tr>
<td>Special Contribution</td>
<td></td>
</tr>
<tr>
<td>A. Burns, B. Dobbing, T. Vardanega</td>
<td></td>
</tr>
<tr>
<td>“Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems (Part 2)”</td>
<td>110</td>
</tr>
<tr>
<td>Ada-Europe Associate Members (National Ada Organizations)</td>
<td>128</td>
</tr>
<tr>
<td>Ada-Europe Sponsors</td>
<td>Inside Back Cover</td>
</tr>
</tbody>
</table>
Quarterly News Digest

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

Contents

Ada-related Events 77
Ada-related Resources 80
Ada-related Tools 80
Ada Inside 83
Ada and other Languages 84
Ada Practice 86
Ada in Jest 89

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

FOSDEM 2019 post hoc summary

From: dirk@orka.cs.kuleuven.be
(Dirk Craeynest)
Date: Mon, 25 Feb 2019 07:04:32 -0000
Subject: FOSDEM 2019 Ada Developer Room - presentations & videos online
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

---------------------------------------------------
** Presentations, videos, pictures available online **
- 9th Ada Developer Room at FOSDEM 2019
- Saturday 2 February 2019
- Université Libre de Bruxelles (ULB), Solbosch Campus, Room AW1.125
- Avenue Franklin D. Roosevelt Laan 50, B-1050 Brussels, Belgium
- Organized in cooperation with Ada-Europe

Call for Papers

DeCPS workshop in Warsaw

From: dirk@orka.cs.kuleuven.be.
(Dirk Craeynest)
Date: Wed, 3 Apr 2019 22:19:24 -0000
Subject: DeCPS 2019 - Dependable and Cyber-Physical Systems Engineering
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

--- Scope ---
In recent years, the Internet of Things (IoT) has experienced an extraordinary development with a broad impact on society; however, there is still a gap between the physical world and the cyber one. Cyber Physical Systems (CPS) constitute a new class of engineered systems, integrating software control and autonomous decision making with signals from an uncertain and dynamic environment. Internet transformed the way people interact with engineered components, but the way people interact and deal with engineered systems is still an open question. Internet transformed the way people interact and deal with engineered components, but the way people interact and deal with engineered systems is still an open question.
systems. For this type of systems, it is necessary not only ensuring the safety of physical devices but also other factors such as information about customers, suppliers, and organizational strategies need to be secured. In the context of cyber systems, the Artificial Intelligence (AI) technologies can contribute to manage a huge amount of heterogeneous data that come from different sources without human intervention. To deliver certification, standards for machine safety are highly recommended as they give confidence to the regulatory. The generic standard for safety-related hardware and software might be applicable, however, due to increasing autonomy of robots there is still a potential for evolution of such regulations or standards. The proper combination of AI, CPS and IoT is therefore fundamental.

CPS are considered a disruptive technology which will transform the traditional manufacturing into Industry 4.0 solutions, and are used in a very wide spectrum of applications: smart mobility, autonomous driving, digital healthcare, smart grids and buildings, mobile co-operating autonomous robotic systems, digital consumer products and services. "In conclusion, the emerging Digital (R)-evolution relies heavily on Embedded Intelligent Systems technologies in domains where it is paramount that Europe takes leadership role" (Laila Gide, "The pathway to digital transformation: an opportunity for Europe", ARTEMIS Magazine 20 May 2016).

This workshop aims to provide a platform to industrial practitioners, researchers and engineers in academia to exchange of their ideas, research results, experiences in the field of dependable and cyber physical systems engineering, both a theoretical and practical perspective. To foster visibility and interaction, participation in the workshop will be also open to conference participants (at no extra cost).

--- Topics of interest ---

The topics of interest includes, but are not limited to:

* Vehicle of the Future
* Transport and Mobility
* Industry 4.0 in transportation sector
* Security and comfort of the end-user
* Human/Machine Interaction
* Safety and Security
* Industrial experiments and case studies
* Integration of Internet of Things and Cloud Computing
* Evolution of standards and certification processes
* Impact of Artificial Intelligence in CPS

The workshop will also include contributions from relevant projects in the domain, such as Future Factories in the Cloud (FiC), Productive 4.0, AMASS, ENABLE-S3, SafeCOP, SCOTT, etc.

--- Paper submission ---

Submission of regular papers (4 pages, AUJ style) at the following page: https://easychair.org/conferences/?conf=decps2019

The post-workshop proceedings will be published in the Ada User Journal (http://www.ada-europe.org/auj/guide/).

--- Important dates ---

* Submission deadline: 30 April 2019
* Notification to authors: 17 May 2019
* Workshop: 14 June 2019
* After-workshop final version: 15 September 2019
* Publication in Ada User Journal: December 2019

--- Steering Committee ---

* Daniela Cancila, CEA LIST, France
* Martin Tomgren, KTH Royal Institute of Technology, Sweden
* Alessandra Bagnato, SOFTEAM, France
* Cristina De Luca, Infineon Technologies Austria AG Austria
* Silvia Mazzini, INTECS Italy
* Laurent Rioux, Thales, France
* Barbara Gallina, Mälardalen Univ., Sweden
* Luis Miguel Pinho, Polytechnic Institute of Porto, Portugal

Dirk.Craeynest@cs.kuleuven.be, Ada-Europe 2019 Publicity Chair

Ada-Belgium Spring 2019 Event

From: dirk@orka.cs.kuleuven.be.
(Dirk Craeynest)
Date: Sun, 5 May 2019 19:35:53 -0000
Subject: Ada-Belgium Spring 2019 Event, Sun 12 May 2019
Newsgroups: comp.lang.ada, fr.comp.lang.ada, be.comp.programming

Ada-Belgium Spring 2019 Event
Sunday, May 12, 2019, 12:00-19:00
Wavre area, south of Brussels, Belgium
including at 15:00
2019 Ada-Belgium General Assembly
and at 16:00
Ada Round-Table Discussion

---

*** Announcement

The next Ada-Belgium event will take place on Sunday, May 12, 2019 in the Wavre area, south of Brussels. For the 12th year in a row, Ada-Belgium organizes their "Spring Event", which starts at noon, runs until 7pm, and includes an informal lunch, the 26th General Assembly of the organization, and a round-table discussion on Ada-related topics the participants would like to bring up.

*** Schedule

* 12:00 welcome and getting started (please be there!)
* 12:15 informal lunch
* 15:00 Ada-Belgium General Assembly
* 16:00 Ada round-table + informal discussions
* 19:00 end

*** Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Thursday, May 9, 21:00, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2019 Event", so you can get precise directions to the place of the meeting. Even if you already responded to the preliminary announcement, please reconfirm your participation ASAP.

If you are a member but have not renewed your affiliation yet, please do so by paying the appropriate fee before the General Assembly (you have also received a printed request via normal mail). If you are interested to join Ada-Belgium, please register by filling out the 2019 membership application form [1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2019 with all member benefits [2]. Early enrollment ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events, as well as some back issues of the Ada User Journal [3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members. (Please indicate in the above-mentioned registration e-mail that you’re interested, so we can bring enough copies.)

Looking forward to meet many of you!

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the lunch are also welcome; they can choose to join the organization or pay the sum of 15 Euros per person to the Treasurer of the organization.

**General Assembly**

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board [4]. See the separate official convocation [5] for all details.


**Ada Round-Table Discussion**

As in recent years, we plan to keep the technical part of the Spring event informal as well. We will have a round-table discussion on Ada-related topics the participants would like to bring up. We invite everyone to briefly mention how they are using Ada in their work or non-work environment, and/or what kind of Ada-related activities they would like to embark on. We hope this might spark some concrete ideas for new activities and collaborations.

**Directions**

To permit this more interactive and social format, the event takes place at private premises in the Wavre area, south of Brussels. As instructed above, please inform us by e-mail if you would like to attend, and we’ll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you!

Dirk Craeynest, President Ada-Belgium
Dirk.Craeynest@cs.kuleuven.be

---

Acknowledgements

We would like to thank our sponsors for their continued support of our activities: AdaCore, and KU Leuven (University of Leuven).

If you would also like to support Ada-Belgium, find out about the extra Ada-Belgium sponsorship benefits:


---

Ada-Core

Ada-Europe 2019

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Date: Thu, 9 May 2019 05:47:27 -0000
Subject: 24th Int. Conf. Reliable Software Technologies, Ada-Europe 2019
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Call for Participation

*** PROGRAM SUMMARY ***

24th International Conference on Reliable Software Technologies - Ada-Europe 2019
11-14 June 2019, Warsaw, Poland

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

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

*** Online registration open ***

*** Early registration discount until May 20 ***

*** Extensive info available on conference web site ***

*** Highly recommended to book your hotel ASAP ***

The 24th International Conference on Reliable Software Technologies - Ada-Europe 2019 visits Poland, for the first time, and is hosted in Warsaw from the 11th to the 14th of June. The conference is the latest in a series of annual international conferences started in the early 80’s, under the auspices of Ada-Europe, the international organization that promotes knowledge and use of Ada and Reliable Software in general, into academic education and research, and industrial practice.

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

Extensive information is on the conference web site, such as an overview of the program, the list of accepted papers and industrial presentations, and descriptions of workshops, tutorials, keynote presentations, and social events. Also check the conference site for registration, accommodation and travel information. The 12-page Advance Program brochure is available there as well.

---

The 2019 edition of the conference features a number of important innovations:

- lower registration fee for conference, unified for all participants;
- further reduced fee for all authors;
- lower registration fee for all tutorials;
- journal-based open-access publication model for peer-reviewed papers;
- an educational tutorial offered especially for those new to Ada;
- more compact program with two core days (Wed & Thu); tutorials on Tuesday, then exhibition opening mid-afternoon, followed by welcome aperitif for all participants;
- full-day DeCPS workshop on Friday (complementary with registration).

Quick overview

- Tue 11: tutorials, opening exhibition + AE GA, welcome reception
- Wed 12 & Thu 13: core program
- Fri 14: workshop

Proceedings

- peer-reviewed papers in open-access journal
- industrial presentation and tutorial abstracts in Ada User Journal

Conference & Program Chair
- Tullio Vardanega, University of Padua, Italy tullio.vardanega at unipd.it

Keynote speakers
- Tucker Taft, AdaCore, USA, "A 2020 View of Ada"
- other keynote to be confirmed (see conference web site)

Workshop (full day)
- 6th International Workshop on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" (DeCPS 2019)

Tutorials (full day)
- "Controlling I/O Devices with Ada, using the Remote I/O Protocol" Philip Munts, Sweden
- "An Introduction to Ada" Jean-Pierre Rosen, Adalog, France

Papers and Presentations

- 9 refereed technical papers
- 8 industrial presentations and experience reports
- a speaker's corner on "Experience from 40 years of teaching Ada"
Vendor exhibition and networking area
- area features exhibitor booths, project posters, reserved vendor tables, and general networking options
- 4 companies already committed:
  AdaCore, PTC Developer Tools, Rapita Systems, Vector; some exhibition slots still available
- vendor presentation sessions in core program

Social events
- each day: coffee breaks in the exhibition space and sit-down lunches offer ample time for interaction and networking
- Tuesday afternoon: opening of exhibition & Ada-Europe General Assembly, Welcome Aperitif on terrace overlooking Warsaw Airport
- Wednesday evening: transportation to restaurant in town where Chopin was born, banquet with Polish cuisine, drinks, and live piano music
- Best Paper and Best Presentation awards will be handed out

Registration
- online registration is open at <https://registration.ada-europe.org/index.html>
- early registration discount until Monday May 20, 2019
- special low fee for authors
- discount for Ada-Europe, ACM SIGAda, SIGBED and SIGPLAN members
- extra discount for students
- registration includes coffee breaks and lunches
- full conference registration includes all social events
- tutorial fees substantially reduced
- payment possible by credit card or bank transfer
- see registration page for all details

Promotion
- recommended Twitter hashtags:
  #AdaEurope and/or #AdaEurope2019
- 12-page Advance Program brochure online at http://www.ada-europe.org/conference2019/AD2019%20AP.pdf

We look forward to seeing you in Warsaw in June 2019!

Our apologies if you receive multiple copies of this announcement. Please circulate widely.

Dirk Craeynest, Ada-Europe’2019 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

*** 24th Intl. Conf. on Reliable Software Technologies - Ada-Europe’2019
June 11-14, 2019 * Warsaw, Poland *
www.ada-europe.org/conference2019

Ada-related Resources

Ada on Social Media

From: Alejandro R. Mosteо
<amosteo@unizar.es>
Date: Thu May 23 2019
Subject: Ada on Social Media

On March 12, 2019, Maxim Reznik created an English-language Telegram chat group, called "Ada", with description "Ada Programming Language and related technologies". It can be joined at https://t.me/ada_lang

On other front, the Google+ Ada Community seems to no longer exist.
Ada groups on various social media:
- LinkedIn: 2,813 (+101) members [1]
- Reddit: 2,243 (+343) members [2]
- StackOverflow: 1,183 (+183) watchers [3]
- Freenode: 87 (-17) users [4]
- Gitter: 42 (-15) people [5]
- Telegram: 47 (new!) users [6]
- Twitter: 61 (-2) tweeters [7]

[1] https://www.linkedin.com/groups/114211/
[4] #Ada on irc.freenode.net
[6] https://t.me/ada_lang

Language popularity rankings

From: Alejandro R. Mosteо
<amosteo@unizar.es>
Date: Thu May 23 2019
Subject: Ada in language popularity rankings

- TIOBE Index: 36 (0.326%) [1]

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

ADA-related Tools

Debugging Ada programs

From: "Randy Brukardt"
<randy@rrsoftware.com>
Date: Tue, 2 Apr 2019 17:07:24 -0500
Subject: Re: Intervention needed?
Newsgroups: comp.lang.ada

Does anyone spend much time in a debugger when writing Ada? Almost all of the time I do it is to track down compiler bugs (hopefully something that the average Ada user doesn't do often).
With the default exception information, there is little need to debug anything the majority of the time.
Certainly, moving detection to compile-time is even better. But I don't see that changing the mostly non-existent use of debuggers much.
I am using tracing, but there are a few cases where debugger could be easier to use. In the debugger you could inspect the states of variables and of other tasks. And you don't need to modify the code. It is quite often that I have to add, in addition to "standard" tracing, some more extensive tracing which I remove later.

In none of my projects GDB works. I never tried to figure out if that is related to the number of compilation units or number of library projects involved.

When you click Debug->Initialize->your-main-program in GPS and debugger does not start you know you reached the point.

In none of my projects GDB works.

> P.S. It never worked reliable in GPS and I bet it never will.

This is very troubling. I understand the sentiment here that Ada is so good in error prevention that debuggers are not needed at all, but what I find in projects I'm related with is that debuggers are not used for debugging anyway.

The major use for debuggers that I see is in integration testing, where test procedures expect particular values in particular variables (or even exact memory locations) in particular circumstances. The test is successful if such expectations are confirmed. Even for a presumably 100% correct program such a test has to be done if foreseen by project plans.

So, we have another paradox: Ada is so good in error prevention that the community does not care about having a proper debugger, and then the lack of working debugger prevents people from choosing Ada for projects that have rigorous integration testing culture. Part of the paradox is that such projects happen to be safety-critical, where Ada is supposed to be the preferred solution. And then they use C, where debuggers work like a charm.

Again: debuggers are not only for debugging and you better get them working right (by, well... debugging them?).

This is a wider problem of traceability. You have to solve this problem anyway for the coverage analysis, for example. And the solution, whatever you happen to use (like tool-readable labels in source code) will help with debugging, too.

In any case, yes, some projects need the debugger to test individual memory locations. The lack of proper tools is a problem, too.

> How big is "moderate"?

> especially if the scripts should be robust (higher-level) interface.

> with anything else in software, recurring problems can be mitigated by additional code. That is, if testing this way is difficult, then the difficulty is similar for the whole class of similar tests and as such that difficulty can be refractored away to additional utility (library/framework/etc.) with simpler (higher-level) interface.

> not start you know you reached the point.

> from .

> > I don't. I can't remember when I last used gdb or any other debugger, and in my ~30 years of Ada use I estimate that I have used a debugger on perhaps ten occasions.

I can trump that.

I have *never* used a "debugger" in much the same time with Ada.

~30 years ago I raced an experienced programmer who was looking for an error in his code with the DEC Ada debugger, while I inspected his compilation listing. I won.
- Minimum supported version of Visual Studio is now 2017 Update 6 (15.0.27413).
  Please feel free to download the free plugin from the following URL: https://marketplace.visualstudio.com/items?itemName=AlexGamper.VisualAda

**Gnu Emacs Ada mode**

*From: Stephen Leake*

*Date: Sat, 23 Mar 2019 10:25:34 -0700*

*Subject: Gnu Emacs Ada mode 6.1.0 released.*

*Newsgroups: comp.lang.ada*

Gnu Emacs Ada mode 6.1.0 is now available in GNU ELPA. This is a medium feature release; partial file parsing is now supported when using the process parser, and error correction is improved. This means the time spent parsing is independent of the file size, so it is fast enough even on the largest files. The process parser requires a manual compile step, after the normal list-packages installation:

```
  cd ~/.emacs.d/elpa/ada-mode-6.1.0
  /build.sh
```

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

**AdaSubst**

*From: "J-P. Rosen" <rosen@adalog.fr>*

*Date: Fri, 19 Apr 2019 08:47:50 +0200*

*Subject: [Ann] Adasubst 1.6r5 released.*

*Newsgroups: comp.lang.ada*

Adalog is pleased to announce the release of a new version of AdaSubst.

This releases adds a new function: Instantiate. It replaces all generic instantiations with equivalent, explicit code. This is useful if your coding standard disallows generics on the ground that it is "hidden code", or if you use a validation or testing tool that does not handle generics properly.

Adasubst can be downloaded from http://www.adalog.fr/en/components.html#adasubst

And of course, it's free software.

Enjoy!

**Win32 and WinRT Bindings**

*From: alby.gamper@gmail.com*

*Date: Sat, 27 Apr 2019 21:05:21 -0700*

*Subject: ANN: Win32 and WinRT bindings update*

*Newsgroups: comp.lang.ada*

Dear Ada Community

The Win32 and WinRT bindings have both been updated to the latest Microsoft SDK version (10.0.18362). This version corresponds to the 19H1 release of Windows 10.

Packages/Source can be found at
https://github.com/Alex-Gamper/Ada-Win32
https://github.com/Alex-Gamper/Ada-WinRT

Alex

**Simple Components**

*From: "Dmitry A. Kazakov"

*Date: Tue, 14 May 2019 19:05:57 +0200*

*Subject: ANN: Simple Components v4.40*

*Newsgroups: comp.lang.ada*

The software 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 designed tools and various protocols implementations.

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

Changes to the previous version:
- The package OpenSSL was added to provide bindings to OpenSSL;
- The package GNAT.Sockets.Server.OpenSSL was added to support secure servers based on OpenSSL;
- Multiple procedures were added to the package GNAT.Sockets.Connection_State_Machine.ELV_MAX_Cube_Client to support devices topology management and time management;
- Race condition in Object.Release fixed. The profile of the primitive operation Object.Decrement_Count has been modified.

**GtkAda Contributions**

*From: "Dmitry A. Kazakov"

*Date: Tue, 14 May 2019 19:08:07 +0200*

*Subject: ANN: GtkAda Contributions v3.24*

The software extends GtkAda 3.14.15, an Ada bindings to GTK+. It deals with the following issues:
- Tasking support;
- Custom models for tree view widget;
- Custom cell renderers for tree view widget;
- Multi-columned derived model;
- Extension derived model (to add columns to an existing model);
- Abstract caching model for directory-like data;
- Tree view and list view widgets for navigational browsing of abstract caching models;
- File system navigation widgets with wildcard filtering;
- Resource styles;
- Capturing resources of a widget;
- Embeddable images;
- Some missing subprograms and bug fixes;
- Measurement unit selection widget and dialogs;
- Improved hue-luminance-saturation color model;
- Simplified image buttons and buttons customizable by style properties;
- Controlled Ada types for GTK+ strong and weak references;
- Simplified means to create lists of strings;
- Spawning processes synchronously and asynchronously with pipes;
- Capturing asynchronous process standard I/O by Ada tasks and by text buffers;
- Source view widget support;
- SVG images support.

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

Changes to the previous version:
- The package GLib.Time_Zone was added

**GCC 9.1.0 for MacOS**

*From: Simon Wright*

*Date: Wed, 08 May 2019 20:00:57 +0100*

*Subject: ANN: GCC 9.1.0 for MacOS*

*Newsgroups: comp.lang.ada*

GCC 9.1.0 for Mac OS X El Capitan (10.11) is available at https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/9.1.0/ Also runs on macOS Mojave (10.14) and (untested) on Sierra (10.12) and High Sierra (10.13).

******************************************************************************
* DO NOT USE ON EARLIER VERSIONS OF OS X *
******************************************************************************

The native/ directory contains the 9.1.0 x86_64-apple-darwin15 compiler, together with tools from GNAT CE 2018 and various Github and other repositories.
The arm-eabi/ directory contains the 9.1.0 arm-eabi Darwin-hosted cross compiler. GNAT Community 2019

From: Alejandro R. Mosteo
<amosteo@unizar.es>

Date: Thu May 30 13:58:51 CEST 2019
Subject: GNAT Community 2019 released

As seen in several social media sources, the new Community edition of GNAT as arrived and is available for download at: https://www.adacore.com/download
From the release announce at [1] by Nicolas Setton:

We are pleased to announce that GNAT Community 2019 has been released! See https://www.adacore.com/download.
This release is supported on the same platforms as last year:
- Windows, Linux, and Mac 64-bit native
- RISC-V hosted on Linux
- ARM 32 bits hosted on 64-bit Linux, Mac, and Windows

GNAT Community now includes a number of fixes and enhancements, most notably:
- The SPARK language now has support for pointers, a fantastic milestone for the language! See https://blog.adacore.com/using-pointers-in-spark for more information about this new feature.
- The installer for Windows and Linux now contains pre-built binary distributions of Libadalang, a very powerful language tooling library for Ada and SPARK.

Check out the README for some additional platform-specific notes.
We hope you enjoy using SPARK and Ada!


Componolit Ada Runtime 1.0.0

From: u/marc-kd
Date: Tue May 28 2019 14:41:16 GMT+0200 (CEST)
Subject: Componolit Ada Runtime 1.0.0

Newsgroups: reddit:/r/ada/ [1]

https://github.com/Componolit/ada-runtime/releases/tag/v1.0.0

[News Editor - From the above link:]
Generic Ada Runtime - A downsized Ada runtime which can be adapted to different platforms.

The Componolit Ada Runtime 1.0.0 builds upon GCC 8.3 and is compatible with GNAT Community 2019. It provides the following runtime features:
- Interfaces (C)
- Secondary stack
- Exception raising
- 64bit arithmetics
- Unchecked conversion

The following features are DEPRECATED and will be removed in future releases:
- GNAT IO
Parts of the runtime are proven to have no runtime errors:
- Secondary stack allocator
- String handling
Supported platforms:
- Genode
- Linux
[1] https://www.reddit.com/r/ada/ comments/btzk99/ componolit_ada_runtime_100/

Ada Inside

Boeing 737 MAX Software

From: Paul Rubin
<no.email@nospam.invalid>
Date: Fri, 05 Apr 2019 14:16:20 -0700
Subject: Boeing 737 and 737 MAX software

Newsgroups: comp.lang.ada

Does anyone know anything about this? It has been under some criticism lately.
I have heard that the 777 software was almost entirely in Ada. It also sounds as if Boeing's software operation may have slipped in recent years, not good news for the 737 MAX.

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Sat, 6 Apr 2019 21:45:24 +0300

[...]
As I've read more about these accidents than I usually do, I will boldly (and perhaps foolishly) describe how I have understood it. All info is from public sources, I have no insider info. I am not a pilot, and moreover I wrote from recollection of my reading and have no references to give, so reader beware.
> On 19-04-06 20:30, Dennis Lee Bieber wrote:
> > > Unless things have changed severely -- GE Aviation (formerly Smith's Aerospace, formerly Lear Siegler) produces the 737 FMS software (and also the processor boxes).
> > > However, I have the impression (from TV news) the software is functioning /as designed/.
> > All info I have seen agrees with that.
> > Some reports have indicated that Boeing designed the hardware (and corresponding software requirements) such that only one sensor is used for the MCAS subsystem.
> > There are two angle-of-attack (AoA) sensors, one on each side of the nose. They feed two redundant computers, each able to run MCAS. Normally only one MCAS instance is running and it uses only its "own" AoA sensor.
> > The original design of MCAS gave it rather little control authority, which is probably why this single-sensor approach was accepted.
> > -- and a fault in that sensor results in MCAS attempting to prevent a (non) stall by pushing the nose down.
> > Yes, but MCAS does not apply a temporary nose-down command -- as if pushing the stick forward -- it changes the pitch trim, the overall angle of the horizontal stabilizer, giving the plane a permanent tendency to dive. This trim change can be overridden by the pilots, but only if they notice that it has happened.
> > In the original MCAS design, one activation of MCAS changed the pitch trim by a small amount, at most 0.6 degrees IIRC, and this limit was reported in the MCAS design documentation to the authorities. During testing, Boeing found that it was not enough, and they increased it quite a lot, to over 2 degrees IIRC. One source I read claimed that this change was _not_ updated in the documentation shown to the authorities.
> > Moreover, by design MCAS would repeat this trim change, with a certain minimum interval, as long as the AoA sensor reading remained too large and indicated a risk of stall. This iteration should converge and stop if the sensor is working, but if the sensor fails and is stuck at a high AoA (the false value reported in the second accident was around 60 degrees, IIRC) then MCAS will incrementally and cumulatively keep increasing the pitch trim and the diving tendency. If the pilots do not understand what is happening, they will find it ever harder to counteract the "dive" trim with stick inputs.
> > Some hints in the news that Boeing is changing the requirements (well, in truth, the news only says Boeing is changing the software) to have MCAS cross-reference with other flight parameter data -- and making an optional bit of hardware (additional sensors) standard.
> > AIUI the modified MCAS will read both AoA sensors and will disable itself if they disagree, and the disagreement will also be reported by a display. This display is the new piece of HW which used to be an option. There are no new sensors, AIUI.
I believe Boeing are also changing the minimum interval between MCAS activations -- perhaps even allowing only one activation -- so as to prevent a cumulatively increasing "dive" trim.

In summary, it seems to me that the criticality of MCAS, and thus the need for redundant sensors, was not realized for two reasons: (1) in its initial design, MCAS command authority was small, and (2) the possibility of multiple repeated commands (due to a stuck sensor) and the resulting large cumulative command (large change of pitch trim) was not considered.

A kind of "criticality creep".

From: Dennis Lee Biecher
From: <wlfrued@ix.netcom.com>
Date: Fri, 12 Apr 2019 18:15:07 -0400

On Fri, 12 Apr 2019 00:46:31 -0700, trannngocduong@gmail.com declaimed the following:

> I know nothing about the software. But I don't think it is written in Ada. If it was, programmers must have chosen a wrong subtype.

It's Ada... (In the past, I was doing maintenance on the FMS "BootROM" code -- which, while not the actual runtime flight software, is responsible for doing CRC checks of the software and databases, reading new software from data loaders, and loading which application is to run based upon external settings. The FMS software links with the same base "OS".

> On the issue of Ada subtypes, it seems to me that if the SW specification, design and coding considers sensor faults (as it of course should), the normal approach for such critical SW.

One of the criticisms of the decisions leading to the MCAS software is that the software is certified only at DO-178B level C, defined as software whose consequences are (https://en.wikipedia.org/wiki/DO-178B):

- Major – Failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries) or significantly increases crew workload (safety related)
- This is instead of level A (catastrophic, the whole plane can be lost), or level B (hazardous, people can be injured). The rationale was that at worst MCAS going wrong would change the nose pitch by a few degrees and then the pilot could fix it. They didn't consider the possibility of it activating over and over again, tilting a few more degrees each time.
- Since the software was treated as level C, its development and certification process was less rigorous than what it would have gotten at a more critical level.
- Certifying and developing this system at level C instead of level A was itself obviously some kind of process failure. I believe finding out how that happened is one of the investigation's objectives.

Ada and other Languages

**Pointer Ownership, Containers and Cursors in Ada, Rust, SPARK**

From: Randy Brukardt
From: <randy@rrsoftware.com>
Date: Mon, 18 Mar 2019 18:36:15 -0500

My understanding is that the SPARK people are far into designing ownership contracts for Spark.

It's also possible that Ada 2020 will have a form of pointer ownership. (Unfortunately, we didn't make any conclusions on that during yesterday's meeting, so it's still in limbo, and we're getting very close to the finish line.) The current problem is that in Ada 2020 as it stands, it's not possible to write a containers implementation in pure Ada. You'd have to have some implementation hack to turn off some of the Legality Rules. Tucker has designed a solution, based on an ownership mechanism, but as it is new and barely vetted, it's unclear what we will do with it ultimately. Note that this solution will not provide the perfect safety that you would get with SPARK or Rust, but it would form the foundation of the SPARK solution and it would clearly catch a lot of issues with using pointers to implement ADTs. (And there is little other reason to use pointers, IMHO.)

From: Randy Brukardt
From: <randy@rrsoftware.com>
Date: Mon, 18 Mar 2019 19:12:29 -0500

But a better question is whether the Rust borrow checker allows building proper ADTs for most data structures. Most of Tucker's proposals didn't have a safe way to build typical data structures like a doubly-linked list or the parent pointer of a tree structure. Leaving out these backward pointers means adding a substantial performance degradation for (possibly) common operations like node deletion. Depending on what you're doing, that could be a non-starter. I haven't had a chance to actually look at Rust's actual rules; Ada is hard enough and as we're in the home stretch for Ada 2020, I literally don't have time for much else. (Probably shouldn't be answering this message...
Tucker's latest proposal does address the back pointer problem. So at least that can be done with checks.

[...]

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Tue, 19 Mar 2019 18:01:19 -0500

[...]

Some background here: The basic idea behind pointer ownership is to prevent various issues by enforcing an invariant -- that each allocated object is stored in exactly one pointer object. This is enforced with a variety of runtime and compile-time rules.

Now, it's clear that one can't even walk a data structure that way, so the idea of "borrowing" a pointer for a limited time was invented. Such borrowing has to be done in carefully controlled ways in order to keep it being safe -- for instance, no one can read or write the original pointer which it is being borrowed from.

Multiple long-lived pointers that point at a single object are simply not allowed. In part, that's done by making assignment either illegal or a move (where the source is nullled when the pointer is assigned).

For something like a cursor, that means that Rust-pointers couldn't be used to create the object. The entire point of a cursor is that it is a long-lived handle to a specific element in a larger data structure. One can't null out part of the data structure to create the handle, and if the assignment is banned completely, you could never create a valid cursor object in the first place.

There are similar issues with back pointers in a data structure, as you might guess.

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Tue, 19 Mar 2019 18:13:45 -0500

[...]

A cursor is a handle accessing an element of a larger container, nothing more or less than that. The primary usage is to connect data structures made up of multiple containers. For instance, consider a compiler symbol table. There is a tree structure that represents each of the declarations and their scopes, and a map structure that represents a mapping of names to nodes of the tree. The contents of that map is going to be tree cursors, each representing a declaration with a particular name.

[...]

The value of cursors is that they can be implemented by a range of abstractions with a range of checking, from array indices (as in the bounded containers and the vector) to pointers with a variety of schemes from no dangling checking to the bulletproof controlled cursor scheme.

[...]
race conditions, no matter how crazy you move stuff around or create new parallelism. Additionally the ownership concept lead to many libraries typecast encoding their API which makes missing them a near impossibility (at compile-time) while Ada mainly catches those misuses at runtime via exceptions.

[...]
[1] "Why Rust was the best thing that could have happened to Ada": https://www.reddit.com/r/ada/comments/7wzrqi/why_rust_was_the_bestThing_t hat_could_have/

From: "Randy Brukardt"
Date: Thu, 14 Mar 2019 17:41:12 -0500

Obviously, if your existing code isn't documented properly as to what needs to be task-safe, then refactoring isn't going to work very well. Refactoring bad code is just going to give you bad code. :-) And almost all code in any language is bad code, because at some point people turned to "just make it work" mode, and stopped doing the things necessary for the code to be understandable. Using Ada helps, but surely doesn't eliminate this practice.

In any case, Ada 2020 is very much about addressing this point. The new Nonblocking and Global contracts make it possible to declare tasking and memory side-effects, and the "conflict check policies" allow using that to prevent data races. (Note that there is a difference between a "data race", and "race conditions"; there are plenty of race conditions that aren't data races, and no programming language can statically prevent the latter, since they're caused of a sequence of operations. Well, other than not having any task interactions in the first place. :-)

In addition, conflict checks are enabled by default on the new parallel constructs, so you have to work at causing problems. (The parallel constructs are safer anyway, since they do not allow blocking, so there aren't any rendezvous and entry calls to worry about.) And they can be enabled on tasks as well (not done by default for the obvious reason of compatibility - but also for capability, tasks should mainly be used in Ada 2020 when one needs rendezvous and other constructs that can't be checked at compile-time).

The issue with this is that a dereference of an access value is almost always going to cause a conflict and thus be illegal. And the contracts for the containers are designed so that they can be used in parallel operations (presuming the actual parameters to the instance allow that). This means that no access types can be used to implement the containers, which is nonsense for the unbounded and indefinite containers. The ownership stuff is a proposal to limit that in the case of building ADTs, including the containers.

From: Olivier Henley
Date: Wed, 13 Mar 2019 06:23:59 -0700

Thanks to those who brought 'material' to the discussion.

a. The Rust thread is now closed and we did not slide into a flame war. Very good.

b. We definitely enlightened a whole bunch. You have no idea how many Rustaceans do not even know Ada exists. After all, awareness and politics are important. Very good.

c. From Randy's post, I find it exciting to see that this 'episode' is of actuality regarding Ada202X. Very good.

Thx

Ada Practice

Interviews to Ada Practitioners

From: Alejandro R. Mosteo
Date: Fri, 24 May 2019
Subject: Interviews to Ada enthusiasts

Tomek Walkuski
<tomek.walkuski@gmail.com>, co-founder at 98elements [5], is running a series of interviews [1] to people from the Ada community. Here is a list with a few extracted words from each interview since the last AUJ Issue:

- Fabien Chouteau interview [2]: "My name is Fabien Chouteau, I am embedded software engineer at AdaCore, hobbyist in electronics, instrument making and woodworking. [...] A couple years ago I started the Ada Drivers Library project, at first it was just a way to have fun with an ARM Cortex-M micro-controller board and see how Ada can be used on such hardware. It became a one stop shop for getting started in embedded Ada programming and sparked many other projects [...]"

- Edward Fish interview [3]: "I was introduced to Ada in one single class, Programming Languages, which did a high-level introduction/survey of various languages and instantly felt at-home. It did raise the question as to why a lot of the features aren't common in more languages [...]"

- Stéphane Carrez interview [4]: "Later I created another computer board based on 68HC11 and to use it I also did a complete port of the GNU compiler, the GNU binutils and the GNU debugger. My work was integrated in the FSF sources in 2000. The GNAT Ada compiler was working! I was able to run a small Ada program that fit in less than 256 bytes!"

[1] https://tomekw.com/tag/interview/

Integer type with gaps

From: mario.blunk.gplus@gmail.com
Date: Fri, 29 Mar 2019 09:10:40 -0700

Subject: type definition for an integer with discrete range

Newsgroups: comp.lang.ada

Hello,

I'm looking for a way to define a type that runs from let say -100 to +100 with gaps of 5 width. Important is to make sure that a value like 7 cannot be assigned to the type.

something like:

```ada
type number is new integer range -100 .. 100;
```

-- or

```ada
subtype number is integer range -100 .. 100;
```

-- with this special thing or something like

```ada
-- that:
for number small use 5; -- cannot applied
-- here, works with fixed point types only
```

Thanks!

From: Simon Wright
Date: Fri, 29 Mar 2019 21:24:53 +0000

[...]

What about this?

```ada
pragma Assertion_Policy (Check);
with Ada.Text_Io; use Ada.Text_Io;
procedure Type_Integer is
  subtype Number is Integer range -100 .. 100;
  begin
    V : Number;
    begin
      V := 0;
      Put_Line ("0't image is " & V'Image);
      V := -50;
      Put_Line ("-50't image is " & V'Image);
      V := 42;
      Put_Line ("42't image is " & V'Image);
    end Type_Integer;
end
```

Executing gives

```ada
$ ./type_integer
0't image is 0
-50't image is -50
```

raised SYSTEM ASSERTIONS.

ASSERT_FAILURE : Dynamic_Predicate failed at type_integer.adb:12

Gauss Error Function in Ada

From: leov@gammawizard.com
Date: Mon, 1 Apr 2019 09:28:58 -0700
Subject: Erf() function in ADA
Newsgroups: comp.lang.ada

Greetings, I have been looking into reimplementing a collection of numerical heavy code from R/C++ into ADA and so far things seem doable. My only question is about the support for the error function and in particular the complementary error function erf(). I assume this is library dependent so I would appreciate any information if erf() is part of the ADA standard library or perhaps provided by GNAT in some form?

From: gautier_niouzes@hotmail.com
Date: Mon, 1 Apr 2019 10:01:07 -0700

You can get easily the error function from the Phi function which is available in the following library:

http://mathpqa.sourceforge.net/

From: gautier_niouzes@hotmail.com
Date: Tue, 2 Apr 2019 03:39:57 -0700

A few random remarks...
1) For further references: there is now in Mathpqa (rev. 153+) a separate Erf_function package. Since Phi_function.Phi uses Erf(x) anyway, it's better to have access to Erf directly.
2) About the Numerical Recipies: be careful, some versions support only 7-8 digits (single precision), so numerical errors cumulate very quickly.
3) Some good stuff can be found in the Algib and Cephes libraries, in C, Fortran or Pascal
4) Simple special functions (with one parameter) could well be in an official Ada.Numerics.Generic_Special_Functions (low maintenance effort for compiler vendors)
5) Don't forget to check:
https://www.adacit.org/ada-resources/tools-libraries/
6) Perhaps the Alire system has some math packages?

Porting GNAT bare-board runtime to a new target

From: Daniel Way
<p.waydan@gmail.com>
Date: Sun, 7 Apr 2019 19:13:07 -0700
Subject: Understanding GNAT Bare Board Run-time for Cortex-M
Newsgroups: comp.lang.ada

I'm trying to port the bare-board GNAT run-time to a Coretex-M0+ (NXPKV11Z7) processor. I'm new to concurrency and have been reading through the run-times for the STM32 targets to understand how the tasks and protected objects are implemented,

however, there seems to be a web of dependencies between the different packages and wrappers of wrappers of wrappers for types and subprograms.

* Is there any tool available to scan through the source code and generate a graphical call graph to help visualize the different dependencies?
* Has anyone on the forum successfully ported a bare-board run-time? What was your experience and do you have any tips?
* Is porting the run-time just a matter of updating the linker, a few packages, and a GPR script, or is there some fundamental implementation changes to consider?

Thank you,
Daniel

From: Simon Wright
<simon@pushface.org>
Date: Mon, 08 Apr 2019 08:36:59 +0100

Daniel Way writes:

> I'm trying to port the bare-board GNAT run-time to a Coretex-M0+ (NXPKV11Z7) processor. I'm new to concurrency and have been reading through the run-times for the STM32 targets to understand how the tasks and protected objects are implemented, however, there seems to be a web of dependencies between the different packages and wrappers of wrappers for types and subprograms.

Yes.

> * Is there any tool available to scan through the source code and generate a graphical call graph to help visualize the different dependencies?
> Pass.

> * Has anyone on the forum successfully ported a bare-board run-time? What was your experience and do you have any tips?

AdaCore have published a guide for porting their runtime[0].

GNAT CE 2018 includes a ravenscar-sfp-microbit runtime.

My Cortex GNAT RTS[1] is based on FreeRTOS[2] and includes an RTS for the nRF51 as found in the BBC micro:bit.

That's a cortex-m0, but as far as I can see [3] the differences from the m0+ are minimal.

The main issue I had was with the clock; the nRF51 doesn't have a system tick, instead I had to use RTC1 (I think AdaCore used RTC0).

> * Is porting the run-time just a matter of updating the linker, a few packages, and a GPR script, or is there some fundamental implementation changes to consider?

That would be it (also the runtime.xml file) but the problem is identifying _which_ packages to change! I wouldn't expect many from the microbit RTS, it's likely to be clock setup and interrupt naming. It would help if you had an SVD to generate the board peripheral dependencies.

[0] https://github.com/AdaCore/ bb-runtimes/tree/community-2018/doc/porting_runtime_for_cortex_m

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Mon, 8 Apr 2019 10:46:56 +0300

On 19-04-08 05:13 , Daniel Way wrote: [...]

> * Is there any tool available to scan through the source code and generate a graphical call graph to help visualize the different dependencies?
> * Has anyone on the forum successfully ported a bare-board run-time? What was your experience and do you have any tips?

In my last project, I ported the small-footprint Ravenscar run-time for the SPARC architecture from the generic off-the-shelf AdaCore version to a specific SPARC LEON2 processor embedded in an SoC for processing satellite navigation signals, the AGGA-4 SoC.

My advice is to first understand the differences between the original target processor and the new target processor, especially in these areas:

- Basic processor architecture, and especially if there is some difference in the instruction set or in the sets of registers that must be saved and restored in a task switch. In my case there was no difference, so I did not have to modify the task-switch code nor the Task Control Block structure. For porting across various models of the same processor architecture, perhaps the most likely difference is in the presence or absence of a floating-point unit and dedicated floating-point registers.
- The HW timers. In my case the RTS used two HW timers, and there were some differences: the bit-width was different (32 instead of 24) and the HW addresses and interrupt numbers were different. The corresponding parts of the RTS had to be adapted, but in my case the changes were small, and the logic of the code did not change.
Heart of Darkness

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Sat, 20 Apr 2019 17:58:48 +0200
Subject: Re: Anonymous Access and Accessibility Levels

Le 20/04/2019 à 17:29, Jere a écrit :

> I was trying to get a bit better at understanding how accessibility levels
> work with respect to anonymous access types. I have GNAT to test out things,
> but I think I am running into various bugs, so I am not seeing the exceptions
> or compilation errors I would expect. It could also be that I misunderstand
> the rules (They are difficult somewhat).

In my tutorial about memory management, I explain that there are 34
special cases in 3.10.2 (AKA "heart of darkness"). Enter at your own risk.

From: "Randy Brukardt"
Date: Wed, 24 Apr 2019 18:27:52 -0500

I suspect that accessibility implemented by compilers is essentially whatever
the ACATS tests require. I know that I've never spent time on it in Janus/Ada
beyond that -- it simply isn't worth self-inflicted pain. Thus, my advice is that
accessibility works like one would expect in basic cases, and do not go beyond
cases unless you like pain.

From: "Randy Brukardt"
Date: Mon, 22 Apr 2019 17:11:19 -0500

As always, I suggest the following rules:
(1) Do not use anonymous access types unless you absolutely need one of the
special capabilities that can only be done with them.
(2) Under no circumstances, do anything
dynamic accessibility works like one would expect
in basic cases, and do not go beyond
cases unless you like pain.

> * Is porting the run-time just a matter of
> updating the linker, a few packages,
> and a GPR script, or is there some
> fundamental implementation changes to
> consider?

If you are porting from one
implementation of the same architecture
info another (e.g. ARM Cortex
M4 with the Thumb-1/2 instruction
sets, if I understand right), IMO it is
unlikely that any fundamental changes are
required. However, if there are
differences in the instruction set (with
M0+ omitting some instructions available
large members and perhaps used in the
original RTS) be sure to use the correct
target options for the compiler so as to
avoid generating code that will not run on
the M0+. If there is a major difference in
instruction sets (say, porting from Thumb-
2 to Thumb-1) you will have to review
and perhaps modify all the assembly-
language RTS parts, and all assembly-
language code insertions in the Ada RTS
code, and all the code in court0.S.

> * Is the memory layout. Where in the address
> space is the ROM (or flash), where is the
> RAM, where are the I/O control
> registers? Any differences in the layout
> must be implemented in the linker
> command script, which in my case was a
> file called leon.ld. The Ada RTS code
> probably does not have to change for
> this reason, and did not change in my
> case.

Once all that is sorted out, you will probably
have to modify the start-up assembly-language code, which in my
case was in the file court0.S. This deals with
HW initialization (clearing registers,
stopping any I/O that might be running,
disabling interrupts, etc.) and SW
initialization, which means to set up the
stack for the environment task and then
enter the body of that task.

> * Is porting the run-time just a matter of
> updating the linker, a few packages,
> and a GPR script, or is there some
> fundamental implementation changes to
> consider?

If you are porting from one
implementation of the same architecture
info another (e.g. ARM Cortex
M4 with the Thumb-1/2 instruction
sets, if I understand right), IMO it is
unlikely that any fundamental changes are
required. However, if there are
differences in the instruction set (with
M0+ omitting some instructions available
large members and perhaps used in the
original RTS) be sure to use the correct
target options for the compiler so as to
avoid generating code that will not run on
the M0+. If there is a major difference in
instruction sets (say, porting from Thumb-
2 to Thumb-1) you will have to review
and perhaps modify all the assembly-
language RTS parts, and all assembly-
language code insertions in the Ada RTS
code, and all the code in court0.S.

HTH. I think others on this group have
more experience with ARM Cortex run-
time systems and can probably offer
better advice.

License woes

[Often, when Ada compilers are
discussed, the licensing model of GNAT
arises in conversation. What follows
discusses the limitations imposed by the
pure GPLv3 license of the GNAT runtime
in Community editions —News Editor.]

From: Maciej Sobczak
Date: Mon, 27 May 2019 23:43:06 -0700
Subject: Re: Needed - Ada 2012 Compiler.

On Tuesday, May 28, 2019 at 1:25:03 AM UTC+2, Optikos wrote:

> Hence why Alex was correctly
> indicating that GPL Community
> Edition forestalls most practical forms
> of commercial business activity

I have an impression that nowadays “most
forms of commercial business activity”
involve setting up an account for
accessing whatever on-line service.
This is why most apps today are free,
anyway. In this context, GPL license on
the app is not a problem at all.
No, I do not applaud the GPL licensing. I
only state that the landscape of
“commercial business activity” has
significantly changed from what it was
say two decades ago.
I also think that you are overestimating
the willingness of customers to engage in
further business activity of reproducing
and re-selling what they have bought from
you. This concept is being demonized
since ever, but I don't think it has any
bigger significance than a “traditional”
counterfeiting.
No, I don't applaud GPL as a licensing
scheme. I just don't consider it to be a
showstopper.

> by entirely prohibiting AdaCore-esque
dual licensing

Wrong. You can write your program (or a
library) and sell it in the form of source
code with whatever license you wish and
library) and sell it in the form of source

compiler that you have used to verify (!) your product has no impact on the licensing of your source code. Thus, dual- or closed-licensing is still possible. Feel free to complete this scheme with any kind of NDA or other forms of legal agreements with your customers.

From: Maciej Sobczak
<see.my.homepage@gmail.com>
Date: Tue, 28 May 2019 22:54:03 -0700

[... ] let's go back a little to better understand the workflow.
1. You write some code. It can be a standalone app or a library.
2. You can put whatever license you wish on your source code.
3. You can deliver it (the source code!) to your users with that license.

Finished.

OK, so you think it might be a good idea to verify this code a little bit before selling it to your customers - you know, test it or at least check whether it compiles at all. So you add an additional points to the scheme above:
1a. You compile your code with whatever compiler you have.

1b. You run your tests or perform whatever other verification activities to make sure that your product has an expected quality level.

These two points have no impact on points 2. and 3. above.

I will agree that this scheme is not satisfactory for the case of applications distributed via App Stores, or for users who don't want to be involved in technical activities like compiling something on their own - this is understandable, and in such cases a turn-key product needs to be delivered. But it is a very satisfactory scheme for the case of libraries, which become included in this kind of workflow on the user side anyway.

__Ada in Jest__

**Lightening the mood in serious discussion**

From: Jeffrey R. Carter
<spam.jrcarter.not@spam.not.acm.org>
Date: Tue, 12 Mar 2019 16:41:28 -0500
Subject: Re: Intervention needed?
Newsgroups: comp.lang.ada

I have no desire to register to post on a [Rust] forum full of people who like to use pointers. It's bad enough being on one full of people who like to use anonymous access types.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 22 Mar 2019 15:09:36 +0100

Le 22/03/2019 à 12:10, Lucretia a écrit :

>> He [a Rust forum poster] also told me that Ada compilers aren’t allowed to do certain kinds of optimizations that for example c, c++ (and Rust and other PL via LLVM) are doing.

> > How true is this?

In Ada, the principle is that the compiler has an obligation of result, i.e. that the "external effect" (see 1.1.3(9)) of the compiled program must be the same as the effect defined by the canonical execution.

Basically, this means that the compiler can do any optimization provided the result is correct. Going farther than that would mean allowing the compiler to generate incorrect programs... Maybe that's what C/C++ compilers are doing ;-)