ADA USER JOURNAL

Volume 28
Number 1
March 2007

Contents

Editorial Policy for Ada User Journal

Publication

Ada User Journal – The Journal for the international Ada Community – is published by Ada-Europe. It appears four times a year, on the last days of March, June, September and December. Copy date is the first of the month of publication.

Aims

Ada User Journal aims to inform readers of developments in the Ada programming language and its use, general Ada-related software engineering issues and Ada-related activities in Europe and other parts of the world. The language of the journal is English.

Although the title of the Journal refers to the Ada language, any related topics are welcome. In particular papers in any of the areas related to reliable software technologies.

The Journal publishes the following types of material:

- Refereed original articles on technical matters concerning Ada and related topics.
- News and miscellany of interest to the Ada community.
- Reprints of articles published elsewhere that deserve a wider audience.
- Commentaries on matters relating to Ada and software engineering.
- Announcements and reports of conferences and workshops.
- Reviews of publications in the field of software engineering.
- Announcements regarding standards concerning Ada.

Further details on our approach to these are given below.

Original Papers

Manuscripts should be submitted in accordance with the submission guidelines (below).

All original technical contributions are submitted to refereeing by at least two people. Names of referees will be kept confidential, but their comments will be relayed to the authors at the discretion of the Editor.

The first named author will receive a complimentary copy of the issue of the Journal in which their paper appears.

By submitting a manuscript, authors grant Ada-Europe an unlimited license to publish (and, if appropriate, republish) it, if and when the article is accepted for publication. We do not require that authors assign copyright to the Journal.

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

Ada User Journal is one of the ways in which people find out what is going on in the Ada community. Since not all of our readers have access to resources such as the World Wide Web and Usenet, or have enough time to search through the information that can be found in those resources, we reprint or report on items that may be of interest to them.

Reprinted Articles

While original material is our first priority, we are willing to reprint (with the permission of the copyright holder) material previously submitted elsewhere if it is appropriate to give it a wider audience. This includes papers published in North America that are not easily available in Europe.
We have a reciprocal approach in granting permission for other publications to reprint papers originally published in *Ada User Journal.*

Commentaries

We publish commentaries on Ada and software engineering topics. These may represent the views either of individuals or of organisations. Such articles can be of any length – inclusion is at the discretion of the Editor.

Opinions expressed within the *Ada User Journal* do not necessarily represent the views of the Editor, Ada-Europe or its directors.

Announcements and Reports

We are happy to publicise and report on events that may be of interest to our readers.

Reviews

Inclusion of any review in the Journal is at the discretion of the Editor.
A reviewer will be selected by the Editor to review any book or other publication sent to us. We are also prepared to print reviews submitted from elsewhere at the discretion of the Editor.

Submission Guidelines

All material for publication should be sent to the Editor, preferably in electronic format. The Editor will only accept typed manuscripts by prior arrangement.
Prospective authors are encouraged to contact the Editor by email to determine the best format for submission. Contact details can be found near the front of each edition.
Example papers conforming to formatting requirements as well as some word processor templates are available from the editor. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

In the foreword to volume 28 of the Ada User Journal, which commences with the present issue, our memory goes to one very good and one very sad news that opened the year 2007 by occurring in the short span of two calendar days: On January 24, the Amendment to ISO/IEC 8652 (which we call and will keep calling 'Ada 2005' in the vernacular) was finally approved by ISO. All nations who participate in WG9 voted to approve and no comments were submitted, which were definite tokens of excellent technical work and also cohesive and determined national support. And that was a very good news indeed. (For the record, it took in fact another 7 weeks, until March 9, for the Amendment to be finally published on the ISO calatogue. But then it was.) The sad news came only two days after that. On January 26, Jean Ichbiah, the man who designed Ada, passed away at 66. I have read many nice words about Jean Ichbiah from various authorities who wrote about him, and more you will read for yourselves in the News section of this issue. I have to say however that I was most impressed by some passages of the obituary that the Boston Globe run about him shortly after his death. I wish to share some excerpts of that with you in this editorial. Jean said in 1984 in an interview to the CACM celebrating the birth of Ada: "I see myself really as an architect, […] My work was not to invent new things; it was not research work, it was architectural work. I had to integrate the best available materials to construct the building that would best suit the requirements of the users." He viewed the Ada language "as a cathedral with all the architectural lines interwoven in a harmonious manner," I find this is a very good way of remembering who Jean Ichbiah was to the Ada community. It is so very sad that he passed away. It is reassuring and conforting though that his creature is still well and alive some 30 years after he took it on himself to design it.

Returning to the more mundane task of illustrating the contents of the issue and the plans for the remainder of the volume, I am pleased to welcome the contribution of Romain Berrendonner and Cyrille Comar of the Paris offices of AdaCore, who report on the development and experimental use of a software infrastructure commissioned by the European Space Agency for benchmarking the space and time performance of Ravenscar technology and applications. The report on that project constitutes the technical matter of the present issue. Future issues will have reports from the IRTAW-13 workshop, which will take place in April at Woodstock, Vermont, USA (cf. the technical program in this issue) and staggered proceedings of the industrial track of the Ada-Europe 2007 conference, which will be held in Geneva, CH, in June. The rest of the issue contains news, and calendar events of interest to the Ada community, as usual, gathered for you from our News and Calendar editors.

Tullio Vardanega
Padova
March 2007
Email: tullio.vardanega@math.unipd.it

News

Santiago Urueña

Technical University of Madrid (UPM). Email: Santiago.Uruena@upm.es

Contents

Ada-related Organizations

Ada-Belgium — Ada 2005 Approved

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.ac.be>
To: ada-belgium@cs.kuleuven.be
Date: Wed, 24 Jan 2007 21:59:57
Subject: Final ISO/IEC ballot approves Ada amendment

I am very pleased to announce that JTC1 (the joint technical committee of ISO and IEC on information technology) approved the Amendment to ISO/IEC 8652 (the Ada standard) as submitted by SC22/WG9 (the Ada Standardization Working Group).

The Amendment was approved by a vote of 17-0-5 (yes-no-abstention). All nations who participate in WG9 voted to approve. No comments were submitted. The voting process is now completed. The only remaining step is actual publication of the Amendment by ISO.

As mentioned earlier, through your Ada-Belgium membership, you helped Ada-Europe to sponsor the production of the Ada 2005 Language Reference Manual (LRM), the Annotated Ada 2005 Language Reference Manual (AARM), and the Rationale for Ada 2005. Chapters of the latter were published in the Ada User Journal issues that you received in 2005 and 2006. Ada-Europe also made the publication possible of the Ada 2005 LRM in Springer's Lecture Notes in Computer Science series (LNCS).

All those documents remain available online in various formats at <http://www.adaic.com/standards/ada05.html>

SC22/WG9 is already looking at the future and several activities are currently underway:

The ARG (Ada Rapporteur Group) now focuses on (in decreasing order of priority)

- developing a revision of ISO/IEC 15291 (the ASIS standard), on the one hand to bring it in sync with the new Ada 2005 standard and on the other hand to provide a semantic interface at a higher level of abstraction (i.e. easier to use);

- responding to Defect Reports and/or Ada Issues on ISO/IEC 8652 (the Ada standard);

- developing Technical Reports or Standards improving the Ada libraries, notably with respect to containers; and

- considering proposals for extending the language.

The HRG (Annex H Rapporteur Group) focuses on

- revisiting ISO/IEC 15942 (the report "Guidelines for use of Ada in High Integrity Applications") with a view to updating it for Ada 2005.

The new PRG (Ada-POSIX Binding Rapporteur Group) focuses on

- maintaining ISO/IEC 14519 (the Ada Binding to POSIX), as since this document was standardized there have been 2 revisions of Ada and 2 of POSIX.

So you see, lots of activities are going on, and we hope that through your Ada-Belgium membership you will continue to support them and that you will be looking forward to be kept informed, among others via the 3-monthly Ada User Journal published by Ada-Europe.

We'd like to thank all our members who promptly paid their membership renewal for the year 2007 upon receipt of the invoice earlier this month. If you haven't paid yet, we would appreciate it if you could settle the invoice as soon as possible.

Thanks once more for your support and interest.

As always, I will keep you informed of further progress.

Dirk Craeynest

ISO/IEC JTC1/SC22/WG9, Head of Delegation, Belgium

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

Disclaimer: http://www.kuleuven.be/cwis/email_disclaimer.htm

[See also "ARA — Technical Work on Ada 2005 Standard Completed" in AUJ 27-2 (Jun 2006) p.69 and "Ada 2005 Published by ISO" in this issue —su]

Ada 2005 Published by ISO

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 14 Feb 2007 11:47:59 −0600
Subject: Re: Ada 2005 ISO approval yet?
Newsgroups: comp.lang.ada

> When will Ada 2005 become official?

Not until it is published by ISO. How long that will take is anyone's guess (sometimes, it has been over a year). All of the approval votes have finished, though, so the remaining wait is purely administrative.

Randy Brukardt, ARG Editor

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Date: Sun, 28 Jan 2007 20:16:32
Subject: Re: Ada 2005
Newsgroups: comp.lang.ada
Summary: Ada amendment approved by ISO/IEC

[…] Although the final ISO/IEC ballot recently approved the Ada amendment, it only becomes an *official* ISO standard upon publication. And that hasn't happened yet.

But for all practical reasons, the updated Ada language definition has been accepted at the highest ISO level and thus can be considered standardized.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Organization: Adalog
Subject: 9 Mars 2007, une date =?ISO-8859-1?Q?=E0_retenir?=
Date: Mon, 12 Mar 2007 10:09:30 +0100
Newsgroups: fr.comp.lang.ada

[Translated from French. —su]

Message from Jim Moore:

The amendment to the Ada language standard was published on March 9:

http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=45001

« Ada 95 *est mort, vive* Ada 2005! »

ARA — Ada Conformity Assessment Test Suite

December 20, 2006

Update: Ada Conformity Assessment Test Suite

ACATS Modification List 2.5N and the associated test files have been posted.

ARA — Jean Ichbiah passes away

http://www.adaic.org/news/ichbiah.html

Photo: http://www.adaic.org/news/images/ichbiah.gif

Jean Ichbiah (1940–2007)

Jean Ichbiah, from Burlington (Massachusetts), the chief designer of the Ada computer programming language, died on January 26, 2007, after a battle with cancer.

Jean David Ichbiah was born in Paris in March 25, 1940. He was a second generation Frenchman, the grandson of Sephardic Jewish immigrants from Greece and Turkey. During World War II his family was hidden on an estate in southern France to escape Nazi persecution.

Mr. Ichbiah attended the prestigious French engineering school École Polytechnique in Paris, majoring in Civil Engineering at the École des Ponts et Chaussées, after serving in the French army in Germany. In 1964 he married Marianne (née Kleen). Soon after his marriage Mr. Ichbiah enrolled as a doctoral student at the Massachusetts Institute of Technology, obtaining a PhD in Civil Engineering and Operations Research in only two years.

Returning to France in 1967, Mr. Ichbiah was employed as a computer scientist by the then recently formed company CII-Bull, conceived by President de Gaulle to give France a leading edge in the computer industry. It was at CII-Bull, later associated with Honeywell U.S., that Mr. Ichbiah did his outstanding work as the chief designer of Ada, a computer programming language sponsored by the U.S. Department of Defense to incorporate the best features from the Babel of computer languages that predominated in the 1970s.

The development of Ada, which was standardized in 1983 in the U.S. and later internationally under ISO, advanced the state of the art in language design and led to significant cost savings in software development. Since its inception Ada has been used for a broad range of applications ranging from aircraft avionics to payroll processing, and it is especially attractive for high-integrity systems with requirements for safety and/or security.

As chief designer of Ada, Mr. Ichbiah succeeded in combining three main goals into a practical language: program reliability, readability, and efficiency. Mr. Ichbiah's colleagues and collaborators have described him as a brilliant, tenacious leader capable of developing a consensus among several proposals for solving tricky technical problems.

In 1980 Mr. Ichbiah left CII-Honeywell-Bull to found the Alsys (Ada Language Systems) company. As its CEO, he continued his work on Ada and hired an international team of over one hundred computer scientists to implement Ada development toolsets on a variety of platforms ranging from PCs to mainframes. Alsys had offices in the U.S., France, England, Germany, and Japan and was ultimately acquired by Thomson in 1991. Since 1993 the Ichbiah family has owned Textware Solutions of Burlington, MA, a company they created when Jean developed an innovative fast text entry system for PCs and a virtual keyboard layout (Fitaly) optimized for handheld computers.

Jean Ichbiah was a member of the French Legion of Honor and the French Academy of Sciences, and he received the "Grand Prix de la Technologie" from the City of Paris. He was awarded a Certificate of Distinguished Service from the U.S. Department of Defense for his work on Ada, and he also received an ACM SIGAda Award for Outstanding Ada Community Contributions.

Jean and Marianne Ichbiah became American citizens in 2001. In a recent article by Mr. Ichbiah published by the French Academy of Sciences, he extolled American research and entrepreneurship suggesting them as a models for the French universities and research institutes.

Mr. Ichbiah is survived by his wife Marianne of Burlington, MA, and also three children and six grandchildren all living in France. His son, Emanuel Ichbiah, is an independent computer consultant; his two daughters Helena and Myriam are respectively a graphic art designer and an executive at l'Oreal.

In a 1984 interview with the Association for Computing Machinery, Mr. Ichbiah was asked to express his feelings about the language he had masterminded. The response is befitting of a designer trained in civil engineering and becoming a preeminent computer scientist: I see Ada as a cathedral, with all the architectural lines interwoven in a harmonious manner. I would not do it differently if I had to do it over again.

[See also "A tribute to Jean Ichbiah" in this issue —su]

A tribute to Jean Ichbiah

From: Joyce Tokar <tokar@attglobal.net>
Organization: Pyrrhus Software
Date: Sun, 28 Jan 2007 13:47:16 −0700
Subject: In Remembrance of Jean Ichbiah
Newsgroups: comp.lang.ada

With great sadness I learned from Ben Brosgol that Jean Ichbiah passed on Friday, 26 Jan 2007

Ben said that Jean had a brain tumor around a year ago, and he had a serious fall last Autumn in which he fractured his skull. He was in rehab for several months, and I believe that he had been home (in Burlington, Mass.) for several weeks.

Funeral services will be held on Tuesday, January 30, at 12:30 pm:

Temple Shalom Emeth 16 Lexington Street Burlington, Mass.

To quote John Barnes:

"Jean had an amazing understanding of the basic concepts concerning what programming was really about. Ada may have its flaws but it is a damn sight better than anything else I know.

Jean will be remembered as the inspiration for ideas which have driven many of our careers."

From: Ian Caldwell <iccaldwell@bigfoot.com>
Date: Mon, 29 Jan 2007 20:32:43 GMT
Subject: Re: In Remembrance of Jean Ichbiah
Newsgroups: comp.lang.ada

Jean through creating Ada changed my life. One of my best periods of employment was when I worked for Alsys. It's a sad loss.

Ian Caldwell

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Mon, 29 Jan 2007 01:16:42 GMT
Subject: Re: In Remembrance of Jean Ichbiah
Newsgroups: comp.lang.ada

I'm very sorry to hear this.

I heard Jean say once that Ada was basically an evolution of LIS, which he developed in the early 70s. Ada 83 was ahead of its time, and clearly Jean even further ahead of his time.

Jeff Carter

From: Jinho Barc <jino@yahoo.co.kr>
Date: 5 Feb 2007 17:25:17 −0800
Subject: Re: In Remembrance of Jean Ichbiah
Newsgroups: comp.lang.ada

I think his 1983-born child "Ada" was a beautiful, elegant and tremendous (expressive power / language complexity) ratio-ed programming language.

I'm sorry for this sad news. Rest in peace.

From: "Beliavsky" <beliavsky@aol.com>
Date: 15 Mar 2007 15:58:50 −0700
Subject: Re: FYI — Lead Designer of Ada Dies
Newsgroups: comp.lang.ada

Steve Lionel, for many years a Fortran compiler developer, once worked on an

Ada compiler, and his tribute to Jean Ichbiah and the Ada language are at http://softwareblogs.intel.com/2007/03/05/a-farewell-to-jean/ , copied below.

By Steve Lionel (6 posts) on March 5th, 2007 at 8:38 am

"If you asked me what my favorite programming language is, you might be surprised when I don't say Fortran. No, my favorite is Ada, the language named for the first computer programmer and the result of an international competition sponsored by the US Department of Defense. Jean Ichbiah, the creator of the "Green" language which became Ada, died January 26 at the age of 66.

I met Jean, briefly, back in 1984 when I was working on DEC's VAX Ada compiler project. In March of 1984 I had the delightful task of traveling to Versailles, France, to deliver to Ichbiah's company Alsys a magtape containing the first beta test version of VAX Ada. I spent a week with the Alsys team helping them shake out the compiler, which went on to be one of the most highly regarded implementations of the language. My main assignment from 1983 through 1988 was project leader for VAXELN Ada, a variant which ran on VAX systems under the real-time and embedded OS VAXELN, created by Dave Cutler just before he left DEC for Microsoft. In August 1988 I then joined the VAX Fortran compiler team.

Ada was an elegant and full-featured language with extremely expressive declaration features, multitasking, exception handling, a module facility with intelligent separate compilation and much more. The language gave the programmer the ability to tell the compiler what was allowed and not allowed to happen in the program and this enabled the compiler to do checking at a level rarely seen in other languages. I liked to say that if you could get an Ada program to compile, it would probably run correctly the first time. This, of course, was one of the things that the DoD wanted.

The DoD mandate that Ada must be used in defense contracts was both a blessing and a curse for Ada. A blessing in that it jumpstarted the widespread use of the language, but a curse in that many developers were dragged kicking and screaming into the world of Ada and non-defense programmers often avoided Ada specifically because of the DoD connection. After ten years, the screaming became loud enough that the DoD dropped the Ada mandate, and Ada use pretty much dropped out of sight. The original Ada 83 language was updated to Ada 88 and again in 1995, but DEC and most other vendors did not update their implementations.

What's the relevance of Ada to Fortran? Some of the major Fortran 90 features, such as modules and generics, are derived at least in part from Ada. Fortran's separate compilation model made it difficult to implement one of Ada's most elegant module features, IS SEPARATE, which permitted the implementation of a module procedure to be compiled separately from its declaration. The "submodules" proposal for Fortran 2008 finally brings that to the language.

So what's my second favorite language? SNOBOL."

[See also "Jean Ichbiah passes away" in this issue —su]

ARA — Ada helps a winner of Sun's Open Performance Contest

http://www.adaic.com/news/perfcont.html
Date: February 12, 2007

The Ada Resource Association congratulates Karl Nyberg of Grebyn Corporation, one of the winners of a T1000 server of a Sun Microsystems Sun Fire T1000 server valued at approximately US $15,000 in Sun Microsystems Open Performance Contest.

Karl's evaluation of the T1000, based upon his research "A Constructive Approach To Integer Factorization" against the RSA Factoring Challenge, was written in Ada. The application was implemented with many tasks working on parts of the problem simultaneously. Karl chose Ada for this project because of the elegance and simplicity of the Ada tasking model and select / accept statements. These constructs made mapping the work to multiple cores relatively simple, and allowed testing versions of the application on multiple platforms, including commodity PCs as well as the T1000, without modification.

It was very important to Karl to have a functional implementation of his algorithm quickly, so he could concentrate on performance improvements as his research progressed and as additional capabilities of the T1000 were understood and taken advantage of. Ada contributed to this goal.

Karl notes that “Ada just works out of the box and allows me to focus on the task at hand and write code that does what I mean for it to do rather than have to try to write code to convince the compiler to do what I want.” You can read more about Karl's research and his use of Ada in this contest on his website:

http://www.grebyn.com/t1000/

Ada-related Events

[To give an idea about the many Ada-related events organized by local groups, some information is included here. If you are organizing 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. —su]

April 17–19 — 13th International Real-Time Ada Workshop

From: Ben Brosgol <brosgol@adacore.com>
Date: 28 Dec 2006 02:08:55 −0500
Subject: Call for Participation: 13th International Real-Time Ada Workshop (Vermont, Apr 07)
Organization: AdaCore
Keywords: Ada, conferenceT
Newsgroups: comp.lang.ada,comp.realtime,comp.arch.embedded

IRTAW-13

17–19 April 2007

Woodstock, Vermont USA

For over 20 years the International Real-Time Ada Workshop series has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions. Well known for its high technical quality, the IRTAW series has attracted participation from key members of the Ada and real-time communities worldwide:

The next workshop in the IRTAW series will be held at the Woodstock Inn, in Woodstock, Vermont, in the northeast US, during 17–19 April 2007. To learn more about this event, including information about topics of interest and how to submit a paper, please read the Call for Participation (www.adaresource.org/irtaw13/cfp-irtaw13.pdf). The IRTAW-13 proceedings will be published in the August 2007 issue of ACM SIGAda's Ada Letters.

Please note that the deadline for submitting position papers is 12 January 2007.

Juan Antonio de la Puente, Technical University of Madrid (Program Chair), jpuente@dit.upm.es

Ben Brosgol, AdaCore (Local Arrangements Chair), brosgol@adacore.com

[See also "Sep 15–19 — 12th International Real-Time Ada Workshop" in AUJ 24-4 (Dec 2003), p.196. —su]

June 25–29 — Ada-Europe 2007

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Subject: 2nd CfIP, Reliable Software Technologies, Ada-Europe 2007
Date: Sun, 24 Dec 2006 15:58:02

Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Summary: 17 days until submission deadline!
Keywords: Conference,tutorials,reliable software,Ada,industry,LNCS,Geneva,ISO
Newsgroups: comp.lang.ada,fr.comp.lang.ada,comp.lang.misc

This call for industrial presentations is specifically targeted to those of you who either work in industrial projects (possibly Ada-related) where reliable software technologies are important, or know people working in such projects.

Please think for a moment what others might learn from the experience gained in those projects, and get a one-page presentation overview submitted by January 10th, i.e. 2.5 weeks from now.

Many projects could report a lot of valuable experience: sharing it with others benefits the whole community and might provide useful feedback as well.

We're looking forward to receive many interesting presentations.

Best wishes for the new year,

Dirk Craeynest, Ada-Europe'2007 Publicity Co-chair

2nd Call for Industrial Presentations

12th International Conference on

Reliable Software Technologies — Ada-Europe 2007

25–29 June 2007, Geneva, Switzerland

http://www.ada-europe.org/conference2007.html

Organized, on behalf of Ada-Europe, by Ecole d'Ingénieurs de Genève in cooperation with ACM SIGAda

General Information

The 12th International Conference on Reliable Software Technologies (Ada-Europe 2007) will take place in Geneva, Switzerland. Following the usual style, the conference will span a full week, including a three-day technical program and vendor exhibitions from Tuesday to Thursday, along with parallel workshops and tutorials on Monday and Friday.

Call for Presentations

In addition to the usual call for papers, and considering the success achieved in the previous conferences, we are having a call for presentations primarily aimed at industrialists who have valuable experience to report but who do not wish to write a complete paper.

This separate call for presentations is made for Experience Reports from Industrial Projects and/or Experiments, Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics and Experience Reports on Education and Training Activities, with bearing on any of the conference topics.

See below for further details.

Schedule

10 January 2007: Submission of presentation proposals

31 January 2007: Notification to authors

8 May 2007: Presentation material required

25–29 June 2007: Conference

Submission of Presentations

Presenters are invited to submit a one-page overview of the proposed presentation to Dominik Madon (dominik.madon@hesge.ch) by January 10th 2007. The Industrial Committee will review the proposals.

The authors of selected presentations shall prepare their final presentation, together with a short abstract (max 10 lines), by 8th May 2007; they should aim at a 20 minutes talk. The authors of accepted presentations will also be invited to derive articles from them, for publication in the Ada User Journal.

Exhibition

Commercial exhibitions will span the three days of the main conference. Vendors and providers of software products and services should contact the Exhibition Chair Neville Rowden (neville.rowden@siemens.com) as soon as possible for further information and for allowing suitable planning of the exhibition space and time.

Conference Topics

In the last decade the conference has established itself as an international forum for providers and practitioners of, and researchers into, reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a variety of application domains. The program will allow ample time for keynotes, Q&A sessions, panel discussions and social events. Participants will include practitioners and researchers from industry, academia and government organizations interested in furthering the development of reliable software technologies. To mark the completion of the technical work for the Ada language standard revision process, contributions that present and discuss the potential of the revised language are particularly sought after.

For papers, tutorials, and workshop proposals, the topics of interest include, but are not limited to:

- Methods and Techniques for Software Development and Maintenance: Requirements Engineering, Object-Oriented Technologies, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues

- Software Architectures: Patterns for Software Design and Composition, Frameworks, Architecture-Centered Development, Component and Class Libraries, Component-Based Design

- Enabling Technology: CASE Tools, Software Development Environments and Project Browsers, Compilers, Debuggers and Run-time Systems

- Software Quality: Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems

- Critical Systems: Real-Time, Distribution, Fault Tolerance, Information Technology, Safety, Security

- Distributed Systems: Reliability, Security, Trust and Safety in Large Scale Distributed Platforms

- Mainstream and Emerging Applications: Multimedia and Communications, Manufacturing, Robotics, Avionics, Space, Health Care, Transportation

- Ada Language and Technology: Programming Techniques, Object-Oriented, Concurrent, Distributed Programming, Bindings and Libraries, Evaluation & Comparative Assessments, Critical Review of Language Enhancements, Novel Support Technology, HW/SW platforms

- Experience Reports: Experience Reports, Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics, Experience Reports on Education and Training Activities with bearing on any of the conference topics

Organizing Committee

Conference Chair

Nabil Abdennadher, University of Applied Sciences, Geneva, Switzerland, nabil.abdennadher@hesge.ch

Industrial Committee Chair

Dominik Madon, University of Applied Sciences, Geneva, Switzerland, dominik.madon@hesge.ch

Industrial Committee Members

Bouali Amar, Esterel Technologies
Chapman Rod, Praxis HIS
Denker Peter, Parasoft GmbH
Devuns Olivier, Aonix
Gasperoni Franco, AdaCore
Leroy Pascal, IBM Rational
Stråhle Rei, Saab Systems
Thom Francis, Artisan Software
Abdennadher Nabil, Conference Chair
Plödereder Erhard, Ada-Europe (President)
Craeynest Dirk, Ada-Europe (Vice-President)

Conference Organization

Conference Chair

Nabil Abdennadher, University of Applied Sciences, Geneva, Switzerland, nabil.abdennadher@hesge.ch

Program Co-chairs

Nabil Abdennadher, University of Applied Sciences, Geneva, Switzerland, nabil.abdennadher@hesge.ch

Fabrice Kordon, University Pierre & Marie Curie, France, Fabrice.kordon@lip6.fr

Tutorial Chair

Dominik Madon, University of Applied Sciences, Geneva, Western Switzerland, dominik.madon@hesge.ch

Exhibition Chair

Neville Rowden, Siemens Switzerland, neville.rowden@siemens.com

Publicity Co-chairs

Ahlan Marriott, White-elephant, Switzerland, Ada@White-elephant.ch

Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium, Dirk.Craeynest@cs.kuleuven.be

Local Chair

Régis Boesch, University of Applied Sciences, Geneva, Switzerland, regis.boesch@hesge.ch

Ada and Education

Ada wikibook to be published

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Wed, 07 Feb 2007 13:22:04 +0100
Subject: [wikibooks] Ada Programming nominated for "Wikipublish"
Newsgroups: comp.lang.ada

The Wikibook "Ada Programming" [1] has been nominated to be published and distributed by the Wikipublish Wikiproject [2]. In fact: "Ada Programming" is the first book to be published.

Work is already on the way — with the first step of converting the book into LaTeX and then into PDF [3] which looks very nice indeed.

Still it is not to late to contribute — especially the reference section for pragmas and attributes could do with some additional work.

Or help out in the publication itself — You know Ada and LaTeX then go ahead.

[1] http://en.wikibooks.org/wiki/Ada_Programming

[2] http://en.wikibooks.org/wiki/Wikibooks:Wikipublish

[3] http://upload.wikimedia.org/wikibooks/en/9/9c/Dragontamer_Ada.pdf

Ada 95 training course

From: bex <andy.bissell@objektum.com>
Date: 29 Jan 2007 04:40:05 −0800
Subject: Ada 95 training in the UK — 14/02/07 — small group, limited space
Newsgroups: comp.lang.ada

We are running an Ada 95 training course in SE London, UK from 14th–16th Feb. 2007 for a small number of delegates. If you wish to make the transition from Ada 83 to Ada 95 or learn Ada 95 from scratch please see the link below:

http://www.objektum.com/objektum/indexcourse.asp?id=451

This course has previously been delivered to BAE Systems, MBDA, etc with excellent feedback.

Happy learning!

Praxis HIS — Q3 2007 Courses

Subject: Public Course Dates for 2007 — UK
URL: http://www.praxis-his.com/sparkada/training.asp

Course 1 — "Software Engineering with SPARK"

10th –13th September 2007 at the Praxis Offices in Bath. Download the booking form here.

Course 2 — "Black-Belt SPARK"

18th –20th September 2007 at the Praxis Offices in Bath. Download the booking form here.

DDC-I — Training Workshops SCORE Compilers

Hands-On Training Workshops Available for Developers using SCORE Compilers for Wind River Workbench

Phoenix, AZ. January 15, 2007. DDC-I, a leading supplier of development tools for safety-critical applications, offers "hands-on" training for customers using SCORE® compilers under Wind River Workbench. Jump Start and Jump Start Plus from DDC-I are intense multi-day workshops that provide different levels of assistance from a formal toolset introduction to advanced run-time system tailoring.

"Our training workshops are designed for engineering teams who want to maximize their productivity from day one," said Jennifer Sanchez, Manager of Marketing Communications for DDC-I. "We want our customers to know they made the right choice, and be able to see the value of their investment immediately. It's just another example of our ongoing commitment to exceptional customer service."

The integration of SCORE Compilers into the Wind River Workbench environment enables developers to utilize SCORE tools to develop mixed Ada, C, and Embedded C++ applications for deployment on VxWorks target systems. The result is a fully integrated solution which addresses all aspects of safety-critical application development.

SCORE currently supports VxWorks 6.3 under Wind River Workbench 2.5. Later this month, DDC-I will announce support for Wind River's latest version — VxWorks 6.4 under Wind River Workbench 2.6 which was just released.

Jump Start and Jump Start Plus training workshops are available immediately. Pricing for Intro Classes start at $5000 for DDC-I Atlas Advantage customers and is free for DDC-I Atlas Premium customers.

Ada-related Tools

Units of measurement for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 16 Jan 2007 20:30:29 +0100
Subject: ANN: Units of measurement for Ada v2.2
Newsgroups: comp.lang.ada

This new version contains a GTK+ (GtkAda based) widget for interactive unit selection. A corresponding dialog is provided as well.

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

[See also "Updates for Fuzzy sets for Ada, and Simple components" in AUJ 27-2 (Jun 2006) p.72 and "Units of measurement" in AUJ 26-2 (Jun 2005) p.75. —su]

NXTAda — Lego Mindstorms NXT

From: Jeffrey Creem <jeff@thecreems.com>
Date: Tue, 27 Feb 2007 23:02:24 −0500
Subject: NXTAda
Newsgroups: comp.lang.ada

I setup a Sourceforge project for controlling the Lego Mindstorms NXT device via Ada. The project does not currently intend to do a compiler port. It allows you to write code on your host computer (currently only Windows though I think a port to x86 Linux or x86 OS X would be pretty easy) to control the NXT module remotely via Bluetooth.

The code is very raw and only in SVN at the moment (no tar/zip releases yet) but I thought it was worth posting in case someone else was thinking of working on it.

http://nxtada.sourceforge.net/

GTKAda contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 16 Jan 2007 20:42:21 +0100
Subject: ANN: GtkAda contributions v1.4
Newsgroups: comp.lang.ada

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

Additions:

1. A fully annotated example of custom tree view model developing

2. Widget and dialogs for measurement unit selection (requires Unit of Measurement for Ada).

[See also same topic in AUJ 27-3 (Jun 2004), pp.136. —su]

Qt4Ada — Qt 4 bindings

From: Yves Bailly <kafka.fr@laposte.net>
Subject: Ada2005 binding to Qt4
Date: Sun, 10 Dec 2006 15:40:43 +0100
Newsgroups: comp.lang.ada

I'm quite pleased to announce the 0.1.0 release of Qt4Ada, a hand-made thick binding in Ada 2005 to Qt 4.2.x. See http://qt4ada.sourceforge.net for more details, grab the archive from

http://sourceforge.net/project/showfiles.php?group_id=173821&package_id=199116&release_id=470160

Qt4Ada is still in early stages, so it still lacks 95% of Qt features. However the 14 tutorials have been re-coded in pure Ada, so I guess it's already usable for very small and simple programs.

From now on:

- my primary goal will be to provide as much widgets as possible, which will be achieved by re-coding the "widgets" examples;

- the building structure needs improvements, so I'll start by trying to recreate a (basic) qmake-like tool;

- the signals/slots implementation mostly works, though it's not quite satisfactory, as already pointed by Vadim — in fact, it's the whole meta- objects structures that should be ported to Ada.

As I don't have much time to work on Qt4Ada, choices have to be made. The last point (about signals/slots), while annoying, is not my top priority for now. However any actual and concrete contribution would be much appreciated.

This library is released under the CeCILLv2 license, a French, GPLv2-compatible, open-source license (see http://www.cecill.info/index.en.html). I still have plans to provide an alternative license, allowing to use Qt4Ada in closed-source software, much like Qt is itself distributed. This double- licensing should not be done in the near future, but any comments or ideas about it are welcome.

Grab it, compile it, enjoy it (I hope) and provide feedback!

Please don't be too hard, I still lack experience in Ada programming ☺

[See also same topic in AUJ 27-3 (Jun 2004), pp.138–139. —su]

log4ada — log4j bindings

From: xavier <xavier@ipnnarval.in2p3.fr>
Date: Tue, 06 Feb 2007 15:53:08 +0100
Subject: log4ada
Newsgroups: comp.lang.ada

I have just started a project to connect to a java log server: log4j. The library is called log4Ada. It is hosted on the monotone server of Ada France (thanks Ada France, Thanks to Ludovic Brenta): org.log4ada

Two tests are available: test_console and test_socketappender (to connect to a log4j server).

This library is released under GPL, help yourself !

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 07 Feb 2007 10:58:28 +0100
Subject: Re: log4ada
Newsgroups: comp.lang.ada

> May I ask where do I find log4ada? Thanks.

It is a branch named org.log4Ada in Ada-France's Monotone database. I have just published an English version of the article that gives all the details; see

http://www.ada-france.org/article131.html

G2F_IO — ImageMagick bindings

From: Ali Bendriss <ali.bendriss@dementia.ion.ucl.ac.uk>
Subject: ANNOUNCE: G2F_IO an Ada 95 binding to the ImageMagick C API
Date: Mon, 8 Jan 2007 12:52:53 +0000
Newsgroups: comp.lang.ada

G2F_IO implement an Ada 95 binding to a subset of the low-level MagickCore library. Recently Olivier Ramonat and Pascal Obry have contributed to binding to make it running with the current version of Image Magick (6.x).

With this binding, it's now possible to

- Read/Write image

- Set/Get some image attribute (format, text, …)

- Create a new image (Canvas)

- Resize images and produce thumbnail

- Work at the Pixels level (read/write on the cache-view of the image)

The Project is now hosted on GNA: https://gna.org/projects/g2f/ You can download the current source code using subversion

- Checkout over SVN protocol (TCP 3690):
svn co svn://svn.gna.org/svn/g2f/trunk g2f

- Checkout over http:
svn co http://svn.gna.org/svn/g2f/trunk g2f

You may read this if you want to know more about ImageMagick: http://imagemagick.org/script/architecture.php

The home page will be updated soon http://home.gna.org/g2f/

Enjoy!

PS: I would like to create a high level API on top of G2F_IO (more Ada/Ada 2005? like) if you want to join please help yourself: https://mail.gna.org/listinfo/g2f-developers/

[See also "AdaMagick — ImageMagick Bindings" in AUJ 25-2 (Jun 2004), p.51. —su]

pgAda — PostgreSQL bindings

From: Maciej Sobczak <maciej@msobczak.com>
Date: Tue, 23 Jan 2007 15:30:47 +0100
Subject: Ada and PostgreSQL
Newsgroups: comp.lang.ada

I must have been quite a pain for you recently with my questions and nit-picking ;-), but at the end I have finished my exercise and built a *very* simple Ada client library for PostgreSQL. You can find it here:

http://msobczak.com/prog/bin/pgAda.tar.gz

Please consider it as a starting point for what should be a *true* database library, but there are also chances that in simpler projects it might be exactly what is needed.

Your comments are of course welcome.

[See also same topic in AUJ 25-3 (Sep 2004), p.123. —su]

Self Booting Hello World

From: freejack <freejack@tds.net>
Date: 1 Feb 2007 16:52:29 −0800
Subject: Self Booting "Hello World" Ada example code?
Newsgroups: comp.lang.ada

Has anyone written a simple self booting (i.e. off a floppy or some such) "Hello World" example program strictly in Ada?

I've been googling around the web to see how this is done, and haven't found any actual code.

I'd like to play around with doing bare-bone Ada hacking. Maybe even write a small hobby kernel. Just looking for some example code to get me started.

Any pointers would be appreciated.

From: Pascal Obry <pascal@obry.net>
Date: Fri, 02 Feb 2007 08:14:33 +0100
Subject: Re: Self Booting "Hello World" Ada example code?
Newsgroups: comp.lang.ada

Yes, look in the archive for the Toy Lovelace project. You'll find reference to it or wait for Xavier Grave response in this group.

Ada-related Products

AdaCore — GNAT Pro 6.0.1

Date: March 7, 2007
Subject: AdaCore Announces First to Market Full Ada 2005 Development Environments
RSS: www.adacore.com/category/press-center/feed/

Wednesday March 7, 2007

AdaCore Announces First to Market Full Ada 2005 Development Environments

NEW YORK and AMSTERDAM, Netherlands, March 7, 2007 — Avionics Exhibition and Convention — AdaCore, provider of the highest quality Ada tools and support, announces the first to market Ada 2005 language development environment, with the release of GNAT Pro version 6.0.1. Ada 2005, ISO/IEC 8652, was formally approved by ISO SC22/WG9 in January 2007. From the start AdaCore has actively participated in the ISO language standard revision process. This has enabled us to be at the forefront in supporting our customers and their use of the new Ada 2005 language standard.

"AdaCore has established a strong reputation of providing the industry's highest quality Ada tools and support for our customers," said Robert Dewar, President of AdaCore. "We are proud to be the first company to provide complete support for Ada 2005. AdaCore is now unique in the industry as the only vendor to support all three ISO versions of the Ada language. We support our customers working on existing long-lived Ada 83 systems. We support development teams using the current Ada 95 language. And we are now the first to support customers who want to start using the new ISO Ada 2005 language standard."

Ada 2005 is a refinement on an already strong foundation. The original Ada 83 language version introduced new programming language concepts including built in exception handling, generic program templates and multi processing tasks. Ada 95 added to this foundation by adding new deterministic task communication and Object Oriented programming features, making it the first ISO OO language standard. It also added special needs annexes to meet different industries' requirements, such as the Safety and Security Annex. This annex in particular standardized capabilities to further support an area where the language had already proven itself to be extremely valuable. This support has made Ada a leading language for avionics safety critical systems, such that it is now in use on almost every modern military and commercial aircraft flying or under development.

The new Ada 2005 language offers significant enhancements to software developers in several areas. Improvements in the language's Object-Oriented Programming features include the addition of Java-like interfaces and traditional "object.operation" syntax. More flexible program structuring allows mutually dependent package specifications and makes it easier to interface with languages such as Java. Real-time system support includes additional task dispatching policies such as Earliest Deadline First, execution-time clocks, and handlers for task termination. The concurrency and object-oriented features are successfully unified through a new interface feature that allows implementation through either a sequential or concurrent type.

Support for safety and security is enhanced with the inclusion of the Ravenscar Profile (a tasking subset that is amenable to safety certification), syntax that avoids some common Object-Oriented Programming errors with inheritance, and a mechanism for defining language profiles. Other enhancements increase the language's general expressiveness, for example by allowing nested subprograms to be passed as run-time parameters, and by extending the predefined environment with new functionality, such as a Containers library.

The 6.0.1 release also includes an enhanced version of the GNAT Programming Studio (GPS) IDE. GPS 4.1.0 offers programmers improved usability and efficiency through an advanced Outline View complete with new design and new features. Python and pygtk enable powerful scripting and customized dialog capabilities and are now supported on all platforms. Developers can make use of a wider range of plug-ins more effectively from within GPS thanks to enhanced support. A more intelligent smart completion engine coupled with automatic fixing for more compiler messages enables an all round smoother development process.

Pricing and Availability

Pricing for GNAT Pro subscriptions starts at $14,000. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a modern programming language designed for large, long-lived applications where reliability, efficiency and safety are critical. AdaCore's flagship product is GNAT Pro, which comes with expert online support and is available on more platforms than any other Ada technology. AdaCore has customers worldwide; see http://www.adacore.com/home/company/customers/ for more information.

Use of Ada and GNAT Pro continues to grow in high-integrity and safety-critical applications, including commercial and defence aircraft avionics, air traffic control, railroad systems, financial services and medical devices. AdaCore has North American headquarters in New York and European headquarters in Paris. www.adacore.com

Press Contact

Jessie Glockner Rainier Corporation (for AdaCore) Tel: 978-464-5302 x140 e-mail: adacore@rainierco.com

From: Romain Berrendonner <berrendo@adacore.com>
To: announce@adacore.com
Date: Thu, 15 Feb 2007 18:53:43 +0100
Subject: [AdaCore] Announcing the immediate availability of GNAT Pro 6.0.1

AdaCore is pleased to announce the immediate availability of GNAT Pro 6.0.1.

GNAT Pro 6.0.1 is a major release introducing many new features, notably, complete support for Ada 2005 and a new code generator for most platforms. Other improvements and new features are described in the release note section in GNAT Tracker and in the files features-Ada 2005 and features-60 distributed with the release.

GNAT Pro 6.0.1 is available for the following platforms:

alpha-tru64
ia64-hp_linux
ia64-hpux
ia64-sgi_linux
pa-hpux
ppc-aix
mips-irix
sparc-solaris
sparc64-solaris
x86_64-linux
x86-linux
x86-solaris
x86-windows
ppc-elf-windows
ppc-elf-solaris
ppc-vxw-solaris

Other platforms will follow in the coming weeks.

Further announcements:

Please note that with the introduction of the remote programming function, the

GNAT Programming Studio IDE is now available to all customers (except OpenVMS) for use on your local Windows or GNU/Linux machines. For more information on this innovative capability, please visit:

www.adacore.com/home/gnatpro/remote-programming

Support is available for two separate native SPARC Solaris platforms:

- 32-bit SPARC Solaris
- 64-bit SPARC Solaris.

Both products includes a GNAT shared run-time.

Support for XML/Ada, the Ada library for processing XML streams, is now included as part of the general GNAT Pro subscription package. If you are interested in adding support for XML/Ada to your account, please contact sales@adacore.com.

All distributions can be downloaded as usual using GNAT Tracker. We encourage you to install and start using this latest version of the GNAT Pro tool suite. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email to the usual report@adacore.com address.

[See also "AdaCore — GNAT Pro Preview release" in AUJ 27-3 (Sep 2006) p.142 and "AdaCore — GNAT Pro 5.04a1" in AUJ 27-3 (Sep 2006), p.143. —su]

AdaCore — gprmake improved

New features for multi-language tool
http://www.adacore.com/2007/01/15/new-features-for-multi-language-tool/

Monday January 15, 2007

AdaCore's multi-language program build tool, gprmake, has been updated to provide a number of important enhancements. These include:

- support for Ada, C and C++ by default

- support for multi-language libraries

- support for new languages and/or toolchains through configuration files

- independence from a specific GNAT Pro version (that is, the same gprmake will work with different GNAT Pro releases)

A beta program for the new gprmake will be initiated later in 2007 and will be open to all AdaCore customers.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Organization: Adalog
Subject: AdaControl V1.6 released
Date: Wed, 06 Dec 2006 16:44:39 +0100
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of version 1.6r8 of AdaControl, the free rule checker for Ada.

Thanks to the support of our new customer SAGEM-DS and contributions from R. Toy, AdaControl now offers 216 possible checks.

Of special interest are rules to check that header comments match a given pattern, indication of possible false positive and false negative due to non-statically analyzable constructs, fine definition of constructs allowed in entry barriers (including the one of the Ravenscar profile), even better integration into GPS, and much much more.

As usual, AdaControl is provided under the GMGPL license, and can be downloaded from http://www.adalog.fr/adacontrol2.htm.

AdaControl is a commercial product of Adalog; for information about support and assistance with AdaControl or more generally issues related to coding rules enforcement, please write to info@adalog.fr

http://www.adalog.fr

Aivosto — Visustin v4

Visustin v4 Flow chart generator

Visustin icon Visualize your source code with flow charts. Open up your code in Visustin to see its flow chart or UML Activity Diagram.

Visustin is the ideal diagramming tool for software developers and document writers. Visustin diagrams Ada, ASP, assembly language, BASIC, C/C++, C#, Clipper, COBOL, Fortran, Java, JSP, JavaScript, LotusScript, Pascal/Delphi, Perl, PHP, PL/SQL, PowerScript, PureBasic, Python, QuickBASIC, REALbasic, T-SQL, VB, VBA, VB.NET and Visual FoxPro code.

Save your documentation efforts by automatic charting! Visustin reverse engineers your source code and visualizes it as flow charts or UML Activity Diagrams. Visustin reads the if and else statements, loops and jumps and builds a diagram — fully automated. No manual drawing is required. Your existing code is all you need. If you see a real complex case, print it out as a big mosaic and hang it on your wall.

Diagrams help you know your code. With your in-depth knowledge you fix bugs and write improvements faster than ever before. Review algorithms. Verify program logic. Document complex functions. Restructure incomprehensible code.

Automated layout. Visustin creates an optimal visual layout automatically. Just hit one key and you're done — no need to adjust the charts.

Flow charts include all of your code, optionally the comments as well. Create large master charts or small charts with just the important logic.

UML Activity Diagrams do the same in UML style. New! Pick your preference or do both UML and flow charts.

Multi-page print. Print large flow charts on multiple pages, or squeeze to fit on one sheet.

Save graphs. Use flow charts in your project documentation in GIF, PNG, BMP, JPG, PCX, TGA, PPM, PGM, WMF, EMF or PS image format.

Visio export [Pro Edition] Export your flow charts to Visio 2002/2003. Save your drawing efforts by converting code to Visio diagrams. Edit and adjust the charts. More about Visio exportPopup link

Bulk charting [Pro Edition] Flowchart all your source files in one run. Also exports Visio .vsd files.

PowerPoint export. Create flow chart slide shows. New!

Web publish. Save flow charts as web pages or MHT web archives.

Word export. Create flow chart .doc's. New!

Aonix — ObjectAda RAVEN for PikeOS

Aonix Releases ObjectAda® Real-Time RAVEN™ for PikeOS

Aonix is pleased to announce the release of ObjectAda Real-Time RAVEN V8.2 for Intel-based Linux platforms targeting PowerPC/PikeOS. ObjectAda's significantly enhanced compiler and debug technology works with PikeOS, a product of SYSGO, the European vendor of reliable device software. PikeOS is a real-time separation microkernel technology for safety-critical systems, which allows separation of costly DO-178B Level A certifiable code from other portions of the application. The implementation makes full use of PikeOS' virtualization capabilities, thus allowing real-time applications such as Ada and traditional Linux to run reliably side by side in different partitions.

PikeOS controls microkernel access to the hardware and allows multiple software partitions to execute on a single CPU with strict separation between them. Each partition can either run application programs or an entire operating system such as Linux, POSIX or ARINC653. This flexibility enables control of applications running under these uniquely different systems to execute in parallel with visualization software under Linux or Java™. Sophisticated, but potentially untrusted applications such as Linux, can be separated from critical components and can therefore be integrated in a safety-critical system. If certification is required,

only the safety-critical components need to be certified.

"The safe code structures of ObjectAda Real-Time RAVEN are invaluable to safety-constrained applications," states Torsten Voegler, marketing manager at SYSGO. "By combining the Aonix safety-critical development and runtime environment with our multipartition strategy, developers are empowered to build more elegant systems with standard, off-the-shelf components and still meet stringent certification requirements. The combination of ObjectAda Real-Time RAVEN and the PikeOS microkernel provides the solution many of our customers have requested."

ObjectAda Real-Time RAVEN for Intel/Linux targeting PowerPC/PikeOS is an embedded Ada development system that allows engineers to build applications in a Linux environment for deployment in appropriately configured partitions of PikeOS running on a PowerPC platform. The product consists of a fully compliant ACATS 2.5 Ada 95 compiler with supporting tools including a build/bind tool, library tool and debugger, and delivered with a predefined program library which conforms to the Ravenscar profile subset of the full predefined language. It is compatible with PikeOS 1.3 and the PowerPC OEA CPU architecture.

Aonix — AonixADT 3.2

Aonix Delivers ADT, an Eclipse-based Ada IDE for Windows, Linux, and Solaris Platforms

Aonix is pleased to announce the release of AonixADT™ Version 3.2.1 —an Eclipse-based Integrated Development Environment (IDE) for the Ada language. Building on the wealth of available plug-ins for Eclipse, Aonix has further extended the AonixADT (Ada Development Toolkit) to support ObjectAda Versions 8.2, and 8.3, and GNAT version 5.03a+.

In addition to adding support for Sparc Solaris and Intel Linux platforms, the latest major release of AonixADT includes improved Code Assist functionality, enhanced debugger support including low-level debugging, debugging of already running processes and extended breakpoint functionality, configurable toolchain customization and configurable file creation wizards.

AonixADT provides Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, along with a complete Ada debugger interface. ADT project awareness allows full library hierarchy manipulation and Ada program units can be conveniently inserted or removed from Ada projects. The language-sensitive editor provides complete language awareness with syntax color coding and template completion. Symbolic debugging is integrated within the Ada-language sensitive editor. The build interface offers complete access to the Aonix ObjectAda compile and build capabilities.

Aonix — ObjectAda RAVEN for PowerPC

Aonix Enhances ObjectAda Real-Time and Safety-Critical Products New features provide "no-cost" Eclipse plug-ins to Embedded Developers

Embedded World, Nürnburg, Germany, February 13, 2007

Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda Real-Time RAVEN V8.3 for Windows platforms targeting PowerPC. ObjectAda Real-Time Raven implements the Ravenscar profile, a restricted subset of the standard Ada runtime environment for applications requiring safety certification or a high-level of confidence in proven and fully tested runtime execution predictability.

ObjectAda Real-Time RAVEN V8.3 continues the Aonix legacy of delivering certifiable applications to both commercial and government safety-critical projects in avionics, space, high-speed rail, and nuclear industries. Aonix gained its solid reputation in the safety-critical field by designing tools that comply with market standards and has provided safety-critical solutions to a myriad of commercial and defense projects including International Space Station, Boeing 777, Rafale Multi-Role Combat Fighter, C130-J Hercules, Airbus A 330-340, and NH90 Helicopter. For systems not requiring formal certification, ObjectAda Real-Time Raven provides the assurance that the Ada runtime used in resource-constrained systems has been rigorously proven and tested.

ObjectAda Real-Time RAVEN V8.3 allows developers to choose between the traditional Aonix IDE for development and the new AonixADT™ Eclipse plug-in. Geared to maximize developer ease and efficiency, AonixADT incorporates Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, and a complete Ada debugger interface, enabling Ada developers to enjoy state-of-the-art interface capabilities. AonixADT is being added to ObjectAda at no additional cost.

"Continued improvement of the ObjectAda product in support of the Ravenscar profile demonstrates Aonix's commitment to support complex and rigorously stressed mission-critical systems with standard software development platforms," noted Gary Cato, Aonix Director of Strategic Alliances. "Full Eclipse support for hard real-time and safety-critical embedded development adds another great development asset to our product line that our customers are eager to use. Such standard platforms take COTS integration to a new level of easy to use and cost-effective solutions."

AonixADT implements plug-ins compatible with Eclipse standards V3.1, 3.1.1 and 3.1.2. Eclipse is an open-source software development project dedicated to providing a robust, full-featured, commercial-quality, industry platform for the development of highly integrated tools. A strong supporter of the Eclipse Foundation, Aonix has gone to great lengths to lock-step with evolving Eclipse specifications. Aonix has released ADT plug-ins with ObjectAda V8.2 native development products for Windows, Intel/Linux, and Sparc/Solaris platforms.

Shipping and Availability

ObjectAda Real-Time Raven for Windows platforms targeting the PowerPC processor family is immediately available. Prices range from $15,000 to $30,000 in the U.S. depending on bundle options plus runtime license fees. Quantity discounts are available. DO-178B certification materials are available and priced based on board support package and other project-specific requirements.

About Aonix

Aonix offers mission- and safety-critical solutions primarily to the military and aerospace, telecommunications and transportation industries. Aonix delivers the leading high-reliability, real-time embedded virtual machine solution for running Java™ programs deployed today and has the largest number of certified Ada applications at the highest level of criticality. Headquartered in San Diego, CA and Paris, France, Aonix operates sales offices throughout North America and Europe in addition to offering a network of international distributors. For more information, visit www.aonix.com.

Aonix — "zero-cost" license model ObjectAda for Linux

Aonix Shatters Ada Price Barrier for Linux

Eclipse-based ObjectAda for Linux Available with No-Cost Licensing

San Diego, January 31, 2007

Aonix®, a provider of solutions for safety- and mission-critical applications, announced a new "zero-cost" license model for its ObjectAda for Linux product. Recognizing the expectations of the Linux community, Aonix has introduced pricing for ObjectAda 8.2 for Linux that emulates what is used for its industry-leading PERC product line. This price model focuses on customer service packs rather than development licenses.

ObjectAda 8.2 for Linux provides a full complement of mature Ada technologies with an integrated Eclipse-based environment for Red Hat Enterprise Linux, Fedora Core and most equivalent x86 Linux distributions.

In conforming ObjectAda for Linux to the new Aonix service-pack model, Aonix has chosen to extend the zero-cost licensing of its best-selling PERC product line used by real-time Java developers to Aonix Ada developers working on the Linux platform. ObjectAda for Linux provides a robust development environment that includes a fully validated optimizing compiler, library manager, runtime, configuration management integrations, life-cycle tools, and a productivity toolset that includes an editor, browser, and debugger. Both a traditional Aonix IDE and the AonixADT plug-in set for Eclipse are provided. With this new pricing, Linux developers are able to choose a service-pack or a traditional user-based model.

“Aonix is committed to delivery of products to our customers in a way that fits the operational mode of the development community,” said Dave Wood, Aonix VP Marketing. “Linux developers are accustomed to service packs, a pricing model that we have successfully used with our PERC product line. Notably, this pricing approach makes commercial-grade Ada available to the wider Linux audience, much in the same way that our groundbreaking ObjectAda for Windows did for the Windows community.”

Since Aonix launched it in 1996 with a Visual C++ style IDE and pricing far below any previously seen for commercial Ada products, ObjectAda for Windows has been by far the top-selling Ada environment. Its installed base now includes many tens of thousands of units. The AonixADT Eclipse environment and service-pack pricing extend similar benefits to the Linux community.

AonixADT incorporates Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, and a complete Ada debugger interface, enabling Ada developers to enjoy state-of-the-art interface capabilities geared to maximize developer ease and efficiency. Developers can focus on building applications, not on integrating tools since AonixADT also retains a large set of existing plug-ins for third-party tools, including support for source-code configuration management.

Shipping and Availability

ObjectAda for Linux V8.2 is available immediately. There is no charge for annual development licenses with an active service plan. Service plans are priced at $3000 for a single user or $12,000 for a 5-user service pack.

From: Dave Wood <dave.wood@aonix.com>
Date: 26 Feb 2007 09:22:00 −0800
Subject: Re: recent changes in compiler pricing
Newsgroups: comp.lang.ada

In the traditional price model, there is a price paid both for the development license and also a price paid for annual support. In the new price model, there is no charge for the development license. Hence, the development license is zero cost. A support subscription, however, is not free, nor is it claimed as such.

For those who prefer a perpetual development license with an optional support contract, we continue to offer the traditional model as well.

If you are interested in doing a what-if on which model works best for your situation, I'd direct you to an Aonix account manager. If you're interest in the subject is more "academic", feel free to email me directly as ordinarily I don't monitor this group.

Dave Wood, VP Marketing, Aonix

DDC-I — SCORE IDE for TMS320C40 DSP

DDC-I Announces Availability of SCORE Integrated Development Environment for TMS320C40 DSP

Provides seamless upward migration path from Ada 83 to mixed Ada 95/Embedded C++ for legacy C40 code

Phoenix, AZ. December 4, 2006. DDC-I, a leading supplier of development tools for safety-critical applications, today announced the availability of its SCORE® Integrated Development Environment (IDE) for Texas Instrument's TMS320C40. The SCORE IDE makes it easy for C40 developers to take existing Ada 83 programs developed for the C40, upgrade them using a mixture of Ada 95 and Embedded C++, and deploy them on a royalty-free Ada 95 run-time system. The SCORE IDE also makes it easy for C40 developers to migrate their code to other processors such as the PowerPC and X86, with the unique ability to debug multiple targets and languages at the same time.

"There has been a lot of Ada 83 code developed for the C40, particularly in defense applications," said Bob Morris, president and CEO of DDC-I. "SCORE provides a modern, best-in-class mixed language development environment that makes it easy for C40 developers to upgrade their Ada 83 code and take advantage of the latest Ada 95 and Embedded C++ technology. SCORE also makes it easy for developers to migrate existing C40 code to new processors."

To support the C40, DDC-I has developed a new C40 compiler, code generator, and disassembler. The SCORE IDE provides full JTAG multiprocessor debugging for the C40, including trace and the ability to monitor all registers. SCORE also provides a PC-based C40 instruction set simulator.

SCORE® is a mixed-language, object-oriented IDE for developing and deploying safety-critical applications. SCORE provides optimizing compilers for Ada, C, Embedded C ++, and Fortran77, all of which pass the applicable ACATS, PlumHall, Perennial, and FCVS compiler validation suites.

The SCORE® IDE features an intuitive GUI with industry leading features such as a color-coded source editor, project management support, and automated build/make utilities. SCORE's mixed-language, multi-window, symbolic debugger recognizes C/EC++, Ada and Fortran syntax and expressions, and can view objects, expressions, call chains, execution traces, interspersed machine code, machine registers, and program stacks. The debugger supports full Ada-level debugging, including constraints, attributes, tasking, exceptions, break-on-exception and break-on-tasking events. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

SCORE provides versatile run-time target options, including a bare run-time system certifiable to Level A of the FCC DO-178B standard, and an enhanced bare run-time system for simulated and emulated environments.

About DDC-I, Inc.

DDC-I, Inc. is a global supplier of software development tools, custom software development services, and legacy software system modernization solutions, with a primary focus on safety-critical applications. DDC-I's customer base is an impressive "who's who" in the commercial, military, aerospace, and safety-critical industries. DDC-I offers compilers, integrated development environments and run-time systems for C, Embedded C++, Ada, JOVIAL and FORTRAN application development.

DDC-I — SCORE Compilers for Workbench 2.6 and VxWorks 6.4

DDC-I Announces Support for Enhanced Wind River Workbench and VxWorks

SCORE Compilers Available for Wind River Workbench 2.6 and VxWorks 6.4

Phoenix, AZ. February 13, 2007. DDC-I, a leading supplier of development tools for safety-critical applications, today announced support for Wind River Workbench 2.6 and VxWorks 6.4, both of which were released in December 2006.

The integration, which marks three generations of SCORE support for VxWorks, enables developers working within Wind River Workbench to utilize SCORE tools to develop mixed Ada, C, and Embedded C++ applications for deployment on VxWorks target systems.

"The latest release of Wind River Workbench contains significant enhancements that simplify the development process and provide access to the latest Eclipse technology," said Bob Morris, president and CEO of DDC-I. "We believe these enhancements will be very attractive to Workbench developers who want to utilize our SCORE compilers to create mixed-language, safety-critical applications targeting both VxWorks and bare board run-time systems."

"Wind River is committed to enriching our customers' overall development experience by increasing interoperability with other software tools and fostering collaboration among hardware engineers, software developers and testers within a project team," said Andrew Lyons, director of tools product management at Wind River. "The integration of DDC-I's SCORE tools with our enhanced Workbench development suite makes it easier than ever for developers using a mix of C, C++ and Ada to create reliable, optimized code for a broad range of safety-critical applications targeting VxWorks systems."

Wind River Workbench 2.6 provides a number of significant enhancements, including support for the new Eclipse 3.2.1 framework. To support existing Eclipse users, Workbench can now be installed as a set of plug-ins to an existing 3.2 installation, thereby enabling users to preserve existing Eclipse projects and configurations. Workbench 2.6 also features new plug-ins for VxWorks, including support for on-chip debugging, enhanced performance for projects with a large number of files, and an enhanced kernel object viewer.

SCORE provides optimizing compilers for Ada, C, and Embedded C ++, all of which pass the applicable ACATS, PlumHall, Perennial, and FCVS compiler validation suites. To support VxWorks, DDC-I has mapped its own bare run-time system to VxWorks, including all system calls, multitasking, and interrupt processing facilities.

Green Hills — AdaMULTI 5.0

Green Hills Software Announces Version 5.0 of its Compiler Suite

New Optimizations Further Lead in Generating the Best Code SANTA BARBARA, CA — January 31, 2007— Green Hills Software, Inc., the technology leader in device software optimization (DSO) and real-time operating systems (RTOS), today announced the release of its next-generation compiler technology, part of Green Hills' integrated development environment MULTI version 5.0.

"For twenty-five years, Green Hills Software has worked diligently to advance its compiler technology, incorporating novel techniques that enable software developers to minimize memory footprint and power usage as well as maximize execution speed of their code," commented David Kleidermacher, chief technology officer of Green Hills Software. "This in turn translates into reduced production cost and higher performance of our customers' end products. At the same time, companies rely on Green Hills' compilers for the success of their products. Green Hills compilers build the code that runs many automotive drive trains, aircraft engines, medical devices, and other critical systems."

The MULTI 5.0 compiler yields a significant code density and speed improvement over the previous generation of Green Hills' compilers, including a 14% performance improvement in the EEMBC Telecom benchmark. EEMBC is an independent consortium of microprocessor manufacturers that runs and independently certifies compiler benchmark scores on all the leading embedded processors used today. Green Hills has long been the predominant compiler selected by microprocessor vendors to demonstrate the highest possible performance for their products.

Optimizations

Some of the most important optimization advancements are inter-procedural: whereas traditional compilers process one source code file at a time, the Green Hills compilers are able to examine the entire application program in order to locate optimization opportunities. The new compiler also provides greatly enhanced support for profile-driven optimization: the run-time profile of the user application can be fed back into the compiler which will then optimize based on the specific run-time characteristics of the application. This enables the Green Hills compiler to tune itself according to the real-world execution environment that is most important to the fielded end product.

The new compiler also provides a number of new C++ optimizations, including enhancements in the efficiency of exception handling and dramatic code density improvements in programs that make heavy use of virtual methods, something commonly found in complex applications such as software defined radios. Finally, the MULTI 5.0 compiler has a wide range of new optimizations targeting specific microprocessor families, including Power® Architecture, ARM, MIPS, V850, ColdFire, Intel® IA-32, and Blackfin.

Compile Speed

Using MULTI's integrated distributed build system, the Green Hills compilers can build programs in parallel across the underutilized workstations on a corporate network. Distributed compilation is easy to configure and use, with all the details of source code distribution and parallel compile management performed on behalf of the user. The result is a typical 30 to 80% decrease in compilation time for a full project build.

Standards and Reliability

The Green Hills compiler was the first compiler for embedded systems to achieve 100% conformance to ANSI/ISO standards for C and C++. In addition, the new compiler supports the latest C99 specification and the latest MISRA C standard. The Green Hills compilers are tested against industry standard validation suites, including Plum Hall, and are also tested against the industry's most proven and extensive regression test suite.

Availability

Green Hills compilers for C, C++, and Ada are available today.

About Green Hills Software

Founded in 1982, Green Hills Software, Inc. is the technology leader in device software optimization (DSO) and real-time operating systems (RTOS) for 32- and 64-bit embedded systems. Our royalty-free INTEGRITY® and velOSity™ real-time operating systems, µ-velOSity™ microkernel, compilers, MULTI® and AdaMULTI™ integrated development environments and TimeMachine™ tool suite offer a complete development solution that addresses both deeply embedded and high-reliability applications. Green Hills Software is headquartered in Santa Barbara, CA, with European headquarters in the United Kingdom. Visit Green Hills Software on the web at www.ghs.com.

Green Hills, the Green Hills logo, MULTI, INTEGRITY, velOSity, µ-velOSity, AdaMULTI and TimeMachine, are trademarks or registered trademarks of Green Hills Software, Inc. in the U.S. and/or internationally. All other trademarks are the property of their respective owners.

Headway Software — Structure101 for Ada

From: pth81500@gmail.com
Subject: Analyzing & Measuring the Architecture or Structure of Your Ada Code
Date: 11 Jan 2007 00:29:50 −0800
Newsgroups: comp.lang.ada

We are preparing the first release of Structure101 for Ada, and we would be

grateful for your feedback on the early access release.

As the name suggests, Structure101 for Ada is all about understanding, controlling and measuring the quality of your Ada software structure (or architecture). So if you have any old Ada code lying around ;-), that you would like to understand better, give Structure101 for Ada a whirl and let us know what you think.

Structure101 for Ada is a third generation structural analysis product, replacing its predecessor, Headway Review. It is mature technology that is easy to get started with and which delivers immediate benefits. At least that's what the Java folks are telling us!

It can be downloaded from here, http://www.headwaysoftware.com/downloads/structure101/ada.php. You will need to pick up the GNAT based parser and required version of GNAT at the bottom of the page, as you will need them to generate the input for Structure101 from your Ada code. With the current release your code is required to compile (although not necessarily run) with GNAT.

Note: Once you have downloaded all the required software you can grab a license key by clicking on the link on the left hand side menu of the downloads page or simply drop us an email.

Your help and feedback would be very much appreciated.

[See also "Headway Software — Headway reView" in AUJ 27-3 (Sep 2006), p.144–145. —su]

McKae Technologies — Avatox 1.4

From: Marc A. Criley <mc@mckae.com>
Organization: McKae Technologies
Subject: Announce: Avatox 1.4 now available
Date: Wed, 13 Dec 2006 19:23:40 −0600
Newsgroups: comp.lang.ada

Avatox (Ada, Via Asis, To Xml) is an application that traverses an Ada compilation unit and outputs the ASIS representation of that unit structured as an XML document (Avatox Xml Format, .axf). The format of the XML in the document can be configured, and supplemental source annotations can be generated.

Changes since version 1.3:

- Fixed a bug that caused duplicated attributes for some elements.

- Added additional axfPoint element types: axfNumber and axfScope.

From: Marc A. Criley <mc@mckae.com>
Organization: McKae Technologies
Date: Sun, 14 Jan 2007 19:48:50 −0600
Subject: Announce: Avatox 1.5 now available
Newsgroups: comp.lang.ada

Changes since version 1.4:

- Now accepts multiple filenames on the command line, including wildcarded filenames.

- Can direct that the supporting units ("withed" units) or closure units of the explicitly specified units be collected and transformed into AXF as well.

- Compilation units identified as supporting or closure units can be filtered by regexp or file-style wildcard filtering.

- Included an XSL stylesheet, deleteCLInfo.xsl, to declutter an AXF file by removing all line and column information from elements. […]

From: Marc A. Criley <mc@mckae.com>
Date: Sat, 17 Feb 2007 10:29:19 −0600
Subject: Announce: Avatox 1.6 now available (now with built-in XSLT)
Organization: McKae Technologies
Newsgroups: comp.lang.ada

[…] The ability to perform XSL style sheet transformations is now built-in so that transformations can be immediately performed on the generated AXF files.

Changes since version 1.5:

- Added built-in support for XSL transformations by specifying a stylesheet and associated options on the command line.

- Provides finer control over axfPoint element generation. Previously it was all or nothing, now specific axfPoint element kinds can be selectively generated.

- Fixed a bug when doing multiple file generation and a units' specs and bodies were among those for which AXF was being generated.

(For those who may be interested in performing XSL stylesheet transformations in Ada, the McKae.XML.XSL.* file hierarchy contains a (very) minimal binding to libxslt sufficient to take an XML file and a file containing a stylesheet and perform the transformation.)

Avatox 1.6 is available at www.mckae.com/avatox.html.

Marc A. Criley, McKae Technologies

[See also "Avatox — Ada To XML" in AUJ 27-4 (Dec 2006), p.201. —su]

Praxis HIS — SPARK Toolset 7.4

From: Rod Chapman <roderick.chapman@gmail.com>
Subject: ANN: SPARK 7.4 now available
Date: 9 Jan 2007 06:28:01 −0800
Newsgroups: comp.lang.ada

Praxis are pleased to announce the immediate availability of release 7.4 of the SPARK language and toolset.

Full details, including the toolset release note, are available from www.sparkada.com as usual.

Professional, supported customers will receive upgrades immediately. Upgrade packages for for readers of the SPARK Textbook are also available by download from http://www.praxis-his.com/sparkada/sparkbook.asp

Highlights of this release include:

- New "accept" annotation system to indicate that a particular error or warning is expected and justified.

- New "Always_Valid" assertion to indicate that the values read from an external input are trustworthy.

- Obsolete SPARK83 floating-point attributes are now acceptable in SPARK95 mode.

- Better error messages for common syntax and semantic errors.

- Complete re-implementation of VC Generation for single- and multi-dimensional unconstrained array parameters. Supporting improvements in the default invariant generator and Simplifier.

- Conditional data- and information-flow anomalies are now reported as errors not warnings.

- Support for System.Bit_Order and System.Default_Bit_Order in the configuration file.

- The Examiner now issues a warning if an Ada 2005 reserved word is used as an identifier in SPARK95 mode.

- The Simplifier's handling is user-defined and Examiner-generated proof-rules has been unified and improved.

- The Simplifier has a new family of proof tactics for enumerated and integer inequalities where transitivity of the relational operators is involved.

- The implementation of the /p=N (multiprocessor) switch in SPARKSimp has been re-implemented to make much better use of all the available processing resources on multi-core or multi-processor machines.

- SPARKFormat now has options to reformat the own, initializes and inherit annotations.

- SPARKMake can now produce with absolute or relative pathnames in the generated index and meta-files.

[See also "Praxis HIS — SPARK Toolset 7.31" in AUJ 27-2 (Jun 2006), p.77. —su]

Ada and CORBA

GNACK — GNU Ada CORBA Kit

From: Oliver M. Kellogg <okellogg@freenet.de>
Date: 31 Dec 2006 05:16:05 −0800
Subject: CORBA Ada bindings
Newsgroups: comp.lang.ada

Version 1.2 of the GNU Ada CORBA Kit (GNACK) has been released. Main feature of this version is the switch from ORBit-1 to to ORBit-2.

http://sourceforge.net/projects/orbitada

[See also "GNACK and ORBit" in AUJ 23-2 (Jun 2002), p.76. —su]

Ada and GNU/Linux

ARM in texinfo

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 1 Feb 2007 17:07:18 −0600
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

> Is there a texinfo version of the Ada 2005 Reference Manual? I have it on paper (courtesy of Ada-Europe) but I'd like to be able to search it from within Emacs:)
> If no such version exists, is anyone working on one, or considering working on one?

Stephen Leake did the module for converting the RM source to Texinfo as an add-in the processing program for the RM+Corrigendum version. I didn't try to maintain that module while working on Ada 2005, because I don't know anything about Texinfo. I did talk to Stephen about it at one point, and we agreed it would be best to wait until the RM was finished. Of course that has happened; I don't know if he is still planning to update his Texinfo module.

In the mean time, you'll have to live with the HTML version and its search engine. I just leave a browser page open to it at all times (saves reloading it repeatedly).

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sat, 10 Feb 2007 22:09:06 −0500
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

> I found the sources, at http://www.ada-auth.org/arm.html
> I've downloaded them, and got an initial compile. Running it on the ARM sources dies with an error in my info code. There are several new dispatching functions I need to implement, so it will be a while. But I'll work on it.

Well, it turned out to be easier than I thought. I've posted an initial draft of the ARM and AARM in Info format on my website; http://stephe-leake.org/ada/arm.html

I looked at the places where I needed to add new code, and spot-checked a few other things. But I have _not_ read the whole info output.

I'd like people to compare a couple sections to the other formats, or just read some sections, and see if there are any obvious problems.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 10 Feb 2007 21:56:14 −0600
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

I'm not too surprised. Many of the new capabilities are used in the Rationale, or the ASIS Standard, or in RR's manuals (that is, not in the RM). For instance, I don't think there are any pictures (images) in the RM, while there are some in all of the other documents.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Mon, 05 Feb 2007 07:55:34 −0500
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

> Thank you! That's a great service to the Ada community.

You're welcome. It's nice to be appreciated :).

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 5 Feb 2007 05:16:21 −0800
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

I'd like to second Bob's kudos, not only for myself who uses the info version of the ARM almost daily, but also on behalf of all users of Debian who get it as part of the ada-reference-manual package (I'm not the maintainer of that package; Florian Weimer deserves the credit for that).

From: Georg Bauhaus <bauhaus@arcor.de>
Date: Mon, 05 Feb 2007 15:10:41 +0100
Subject: Re: Ada Reference Manual in texinfo format?
Newsgroups: comp.lang.ada

Thanks on behalf of the users of Ubuntu GNU/Linux, too, who also use the nice Debian packages.

Ada in Debian's Popularity Contest

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 5 Feb 2007 08:54:25 −0800
Subject: Ada is popular after all
Newsgroups: comp.lang.ada

I quickly looked at the results of Debian's Popularity Contest, which ranks packages in Debian according to their popularity. (This ranking helps choose which CD-ROM or DVD-ROM each package ships on. There are currently 23 CD-ROMs or 3 DVD-ROMs for i386 alone.)

I only looked at the "zen master" languages and at the "votes" column in the popularity contest results. Here is what I found:

Language	Package	Votes
Ada	gnat	98
Pascal	fp-compiler	65
Pascal	gpc	55
Eiffel	smarteiffel	20
Modula-2	m2c	4
Oberon	oo2c	1

Granted, the two Pascal compilers combined beat GNAT, but just look at the graph on [http://popcon.debian.org] for the evolution since 2004. Something's happening.

Another thing that makes Ada trendy nowadays is the enduring series of articles by Yves Bailly in GNU/Linux Magazine France. The December issue contains article #14 in the series, ending with a mention of "the next article"…

From: Yves Bailly <kafka.fr@laposte.net>
Subject: Re: Ada is popular after all
Date: Mon, 05 Feb 2007 18:54:58 +0100
Newsgroups: comp.lang.ada

It seems we're many to work hard to promote Ada in some way or another, so in the long run there's hope ☺

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 05 Feb 2007 21:23:32 +0100
Subject: Re: Ada is popular after all
Newsgroups: comp.lang.ada

> [Package 'gnat'] is an unidentified version, presumably the default. There's also
> gnat-4.1 97
> gnat-3.3 5
> gnat-3.4 2
> gnat-3.2 1
> gnat-4.0 1
> How do they fit into this?

Indeed, gnat is the default, per Debian Policy for Ada. In Sarge, that was GNAT 3.15p but now in Etch, it is almost empty and provides just one symbolic link: /usr/bin/gnatgcc -> gcc-4.1. Most importantly it depends on the actual compiler package, gnat-4.1.

The other versions (3.3 .. 4.0) are older and no longer provided in Debian. They were never supported anyway.

Ada and Microsoft

Ada in Windows Vista

From: Wiljan Derks <Wiljan.Derks@zonnet.nl>
Subject: Re: GNAT and Vista?
Date: Mon, 18 Dec 2006 20:26:37 +0100
Newsgroups: comp.lang.ada

> Has anyone already experience with GNAT on Windows Vista?
> Does it work? And even such tools like GPS, GVD, AdaGIDE?

I did a test driver with Vista RC2 with GNAT and our own software. Looks like Vista is highly compatible with XP. I was able to build all our software using GNAT on Vista. I also was able to use our device drivers build for XP on Vista without trouble.

There was some trouble however:

- GCC does not seem to be able to find the compiler gnat1.exe. I fixed this by adding the containing directory to the path.

- GPS did not start properly but was still usable.

- My own software had some problems with the service interface on Vista.

For the rest all my software seems to be running fine. That includes quite some GUI code based on GWindows, some web servers based on AWS.

My conclusion is that it should be easy to switch to Vista.

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: GNAT and Vista?
Date: Mon, 18 Dec 2006 16:22:52 +0100
Newsgroups: comp.lang.ada

> Some friends told me that Win32-API based applications are no longer supported on Vista (only dot Net).

The Win16-API is not. Win32-API should be OK.

References to Publications

Aonix AONews

[Extracts from the table of contents. See elsewhere in this news section for selected items. —su]

Welcome to the first Aonix Newsletter of 2007! We have some exciting news and interesting articles in this issue that we hope you will find useful and enjoyable.

Hot Topics in this issue: […]

- New product releases, featuring PERC Ultra 5, the premiere virtual machine for embedded Java™ developers, ObjectAda v8.2 with support for the PikeOS RTOS, and AonixADT v3.2.1, the Eclipse plug-in technology for Ada development.

- Aonix Customers and how they're using Aonix development and execution technologies including Lockheed Martin Aegis program and FKI Logistex material handling control system.

- Partner highlights including Concurrent, ProSyst and Wind River Systems.

…and much more

Embedded Technology Journal

Subject: Remote programming in the Embedded Technology Journal
Date: Tuesday December 19, 2006
RSS: www.adacore.com/category/press-center/feed/

The latest edition of the Embedded Technology Journal includes an interesting article on IDE considerations for remote programming. You can find more information on the use of this technology with GNAT and GPS by clicking here, or by contacting sales@adacore.com.

Military Embedded Systems Magazine

SPARK in Military Embedded Systems Magazine

Military Embedded Systems magazine features SPARK in "Building secure software: Your language matters!" co-authored by SPARK team's Roderick Chapman and AdaCore's Robert Dewar. (A recent version of Adode PDF Reader is required to open the PDF of this article).

EE Times Online

Subject: Ada enhances embedded-systems development
Date: Tuesday December 19, 2006
RSS: www.adacore.com/category/press-center/feed/

Check EE Times Online, at

http://www.industrialcontroldesignline.com/showArticle.jhtml?articleID=196800890

AdaCore — The Military Technologies Conference 2007

http://www.adacore.com/2007/01/05/the-military-technologies-conference-2007/

Friday January 5, 2007

The Military Technologies Conference 2007

AdaCore CEO, Robert Dewar will give a talk entitled "Why High Integrity Software Requires Open Source Tools."

AdaCore presentations in Ada Europe 2007

From: AdaCore Press Center
Date: Tuesday February 13, 2007
Subject: Ada Europe 2007, 12th International Conference on Reliable Software Technologies
RSS: www.adacore.com/category/press-center/feed/

AdaCore co-authored papers:

- "Precise Garbage Collection"

Francisco García, Javier Miranda and José Fortes Gálvez

- "Implementation of new Ada 2005 real-time services in MaRTE OS and GNAT"

Mario Aldea-Rivas and Jose F. Ruiz

Industrial session:

- "Towards Certification of Object-Oriented Code with the GNAT Compiler"

Javier Miranda

Tutorials:

- "Building interoperate distributed applications with PolyORB"

Thomas Quinot

AdaCore will also be exhibiting at this event.

AdaCore — Software Technology Conference (SSTC 2007)

Subject: Software Technology Conference (SSTC 2007)
Date: Tuesday February 13, 2007
RSS: www.adacore.com/category/press-center/feed/

18–21 June 2007, Tampa Bay, Florida, USA

Ben Brosgol will be giving a talk entitled "Designing High-Security Systems: A Comparison of Programming Languages".

AdaCore will also be exhibiting at this event.

AdaCore — DASIA 2007

Subject: DASIA 2007
Date: Friday March 9, 2007
RSS: www.adacore.com/category/press-center/feed/

29th May – 1st June 2007, Naples, Italy.

Jose Ruiz will present a talk on "Preventing Stack Overflow using Static Analysis" (paper written by Jose F. Ruiz, Eric Botcazou, Olivier Hainque, and Cyrille Comar).

Ada Inside

Tomahawk Cruise Missile Mission Planning
*http://www.aonixnews.com/jan07/inthefield.htm#Tomahawk_*

Boeing Selects Aonix ObjectAda for Tomahawk Cruise Missile Mission Planning Software

As a historical leader in mission critical Ada technologies, Aonix is pleased to announce the selection of ObjectAda by Boeing for the Tomahawk Cruise Missile program. Boeing plans to use Aonix's ObjectAda for Windows for ongoing software development and for migration tasks on the Tomahawk Mission Planning (TMP) Software Platform. Boeing's interest in Aonix's ObjectAda for Windows hinges on several technical factors, including its full compatibility with Microsoft's .NET platform.

Facing legacy obsolescence and diminishing support for their existing Ada development environment, Boeing's TMP group initiated a full-scale evaluation of available Ada compiler and tool solutions. Their challenge was to find an Ada vendor with compiler technology able to support a very large Ada source code base, meet stringent performance and functionality requirements, and efficiently support a large software development team. In order to port a large code base without requiring a large investment of new engineering resources, Boeing's TMP group needed a multilanguage development environment to accommodate existing C, Fortran, and .NET software assets.

"Aonix rose to all the challenges we laid out," noted Dan Turpin, TMP Systems Engineer. "They accommodated specific requirements critical to our success, such as performing specific debugger and compiler performance improvements that we needed."

Similarly, Ben Ralston, TMP's compiler technical evaluator, stated, "As an engineer, I realize it was no small feat to accomplish technical changes of this magnitude to their compiler, especially in such a short time frame. Aonix met all of our objectives."

In integrating current Windows improvements with the Aonix Ada 95 compiler, Aonix has delivered enhancements to the object code and symbolic debugging information generation and provided full compatibility with the Microsoft Visual Studio .NET 2003 development tools. Recognizing the growing number of large-scale Ada projects, ObjectAda for Windows offers dramatic performance improvements for developers linking executable files or initiating debugging sessions for large programs. As part of the ObjectAda family, ObjectAda for Windows allows developers to choose between the traditional Aonix IDE for development and the new AonixADT™ Eclipse plug-in. AonixADT incorporates Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, and a complete Ada debugger interface, enabling Ada developers to enjoy state-of-the-art interface capabilities geared to maximize developer ease and efficiency.

C-130 AMP aircraft

Tuesday December 12, 2006
AdaCore Celebrates C-130 AMP's Maiden Flight

New York and Paris, December 12, 2006 — AdaCore joins The Boeing Company, Smiths Aerospace and Wind River Systems in celebrating the successful first flight of the C-130 Avionics Modernization Program (AMP) aircraft, which took place on September 19. AdaCore serves as a key member of Smiths Aerospace's development team for the C-130 AMP's Mission Processor (MP). The MP provides primary computing capability for the cockpit display generation, and extensive video processing, which supports the manipulation and distribution of new and legacy video sources to all aircraft displays. The U.S. Air Force initiated the C-130 AMP to standardize configurations, lower the cost of ownership, and increase survivability of its aging C-130 aircraft. It is the most comprehensive C-130 avionics modification ever conducted.

The MP's critical infrastructure software is a combination of Wind River's Platform for Safety Critical ARINC-653 real-time commercial operating system and development tools, AdaCore's GNAT Pro Ada 95 compiler and development environment for PSC ARINC-653 (including GPS) and Smiths Aerospace's infrastructure software. The Software Common Operating Environment (SCOE) delivery provides the C-130 AMP team with an ARINC-653 software partitioned operating system, as well as a full set of "partitioning aware" tools to support software development and debug for the PowerPC.

As part of the Smiths Aerospace contract, AdaCore ported the compiler, tools and run-time libraries to work with PSC ARINC-653. AdaCore also provided an Ada binding to the ARINC-653 APEX facilities for partitioned operating systems, as provided in PSC ARINC-653. New debugging modes were supported as well. The company developed an Ada run-time library certifiable to avionics safety standard DO-178B Level A, and worked with Verocel to develop certification evidence for it.

"AdaCore was specifically selected by Smiths' mission processor development team for its superior technical expertise with both Ada compilation systems and with avionics application development environments," said Dudrey Smith, chief software technologist at Smiths Aerospace. "With fast compilation speed, quality code generation, and an extensive set of switches and pragmas, AdaCore's GNAT Pro specifically addressed the mission processor's need for mission-critical robustness and flexibility."

"The MP project is a perfect illustration of how mission-critical aerospace systems should be architected," said Robert Dewar, CEO of AdaCore. "Turnkey avionics configurations, like the SCOE, facilitate the integration of future upgrades and minimize the impact of obsolescence."

SAAB Signs Corporate Wide Software License

Subject: March 7, 2007
Date: SAAB Signs Corporate Wide Software License with AdaCore
RSS: www.adacore.com/category/press-center/feed/

AMSTERDAM, Netherlands, March 7, 2007 — Avionics Exhibition and Convention — Agreement delivers volume subscription to leading Ada development environment across Saab Group

Saab, one of the world's leading high-technology companies, today signed a corporate wide licensing agreement with AdaCore. Saab will adopt AdaCore's GNAT Pro development environment for projects across the organisation and will standardise on GNAT Pro.

The €340,000 annual agreement provides Saab with a volume subscription for all its developers, rather than access to AdaCore tools and support on a project by project basis. GNAT Pro is available to Saab on a number of architectures and has been ported to more platforms, both native and embedded, than any other Ada technology.

The Ada programming language is designed specifically for large, long-lived applications where reliability, efficiency and safety are critical. AdaCore has been closely involved with the Ada language since its inception and its GNAT Pro development environment combines market-leading technology with an expert support system to provide a natural solution where efficient and reliable code is critical.

The agreement, signed by AdaCore's Swedish distributor Asplund Data AB, demonstrates the importance of the Ada programming language to Saab. By using Ada Saab benefits from a high-integrity and high-quality programming language that enables it to develop safety-critical systems for projects across the whole group

Saab has benefited from AdaCore's combination of an advanced software development environment with expert support for over 7 years. It is currently being used by around 250 developers on

projects that range from the Electronic Warfare System (EWCS) for the JAS 39 Gripen fighter to the GFORCE ship control system and Taurus, MPS and METEOR missiles. Divisions such as SaabTech, Saab Bofors Dynamics, Saab Microwave Systems, Saab Avitronics, and Saab Systems Pty Ltd are currently using Ada and GNAT Pro. By now standardising on AdaCore Saab will cost-effectively extend these benefits across the entire group.

"This agreement demonstrates the increasing importance of Ada to Saab's development of world-leading defence and avionics systems," commented Franco Gasperoni, Managing Director, AdaCore. "This is part of a growing trend of companies adopting AdaCore as a corporate standard. By moving to an organisation-wide licensing model, they are now benefiting from significant efficiencies and cost-savings as well as access to our advanced tools and support."

About Saab

Saab is one of the world's leading high-technology companies, with its main operations focusing on defense, aviation and space. The Group covers a broad spectrum of competence and capability in systems integration.

International Space Station (ISS)

From: R. B. Love <rblove@airmail.net>
Date: Sat, 17 Feb 2007 17:41:34 −0600
Subject: Ada downsizing in space
Newsgroups: comp.lang.ada

[..] I have to believe that the International Space Station (ISS) was one of the biggest Ada projects in the world, employing people in several nations writing Ada. NASA has decreed that there must be a 15% reduction in spending on ISS and Boeing responded Friday with layoff notices going to between 140 and 180 people. A good many of them are Ada programmers.

All the work I see being done for CEV is C or C++. LockMart, the same people who spiked Ada for with the Secretary of the Air Force on SBIRS, seems determined to make everything C++.

Now some of us will be employed for years maintaining existing ISS code. The transition from development to maintenance had to come someday.

It would be very interesting to hear about new, large Ada projects anywhere. Does someone still maintain a list? […]

From: Jeffrey Creem <jeff@thecreems.com>
Date: Sun, 18 Feb 2007 09:58:33 −0500
Subject: Re: Ada downsizing in space
Newsgroups: comp.lang.ada

[…] NASA appears to have abandoned Ada around 10 years ago (That is not to say that nothing was being done in Ada — just most high visibility things that you'd hear about were not done in Ada). I think it was part of their Better, Faster, Cheaper (Choose any 0 of them) plan. […]

Boeing appears to at least maintain some interest in Ada as the C-130 and 7E7 announcements indicate.

Since Ada is no longer "buzzword compliant" I don't think (most) people using it are really into press release engineering anymore.

From: R.B. Love <rblove@airmail.net>
Date: Sun, 18 Feb 2007 11:15:46 −0600
Subject: Re: Ada downsizing in space
Newsgroups: comp.lang.ada

> Do you have first hand knowledge that ISS has a lot of Ada?

[…] There are between 45–50 flight computers on board the ISS on various US components. They are all programmed in Ada. I believe the Russian flight computers use C. The onboard, hand held PCs are mostly C/Linux. The large trainers for ISS use Ada almost exclusively. That was another 1–2 dozen programmers. Some of the foreign trainers use Ada.

Indirect Information on Ada Usage

[Extracts from and translations of job-ads and other postings illustrating Ada usage around the world. —su]

Job Description:

Looking for a software project manager for a large airspace management defense system. Will be responsible for project planning, statusing, reporting and staffing. Will have a lot of interfacing with the program office, customers, engineering managers and software developers. Requires knowledge of earned value management process (stated on resume) and detailed planning with PERT charting tools. Must have software development experience, as there will be involvement in project technical decision making. Needs excellent presentation skills. May require occasional travel internationally.

Required Skills:

Must have experience in software development and management of large software-intensive programs. Excellent communication skills, both verbally and in writing.

Desired Skills:

Hands on experience with real-time software and military programs is a plus. Experience with Ada programming language.

Required Education(including Major):

BS Computer Science or Engineering.

Job Description:

Software Engineer needed for software development and integration position on the Airspace Command and Control (AC2) Product Line (APL). Engineer will participate in all aspects of the APL software development process for the surveillance components including Track Management, Identification, Tracking, and Airspace Management.

Initial responsibilities will be to learn the application's surveillance domains and the APL software development environment. After initial training, opportunities in the following will be available for multiple projects:

- Update existing surveillance components
- Design new capabilities and enhancements
- Implement and Integrate software upgrades and changes
- Support software deployment throughout the world

Candidate should have some familiarity with ground based radar systems and a desire to learn how AC2 systems interface with these radars and other data sources to produce an integrated air picture. Candidate will be developing software in the Ada programming language. No relocation costs.

SECRET security clearance preferred, Eligibility required.

Required Skills:

2–3 years experience developing real time software. Experience with Object Oriented Design and Ada programming languages. Experience with Unix and Linux operating systems. Sound reasoning, keen attention to detail and the ability to deliver reliable software products. Willingness to travel under business trip status for site integration tasks. Number of trips per year will vary but the trips are generally two to four weeks in duration.

Desired Skills:

Ability to apply Object Oriented Design techniques to the large scale software-based product line environment. Prior experience with large software applications. Prior experience with surveillance systems. Applied mathematics experience. Experience with Rational APEX environment.

Required Education (including Major):

BS Computer Science.

Job Description:

Looking for a software engineer who is interested in developing HMI (Human Machine Interface) code for multiple air defence programs. Candidate will develop code, test, and integrate. Candidate will

be developing code using Ada, C, C++, and Java. Must be able to travel internationally. Business trips can be expected to last 1–3 weeks and can occur several times a year. Candidate must be able to obtain a secret clearance (no dual citizens). Preference will be given to candidates with an existing secret clearance.

Required Skills:

Must have 2 years (post graduate) experience working on large software programs. Must have at least 1 year Ada programming experience and 1 year Java programming experience. Must also have C/C++ and UNIX experience. Must have good communication skills, both verbally and in writing. Willingness to travel under business trip status for site integration tasks.

Desired Skills:

GUI experience. Knowledge of X Windows/Motif. Experience with Rational APEX environment.

Required Education (including Major):

BS Computer Science.

Job Description:

[…] Looking for Senior System / Software Engineers responsible for design and development of data link systems for military applications. Responsibilities include developing concepts of operation, interoperability and technical execution, judgment of a variety of technical inputs from subject-matter experts, customer interface and technical presentations. Will participate in Ada software design, development and maintenance for the Links Software component of APL. Links consists of capabilities which interface with adjacent air defence systems, radar sites and supporting aircraft while maintaining system-wide integrity. Participate in full life cycle software development — design, code, unit test, integration and maintenance. Perform software problem investigation and resolution. Produce documentation of software artefacts. Desired for candidate to be a self-starter who routinely acts independently to uncover and resolve issues on programs and is able to communicate well with software and system engineering disciplines in evaluating potential software issues. Excellent problem-solving ability. Will need to multitask in order to support multiple programs and priorities.

Candidate should have hands on experience or knowledge of integrating hardware equipment in the lab. Where necessary will work with the customer. Understanding of military tactical data links such as Link 11A, 11B, 16 required and military certification processes such as AFSIT, OT&E, DT&E desirable. The candidate must be able to work well under pressure and work long hours occasionally to meet schedules. Relocation not included.

Required Skills:

The candidate must have at least 7–15 years of experience on large software intensive systems. Experienced with UNIX or LINUX operating systems. Experienced with external Interfaces in the Links area. Must have domain knowledge of Links applications in Air Defence systems. Strong appreciation for, or experience in, software engineering and configuration management practices and procedures. Good communication skills and ability to work well in a team environment. L11/11B and/or L16 experience required either in SW development or integration. Must have an existing secret clearance.

Desired Skills:

Candidate should be familiar with the SW development lifecycle, Rational APEX. Proficiency in Ada a plus; C++, Java helpful. Candidate to be familiar with requirements flowdown and testing in a lab environment. Proven ability to work within a large and diverse organization. Link Simulator experience highly desired. Understanding of military certification processes such as AFSIT, OT&E, DT&E desirable. Ability to travel to support the on-site testing a plus.

Required Education (including Major):

BS Computer Science.

Job Description:

Looking for a Software Requirements Specification (SRS)/Interface Requirements Specification (IRS) Engineer for the Software Requirements team in the APL (Air Defense Systems Product Line) department. Candidate must have experience in generating either SRS and IRS specifications for large software intensive systems. […]

Requirements team supports the software generation. At the beginning of a project, the System Specifications are analyzed and then broken down into lower level specifications that the software engineers can code to. We are looking for engineers for software requirements team. This is not a software development position or a hardware position. No relocation costs.

Required Skills:

Must have 5–7 years of professional experience in an aerospace/defense oriented engineering environments in any of the following disciplines: software requirements development, SRS/IRS writing, software requirements analysis, software development, systems engineering, and tactical data links/communications. Candidate must have experience in generating either SRS and IRS specifications for large software intensive systems. Eligibility for a US SECRET security clearance required (NO dual citizens) must be able to work on multiple projects concurrently. Excellent written and verbal communication is required.

Desired Skills:

Hands on experience with real-time software and military programs is a plus. Proficiency with the DOORs requirements management tool is desired. Familiarity with military command and control systems (C2), tracking software and algorithms, and surveillance and identification (ID) functionality in C2 systems is also a plus. Existing US SECRET clearance preferred. Experience with UML, C, C++, or Ada languages, Linux and Unix operating system is desired.

Required Education (including Major):

BS/MS/MA in Electrical Engineering, Mathematics, Physics, and/or Computer Engineering.

Job Description:

This position will focus on a broad spectrum of Systems Engineering tasks on a large, air defense command and control system software development, integration, and evaluation project. Working directly with principal systems engineers, this individual will support and perform tasks to include requirements analysis and maintenance, system design, software system integration, and system test and evaluation. This position has no hardware content or Automated Test Equipment (ATE) involvement. The ideal candidate is a Systems Engineer who has experience and familiarity with software development, code, and test. The software system includes code written primarily in C/C++ and ADA, as well as Government-furnished code in UML, operating in Linux and Unix environments.

For system requirements analysis and maintenance, this individual will translate

customer needs into well-written requirements from which systems and subsystems can be architected and designed. This task also involves maintaining system-level requirements with the DOORs tool throughout the product life-cycle and assessing the system impact as requirements change.

In support of system integration and test/evaluation, the individual will assist principal systems engineers with development and implementation of software system integration plans and the allocation of system level requirements to tests, developing test plans and procedures, conducting tests, and documenting the results in test reports and presentations. This individual will be tasked with developing and executing software system performance evaluation plans and procedures and collecting/analyzing software technical performance metrics. In the performance

of these duties, the individual will be expected to become a subject matter expert in one or more of the functional areas comprising command and control software systems, including tracking software and algorithms, surveillance and identification (ID), and/or system control.

This task will involve troubleshooting challenging problems of broad technical scope which arise as a result of the system requirement analysis, integration, test and performance evaluation. Accordingly, this individual may be required to seek and synthesize expert advice from other TRS engineers or suppliers/vendors in order to recommend corrective actions.

This position may require occasional domestic travel of short duration (3-4 days) to customer facilities to support technical interchange meetings. Occasional long hours (> 40 per week) may be required to meet program needs and schedules.

Required Skills:

Six years experience in an aerospace/defence-oriented, systems engineering environment (requirements analysis and maintenance, system design, software system integration, system test and evaluation, or system performance measurement and analysis). Eligibility for a US SECRET security clearance required (no dual citizens). Although this is a Systems Engineering position, experience with C or C++ programming languages, Linux and Unix operating systems is required. Experience with coding, compiling, and testing software is required. Proficiency with the DOORs requirements management tool is required.

Desired Skills:

Familiarity with large-scale software systems, UML and/or Ada code, and software integration and development processes (such as the Spiral development process) is a plus. Familiarity with military command and control systems (C2), tracking software and algorithms, and surveillance and identification (ID) functionality in C2 systems is also a plus. Existing US SECRET clearance preferred. Proficiency with the MS Office Tool Suite (Excel, Powerpoint, and Word) is desired.

Required Education (including Major):

BS Electrical Engineering, Computer Science/Engineering, Systems Engineering, Physics, or Mathematics.

Job Description:

[…] One of our most aggressive projects is producing a web-based, multi-media contact center solution written in Ada. To help us accomplish this goal, we are hiring additional Ada developers to add to our existing staff spread out over 3 countries.

Candidate must be self-motivated, able to take an abstract idea or process, and both design and implement an accurate corresponding solution. Must be able to work both individually as well as part of a team.

Must be able to program for UNIX or Linux operating systems. Linux preferred. Willing to consider this as a full-time remote/telecommuting permanent position. Most of our development staff are remote workers, however, our preference is for someone to eventually relocate to the Phoenix Area.

Some call center knowledge, especially in predictive dialers, would be a definite plus, but not required. So if you are a self-motivated developer, we look forward to you joining our energetic team.

Ada in Context

Physical comparison of Ada 95 and Ada 2005

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sun, 21 Jan 2007 15:29:15 +0200
Subject: A physical comparison of Ada 95 and Ada 2005
Newsgroups: comp.lang.ada

A few days ago I received an eagerly awaited Springer hard-copy of the Ada 2005 LRM (courtesy of Ada Europe, to whom many thanks). Being less busy than usual I have had some time to study this work and compare it to the earlier Springer LRM for Ada 95, with the following results:

	Ada 95	Ada 2005
Weight (g) :	985	1205
Thickness (mm) :	32	27

The height and width are the same, so the overhead in terms of desk-top area is unchanged, which is good. The result of this comparison is thus that Ada 2005 is 44.9% denser than Ada 95, as represented by their respective LRMs. The analogous comparison based on the text of the LRMs was not in scope for this study ☺

Official name of Ada

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Date: Fri, 23 Feb 2007 14:04:02
Subject: The Ada 2005 name (was: Re: Bug in GNAT GPL 2006?)
Newsgroups: comp.lang.ada
Summary: Recommended informal name for latest standard is Ada 2005

I notice that once in a while there appears to be some confusion about how to refer to the latest Ada language definition.

FYI, the internationally accepted recommendation is to use the name "Ada 2005".

Obviously the official name of the language is Ada, but when referring to previous "instances" of the standard, the Ada community has been using Ada 83 and Ada 95 as informal or "vernacular" names.

In an attempt to avoid possible confusion about how to refer to the amended Ada language definition, the ISO working group on Ada (WG9) discussed this issue during its June 2005 meeting in York. Various proposals were made and many arguments were presented. For those of you who are interested: a summary of the discussion is available in the minutes of that meeting [1].

Finally, the following recommendation was accepted *unanimously*:

"Recognizing that ISO's publication date will differ from the date of technical completion in 2005, and recognizing that the term "Ada 2005" is widely used in the community, WG9 recommends that an appropriate vernacular designation for the amended language should be "Ada 2005"." [2]

That's why in most literature, marketing material, communications, etc, we refer to the amended language as "Ada 2005".

This includes e.g. the ARA announcement "Ada 2005 on Track for Formal ISO Approval" [3], Ada-Europe's press release "Technical Work on Ada 2005 Standard Completed" [4], and the title "Ada 2005 Reference Manual. Language and Standard Libraries" of the book recently published by Springer in its LNCS (Lecture Notes in Computer Science) series [5].

[1] <http://www.open-std.org/jtc1/sc22/wg9/n451.htm#VernName>

[2] <http://www.open-std.org/jtc1/sc22/wg9/n451.htm#r5>

[3] <http://www.adaic.com/news/iso-Ada05.html>

[4] <http://www.ada-europe.org/Ada_2005_Press_Release.pdf>

[5] <http://www.springer.com/home?SGWID=5-102-22-173712407-0>

We may all have our personal preferences about how to name things, but in the interest of global understandability and to avoid creating confusion, may I suggest we all abide to that recommendation of the ISO working group on Ada?

So, when referring to the language in general, use "Ada", and when referring to the recently amended language definition in particular, use "Ada 2005".

Future of Ada-POSIX Binding

From: Stephen Michell <stephen.michell@maurya.on.ca>
Date: Thu, 04 Jan 2007 22:09:20 −0500

Subject: Participation in Ada POSIX Binding Working Group
Newsgroups: comp.lang.ada

Ada-POSIX Binding Rapporteur Group Interest

Dear All,

I have been named the Rapporteur of the Ada-POSIX Binding Rapporteur Group by ISO/IEC/JTC1/SC22/WG9. This Rapporteur Group (hereafter called the PRG) is charged with assisting the editor of IS14519 Ada Binding to POSIX in the maintenance of this document.

IS14519 was completed (revised from an earlier version to accommodate Ada 95) and standardized in 1998 (2001 for ISO) reflecting the POSIX version of the time. Since that time there have been 2 revisions of Ada and 2 of POSIX.

The changes to Ada and to POSIX may have significant impact on the binding.

- POSIX has had major revisions, with most of the documents (except the Ada binding and the real time extensions) being folded into a single 4-part document.

- Ada has added new capabilities in areas that directly impact the POSIX binding, adding

- Directories,

- Synchronized, protected and task interfaces,

- Events, Timers, and new task scheduling paradigms

A study that the Canadian delegation did for WG9 (WG9 document N477r) indicates that the existing POSIX C-language interfaces that IS14519 relies upon has not changed in many significant ways. This may mean there is little to be done to the interface.

An alternative position could be that we have an opportunity to bring the Ada-POSIX binding into IS8652. It may be possible to modify the binding so that they are all children of Ada.Interfaces and it may be possible to deprecate interfaces where functionality is now provided by the language itself.

As I currently see the situation, we could do one of the following:

1. Leave it alone

2. Perform a minimal amendment to include new exceptions, possibly changed behaviours, and possibly a few new capabilities

3. Perform an amendment and move all capabilities into Ada.Interfaces but largely leave them alone except to add material from 2 above.

4. Perform a larger amendment or a revision to update Ada's interface to POSIX needs for Ada 2005 to POSIX 2008, possibly deprecating capabilities in favour of Ada's capabilities and possibly adding capabilities if needed.

5. Do an OS interface that replaced POSIX with one that was Windows and POSIX compatible.

What we do will depend upon the level of interest that we have and the perceived need. Options 4 and 5 are significantly larger activities, would need considerably more resources but could have a larger payoff.

To that end, we are having the first meeting of the PRG in Tallahassee, Florida February 19–21. One of the major discussions will be the level of effort and the kind of amendment/revision that we will propose.

If you are interested in participating in the PRG, please contact me directly at stephen.michell@maurya.on.ca

Thanks you

Stephen Michell, Rapporteur,
ISO/IEC/JTC1/SC22/WG9/PRG

Ada.Command_Line and wildcards

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Wed, 21 Feb 2007 21:43:41 +0100
Subject: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

I was surprised to see Ada.Command_Line in GNAT (3.15p and 2006 GPL) serving the command-line argument "*.adb" indeed as N arguments, the list of files with an "adb" extension!

ObjectAda gives the argument as-is. Who is right then ?

It's annoying if the wildcard expansion is done a priori, because I would like to have it done by Ada.Directories, for a generic command-line tool…

From: Gautier de Montmollin <gdemont@hotmail.com>
Subject: Re: Ada.Command_Line and wildcards
Date: 22 Feb 2007 00:16:09 −0800
Newsgroups: comp.lang.ada

[…] I tested both programs on both Windows 98 and XP. In both systems, the OA-compiled program gives "*.adb" and the GNAT- compiled the list of files with ".adb" extension. To Adrian, enclosing with "" works, thanks, but then the syntax differs from the usual one for a command-line tool…

I am just surprised by the GNAT behaviour, for two reasons:

- I did not find (or missed) something about it in the RM (95)

- there is also a GNAT.Command_Line that explicitly intends to do wildcard expansions; so why also GNAT's Ada.Command_Line should do it silently?

From: "Marc A. Criley" <mc@mckae.com>
Organization: McKae Technologies
Date: Wed, 21 Feb 2007 19:13:54 −0600
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

> This is an OS (shell) issue, surely?

I was surprised as well, and I took it to be a shell issue as well.

I found that just wrapping the wildcard in quotes as Adrian suggested passes it in as typed (with tcsh on Linux, anyway). I've got Avatox able to accept either the wildcard or the list of files.

From: "Alex R. Mosteo" <devnull@mailinator.com>
Subject: Re: Ada.Command_Line and wildcards
Date: Thu, 22 Feb 2007 12:02:07 +0100
Newsgroups: comp.lang.ada

It would be in Unix. There, wildcard expansion is a shell matter. Windows only half-faked it.

What I mean is that if your code is to be run outside of a windows platform, you'll not see wildcards in your arguments (unless you quote them, as someone else has pointed elsethread).

But if GNAT is deliberately doing expansion to emulate Unix behavior, I guess that should be documented in Windows platforms…

From: Jean-Pierre Rosen <rosen@adalog.fr>
Organization: Adalog
Date: Thu, 22 Feb 2007 12:19:12 +0100
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

I think it is a GCC feature. Because GCC is often used to port Unix applications to Windows, the GCC library emulates Unix behaviour on Windows.

Too bad that Unix behaviour was wrong in the first place…

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: Ada.Command_Line and wildcards
Date: Sat, 24 Feb 2007 07:34:35 +0100
Newsgroups: comp.lang.ada

[…] The VMS also won't expand wildcard. In fact Unix shells are pretty alone here. And they got it wrong. Bot DOS and VMS will warn you on:

DEL *.*

but not on

DEL Some_File.Txt

From: Maciej Sobczak <maciej@msobczak.com>
Date: Thu, 22 Feb 2007 14:49:37 +0100
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

I don't understand. Shell uses some special characters to make it easier for the user to type commands. (Shells can compete on how well they do this job.) Wildcards are just an example. Consider this:

$ cat *.ads *.adb | wc -l > loc.txt

If you claim that * above should be passed "as is" to the program (cat), so that the program can figure out on itself what to do with it, then you might as well argue that the program should figure out *everything* above. Obviously, that wouldn't be funny.

From: Larry Kilgallen <Kilgallen@SpamCop.net>
Subject: Re: Ada.Command_Line and wildcards
Date: 22 Feb 2007 09:12:05 −0600
Newsgroups: comp.lang.ada

> I claim it would be easier to provide a function that expands parameters, than to force expansion. Or maybe just provide another function that provides the raw parameters.

That is what VMS provides, and what one calls from DEC Ada on VMS. Using the OS-provided code is essential, as interpreting rooted directory multivalued logical names is just too complex for private implementations.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: Ada.Command_Line and wildcards
Date: Thu, 22 Feb 2007 17:20:15 +0100
Organization: Adalog
Newsgroups: comp.lang.ada

> There are three such "other functions":
> $ cat '*.ads *.adb' | wc -l > loc.txt
> $ cat "*.ads *.adb" | wc -l > loc.txt
> $ cat *.ads *.adb | wc -l > loc.txt
>
> So could you please explain why you think the Unix behaviour is "wrong"?

No, that's from the user's side, not from the program's side. It should be up to the program to decide whether "*" is to be interpreted as a wild-card or not.

For example, I have a utility where I pass Ada unit names (not file names), but wildcarding is allowed for the unit names (handled by the program). If by chance I have a file that matches the wildcard in the current directory, I get an absolutely useless parameter. And I hate having to tell the user (even if I am the only known user of the program ☺ that the parameters must be quoted.

From: "Randy Brukardt" <randy@rrsoftware.com>
Subject: Re: Ada.Command_Line and wildcards
Date: Thu, 22 Feb 2007 19:01:22 −0600
Newsgroups: comp.lang.ada

> Actually it is. In
> find . -name '*.txt'
> the program 'find' decides to do the wildcard expansion on '*'. The wildcard expansion the shell does, is just an additional service. You can just not use it, by enclosing every word on the command line in """

You're missing the point. The "Find" program can't decide anything; it has to require the user to quote everything. If the user doesn't quote it, they'll get garbage (precisely why I could never remember how find was supposed to work on Unix — I ended up writing shell scripts to cover up this obnoxious behaviour.

It is never sensible to force clients to be aware of things that they should not logically have to care about. Whether the shell or the program wants to expand file names is completely irrelevant to the user — it simply should not matter to the use of a program.

This is exactly the same reason that anonymous access types are not a usable replacement 'in out' parameters in functions. Using them requires the client to do various handstands (use 'Access, declare the object as aliased) for no benefit at all to the client. That is silly; it breaks encapsulation.

Thus I conclude that the Unix shell behaviour (especially as it is not consistent, in that an exec'd program doesn't get the benefit, so the application has to be prepared to handle wildcards anyway — or be stupidly fragile) is harmful, as it makes the user care about irrelevant implementation details.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: Ada.Command_Line and wildcards
Date: Fri, 23 Feb 2007 12:34:38 +0100
Organization: Adalog
Newsgroups: comp.lang.ada

> But if the programs do the expansion, you can be certain that the expansion will differ from program to program.

Why? Assume that your OS provides a function for doing the expansion.

From: Jacob Sparre Andersen <sparre@nbi.dk>
Subject: Re: Ada.Command_Line and wildcards
Date: Sat, 24 Feb 2007 14:40:18 +0100
Newsgroups: comp.lang.ada

Because some programmers will prefer one of the available functions for doing the expansion, and other programmers will prefer some of the other available functions for doing the expansion.

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: Ada.Command_Line and wildcards
Date: Sun, 25 Feb 2007 17:57:41 +0100
Newsgroups: comp.lang.ada

Well, all VMS programs I know of — apart from GNV [1] applications — expand the same way including […]*.TXT expanding recursively. Something you won't get in Unix where you need to guess which -r, -R, --recursive option gets you recursive behaviour. Actually: none — you will need:

find . -iname "*.TXT" -print0 | xargs --null --no-run-if-empty my_command

(did you always remember to use: "--no-run-if-empty").

Again: the Unix way is convenient at start but does not scale all that well for more complex problems.

[1] Note: GNAT for VMS is GNV application :-/

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: Ada.Command_Line and wildcards
Date: Sat, 24 Feb 2007 14:24:40 −0500
Newsgroups: comp.lang.ada

There's lots of stuff I don't like about VMS, but it does handle wildcards without overflowing some buffer than can never be the right size. And it doesn't suffer from the windows problem of every program having it's own notion of wildcards.

From: "Adam Beneschan" <adam@irvine.com>
Subject: Re: Ada.Command_Line and wildcards
Date: 22 Feb 2007 09:07:52 −0800
Newsgroups: comp.lang.ada

[…] This has been one of my pet peeves with Unix for a long time. With other operating systems I've worked with, you can enter a command like

rename *.ads *.ada

for instance, to rename a bunch of files. Unix makes this difficult. (Yes, I know how to use "foreach" in csh… but still…)

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Mon, 05 Mar 2007 13:16:22 +0100
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

> Get a real shell !

[…] Many essential shell scripts are built around Unix shell details. E.g. software configuration scripts tend to stubbornly use a mix of bash (sic, not sh, not ksh, not SUN sh), m4, sed, C helpers, etc etc. In particular, they require the Unix process model.

See the current difficulties in translating recent GCC in a MinGW environment.

Suppose you want to port some piece of Unix software to some other system, a text processing tool, say. The C source is in fact perfectly portable ANSI C. But it can be real hard to get the C source

through the configure stage only because configuration _depends_ on original style Unix shells and the Unix process model (e.g. piped processes "within" a backtick (yes, I know $(), not the point), result to be assigned to some variable. The GNU "standard" config.guess is such a thing.)

On occasions like these the choice of a shell matters. It is less relevant what you or I would choose, because we don't have a choice! (Other than occasionally ask the developers to consider the consequences of a larger Unix dependence graph when they claim their program is portable to non-Unix systems.) […]

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 06 Mar 2007 13:56:12 +0100
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

> Most (auto-) configuration concepts are really bad hacks. They are hardly Unix design choices. The wrong place to fix that is to change the shell or the process model.

Right. But don't you think that this kind of use of Unix tools must be investigated such that the Unix design choices are taken into account while doing so? What is the reason that the Unix design choices do not help prevent complex, strongly coupled, highly dependent pieces of shell programming? Will REXX programs look the same?

Unlike Ada, Unix favors the *writer* over the reader. No surprise I think. But damn right?

When it comes to echo *.ads, a decision was necessary as to what should happen when there is no matching file name. The choice was not: reflect this fact and produce the empty result. This might have had other consequences somewhere else. Instead, it was to produce the pattern itself—which is very different from a file name. echo(1) is a highly overloaded function, so to speak, very flexible. So flexible indeed that it takes some time to learn how this one command interacts with all sort of things Unix.

That's unlike what ls produces. But it is consistent with ls and with the other uses of echo by 1..* mental indirections. Now, what might be an alternative design choice?

I wouldn't mind programs that just read their arguments unexpanded and call the OS expansion service, an iteration device for example. And if there is no matching thing, then *every* program consistently produces empty results! One difference, then, is that an ls will consistently produce nothing for no argument. This will require more typing when you want something, for example ls -d "*". I don't think any Unix programmer will see the benefits of more typing. But having gone some way through the Ada experience I think this the right thing.

Or with this change, if calls to the expansion service should not be part of Unix programs, why not have

$ ls $(expand "*.ads")

It isn't more difficult to understand, and it doesn't work with large directories either. It requires that shell commanders say what they want, though. (I'm not a fan of large directories when we do have a hierarchical filing system, but again, who am I to suggest that a program be redesigned to use a hierarchy of directories for file storage.)

There was a time when you knew that logging into a Unix system (HP, BSD, etc.) would give you a very basic setup. But you could use the entire Unix Toolbox, without surprises. Reaping the benefits of some shell programming would give you a productive environment.

Today, the new Unix hackers try to give you a "productive" environment by default. You may have to expect AI-driven completion, and lots of aliases, and not only will learning some Unix shell programming still be profitable, but in addition you have to learn how to bridle typical default shell setups. Unix used to be configurable (after mastering the design choices). Now it is pre-configured… I guess the consequence is that many users think they won't need to learn shell programming.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 9 Mar 2007 20:12:01 −0600
Subject: Re: Ada.Command_Line and wildcards
Newsgroups: comp.lang.ada

> I wouldn't want to use Ada as an interactive command language.

Well, I would. Down with all non-Ada syntaxes!! ;-)

Seriously, we did in fact design an Ada command language for our debugger. It works quite well. (I think others have done similar things.) Of course, it's neither full Ada and it allows leaving out "noise" characters. And I believe we required one statement per line (that allows a lot more abbreviations). You can type:

```
Step_Line (Count => 20);
```

or

```
sl 20
```

(abbreviated command name; unneeded parens and semicolon omitted) Of course, if something is ambiguous, you have to use a longer form.

We originally did this because we wanted the macro language for the debugger to be as close to Ada as possible — that makes the macros readable and maintainable. Usually, you use shorter forms from the command line (typing too much awful).

Text Insertions

From: Pascal Obry <pascal@obry.net>
Date: Wed, 21 Feb 2007 20:07:58 +0100
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

> Currently I'm using Ada.Text_IO which means I have to copy the whole thing into memory, insert the line then over write the file with new contents.
> Ada.Direct_IO is not an option (varying string lengths)
> What alternatives should I consider for making insertions faster? (NB retrieval of a line needs to be fairly quick as well).

Use Text_IO OK, but why copy all in memory? Just write to a temp file, delete the original one and rename the temp file.

If you have a lot of modifications to do on the file, then you probably need to read all the file in memory to have good performances. In this case, read file in blocks using Ada.Stream_IO. Do the changes in memory and write it back in blocks using Ada.Stream_IO.

From: Larry Kilgallen <Kilgallen@SpamCop.net>
Date: 22 Feb 2007 08:02:20 −0600
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

Or use an operating system feature that allows insertions. GNAT Ada 95 on VMS is supposed to emulate DEC Ada features, so that should include the Mixed_Indexed_IO package. There will be more overhead in disk files but for large files it is much better for inserting data in the middle.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 21 Feb 2007 14:16:01 −0600
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

An insertion into a file requires that you're going to have to rewrite everything after the insertion anyway. So there isn't any way to do that cheaply if you have to do it one insertion at a time. Thus my recommendation is to not do it — that is, find a better way to accomplish whatever it is you need to do. For instance, create a file listing the insertions as an adjunct to the original file, and do the merge only on rare occasions. That would allow copying the file only rarely, and allows doing a large number of insertions at once.

If you absolutely have do this as you described, Pascal Obry's suggestions are probably the best. My Trash Finder spam filter has to do this to add header lines to stored messages, and it uses Stream_IO to read and write the file (that can be much cheaper than using Text_IO, because it does not need to look for the ends of lines

once it has determined the insertion point).

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 23 Feb 2007 08:51:23 −0500
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

> Thanks everyone for the input.
> I suspected as much that I would have to do some stream_io.
> Unfortunately I can't do things any other way. The requirement is for a text file :(

Just be cause the file is "text" on the disk, doesn't mean you have to use Ada.Text_IO to read and write it.

Ada.Stream_IO reads and writes "text" files perfectly well.

If you were writing this program in C, you would have no choice other than the C equivalent of Ada.Stream_IO, and no one would claim you were not using "text" files.

And what is a "text" file, precisely?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 22 Feb 2007 23:19:58 −0600
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

> Actually Direct_IO is an option, and probably the fastest way to handle the operation.
> It's kind of messy to deal with the content of a text file in a buffer, and will not work on systems with structured files (like VMS) but will work on most modern systems.

That's how you'd do it in Ada 83, but that's an awful lot of unnecessary complication in Ada 95 (not to mention Ada 2007). Just use Stream_IO for this, and you don't need instances to fill and write your buffer. (And you can easily start in the middle of the file and only read part of it if that works for your application.)

I.e.

> Step 1. Determine the initial file size

Use Stream_IO.Size(File).

> Step 2. Allocate a buffer that is the size of the file plus the size of the string you want to add (including a line terminator)

Buffer : Stream_Element_Array (1 .. Size); -- But you can make it bigger.

> Step 3. Create an instance Direct_IO that is the file size

null;

> Step 5. Read the file into the start of the allocated buffer in one gulp.

Stream_IO.Read (File, Buffer, Last);

> Step 6. Insert your string in the buffer (a little tricky, but doable).

Exercise for the reader. ;-)

> Step 7. Create an instance of Direct_IO that is the size of the buffer with the new string.

null;

> Step 8. Write the buffer to a file as one operation.

Stream_IO.Set_Mode(File, Out_File); -- Or Reset.

Stream_IO.Write(File, Buffer);

From: Jacob Sparre Andersen <sparre@nbi.dk>
Date: Fri, 23 Feb 2007 08:53:19 +0100
Subject: Re: Text Processing in Ada 95
Newsgroups: comp.lang.ada

I would first of all consider using POSIX.Memory_Mapping.Map_Memory to get access to the complete file as an in-memory string. Here is a piece of code I wrote recently for that purpose: [see comp.lang.ada —su]

My reason for suggesting that you map the file into memory is that you can avoid messing with buffers, caching and several copies of the file content.

If you need to make lots of insertions, then I would consider mapping the lines into a insertion-friendly data structure such as a linked list. This data structure should keep track of 'First and 'Last for each line in the file. Inserting new lines would simply be a matter of writing the text of the lines to the end of the "Text" string, and inserting a pointer at the appropriate place in the data structure keeping track of the lines.

The costly part of this method is to write back the lines to the file. Since it will have to be done one line at a time. Depending on the number of insertions needed, it may be cheaper simply to do the insertions with plain string slices on "Text".

End_of_File deprecated

From: Adam Beneschan <adam@irvine.com>
Subject: End_Of_File but not really
Date: 7 Dec 2006 10:00:26 −0800
Newsgroups: comp.lang.ada

This is based on some of the things I've been saying on Maciej's thread about the Get_Line problem. I ran this test using GNAT. Note that my input file is a disk file, to avoid the additional issues that arise with interactive files.

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure IOTest is
   Line : String(1..100);
   Last : Integer;
   F    : File_Type;
   EOF  : Boolean;
begin
   Open(F, In_File, "f1");
   loop
      begin
         EOF := End_Of_File (F);
         Get_Line (F, Line, Last);
         if EOF then
            Put_Line ("End_Of_File
returned TRUE but Get_Line did not
raise an exception");
            exit;
         end if;
      exception
         when End_Error =>
            Put_Line ("End_Error
                  raised");
            exit;
      end;
   end loop;
end IOTest;
```

If I try this on Linux, on a file whose bytes are abc<LF><LF>, then the message "End_Of_File returned TRUE but Get_Line did not raise an exception". This strikes me as bizarre — if End_Of_File returns True, then a subsequent read operation should raise an End_Error, but that isn't what's happening. One of these must be true:

(1) The RM does not allow this behavior, and GNAT is broken.

(2) This behavior is what the RM requires. To me, this means the RM is broken — it just doesn't make sense to me that End_Of_File would return True if there is more information in the file to get with Get_Line (even if the information is "the presence of a blank line"). But perhaps it's my own understanding that is broken; perhaps my understanding of what End_Of_File is supposed to do is wrong, even though I think it's what a reasonable person would expect a function called "End_Of_File" to do.

(3) The behavior is implementation-defined according to the RM (because the nature of terminators is implementation-defined), and it's possible to have an implementation that never displays this message and an implementation that is capable of displaying message both conforming to the RM. In this case, though, I'd say (3a) the RM is a little bit broken, because even though the representation of terminators is implementation-defined, it would seem that the definitions of End_Of_File and End_Error ought to conform to each other, so that the above message could never appear, and (3b) GNAT is broken, because even though the RM allows it to implement Text_IO in a way that causes the above message to appear, it shouldn't because it doesn't make sense.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: End_Of_File but not really
Date: Thu, 7 Dec 2006 18:29:52 −0600

Newsgroups: comp.lang.ada

[Number (2) above] is exactly the state. It's the way these things were defined in Ada 83, and required by the ACATS since the very beginning. I'd be very surprised if you found any Ada compiler that doesn't have this behavior (didn't you try it on your compiler to cross-check)?

As I noted before, there is absolutely no chance that this behavior (or any behavior of Text_IO) would be changed, because there's no possible way for such a change to be compatible. If the program "knows" about the "lost" blank line, it might fail badly if the definition was to be changed. So it's 23 years too late to fix Text_IO.

The moral is simply to never use Text_IO.End_of_File, but rather handle End_Error instead. If you try to use End_of_File, you have the potential of "losing" the last line of the file, you still can have End_Error raised if you call Get or Get_Line when there is no <LF> character at the end of the file, your program will run slower, and your program won't be able to read interactively from the keyboard. It's not worth it, no matter what you think about using exceptions for non-errors.

From: Adam Beneschan <adam@irvine.com>
Subject: Re: End_Of_File but not really
Date: 8 Dec 2006 09:02:53 −0800
Newsgroups: comp.lang.ada

[…] It might have made sense to provide a version of Get_Line with an "End_Of_File" OUT Boolean parameter; this version would behave exactly like the other Get_Line except that it would set this parameter to True instead of raising End_Error. That would have satisfied people who don't like exceptions, while avoiding the hokey semantics of the End_Of_File function (and not requiring Text_IO to do any look ahead). Of course, anyone can easily write their own version of Get_Line that works like that. I agree that, based on the discussions in this thread, the Text_IO.End_Of_File function should just be avoided.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: End_Of_File but not really
Date: Fri, 8 Dec 2006 17:02:17 −0600
Newsgroups: comp.lang.ada

> […] I was under the impression that, since the definitions of the terminators were left up to the implementation, it would be possible for an implementation to define them in a way so that things would work "correctly" (i.e. in a way that would make sense to me). But I guess that's not possible, or in any case implementations aren't expected to do this. Thanks.

No, it's not possible. That's because of the definition of writing files.

Recall that closing a file adds (logical) line and page terminators before the file terminator if they are not present. Let's show these terminators as <EOL><EOP><EOF> (the real representation doesn't matter).

Now consider the program:

```
Create (File1, …);
Close (File1);
```

File1 will contain just <EOL><EOP><EOF> after this program.

Now consider the similar program:

```
Create (File2, …);
New_Line (File2);
Close (File2);
```

File 2 will *also* contain just <EOL><EOP><EOF>.

But clearly, when you re-read these two files, you'd expect different behavior. The first should return End_of_File = True immediately and raise End_Error if you call Get_Line, and the second should return End_of_File = False and allow a single call to Get_Line. But these files have the exact same contents! There is no possible way for them to have different behavior, even though they're clearly different from a user perspective.

It appears that Ada 83 chose End_of_File = True and a single call to Get_Line to work in order that the most important feature of each of these files works as expected. (That is, if the last line of a file that you write is a blank line, you can read a blank line; and that a Get(char) following End_of_File = False will always work.) But the result of that is that neither file will read quite as expected! IMHO, it would have been better to define End_of_File such that a subsequent Get_Line would always raise End_Error (it does mean that for Get(char)). But such hindsight is 20-20.

Since these two files are identical to the language, and certainly the ACATS could check that they're considered identical (it makes similar checks, I don't know if this exact one is made), no amount of implementor tricks can make things work sensibly. In our implementation, the two files would indeed have different contents (the first would be empty and the second would have an explicit <CR><LF> [or just <LF>, depending on the target]}. But that doesn't help, because we have to consider them logically the same, since the language does — indeed, it makes it more complex than simply writing out all of the markers. (Depending on all of the markers to be present would make Text_IO useless on files written by something other than Ada, which would be unusable even if it is technically correct.)

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: End_Of_File but not really
Date: Mon, 11 Dec 2006 16:49:46 −0600
Newsgroups: comp.lang.ada

> Actually… couldn't the "Form" option that is passed to Open include an option that changes the end of file behavior? That wouldn't break old code, but would make new code easier.

I suppose, but it wouldn't help much, because there is no "Form" for Standard_Input (it's already open). So the behavior of that can't be changed — but that's where the worst problem is. So it's not clear that a new "Form" would be very useful (remember that the default behavior would still have to be the "bad" behavior).

It's also clear that the bad behavior is inevitable with the current model of terminators. So we'd need a new underlying model, which sounds like a mess. Besides, getting programmers comfortable with using exceptions is a good thing: I/O can fail in many ways other than reaching the end, it's hardly sensible to write any I/O without some handlers. So I'd be more inclined to make the End_of_File function Obsolescent — it's much like the Constrained attribute and specific Suppress (which are already Obsolescent): it seems like they should work, but they don't.

Net result: it's probably not worth the headache.

Normalize_Scalars vs. Initialize_Scalars

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 07 Feb 2007 14:33:29 +0100
Subject: Normalize_Scalars
Newsgroups: comp.lang.ada

[…]

error: some but not all files compiled with Normalize_Scalars

files compiled with Normalize_Scalars

hello.adb

files compiled without Normalize_Scalars

ada.ads
system.ads
s-stalib.adb
s-memory.adb

[…]

gnatmake: *** bind failed.

Wow.

I guess I have to recompile the world to be able to benefit from Normalize_Scalars. Any fast path for dummies or should I be disappointed? ☺

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: 7 Feb 2007 05:59:13 −0800
Subject: Re: Normalize_Scalars
Newsgroups: comp.lang.ada

> I guess I have to recompile the world to be able to benefit from Normalize_Scalars. Any fast path for dummies or should I be disappointed? ☺

In a configuration pragma file I have:

pragma Initialize_Scalars;

-- pragma Normalize_Scalars; -- For all units!

Probably it is a thing to look at.

From: Gautier de Montmollin <gdemont@hotmail.com>
Subject: Re: Normalize_Scalars
Date: 7 Feb 2007 06:15:44 −0800
Newsgroups: comp.lang.ada

BTW

- it seems that Normalize_Scalars is in Ada 95, Initialize_Scalars is GNAT-only.

- combine with -gnatVa and you'll see bugs falling in clouds…

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: 7 Feb 2007 07:36:04 −0800
Subject: Re: Normalize_Scalars
Newsgroups: comp.lang.ada

> Portable code? ☺
> But to be frank, I'm fine with GNAT-only Initialize_Scalars. Thank you for pointing me in this direction.

Anyway, you are not obliged to pollute your code with testing and/or unportable pragmas. You can put them in a file like "debug.pra" and compile with -gnatecdebug.pra

Advantages:

- all your sources will be compiled with the pragma

- you can keep a debug/test version and also have a fast one without changing source.

- you can use also another Ada compiler for release (or not…).

From: Simon Wright <simon.j.wright@mac.com>
Date: Wed, 07 Feb 2007 22:12:13 +0000
Subject: Re: Normalize_Scalars
Newsgroups: comp.lang.ada

> But bear in mind they do opposite things!

Normalize_Scalars: "This pragma ensures that an otherwise uninitialized scalar object is set to a predictable value, but out of range if possible.' (ARM95)

Intialize_Scalars: "This pragma is similar to Normalize_Scalars conceptually …" (GNATRM)

From: "Alex R. Mosteo" <devnull@mailinator.com>
Newsgroups: comp.lang.ada
Subject: Re: Normalize_Scalars
Date: Thu, 08 Feb 2007 11:43:30 +0100

Mmmm, I had the following idea about them:

- Normalize_Scalars should tries to cause an exception as soon as possible, because it initializes to an /invalid/ value.

- OTOH, Initialize_Scalars will use a /valid/ value for the initialization.

So the former should expose bugs and the latter hide them, for programs that don't do proper initializations.

But I've seen after your warn that Initialize_Scalars is tailored by the user, so my remembrances can come from a particular use of this pragma…

Deallocating list of polymorphic objects

From: Michael Rohan <mrohan@acm.org>
Subject: Deallocating list of polymorphic objects?
Date: 30 Nov 2006 15:40:27 −0800
Newsgroups: comp.lang.ada

I would like to construct a list of polymorphic objects that, as part of the list's finalization, deallocates the objects on the list. Basically, I have a vector of pointers to Object'Class. The objects are added to the list via procedures defined for the list, e.g., append an integer, append a floating point. These append procedures allocate objects derived from the base Object type for the type being appended, e.g., Integer_Object, which is private to the list package.

Since I want the deallocation to be dispatching, it needs to take an access parameter which is then converted to a pointer for the object being deallocated, e.g., an Integer_Pointer, and then passed to an Unchecked_Deallocation procedure. […]

However, I have a feeling there is something "bad" about this type of deallocation, probably related to storage pool but I'm not familiar enough with storage pools to be sure.

Would anyone care to comment on how safe/unsafe this deallocation scheme is?

From: Robert A Duff
Subject: Re: Deallocating list of polymorphic objects?
Date: Thu, 30 Nov 2006 19:05:07 −0500
Newsgroups: comp.lang.ada

You don't need to do all that by hand. It's OK to pass access-to-classwide to Unchecked_Deallocation. It will do the necessary dispatching internally.

But, to be safe, you should ensure that the result type of each "new" is the same as the type passed to Unchecked_Deallocation.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Deallocating list of polymorphic objects?
Date: Thu, 30 Nov 2006 19:24:06 −0600
Newsgroups: comp.lang.ada

To deallocate the elements, then just doing it should work fine:

```
procedure Free is new
Ada.Unchecked_Deallocation
(Object'Class, Object_Pointer);
```

Unchecked_Deallocation of the Objects will call Finalize on them, so that any internal cleanup can be done. (That's presuming that Object is also derived from Controlled, but IMHO that should be true of virtually all complex types in new Ada code. Remember that Object can still be abstract even if derived.) You could also uses a separate "Ready-me-for-Deallocation" dispatching routine, but that is neither as safe nor fool-proof as just letting Ada do it: there are special rules in the language that insure that Finalize is always called at least once.

The important thing here is that (using the terminology of the Ada 2007 predefined containers) the container is responsible for deallocating the elements as a whole, but any internal cleanup is the responsibility of the elements themselves. It's not possible (in general) to have objects that deallocate themselves — but that's actually a good thing: an object should be responsible for cleaning its contents up, but only the client can know how that object is going to be used, and thus how the memory should be deallocated. Otherwise you have unnecessary coupling between the object and its clients: the object type cannot be reliably used to declare objects on the stack (or in the predefined containers, or anywhere that non-standard storage pools are used, etc.).

Summary: The objects and the list are separate abstractions and should be kept separate. The list should allocate and deallocation elements (objects); the objects themselves should do any internal cleanup needed.

From: Matthew Heaney <matthewjheaney@earthlink.net>
Date: Fri, 01 Dec 2006 12:33:35 GMT
Subject: Re: Deallocating list of polymorphic objects?
Newsgroups: comp.lang.ada

> Can it be called more than once?

Yes. When you write Finalize you must write it in such as way as to ensure that it can be safely called a second time.

From: Matthew Heaney <mheaney@on2.com>
Subject: Re: Deallocating list of polymorphic objects?
Date: 1 Dec 2006 06:56:38 −0800
Newsgroups: comp.lang.ada

> Could you please throw some paragraph numbers from AARM that are relevant to this? I would like to take a closer look at this subject. The assertion that Finalize has to be safe w.r.t. multiple calls is a very important one and I don't

seem to remember it being mentioned anywhere (including The book).

I just used the search again here:

http://www.adaic.com/standards/05aarm/html/AA-SRCH.html

to search for a page with all of the words "finalize" and "twice" and this page dropped out:

http://www.adaic.com/standards/05aarm/html/AA-7-6-1.html

See RM05 7.6.1, Note 22.

You could also use Google groups to search CLA for posts with "finalize" and "twice" and you'll get other hits. This subject came up a lot after Ada 95 was released.

From: Georg Bauhaus <bauhaus@futureapps.de>
Subject: Re: Deallocating list of polymorphic objects?
Date: Fri, 01 Dec 2006 20:03:57 +0100
Newsgroups: comp.lang.ada

[…] Cohen's Ada as a Second Language has many pointers. One is an index entry, entitled "finalization invoked twice for the same object", which leads to Controlled types, and also to deferred abortion.

From: Matthew Heaney <matthewjheaney@earthlink.net>
Subject: Re: Deallocating list of polymorphic objects?
Date: Fri, 01 Dec 2006 03:52:44 GMT
Newsgroups: comp.lang.ada

The easiest way to do [that list of polymorphic objects] is using the indefinite form:

```
With Ada.Containers.
  Indefinite_Vectors;
package Object_Vectors is new
 Ada.Containers.Indefinite_Vectors
(Object'Class);
```

As others have pointed out, the Ada run-time properly handles deallocation of objects having a class-wide type, so there's nothing special you need to do.

Unicode pitfalls

From: "Hyman Rosen" <hyman.rosen@gmail.com>
Subject: Re: Ada generics
Date: 27 Dec 2006 11:06:36 −0800
Newsgroups: comp.lang.ada

> The "encoding language" is outside the programming language, so it is not the language problem

Remember that Ada wishes to be case-insensitive, so it cannot ignore Unicode issues if it wishes to allow Unicode characters in identifiers. Not to mention "normalization form KC". […]

From: Georg Bauhaus <bauhaus@arcor.de>
Subject: Re: Ada generics
Date: Thu, 28 Dec 2006 18:35:06 +0100
Newsgroups: comp.lang.ada

> Which is a BAD idea, IMO.
Which is a BAD idea, IMO. We cannot know anything about properties of letters in Klingon. As a practical example consider Russian where e can be used (and is) in place of ë see […], but not reverse. Or, maybe we should make Ada compilers capable to detect program written by Germans to consider ü and ue same?

Writing source code is a question of being practical, which is probably not easily formalized… An international character set for portable programs seems to leave only some choices open when they should be practical, does it not? Naturally, mathematical fancies like being complete, free of contradictions, etc. are out of the question when it comes to writing for both humans and computers. What's the point of having a high level language when you are only allowed identifiers that the most simplistic mechanical interpreter can "understand"?

Why is it that programmers become somewhat irrational and impractical when it comes to character sets? They do try to devise all kinds of pattern recognition algorithms, tricky transformations, get the best out of fuzzy measurement procedures, and so on. But not so with character sets. No no, every school child knows that characters must be such and such … (maybe the early exposition to characters is to be held accountable here, everyone is an expert ☺

Anyway, do we have some data that we could discuss that would explain the practical importance of Unicode/casing issues? Or, do we have programmers who are well versed in using a keyboard connected to a computer and still can't write a program that can tell apple characters from orange characters?

GNAT already supports the detection of identifiers that were spelled similarly. In case of errors, it lists their "relatives". Surely a helpful feature, and a proof that practical handling of natural language identifiers is possible. As an example, as you have been referring to German, consider that sharp s, 'ß', is usually written "SS" when capitalized. So "Straße" tends to become "STRASSE". Now if you have a composite word that has

- a 'ß', and

- an 's' right after it,

such as "Maßstab" (= scale, rule, yardstick), then from a simple minded formalist's perspective I could argue:

"Using Unicode is nonsense because there is no 1:1 mapping for the German word 'Maßstab' which will become 'MASSSTAB'. "SSS" is ambiguous, it could be "sß" or it could be "ßs". That's too big a challenge for a compiler write. So leave me alone with your Unicode and case insensitivity."

Is that what computer science has to answer when asked about characters handling?

Challenge: Try to find a significant number of German words that have an 's' before a 'ß'. What's the consequence of your findings? Even if there are ambiguities in other languages, ambiguities are not new to Ada (and C++, IIRC), and they have been addressed.

(It seems that the introduction of Unicode to Scheme 6 has recently made Lisp case sensitive based on arguments such as the one above. To me this shows "practicality" on the part of the language designer, vulgo just compiler writer's laziness.)

If the programmers' representatives (the ARG for example) agree that it is practical to exclude some casing rules or "representation rules", such as "ue" <-> 'ü', I'm perfectly happy. Because the rule *is* practical, it helps work, and to hell with mathematical fancies and game theoretic character shuffling possibilities, when they do not really matter.

> What about parsing the source right to left, or top to bottom?

The writing direction problem is solved. Similarly, it seems possible and practical to connect big endian and little endian computers, and have them cooperate using algorithms. Both exist, as do apples oranges, bananas, and pineapples. We can make nice fruit salads.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada generics
Date: Fri, 29 Dec 2006 20:25:28 −0600
Newsgroups: comp.lang.ada

For what it's worth, Ada says that all three of [Maßstab, Masßtab, and MASSSTAB] represent the same identifier. That's not ideal, but it's the best that we can do without dropping into the character handling mess ourselves.

This is even more interesting when you consider that there are alternative spellings for reserved words. For instance "acceß" is identical to "access". (See 2.3(5.c/2) in the AARM for more examples). We wrestled with that quite a while before deciding that such identifiers had to be illegal (2.3(5.3/2)); we didn't want them appearing in programs in place of reserved words.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada generics
Date: Wed, 3 Jan 2007 19:09:17 −0600
Newsgroups: comp.lang.ada

> Would "acce?" with Greek beta (?) and "if" with Cyrillic ? in it be valid identifiers?

Sure, the upper case of a Greek beta is still a Greek beta, it's not "SS" (and doesn't look anything like "ss", either). I

don't know much about Cyrillic, so I don't know the answer to that (but I suspect you do).

I would guess that you'll want some external style rules to prevent bogus mixing of letters from different character sets. That's not any worse that the style rules for capitalization and indentation that GNAT can enforce.

I've always limited myself to using the characters commonly available on Windows systems (roughly 680 glyphs), and there needs to be something that checks for use of letters that won't necessarily display well. But all of that is outside of the language.

It should be pointed out that one of the reasons for Ada's support of Unicode is that we had a long discussion of how to support Latin-9 (which contains the euro symbol). Eventually, we decided that that way lies madness — at least by using Unicode, there is only one definition to worry about, rather than a set of them. My only regret is that we didn't find a way to include real runtime UTF-8 support in the language: it's wasteful to store everything as 32-bit characters.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada generics
Date: Thu, 4 Jan 2007 19:32:26 −0600
Newsgroups: comp.lang.ada

> My God. A good third of the Latin and Cyrillic glyphs are same. Practically all vowels are. That means that *any* reserved word of Ada can be spelt as a proper identifier!

Yes, and so what? There would be little ambiguity introduced by using (say) "overriding" as an identifier, so the meaning would be obvious to the reader, and it won't confuse the compiler (usually it's more confusing to the writer who didn't remember that some word is reserved). There are some of them that should be avoided, of course, but there aren't many of those.

However, you alluded to a real concern in another message. That is, it's possible to write two different identifiers that look the same. That would be confusing and possibly cause problems. But that's already possible (depending on the font), so it just is a slight expansion of a problem that already exists. And it certainly can be handled with style checkers (identifiers containing mixes of Latin, Cyrillic, or Greek characters are suspicious, as are identifiers differing only by the replacement of Latin characters with Cyrillic equivalents).

If that is a real concern, just insist that all of your programs are edited with a 1984-vintage MS-DOS editor (like I do ;-), and you won't possibly be able to have a problem. Indeed, I expect most programmers will continue to do this (use tools that don't support Unicode), so any new problems will be limited.

> (and of course, there is no any chance to reverse this nightmare…)

I don't see a nightmare, but I do see a need to have decent style rules around the writing of identifiers. That's necessary even in Ada 83, they're just more complex now.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Organization: Adalog
Subject: Re: Ada generics
Date: Fri, 05 Jan 2007 10:08:32 +0100
Newsgroups: comp.lang.ada

☺ This check is already implemented in the wavefront version of AdaControl (not yet in the public version).

From: Georg Bauhaus <bauhaus@arcor.de>
Subject: Re: Ada generics
Date: Fri, 29 Dec 2006 20:39:05 +0100
Newsgroups: comp.lang.ada

> [Computer Science] is all about introducing formal languages in place of natural ones, for obvious reasons.

But Unicode and/or ISO 10646 *are* formal things.

> Corollary: never ever make a formal language (Ada) dependent on a natural one (German). That would make the former natural.

I don't see how identifier rules are natural (not formal), whatever the natural language is that guides the choice of names in a particular program. Because of I18N efforts tool makers can do some work to make programming easier for humans, even if this means supporting more than the most trivial interpretation of character bit patterns.

Take Google as an example of why finding things that were spelled "incorrectly" is so immensely useful. And successful.

From: Georg Bauhaus <bauhaus@arcor.de>
Subject: Re: Ada generics
Date: Sat, 30 Dec 2006 15:53:24 +0100
Newsgroups: comp.lang.ada

> Because these rules are subject of endless chaotic political changes.

I don't know about ISO or ARG political changes — besides the rather interesting glimpses at language debates during Ada 9X in the archives, if you want to call this politics.

But where is the chaos in the simplified Unicode rules that have been adopted for Ada 2005 (or 2007)? You won't need thermodynamics to find out whether or not a given word is an identifier?

Should the characters '1' and 'l' be removed from the Ada standard characters because that's a similar chaos? Should there be a ruling about Finalisation versus Finalization?

> Do you want programming languages acting as Google?

No, by referring to the usefulness of Google search I meant that

- People value Google search service because it finds things, even noticing possible spelling errors, and it overcomes lack of structure of "the Internet".
- Programs have spelling errors, lack perfect structure.
- Program analysis will provide better errors/warnings/info if identifier spelling, syntax, languages, etc. are given the attention they deserve instead of asking humans to always provide proper, clean, simplified input.

Perhaps compilers can profit from a notion of Almost-Homograph. Something like soundex. When "overriding" is missing, this circuitry could warn programmers of too similar identifiers. Or of a possible misspelling of Finalisation.

Or was it Finalization?

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada generics
Date: Thu, 28 Dec 2006 18:09:20 −0600
Newsgroups: comp.lang.ada

> As a practical example consider Russian where e can be used (and is) in place of ë see […], but not reverse. Or, maybe we should make Ada compilers capable to detect program written by Germans to consider ü and ue same?

The Unicode standard has grappled with these issues and produced results which are useful for the vast majority of languages. Surely Ada is not going to repeat that work (and arguments). And Ada is not going to drop case insensitivity and start claiming that "this" and "This" are somehow different.

> They reap what they sowed. Should Ada or C++ go into that mess?

Well, that's irrelevant because they have. Ada 2005 says that the semantics of a program not in Normalization form KC are implementation-defined. (2.1(4.1/2)). That was done because there was concern about programs that are represented differently being treated the same (we originally considered requiring converting into that form).

Similarly, upper case conversion is defined by various Unicode properties (such as Upper Case Mapping) (2.1(5/2)). It should be noted that such conversions aren't necessarily reversible, but that's irrelevant to identifier equivalence. Identifier equivalence is defined in 2.3(5 – 5.3/2).

This is more complicated than the English-only definition, but it was thought to be mandatory to get approval of a new

standard. (This sort of internationalization is being required of all languages: C++ has a number of proposals on the table for handling this as well.) It's also a ramification of case insensitivity — the only alternative would be to completely abandon it, and that would be very bad for compatibility with Ada 95.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Ada generics
Date: Fri, 29 Dec 2006 20:40:46 −0600
Newsgroups: comp.lang.ada

> I don't see why letters of identifiers must be all Unicode letters. I wouldn't allow anything but Latin. In any case it just cannot be open-ended.

Because higher ups at ISO/IEC has said that such things need to be allowed. If you want an ISO/IEC standard, you have to be responsive to their wishes. Personally, I think anything beyond 8-bit characters is going too far (even for strings): if it's not worth doing in English, its not worth doing! [For me, the universe revolves around Madison, WI and everyone should speak (American) English (dropping all of those other archaic languages) so that everyone can communicate without unnecessary barriers. This is very similar to my stand on Ada vs. other programming languages. But I'm not particularly surprised when someone disagrees with any of those positions… ;-)]

And it's not "open-ended". It follows a published standard (Unicode), just like the earlier versions of Ada followed other published standards (ISO/IEC 10646 in the case of Ada 95).

In any case, Unicode identifiers are part of the Ada Amendment. And I would be very surprised if we went backwards on that; such a change would be very incompatible. (I personally don't believe that many programs will use Unicode identifiers, but it's likely to be non-zero, maybe 5%.)

Why is task termination disallowed in Ravenscar?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 29 Jan 2007 20:53:34 +0100
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> The N442 document states that Ravenscar profile forbids task termination. I understand that task termination is the fact of a task finishing its job and completing. Do I understand correctly that Ravenscar programs are by definition running forever? What about programs that are expected to finish?

Yes, it is my understanding as well, and I'm happy with that.

I remember being impressed with Ada because you could write an infinite loop without a faked up condition. The idea being that in Ada the typical infinite loop would normally be terminated by detonation. —Larry Wall

The Ravenscar profile is specifically targeted at high-integrity systems, where infinite loops are, I think, the norm.

From: Matteo Bordin <matteo.bordin@gmail.com>
Date: 30 Jan 2007 06:24:44 −0800
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> Sorry, but I don't see anything in the concept of high-integrity software that would make it a norm. High-integrity software is a set of quality objectives, whereas infinite loops are (or aren't) part of system requirements. These should stay independent, even though I understand that expectations for both often come in pairs.

The Ravenscar profile is aimed to high-integrity real-time systems. A static set of working tasks is a requirement to perform sound feasibility analysis (at least within a given execution mode).

> Still, it looks like I cannot say:
> pragma Profile(Ravenscar);
> in my Hello World program even though this program meets the objectives of the profile. That's not fair! ☺

On real-time kernels supporting the Ravenscar profile, even the main procedure must contain an infinite loop.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 31 Jan 2007 09:12:40 +0100
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> If Ravenscar really requires that the main procedure be non-terminating, I'm happy to learn that. From a very formal point of view I guess this requirement means that the kernel need not implement "await for task termination" even in the environment task.

Indeed, one of Ravenscar's goals is to make the necessary kernel easy to certify to the most stringent safety standards. As with all high-integrity software, the best way to achieve this is to make things small and simple. So, not only does Ravenscar avoid the need to wait for task termination, but also the tasking model (priority ceiling inheritance) avoids the need for locks completely. Imagine a tasking kernel with no mutexes ☺

Ravenscar is beautiful, IMHO.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Tue, 30 Jan 2007 17:48:52 GMT
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> Well, it terminates. What does termination mean in a "high integrity" embedded system — does the hardware go away? ☺ I think if someone wants to shutdown such a system the thing happening is, that every task goes into idle mode and the last thing a controlling task does, is, to display (or otherwise indicate) "you may now shut off power, the countdown to eject the warp core has been stopped" or something like this.

Termination in an embedded system often means the processor no longer has power ☺

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 31 Jan 2007 10:59:03 +0100
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> I wrote a Hello World program and I want to impress my boss telling him that my program complies with Ravenscar recommendations. That sounds much more serious than a plain dumb Hello World program!

A high-integrity "hello world"? With tasking? ☺

```
pragma Profile (Ravenscar);
with Ada.Text_IO;
with Ada.
  Synchronous_Task_Control;
procedure Hello is
   Blocker : Ada.
     Synchronous_Task_Control.
     Suspension_Object;
begin
   Ada.Text_IO.Put_Line("Hello
     Ravenscar!");
   --  loop
     -- null;
    Ada.
      Synchronous_Task_Control.
      Suspend_Until_True (Blocker);
   --  end loop;
end Hello;
```

That should solve your CPU utilisation problem ☺

> OK, back to serious mode.
> One of the Ravenscar objectives is to allow implementations to provide stripped-down runtime when the profile is requested. This is a nice feature, even for programs that are not safety-critical in nature. How does GNAT handle this? Can I expect it to build smaller (faster?) executables when I say pragma Profile(Ravenscar) provided that the program complies to all the restrictions anyway?

I'm not sure how GNAT handles this, and I think it depends on the target. It makes no sense at all to write high-integrity software running on a low-integrity operating system (not to mention low-integrity hardware); the intention is that the high-integrity Ravenscar run-time kernel *is* the operating system.

As a consequence, Ada.Text_IO in a high-integrity system makes little sense, unless you have a high-integrity console driver. Since the console driver would be hardware-dependent, you'd have to write your own to complement GNAT's minimal Ravenscar tasking kernel.

I think that's why, in effect, high-integrity implies embedded.

In low-integrity, non-embedded software, you cannot benefit from the "minimal kernel", "lock-free operation" or "configurable scheduling policies", but you can benefit from other inherent properties of the tasking model, which reduce the opportunities for deadlocks.

PS. Keep in mind that calls to Ada.Text_IO.Put_Line are "potentially blocking", so you cannot call them from a protected object in Ravenscar. See ARM 9.5.1(8, 10), D.13.1(4/2), H.5(5/2).

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Tue, 30 Jan 2007 21:15:18 +0200
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> On real-time kernels supporting the Ravenscar profile, even the main procedure must contain an infinite loop.

[…] I have seen a Ravenscar Ada implementation that requires the main procedure to end with an infinite loop, but I think that is a non-standard requirement. A trivial one, of course.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Wed, 31 Jan 2007 09:53:53 +0200
Subject: Re: Ravenscar — program termination
Newsgroups: comp.lang.ada

> Then what would happen if there were no tasks other than the environment task?

OK, good point. If a Ravenscar program has no tasks (other than the environment task) then the main procedure must not terminate, agreed. (Also there must *be* a main procedure.)

> The RM wording says "all tasks", and that includes the environment task.

If I remember correctly the documentation for that implementation said that it was a non-standard requirement. The reason given for this requirement was very implementation-dependent: the main procedure became the "idle task" for the scheduler, which required that the idle task be always "ready", therefore the main procedure was not allowed to terminate. But of course the documentation may have been wrong to say it was non-standard.

If Ravenscar really requires that the main procedure be non-terminating, I'm happy to learn that. From a very formal point of view I guess this requirement means that the kernel need not implement "await for task termination" even in the environment task.

GNU/Linux Magazine France Ada series

From: Yves Bailly <kafka.fr@laposte.net>
Date: Mon, 05 Feb 2007 18:54:58 +0100
Subject: Re: Ada is popular after all
Newsgroups: comp.lang.ada

[…]

> Another thing that makes Ada trendy nowadays is the enduring series of articles by Yves Bailly in GNU/Linux Magazine France. The December issue contains article #14 in the series, ending with a mention of "the next article"…

Thanks for noting them ☺ I hope you found them valuable enough, you most probably didn't learn anything, but hopefully some might "see the light". I can say that I received numerous feedback for those articles (asking for source code, more details, etc.), already more than for my serie on Qt (C++, 24 on Qt3, 8 on Qt4). So yes, it seems that Ada is rather popular after all.

If you have any comment about the articles, please let me know.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 05 Feb 2007 21:28:09 +0100
Subject: Re: Ada is popular after all
Newsgroups: comp.lang.ada

Just one: they're really good. I bought all issues of GMLF where they appear, except for Jan and Feb 2007 because I couldn't find them in Brussels. Keep up the good work!

At a local GNU/Linux copy party, I once introduced Ada to two students in CS, and noted that "unfortunately Ada is not a fashionable language". One of the students said: "oh yes, it is fashionable, what with all these articles in GNU/Linux Magazine France!"

I hope you find more to say about Ada in future articles. It seems you've now covered pretty much everything about the language, but maybe (just a suggestion) you might like to extend the series with Toy Lovelace, Qt4Ada or GtkAda?

From: Frederic Praca <frederic.praca@freebsd-fr.org>
Date: Mon, 5 Feb 2007 22:05:49 +0100
Subject: Re: Ada is popular after all
Newsgroups: comp.lang.ada

And (another suggestion) why not talking about Open Ravenscar (http://polaris.dit.upm.es/~ork/) or MarteOS (http://marte.unican.es/) which allow to code in Ada for Real Time embedded systems ?

From: Yves Bailly <kafka.fr@laposte.net>
Subject: Re: Ada is popular after all
Newsgroups: comp.lang.ada
Date: Tue, 06 Feb 2007 00:11:53 +0100

About ToyLovelace, I'm trying to convince Xavier Grave (the author) to write something about it by himself. Qt4Ada is not yet enough advanced I'm afraid (but it's improving). And there has been already an article about GtkAda in GLMF: http://www.fdn.fr/~sdescarp/realisations/articles/LM66/LM66_Ada GTK+, le duo gagnant.html Also, I don't know Gtk well enough to write anything sensible about it.

I was trying to write something about Annex E, using Glade, but with no luck. It seems there are some problems in the latest Glade, at least it's what have been said in this forum (the thread starting at "Help with Glade (Annex E) on Windows" by Gene).

About OpenRavenscar or MarteOS (to answer Frederic), I have absolutely *no* experience in real-time or embedded programming, so I won't write anything about it. But if you wish to write something yourself, I'd be glad to introduce you to GLMF's redactor-in-chief. […]

Ada and microcontrollers

From: "Talulah" <paul.hills@uk.landisgyr.com>
Subject: Re: Translating an embedded C algorithm
Date: 17 Jan 2007 05:31:28 −0800
Newsgroups: comp.lang.ada

[…] Since the majority of embedded RT

projects ARE written in C, then the majority audience for the book will be interested in seeing examples written in C. That is no reflection on the "quality" of the language, but is just facing facts. […] This is not a safety-critical product at all, but one which in those volumes must be designed for the lowest cost possible. Hence the use of 4 diodes as a temperature sensor, and hence (this may get more argument) the use of C rather than Ada. Crossing an 8k ROM boundary (the current code is just over 32kbytes) adds 3 pence (UK) to the cost on this microcontroller (Renesas H8/3827). Multiply that by 18 million and the resulting half million UK pounds is the reason I program in C. Thirdly, I am thick-skinned enough to brush off rudeness from newsgroup posters, it is just a shame that you decided not to post any useful comments.

[…] It hopefully will allow readers to see the advantages that Ada can give them, and encourage them to research Ada as a possible language for future products that would benefit from these advantages, but also to understand the disadvantage of code size and speed (hard hat firmly in place after that!).

I understand now that a direct translation of the code into Ada would not be a good illustration of the Ada language, and that an example which performed the same calculation, but added the additional useful features would better serve. I have been sent an example written by a member of your newsgroup which does this, introducing better protection. I will also write a section in the text describing why a direct translation is a bad idea, and demonstrate how the extra features have made the code much more reliable. […]

From: Talulah <paul.hills@uk.landisgyr.com>
Date: 18 Jan 2007 06:19:26 −0800
Subject: Re: Translating an embedded C algorithm
Newsgroups: comp.lang.ada

[…]

> Where we have hard data, they show that Ada reduces development costs by 1/2 over C, reduces post-deployment errors by 1/4, and reduces the cost to fix an error by 1/10.

I accept that may be the case. However, there are no Ada tools for very many microcontrollers. The cost drives the choice of microcontroller, and the microcontroller then drives the choice of development system.

> There is a least one documented case of Ada producing smaller code than hand-optimized assembler.

I love these statistics. There's a Java vendor who reckons the byte code runs faster than C as well. You can prove anything if you have choice over the tools that you use to produce the results. This "hand coded assembler" could mean anything — it could mean taking Ada compiled code and ADDING instructions to it!

Thinking logically, if the assembler coder was any good, he can always produce code of equal size to compiled code, and should always produce tighter code. It just depends what he is trying to prove, and who has sponsored him to do the work!

> Dewar has a number of examples of equivalent Ada and C code that produce identical object code. Thus, the assumption that C is necessary to keep costs down is unsupported, a fact that anyone qualified to choose the language for such SW should know.

When was the last time you wrote an Ada program to run on a microcontroller such as a PIC or ATMega48, i.e. something around the $0.50 price mark? And can you tell me a compiler vendor? C compilers are available for both these devices. Therefore most embedded developers (who are not in the military market and where cost is the greatest issue) cannot choose the language they develop in. […]

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 19 Jan 2007 04:52:22 GMT
Subject: Re: Translating an embedded C algorithm
Newsgroups: comp.lang.ada

> There is Ada for every microcontroller with an ANSI C compiler. See http://sofcheck.com/products/adamagic.html

I've seen the compiler in use at Praxis, precisely to allow high-integrity SW in SPARK to target a microprocessor without a dedicated Ada compiler. It worked much like any other compiler.

From: Warner BRUNS <Warner.Bruns@cern.ch>
Date: Fri, 19 Jan 2007 11:13:47 +0100
Subject: Re: Translating an embedded C algorithm
Newsgroups: comp.lang.ada

I have purchased the AdaMagic Ada to C translator from SofCheck, after trying it out for some weeks. I am using it since several years as my main Ada compiler. I did not get it via a download but I did send an email, stating my requirements, and they could be met.

Checks in Ada and C

From: Larry Kilgallen <Kilgallen@SpamCop.net>
Subject: Re: C compiler warnings
Date: 4 Dec 2006 22:24:24 −0600
Newsgroups: comp.lang.ada

> […] It seems to show that while Ada is going in one direction (strict compiler checking of code), C is going in the opposite direction (assume the developer knows what he/she is doing).

The quote does not say "C" is going in the opposite direction from Ada. It shows at most that a particular implementation of "C" is going that way. The HP (nee Compaq (nee DEC)) C compiler on VMS has been steadily getting more and more checks added to find programmer errors. Of course it can never do so much as an Ada compiler, but it is not the case that all C compilers are decreasing their level of checking.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: C compiler warnings+0100
Newsgroups: comp.lang.ada

> Going in the opposite direction? C's basic design philosophy has always been to make that assumption. It seems to me that the C compilers with extensive warnings are the ones going in an unusual direction.

The initial C design was. But its further evolution has always been in the direction of becoming more contract-based, more like Ada. The difference though was in the treatment of contracts. In C traditionally less attention was paid to enforcing the contract on both parties. The contract (nonull) was assumed on the callee's side, but ignored on the caller's one. The rationale probably was that C usually does not try to enforce the contracts at run-time (Ada does). This can explain why nonull was not attempted to check. It is not fully statically checkable. So why should we bother?

> I wish they'd all get rid of the warnings; then maybe more people would use a well designed language.

I don't think so. They just would use more and more tools instead. It is the tool chains which compensate language deficiencies. Just look around, people are ready to invest into tools, which should by sole magic compensate for everything, from the use of C++ to mismanagement. A tool might cost several thousands of dollars, and it could be dozens of them. Try to sell a compiler for that money. Something is deeply wrong in all this.

Embedded Ada learning

From: "Mike Silva" <snarflemike@yahoo.com>
Date: 22 Feb 2007 16:59:09 −0800
Subject: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

[…] I'm a long-time embedded programmer and a dabbler in Ada (I'd use it more if I could get paid for it). Now I'd like to play around with Ada on a single-board computer. I have no particular goals in mind other than to try something "neat". So, what is likely to be the quickest, most foolproof way for me to get from here to there?

I'm assuming I'll want an OS on the board for the runtime stuff. Would one of the *BSDs or Linux be the way to go? If so and given my intentions, would there be a reason to choose one over the other? My contrary side wants to try a *BSD, but I have no experience in any of them _or_ Linux.

And what about processor family? I was thinking ARM or Coldfire or PPC (something in the MPC5xx family maybe). Again, would there be an Ada- or OS-related reason to choose one over the others?

I did ask an abbreviated version of this question at the bottom of another thread, but I'm hoping this thread will have more visibility. So, is all of this do-able by a mere mortal? Many thanks for any advice!

From: "Mike Silva" <snarflemike@yahoo.com>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: 23 Feb 2007 05:13:13 −0800
Newsgroups: comp.lang.ada

[…] I was asking in terms of what OSes and/or processors might have better support, or fewer gotchas. What I want to avoid is the situation where some port or feature X has not been kept up to date, or is known to have problems. For example, I believe I remember some years back that there was a problem or poor performance with some version of Linux threads. That kind of thing. What I want to do is not accidentally drift so far out of the mainstream that I cause myself grief.

> For getting quickly and easily into embedded Ada, you could try Lego Mindstorms. There's a free Ada => NQC compiler available. This is not the preferred way, of course, but it is a way.

I appreciate that suggestion, but I'd like to work with a 32-bit mainstream processor family. While my goal now is just to play around, if I could use what I learn in some real products down the line so much the better. It's that hopeful thing about maybe getting paid to do Ada (after I first have some fun with it).

From: Steve <steved94@comcast.net>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: Thu, 22 Feb 2007 20:41:57 −0800
Newsgroups: comp.lang.ada

Two free RTOS I am aware of are RTEMS and MaRTE:

http://www.rtems.com/wiki/index.php/RTEMSAda

and:

http://marte.unican.es/

These both work with GNAT.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: Fri, 23 Feb 2007 08:56:00 −0500
Newsgroups: comp.lang.ada

[…] There are several readily available solutions to the requirements as you state them.

They all cost money, several thousands of dollars. You don't say how much money you are willing to spend; is $10k too much?

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sat, 24 Feb 2007 05:45:37 −0500
Subject: Re: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

> As this is just a hobby/learning thing at the moment, $10k is way, way too much. I'd like to keep the cost including SBC under, say, $1000. Do I dream the impossible dream? I hope not, because I'd really like to give this a try and perhaps learn enough to use embedded Ada commercially down the line (at which time somebody else could fork up the $10k).

My main job at work is building a satellite simulator (GDS; http://fsw.gsfc.nasa.gov/gds/). It's a hard real-time system. Some people would say it's not "embedded" because it has an Ethernet connection to a sophisticated user interface, but that's another discussion.

I develop all of the software for GDS on Windows. I've written emulation packages for some of the hardware. I do this because it's easier to debug top level code without the hardware getting in the way, and the development tools (Emacs, GNAT, GDB) work better on Windows than on the target OS (Lynx). Once it's working on the emulator, then I run it on the real hardware. Sometimes it Just Works, sometimes I have to get out the scope and see what's going on. In that case, I try to fix the emulator so I won't have to use the scope again ☺ Using the scope can be fun, but it's always way slower than using gdb or higher-level tests.

So I suggest you take a similar approach. Make up some hardware that you'd like to play with, and write an emulator for it. Then write some code to make that hardware dance. […]

If I was hiring (which I'm not), I'd look for someone who can implement algorithms from simple problem descriptions. That's my biggest need.

Understanding how to use a scope to debug hardware problems is also good, but not as important. It's easier to learn that on the job.

If you want to expand into "real hardware", there are data acquisition and control devices that plug into PCI slots, and come with Windows drivers. I don't use them, but I think they are fairly inexpensive. Anything for Windows is going to be the cheapest solution, because of economies of scale. And they are "real-time" enough to get your feet wet.

From: "Mike Silva" <snarflemike@yahoo.com>
Date: 24 Feb 2007 11:11:04 −0800
Subject: Re: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

> You can do all of that on free software and cheap hardware.

It's that "make the hardware dance" part that seems much more complicated with Ada than with C-plus-an-OS (but the benefits seem much greater as well). That is to say, choosing an underlying runtime enviornment and getting it not only set up on the hardware, but integrated with the GCC Ada compiler. So, ignoring the question of preferred processor families (least amount of unnecessary gotchas), I'm still wondering about which OS is the best choice to get something up and running. Can anybody comment on the relative merits and troubles of running Ada on Linux, one of the *BSDs, and RTEMS?

> Another area to explore is FPGA programming. […] FPGA development relies heavily on simulation, which does not require real hardware.
> If you are ambitious, you can try to tie the ghdl simulator to your Ada code, to allow testing the Ada interface to the FPGA in simulation. I haven't done that yet, but I wish I could.
> Someone who can do both Ada and VHDL would be a very valuable person!

Well, I did pick up a VHDL book a while back. Maybe it's a sign ☺ But first I want to get Ada running on a SBC.

From: "Mike Silva" <snarflemike@yahoo.com>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: 1 Mar 2007 12:22:05 −0800
Newsgroups: comp.lang.ada

[…] I've ended up going down a somewhat different path, for now at least. I've gotten this board http://www.olimex.com/dev/lpc-e2294rb.html because it has just about the right mix of horsepower and features for some ideas I have. I know this board isn't big enough to run Linux or FreeBSD, so I am going to look at Ada on RTEMS instead. But again, thanks for the follow-up, and I am going to look up the UNC90 as well.

Ada and VHDL

From: Jeffrey Creem <jeff@thecreems.com>
Date: Sat, 24 Feb 2007 07:27:01 −0500
Subject: Re: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

> Someone who can do both Ada and VHDL would be a very valuable person!

I'm always surprised that VHDL engineers are not more open to Ada given how close the syntax is. The standard joke where I work is that VHDL is just like Ada except the capslock is always stuck on and comments are apparently forbidden ;)

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk.uk.uk>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: Sat, 24 Feb 2007 22:10:22 GMT
Newsgroups: comp.lang.ada

I came to Ada from VHDL. When I first encountered VHDL, my first though was "Wow! You can say what you mean clearly". Features like user defined types (ranges, enumerations, modular types, multi-dimensional arrays) gave a feeling of clarity and integrity absent from software development languages.

So when I found that you could get the same benefits of integrity in software development from a freely available compiler, it didn't take long to realize what I'd been missing! Ada is without doubt the language at the pinnacle of software engineering, and infinitely preferable to Pascal, C++ or Modula 3 as a first language in teaching.

But I have ever since wondered why the VHDL and Ada communities are so far apart. It seems like such a natural partnership for hardware/software co-development. And there is significant scope for convergence of language features — fixing the niggling and unnecessary differences too. Physical types, reverse ranges, configurations, architectures, defered constants and ultra-light concurrency come to mind from VHDL. And general generics, private types, tagged types, controlled types from Ada (does the latest VHDL have these?)

Perhaps a common denominator language can be devised which has the key features of both, with none of the obsolescent features, and can be translated into either automatically? Something like this might allow a "rebranding" of Ada (i.e. a new name, with full buzzword compliance), and would be ideal to address the "new" paradigm of multicore/multithreaded processor software, using the lightweight threading and parallelism absent from Ada as we know it. For those who know Occam, something like the 'PAR' and "SEQ" constructs are missing in Ada.

While the obscenities of C-like languages thrive with new additions seemingly every month, the Pascal family has withered. Where is Wirth when you need him?

(Don't take it that I dislike C. Or assembler. Both have their legitimate place as low-level languages to get the machine code you want. Great for hardware hacking. Lousy for big teams, complex code.)

One can dream…

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: 25 Feb 2007 05:10:51 −0800
Newsgroups: comp.lang.ada

> Where is Wirth when you need him?

In retirement. He did give the after-dinner speech at the VSTTE conference in Zurich in 2005, and he was brilliant. I wish I could remember exactly what he said about C++ — I think the word "abomination" was in there somewhere… ☺

I met him afterwards and had a brief chance to chat and thank him for his influence on SPARK.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sun, 25 Feb 2007 10:08:57 −0500
Subject: Re: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

I haven't actually studied the additions in VHDL 2003, but I don't think most of these Ada features make sense for VHDL. At least, if you are using VHDL to program FPGAs.

And reverse ranges make things ambiguous, especially for slices of unconstrainded arrays. So I don't want to see those in Ada.

One big problem with VHDL is that it was not actually designed for programming FPGAs; it was designed as a hardware modeling language. People discovered that you can sort of use it for FPGA programming, and it was the only standard language available for that purpose. There are many things that you can say in VHDL that make no sense in an FPGA, so each compiler vendor picks a slightly different subset of VHDL to support for FPGAs, and gives things different meanings.

Why would you want to translate [Ada and VHDL] into each other? The semantics of VHDL are _significantly_ different from Ada. A VHDL process is _not_ an Ada task.

Although I suppose if you decided to use VHDL to write code for a CPU instead of an FPGA, you could decide that they were the same.

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk>
Date: Mon, 26 Feb 2007 21:18:20 GMT
Subject: Re: Preferred OS, processor family for running embedded Ada?
Newsgroups: comp.lang.ada

> Have you looked at AADL?

I hadn't seen this. Interesting.

It looks quite similar in some respects to what I was thinking of. Particularly the emphasis on multiple representations of the underlying program (graphical, XML, plain text etc).

It looks like it draws together aspects of VHDL and Ada without really being based on either. Is it going to be the next Big Thing?

From: Jerome Hugues <hugues@antigone.enst.fr>
Subject: Re: Preferred OS, processor family for running embedded Ada?
Date: Wed, 28 Feb 2007 12:25:22 +0000 (UTC)
Newsgroups: comp.lang.ada

> A lot of people is trying to make this happen ☺ In an nutshell, AADL is a design language at system level; many concepts are inherited from Ada, and you'll find many Ada people involved (Joyce Tokar did the Ada binding), as well as AADL presentations at Ada conferences.

AADL is not just a design language, it also allows you to perform a wide range of checks and code generation on high level models, or some refinements of them.

We, at ENST, are developping Ocarina, that includes an AADL-to-Ada code generator. We got some interesting results in generating Ada code that matches many restrictions from the HIS annex from AADL models.

See http://ocarina.enst.fr/ for more details

Also, Cheddar, the scheduling tool-suite, has some support for AADL, same goes for STOOD from Ellidiss.

Which means, as stated by Jean-Pierre, that the Ada community is also involved in this language, and that links between the two are strong.

Structured exception information

From: Robert A Duff
Subject: Re: Structured exception information
Date: Mon, 15 Jan 2007 12:28:14 −0500
Newsgroups: comp.lang.ada

> If there are problems during the execution of the constructor function, the exception is raised, so that there is no X object in a bad state. How can I pass some error information from the constructor function out, so that it's used when the exception is handled?

There is no good way to do this in Ada. You can attach any information you like to an exception, if you are willing to encode it as a String — but then you lose static type checking. You can put the info in a global variable, but that's bad for several reasons (not task safe, can be accessed outside of any handler …). You can put the info in a Task_Attribute, but that's rather a pain — verbose and inefficient.

From: Robert A Duff
Subject: Re: Structured exception information
Date: Mon, 15 Jan 2007 17:32:37 −0500
Newsgroups: comp.lang.ada

> We did look at this issue when working on the Amendment. The "obvious" answers seem to have issues with visibility and compatibility with existing Ada.Exceptions mechanisms.

If we had done it right in Ada 95, we wouldn't have had the Ada.Exceptions kludge in the first place, so no need to be compatible with it. […]

The attitude about this feature during Ada 9X seemed to be:

1. Folks should not overuse exceptions. (I agree.)

2. Therefore, we should make exceptions painful to use. (Sorry, that does not follow.)

The problem with (2) is: what about the cases where exceptions ARE appropriate? Pushing people in the direction of encoding information un-type-safely as Strings, or using global variables, or whatever is not helpful. It's like removing the guard rail from a dangerous curve in order to make drivers slow down.

The language designer should always assume that programmers are competent — in this case, that they can decide whether exceptions are appropriate in any given case — and not try to prevent people from doing bad things. (Preventing people from doing bad things by accident, however, is Good Language Design.)

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Structured exception information
Date: Tue, 16 Jan 2007 16:36:13 −0600
Newsgroups: comp.lang.ada

> I don't know what the visibility issues were, so I can't comment on that. Do you happen to know which AI this was?

Not off-hand, but it only takes a minute to look up…

It was AI-264, "Exceptions as types". The title alone suggests trouble: the main issue is to provide type-safe data along with exceptions. The minor issue is a better way to deal with sets of exceptions. Neither of those necessarily require making exceptions into types.

My personal feeling is that we solved a lot of the problems that we had with the exception proposal when we dealt with all of the issues that nested tagged types brought up. When we considered AI-264, we hadn't yet gone through that exercise, and the entire thing looked impossible. Having solved some of the related issues, it would be easier to deal with now. It would be even easier if we had a mechanism for user-defined 'Image (which would allow automatic converting to strings to keep the existing Ada.Exceptions routines working).

From: Robert A Duff
Date: Sat, 20 Jan 2007 17:07:18 −0500
Subject: Re: Structured exception information
Newsgroups: comp.lang.ada

[…]

> If you look into the details of "structured exception handling" in other languages and implementations, they have bugs, and fundamental flaws in design.

C++ always had finalization (destructors). Later on, exceptions were added. There was much moaning and gnashing of teeth from implementers, claiming "exceptions are hard to implement properly".

Ada always had exceptions. Later on, finalization was added. There was much moaning and gnashing of teeth from implementers, claiming "finalization is hard to implement properly".

The truth is, the interactions between exceptions and finalization are nasty, and hard to get right.

From: Georg Bauhaus <bauhaus@arcor.de>
Subject: Re: Structured exception information
Date: Wed, 31 Jan 2007 19:58:10 +0100
Newsgroups: comp.lang.ada

[…] I worked on a system involving 3 companies where one style of exception handling was to be silent about them (empty handlers). Another style was to write a few words that could be understood by the programmer who wrote the handler because he knew the context. These were (are, I think the programs are still up and running) communicating programs, no source code was exchanged.

But when the exceptions were reported as strings into some log some of us on all sides were out of luck. Strings are too easy to write. People didn't give them enough attention, and they didn't create exception types either even when this was possible (in Java). And you don't ask for education across company borders if there is a hierarchy. Let alone speak of education!

> The system design says exception handlers only have to add information, never subtract it.

Adding information is good, and the advice also needs to mention that exceptions raised *do* have to provide some information ☺

[…] I'm not so sure what is easier to do, write a good exception message or build up an exception object containing the necessary information about what happened. It seems easier to just write an exception message string. Easy programming. But once it is written, the information is as inflexibly coded as can be: You need parsing if you have to extract it. I need to extract it if and when I cannot change what a 3rd party library is reporting. There just is no "education" or telling them what they should write in exception messages. […]

From: Robert A Duff
Subject: Re: Structured exception information
Date: Fri, 19 Jan 2007 10:45:30 −0500
Newsgroups: comp.lang.ada

Building strings is fine. Requiring clients to parse them is evil — not type safe.

The whole point of exceptions is to separate the detection of potential errors from the handing of them. A well-designed exception mechanism would allow the client to make these decisions:

Is this condition really an error?

If so, is it a recoverable error, or is it a plain old bug?

If recoverable, what should I do?

If it's a bug, should I print out useful information? Useful to the user, or useful to the programmer who wants to fix the bug (or both)? (Example: if the GNAT front end detects a bug in itself, it prints out the line number it was processing at the time, which is useful to the user who wants to find a workaround.)

Granularity of handling — do I want to handle all I/O errors, or just the "disk full" error?

Etc.

Constructing a string at the "raise" point is wrong because it presumes that the client wants to print a string and exit, and it presumes the format of that string. If that were OK, then why have exceptions — why not make the code that detects the error print a string and exit?

Suppose we want to print error messages in French. If the "raise" point constructs a message in English, the client can't make that decision.

Conference Calendar

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

The information in this section is extracted from the on-line *Conference announcements for the international Ada community* at: http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2007

☺ April 03-06 — 13th IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2007), Bellevue, Washington, USA. Topics include: embedded and open real-time systems and computing.

♦ April 17-19 — 13th **International Real-Time Ada Workshop** (IRTAW'2007), Woodstock, VT, USA. Topics include: early experiences in using Ada 2005 for the development of real-time systems and applications; implementation approaches for the new real-time features of Ada 2005; developing other real-time Ada profiles in addition to the Ravenscar profile; implications to Ada of growing use of multiprocessors in development of real-time systems; paradigms for using Ada 2005 for real-time distributed systems; definition of specific patterns and libraries for real-time systems development in Ada; how Ada relates to the certification of safety-critical and/or security-critical real-time systems; current ISO reports related to real-time Ada and new secondary standards or extensions; status of the Real-Time Specification for Java and other languages for real-time systems development, and user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems; lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects.

April 25-27 — **Software & Systems Quality Conferences** (SQC'2007), Duesseldorf, Germany.

☺ May 07-09 — 10th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2007), Santorini Island, Greece. Topics include: Programming and system engineering (ORC paradigms, languages, RT Corba, UML, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, system of systems, etc.); System software (real-time kernels, middleware support for ORC, extensibility, allocation, scheduling, fault tolerance, security, etc.); Applications (embedded systems (automotive, avionics, consumer electronics, etc), real-time object-oriented simulations, etc.); System evaluation (timeliness, worst-case execution time, dependability, fault detection and recovery time, etc.); ...

☺ May 20-26 — 29th **International Conference on Software Engineering** (ICSE'2007), Minneapolis, Minnesota, USA. Theme: "Developing Dependable Software".

- ☺ May 21-22 — 1st **International Workshop on Aerospace Software Engineering**. Theme: "Managing the Complexity of Aerospace Software". Topics include: tools and methods for the effective modeling, analysis, development and maintenance of aerospace software; systems and applications; etc.
- ☺ May 22 — 1st **Workshop on Assessment of Contemporary Modularization Techniques** (ACoM.07). Topics include: Lessons learned from assessing new modularization techniques, Empirical studies, Comparative studies between new modularization techniques and conventional ones, Software metrics and quality models, etc.
- ☺ May 26 — 4th **International Workshop on Software Engineering for Automotive Systems** (SEAS'2007). Topics include: all aspects of software engineering for automotive systems, specifically all facets of integration of independently developed software parts to one system with emphasis on the following aspects: software quality, safety / reliability / robustness, component orientation in embedded systems, maintenance of the

integrated embedded software system and compatibility of its components over the lifecycle, etc.

May 27-30 7th **International Conference on Computational Science** (ICCS'2007), Beijing, China. Theme: "Advancing Science and Society through Computation".

☺ May 27-30 4th **International Workshop on Practical Aspects of High-level Parallel Programming** (PAPP'2007). Topics include: high-level parallel language design, implementation and optimisation applications in all fields of high-performance computing (using high-level tools), benchmarks and experiments using such languages and tools; etc.

☺ May 28-31 5th **Object Oriented Technologies conference** (OOT'2007), Plzen (Pilsen), Czech Republic. Topics include: Software Engineering (software components, large-scale software, multi-language programming); Parallel and Distributed Computing (multithreading, distributed applications, ...); Programming Languages and Techniques (object-oriented techniques, programming paradigms, assertion support); Educational Aspects (teaching object-oriented paradigm, educational software); Software Security; Development on Different Platforms; Industrial Applications of Object Oriented Technologies; etc.

☺ May 29-06/01 **DAta Systems In Aerospace** (DASIA'2007), Naples, Italy.

June 06-08 1st IEEE & IFIP **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2007), Shanghai, China. Topics include: Specification and Validation, Component-based Development, Software safety and reliability, Reverse Engineering and Software Maintenance, Embedded and Real-time Software, Model-driven Development, Parallel and Distributed Computing, Program Analysis, Semantics and Design of Programming Languages, Type Theory, etc.

☺ June 09-16 3rd **History of Programming Languages Conference** (HOPL-III), San Diego, CA, USA.

☺ June 11-14 7th **International Conference on Algorithms and Architectures for Parallel Processing** (ICA3PP'2007), Hangzhou, China. Topics include: Distributed & Parallel Middleware, Parallel Programming Paradigms, Tools & Environments for Parallel & Distributed Software Development, etc.

June 13-15 1st IFAC **Workshop on Dependable Control of Discrete Systems** (DCDS'2007), Paris, France. Topics include: specification, design, implementation and operation of dependable controllers for critical discrete systems.

☺ June 14 PLDI2007 - ACM SIGPLAN **Workshop on Programming Languages and Analysis for Security** (PLAS'2007), San Diego, California, USA. Topics include: the use of Programming Language and Program Analysis Techniques to improve the Security of Software Systems; Language-based techniques for security; Program analysis techniques for discovering security vulnerabilities; Specifying and enforcing security policies for information flow and access control; etc.

June 18-21 **Systems and Software Technology Conference** (SSTC'2007), Tampa Bay, Florida, USA.

☺ June 24-28 **Technology of Object-Oriented Languages and Systems** (TOOLS Europe'2007), Zurich, Switzerland. Topics include: all aspects of object technology and neighbouring fields, in particular model-based development, component-based development, and patterns (design, analysis and other applications); more generally, any contribution addressing topics in advanced software technology.

June 25-27 12th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2007), Dundee, Scotland, UK.

June 25-28 2007 **World Congress in Computer Science, Computer Engineering, and Applied Computing** (WORLDCOMP'2007), Las Vegas, USA.

♦ June 25-29 12th **International Conference on Reliable Software Technologies - Ada-Europe'2007**, Geneva, Switzerland. Sponsored by Ada-Europe, in cooperation with ACM SIGAda.

June 25-29 27th **International Conference on Distributed Computing Systems** (ICDCS'2007), Toronto, Canada. Topics include: all aspects of distributed and parallel computing.

☺ June 27 **DSN2007 - Workshop on Architecting Dependable Systems** (WADS'2007), Edinburgh, Scotland, UK. Topics include: everything related to software architectures for dependable systems, such as: Rigorous design: architectural description languages, ...; Verification & validation; Fault tolerance; System evaluation; Enabling technologies; Application areas: safety-critical systems, embedded systems, ...; etc.

☺ July 01-02 12th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2007), Berlin, Germany. Affiliated with CAV'2007. Topics include: Design, specification, code generation and testing with formal methods; Verification and validation of complex, distributed, real-time systems and embedded systems; Verification and validation methods that aim at circumventing shortcomings of existing methods with respect to their industrial applicability; Tools for the design and development of formal descriptions; Case studies and project reports on formal methods related projects with industrial participation (e.g. safety critical systems, mobile systems, object-based distributed systems); Application of formal methods in standardization and industrial forums. Deadline for submissions: April 6, 2007 (papers).

July 02-06 6th **International Conference on Integrated Formal Methods** (IFM'2007), Oxford, UK.

July 03-05 20th **Conference on Software Engineering Education and Training** (CSEET'2007), Dublin, Ireland.

☺ July 05-08 6th **International Symposium on Parallel and Distributed Computing** (ISPDC'2007), Hagenberg, Austria. Topics include: Parallel Computing; Algorithms, Models and Formal Verification; Tools and Environments for Program Analysis; Task and Communication Scheduling and Load Balancing; Real-time Systems; Distributed Software Components; Real-time Distributed Systems; Security; Fault Tolerance; Applications and Case Studies; etc.

☺ July 09-12 2007 **International Conference on Software Engineering Theory and Practice** (SETP'2007), Orlando, FL, USA. Topics include: all areas of Software Engineering and all related areas, such as: Component-based software engineering; Critical and embedded software design; Distributed and parallel systems; Distribution and parallelism; Education (software engineering curriculum design); Embedded and real-time software; Empirical software engineering and metrics; Evolution and maintenance; High assurance software systems; Interoperability; Legal issues and standards; Object-oriented techniques; Program understanding issues; Programming languages; Quality management; Real-time software engineering; Reliability; Reverse engineering and software maintenance; Software architectures and design; Software components and reuse; Software cost estimation techniques; Software design and design patterns; Software engineering methodologies; Software engineering versus systems engineering; Software policy and ethics; Software reuse; Software safety and reliability; Software security; Software testing, evaluation and analysis technologies; Software tools and development environments; Survivable systems; Technology adoption; Verification, validation and quality assurance; etc.

☺ July 22-25 2nd **International Conference on Software and Data Technologies** (ICSOFT'2007), Barcelona, Spain. In conjunction with ENASE'2007. Topics include: Programming Languages (Object-Oriented Programming, Languages and compilers, ...); Software Engineering (Reliable software technologies, Dependable computing, Software components, Software maintenance, Real-time software, Software economics, ...); Distributed and Parallel Systems; etc.

July 23-25 2nd **International Working Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2007), Barcelona, Spain. In conjunction with ICSOFT'2007. Topics include: Model driven engineering; Software components and component-based software engineering; Generative software development; Evolutionary design; New methodologies, practices, architectures, technologies, tools, metrics; etc. Deadline for registration: May 31, 2007.

☺ July 30-08/03 21st **European Conference on Object-Oriented Programming** (ECOOP'2007), Berlin, Germany. Topics include: all areas relevant to object technology. Deadline for submissions: April 18, 2007 (student volunteers).

☺ July 30 11th **Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts**. Topics include: successfully used exercises, examples, and metaphors; approaches and tools for teaching (basic) object-oriented concepts; teaching refactoring and/or design patterns; misconceptions related to object technology; etc. Deadline for position paper submissions: May 13, 2007.

☺ July 30 — 17th **Doctoral Symposium and PhD Students Workshop**. Topics include: Design Patterns; Components, Modularity; Concurrency, Real-time, Embeddedness, Distribution; Domain Specific Languages, Language Workbenches; Adaptability; Generative Programming; Language Design, Language Constructs, Static Analysis; Language Implementation; Model Engineering, Design Languages; Software Evolution, Versioning; Formal methods; Tools, Programming environments; etc. Deadline for submissions: May 1, 2007.

August 12-15 — 26th **Annual** ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2007), Portland, Oregon, USA.

☺ August 21-24 — 5th **International Symposium on Parallel and Distributed Processing and Applications** (ISPA'2007), Niagara Falls, Ontario, Canada. Topics include: Tools and environments for software development; Distributed systems and applications; Reliability, fault-tolerance, and security; High-performance scientific and engineering computing; etc.

☺ August 21-24 — 13th IEEE **International Conference on Embedded and Real-Time Computing Systems and Applications** (RTCSA'2007), Daegu, South Korea. Topics include: Real-Time Systems (Scheduling, Fault-tolerance, Programming languages and run-time systems, Middleware systems, Design and analysis tools, Formal methods, Case studies, Applications, etc.); Embedded Systems (Scheduling, HW/SW co-design, Embedded system design practices, etc.); etc.

August 25-31 — 2nd **International Conference on Software Engineering Advances** (ICSEA'2007), Cap Esterel, French Riviera, France. Topics include: Advances in fundamentals for software development; Advanced mechanisms for software development; Advanced design tools for developing software; Open source software; Software deployment and maintenance; Software economics, adoption, and education; etc.

☺ August 28-31 — 13th **International Conference on Parallel and Distributed Computing** (Euro-Par'2007), Rennes, France. Topics include: the promotion and advancement of all aspects of parallel and distributed computing, such as support tools and environments, distributed systems, parallel and distributed programming, etc. Deadline for submissions: April 2, 2007 (workshops).

☺ August 28 — **Workshop on Highly Parallel Processing on a Chip** (HPPC'2007). Topics include: (parallel) programming paradigms, languages, libraries, and support tools for efficient and manageable exploitation of highly parallel multi-core architectures; etc. Deadline for submissions: June 22, 2007.

☺ Aug 28 – Sept 01 — **International Workshop on Multicore and Hybrid Systems for Numerically Intensive Computations** (MHSN'2007), Niagara Falls, Ontario, Canada. In conjunction with The 5th International Symposium on Parallel and Distributed Processing and Applications (ISPA'2007). Topics include: parallel programming models, compiler technology, runtime systems and libraries, etc. Deadline for submissions: April 10, 2007.

☺ September 03-07 — 9th **International Conference on Parallel Computing Technologies** (PaCT'2007), Pereslavl-Zalessky, Russia. Topics include: New trends and models in Parallel Programming; All aspects of the applications of parallel computer systems; Languages, environment and software tools supporting parallel processing; General architecture concepts, enabling technologies; Teaching parallel processing; etc.

☺ September 04-07 — **International Conference on Parallel Computing** 2007 (ParCo2007), Juelich & Aachen, Germany. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments. Deadline for submissions: April 8, 2007 (mini-symposia), May 15, 2007 (presentations), July 31, 2007 (full papers).

September 04-07 — 18th **International Conference on Concurrency Theory** (CONCUR'2007), Lisbon, Portugal. Topics include: all areas of semantics, logics, and verification techniques for concurrent systems, related verification techniques and tools, related programming models, etc. Deadline for submissions: April 6, 2007 (abstracts), April 9, 2007 (papers).

☺ September 10-14 — 5th IEEE **International Conference on Software Engineering and Formal Methods** (SEFM'2007), London, UK. The aim is to advance the state of the art in formal methods, to scale up their application in software industry and to encourage their integration with practical engineering methods. Topics include: software specification, validation and verification; programming languages and type theory; program analysis; fault-tolerant computing; embedded systems; real-time and hybrid systems theory; software

architectures and their description languages; CASE tools and tool integration; applications of formal methods and industrial case studies; etc. Deadline for submissions: April 14, 2007 (papers), June 25, 2007 (tutorials).

☺ September 15-19 16th **International Conference on Parallel Architectures and Compilation Techniques** (PaCT'2007), Brasov, Romania. Topics include: Compilers and tools for parallel computer systems; Support for correctness in hardware and software (esp. with concurrency); Parallel programming languages, algorithms and applications; Middleware and run time system support for parallel computing; High performance application specific systems; etc. Deadline for submissions: April 1, 2007 (papers), April 2, 2007 (workshops).

☺ September 18-21 26th **International Conference on Computer Safety, Reliability and Security** (Safecomp'2007), Nuremberg, Germany.

September 20-21 1st **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2007), Madrid, Spain. Incorporating ISESE and Metrics. Topics include: Evaluation and comparison of techniques and models; Reports on the benefits derived from using certain technologies; Empirically-based decision making; Industrial experience in process improvement; Quality measurement and assurance; Evidence-based software engineering; Effort and cost estimation, defect rate and reliability prediction; etc Deadline for submissions: April 13, 2007 (short papers), May 15, 2007 (posters).

September 25 **Ada UK Conference 2007**, Manchester, UK. This UK-based Ada conference is being organised to promote awareness of the Ada 2005 language revision, and to highlight the increased relevance of Ada in safety-critical programming.

☺ September 26-28 3rd **Latin-American Symposium on Dependable Computing** (LADC'2007), Morelia, Mexico. Topics include: Dependability Modeling, Prediction and Evaluation; Dependable Applications; Distributed Systems; Parallel, Clustered and Grid Systems; Real-Time and Embedded Systems; Safety-Critical Systems; Security of Computing Systems; Software Engineering of Dependable Systems; Software Reliability; Software Testing, Validation and Verification; Survivability of Computing Systems; etc.

Sept 30 - Oct 01 7th IEEE **International Working Conference on Source Code Analysis and Manipulation** (SCAM'2007), Paris, France. Co-located with ICSM'2007 Deadline for submissions: April 23, 2007 (abstracts), April 30, 2007 (full papers).

October 02-05 23rd IEEE **International Conference on Software Maintenance** (ICSM'2007), Paris, France. Topics include: software and systems maintenance, evolution, and management. Deadline for submissions: April 6, 2007 (research papers), May 4, 2007 (industrial applications, tool demonstrations, doctoral symposium), May 28, 2007 (working sessions).

☺ October 15-17 1st **Workshop on Advances in Programming Languages** (WAPL'2007), Wisla, Poland. Within the framework of the *International Multiconference on Computer Science and Information Technology* (IMCSIT). Topics include: Compiling techniques; Domain-specific languages; Formal semantics and syntax; Generative and generic programming; Languages and tools for trustworthy computing; Language concepts, design and implementation; Metamodeling and modeling languages; Model-driven engineering languages and systems; Practical experiences with programming languages; Program analysis, optimization and verification; Program generation and transformation; Programming tools and environments; Proof theory for programs; Specification languages; Type systems; etc Deadline for submissions: June 25, 2007 (full papers).

☺ October 16 **International Workshop on Real-Time Software** (RTS'2007), Wisla, Poland. Within the framework of the *International Multiconference on Computer Science and Information Technology* (IMCSIT). Topics include: real-time system development, real-time scheduling, safety, reliability, dependability, fault-tolerance, standards and certification, software development tools, model-based development, automatic code generation, real-time systems curricula, etc. Deadline for submissions: June 25, 2007 (draft papers).

☺ October 21-25 22nd **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2007), Montreal, Canada. Topics include: programmer productivity, secure and reliable software, changing hardware platforms, ultra-large scale systems, improve programming languages, refine the practice of software development, etc. Deadline for submissions: July 2, 2007. (Posters, Demonstrations, Doctoral Symposium, Onward! Films, Student Research Competition, Student Volunteers.)

☺ October 30-31 4th **Workshop on Object-oriented Modeling of Embedded Real-Time Systems** (OMER-4), Paderborn, Germany. Topics include: Architectures/frameworks for platform independent, reusable software components; Formal verification at the model and code level; Software components as products; Software quality; Standards and guidelines (e.g., AUTOSAR, IEC 61508, MISRA, UML, ...); Respective trends in automotive software development; etc. Deadline for paper submissions: July 1..

Nov 04-08 2007 **ACM SIGAda Annual International Conference** (**SIGAda'2007**), Washington, DC, USA. Sponsored by ACM SIGAda (ACM approval pending), in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada Resource Association (Cooperation approvals pending). Topics include: Safety, security and high integrity development issues; Language selection for a high reliability system; Use of ASIS for new Ada tool development; Mixed-language development; High reliability software engineering education; High reliability development experience reports; Static and dynamic code analysis; Use of new Ada 2005 features/capabilities; etc. Deadline for submissions: May 16, 2007 (technical articles, extended abstracts, experience reports, workshops, panel sessions, and tutorials).

November 05-09 18th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2007), Trollhaettan, Sweden. Topics include: Reliability, availability and safety of software systems; Quality/reliability-related security issues; Verification and validation; Industrial best practices; Empirical studies of those topics; etc. Deadline for submissions: April 2, 2007 (abstracts), April 16, 2007 (full papers).

November 07-09 6th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2007), Rome, Italy. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Intelligent software systems design, and software evolution techniques; Software optimization and formal methods for software design; Software security tools and techniques, and related Software Engineering models; End-user programming environment; Software Engineering models, and formal techniques for software representation, software testing and validation; etc. Deadline for submissions: May 15, 2007.

☺ December 03-06 28th IEEE **Real-Time Systems Symposium** (RTSS'2007), Tucson, Arizona, USA. Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies. Deadline for submissions: May 18, 2007.

☺ December 03-06 8th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2007), Adelaide, Australia. Topics include: Formal methods and programming languages, Software tools and environments, Component-based and OO Technology, Parallel/distributed algorithms, Task mapping and job scheduling, High-performance scientific computing, etc. Deadline for submissions: June 5, 2007.

December 10 Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

2008

☺ January 10-12 35th **Annual** ACM SIGPLAN-SIGACT **Symposium on Principles of Programming Languages** (POPL'2008), San Francisco, California, USA. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.

May 07-09 7th **European Dependable Computing Conference** (EDCC-7), Kaunas, Lithuania. Topics include: Architectures for dependable systems; Fault tolerant distributed systems; Fault tolerance in real-time systems; Hardware and software testing, verification, and validation; Formal methods for dependability; Safety-critical systems; Software reliability engineering; Software engineering for dependability; etc. Deadline for submissions: September 20, 2007.

June 13th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2008), Madrid, Spain.

ERB : A Ravenscar Benchmarking Framework

Cyrille Comar, Romain Berrendonner

AdaCore SAS, 8 rue de Milan, F-75009 Paris; Tel: +33 1 49 70 67 16 ; email : {comar,berrendo}@adacore.com

Abstract

This paper first describes the ESA Ravenscar Benchmark (ERB), an execution-time and memory consumption benchmark originally developed for the Ada Ravenscar implementations running on the ERC32 processor. Then, after explaining how difficult it is to compare different tool-chains, three different usages of the framework are show-cased. In the first scenario, ERB is used to compare the evolution of the GCC technology in terms of performance over time. In the second scenario, ERB is used to compare GNAT Pro performance on different target platforms. In the third, ERB is used for monitoring the impact of the day-to-day development of the compiler technology in terms of performance of the generated code. All these activities are of interest for compiler vendors.

Keywords: Ravenscar profile, Benchmarking, ERC32, Compilation technology, performance, code quality.

1 Introduction

The question of selecting a particular development tool-chain is always a difficult one. A number of issues must be taken into account, and they are particularly difficult in long-lasting projects with many constraints, such as space systems. Among those issues, one can cite the choice of programming language, target processor, runtime environment and subset of the selected language developers will be allowed to use.

For ESA, the target processor question has been solved, and a decision has been made to promote the ERC32 family, a range of space-hardened SPARC processors including the ERC32 SPARC V7 processor, and the Leon SPARC V8. The choice of programming language and a particular tool-chain was more delicate, though, due to a rich offer on this target: on the Ada side, AdaCore, Aonix and XGC provide commercial solutions; the ORK tool-chain, a spin-off from the GNAT compiler, is also available unsupported from the University of Madrid; on the C side, the RTEMS environment is available.

ESA was therefore interested in understanding the differences between these tool-chains. The point was not to make absolute performance comparison, but rather to identify noticeable differences between them so that project managers could issue useful programming and architectural guidelines. For this to happen, it initiated the ESA Ravenscar Benchmark (ERB) project and contracted AdaCore to write an Ada 95 benchmark framework compatible with the Ravenscar profile. A number of constraints were placed on the development of the project.

In the first place, ERB is an analytic benchmark, like the PIWG [4]or ACES [5]. This means that the test base aims at testing individual features rather than whole applications. The goal is to truly evaluate the cost/benefit ratio for all tested features, rather than giving a global mark. In the context of a given project, the best approach for improving performance is to use the whole application as a benchmark. For compiler developers however, the situation is different since we are not interested in evaluating the direct compiler performance itself but the code it generates for any kind of applications. Analytic tests therefore enable us to make a link between performance variations and specific language constructs.

Secondly, ERB features both execution time measurement and memory measurements, namely runtime system footprint measurement and stack consumption. Space systems are actually very constrained both in terms of time and memory, and any evaluation of generated code "performance" must take this into account.

Thirdly, ERB is the first Ada benchmark targeting applications following the Ravenscar profile. This profile aims at defining a safe subset of the Ada language suited for use on high-integrity real-time systems. With efficiency and safety of use in mind, it sets restrictions on the tasking and synchronization features that one can use. Memory and execution time efficiency is improved by removing high overhead or complex features, and reliability and predictability are increased by removing nondeterministic and non-analyzable features. It is therefore particularly interesting for embedded real-time applications such as in space systems.

The last important design feature of ERB is that it aims to compare programs written in Ada and C. This is a difficult task as the semantics of the languages are very different, but ESA was nonetheless interested in having an insight into the possible differences between the two languages.

The Final Report of the ERB project [6] answers most of these questions. However, the present paper has a different goal. It aims to explain how compiler vendors can use a benchmarking technology for their own purposes. After making a detailed technical presentation of the ERB benchmark, it goes on to present a number of user-cases. .ERB was actually designed with AdaCore's particular needs in mind. In particular, it was written for maximum portability so that it could be used on the very large set of platforms supported by AdaCore. In the present paper, we showcase results obtained from the original ESA study complemented by internal studies made in other contexts.

2 The ERB Framework

Before looking at how to take advantage of the framework, let us portray its main capabilities. The harness is described first. The measurement methods for execution times, the memory consumption measurement and the test base come next. The end of this section contains a description of the potential difficulties encountered when trying to interpret the results of a benchmarking suite.

2.1 The ERB Harness

The ERB harness is the main program that ERB users invoke to run the test suite whole. It is written in Ada 2005, using a very simple design layout, and calls external programs to carry out tasks like compiling the tests, measuring the footprint, or checking the environment.

The ERB harness is able to carry out a number of different measurements. Three timing methods are implemented, namely the dual-loop, external and semi-external methods. Two memory measurement methods are implemented, one using the static information contained in binary files to provide an estimate of the footprint, and the other using a dynamic watermarking method to compute an estimate of the stack consumption.

In addition, the harness currently supports not only a number of existing Ada 95 ERC32 tool-chains but also C on RTEMS. Comparing two languages, in particular when they are as different as Ada and C, is always a difficult task; the semantics of the languages greatly differs, leading to portability issues with the tests. To work around this, the ERB harness includes a C library emulating most of the Ada semantics for tasking and protected object use, written on top of the RTEMS native threading library. However, it was not possible to develop emulation libraries for all the other Ada 95 features, such as exceptions, generic or object orientation. This choice results in a trade off made with the interests of ESA in mind. In any event, the extensibility of ERB makes such additions possible in the future.

2.2 Footprint Measurements

The first relevant factor in memory management of embedded applications is the footprint of the program, which is defined as the amount of memory required to store the code of the program (often in the `text` section in ELF binaries) and its associated data. Data in ELF binaries is mostly contained in the `.data` and `.rodata` sections, which contains both the initialized variables and the initialized constants. The `.bss` section contains usually the uninitialized data as well as the execution stacks of the various tasks. The heap, used for dynamic memory management, is either defined as a part of the `.bss` or as the remainder of the memory space. It also contains the execution stacks on some systems such as ORK.

ERB is able to provide an estimate of this footprint and it is able to discriminate between the user code contribution on the one hand and the contribution coming from kernel code or runtime code on the other. This information is computed directly from the ELF object files. It is an approximation, since some runtime code might be inlined, but it has proved sufficiently accurate to provide exploitable results.

2.3 Stack Consumption Measurement

ERB is also able to provide an estimate of the stack consumption by a program in a portable fashion through a pattern filling technique, better known as “watermarking”. With this system, the memory is filled with a known pattern, before execution of the test. When execution is completed, the memory is read to determine the areas that have been modified. The memory can be filled either externally, by a debugger, or internally, by calling a support library.

This method has several known inaccuracies, but they are compatible with ERB measurement requirements and can be precisely estimated. These inaccuracies are mainly caused by the effects on the stack of the instrumentation routines themselves. Our findings show that they are between 100 and 220 bytes depending on the target. In addition, it is necessary to put a number of constraints on developers to avoid other potential issues like stack overflow and use of pattern in data structure.

This implementation is fully portable on all Ada implementations. Only one parameter of the instrumentation code is likely to change with each configuration: it indicates whether the stack grows up (from low addresses to high addresses) or down. The code is designed as a general stack usage measurement library usable not only for benchmarking purposes but also for evaluating the stack usage of the tasks of any Ada application. This technique has proven so successful that it has been included as a standard GNAT Pro feature and can now be triggered by `gnatbind`’s `–u` switch.

2.4 Timing Measurement

Timing measurement is a critical factor for hard real-time systems as it makes it possible to compute upper bounds for execution times. In some cases the whole execution time of the application is a sufficient measure; sometimes it is more useful to time specific sequences of code within the application. However, not every part of the code can easily be accessed. For instance, it is difficult to evaluate the initialization and elaboration code in an Ada program because of its implicit nature. In order to provide a complete analysis tool, ERB provides three different timing methods: the dual-loop, the semi-external and the external methods.

External Method. The first method that comes to mind for measuring execution time is very simple. For instance, with the ERC32V simulator, once the test program is compiled, the executable is loaded and run, and one can use the simulator timing facility to retrieve the execution time. In order to make sure that the results are statistically meaningful, the harness can repeat the test a number of times. Unfortunately this approach alone is too coarse-grained for our requirements. It makes no distinction, for example, between user code, elaboration code and finalization code, and just provides a global figure including them all.

Semi-external method. The second strategy that comes to mind is slightly more evolved. Instead of just getting the simulated execution time, the program is instrumented by fetching the time, using the target environment timing facility, at the beginning and at the end of the main program. The harness is responsible for repeating the test a number of times, so that it is possible to make statistically sure that the results are meaningful. This method complements the previous one very well; the instrumentation surrounds the user code and provides timing information naturally excluding the elaboration and finalization code. On the other hand, this kind of measurement is very sensitive to specific inaccuracies that can affect the timing facility of a computer, like the jitter or cache effects. It is also very sensitive to compiler optimizations that can just move out of the instrumented part, some of the code one wants to measure.

Dual-Loop. The most sophisticated method was originally developed as part of the ACES [5] project and demonstrated to be an appropriate solution to avoid the errors we just described. It is based on a careful analysis of the possible systematic inaccuracies related to such timing systems: the jitter, which is basically the random variation of clock precision; and the quantization error, which is mostly caused by the analog-digital conversion of the physical phenomenon used to generate the clock signal.

The first step of the dual loop strategy is to evaluate the combined order of magnitude of these errors, as measured by a program executed on the target. The basic idea of this method is to count the number of elementary ticks that can fit into a known duration of, for example, one second:

```
Time := Clock;
Nb_Elementary_ticks := 0;
while  Time = Clock loop
  Nb_Elementary_ticks := Nb_Elementary_ticks + 1;;
end;
```

An estimate of the combined effect of jitter and quantization is then provided by multiplying the standard deviation of the number of elementary ticks into one second by the estimated duration of an elementary tick.

Once the errors are estimated, the dual loop strategy repeats each test case a number of times, so that the overall execution time is significantly longer than the possible systematic errors estimates. Obviously, this must be gauged against execution time constraints and meaningfulness: a low value such as 10 microseconds is likely to provide unstable measurements, while a large figure like 10.000 would result in impractically long execution times. As the duration of the test is not known in advance, the harness increases the number of iterations until it meets this criterion.

This very short presentation should not hide the fact that the ACES [5] has demonstrated that the dual loop method provides very good results, thanks to a number of advanced heuristics to handle optimizations, processor cache issues and memory paging. But despite those efforts, the dual loop method still has some limits. In particular, it is not well suited for user code that cannot be freely repeated without changing the behaviour of the program, for instance because of a side effect, or for measuring elaboration code. All three are required to make a thorough analysis of some tests, in particular when side effects are present or when information on elaboration is needed.

2.5 The Test Suite

The ERB test suite is divided into 12 different chapters. Each chapter is specialized in a particular kind of testing: arithmetic, tests for estimating the impact of implicit Ada checks, arrays, access types, exception handling, high-level algorithms (DES, FFT, Dhrystone …), iterations and loop tests, miscellaneous tests (in particular measurement of interrupt handling times, estimate of minimal and maximal footprints), object-oriented programming, subprograms tests, tasking and protected objects.

Not all the tests could be ported to C on RTEMS. In particular, we could not afford to develop our Ada semantics emulation library to support other advanced Ada features such as object-oriented programming, generics and exceptions. We preferred to focus on the most interesting aspects of Ada semantics for space applications, which is undoubtedly embodied by the Ravenscar profile.

2.6 Interpreting the Results

Producing results is only a first step. Interpreting them is almost as difficult.

The first difficulty is the amount of data that a benchmarking suite like ERB can produce. In the framework of the ESA study for instance, 15,000 different figures were produced and had to be analyzed. This amount of data comes from the many different parameters that can change at the same time: the tool-chain and the test cases of course, but also the optimization level, the kind of test (stack, footprint, dual-loop, external, semi-external).

In order to analyze so much data, we decided to compare things *ceteris paribus*, meaning that only one parameter could change at the same time. For instance, when comparing GNAT Pro and ORK, we would stick to optimization level O2 on both targets. This is not always easy; for optimization again, there is no warranty that O2 includes the same optimizations on two different tool-chains.

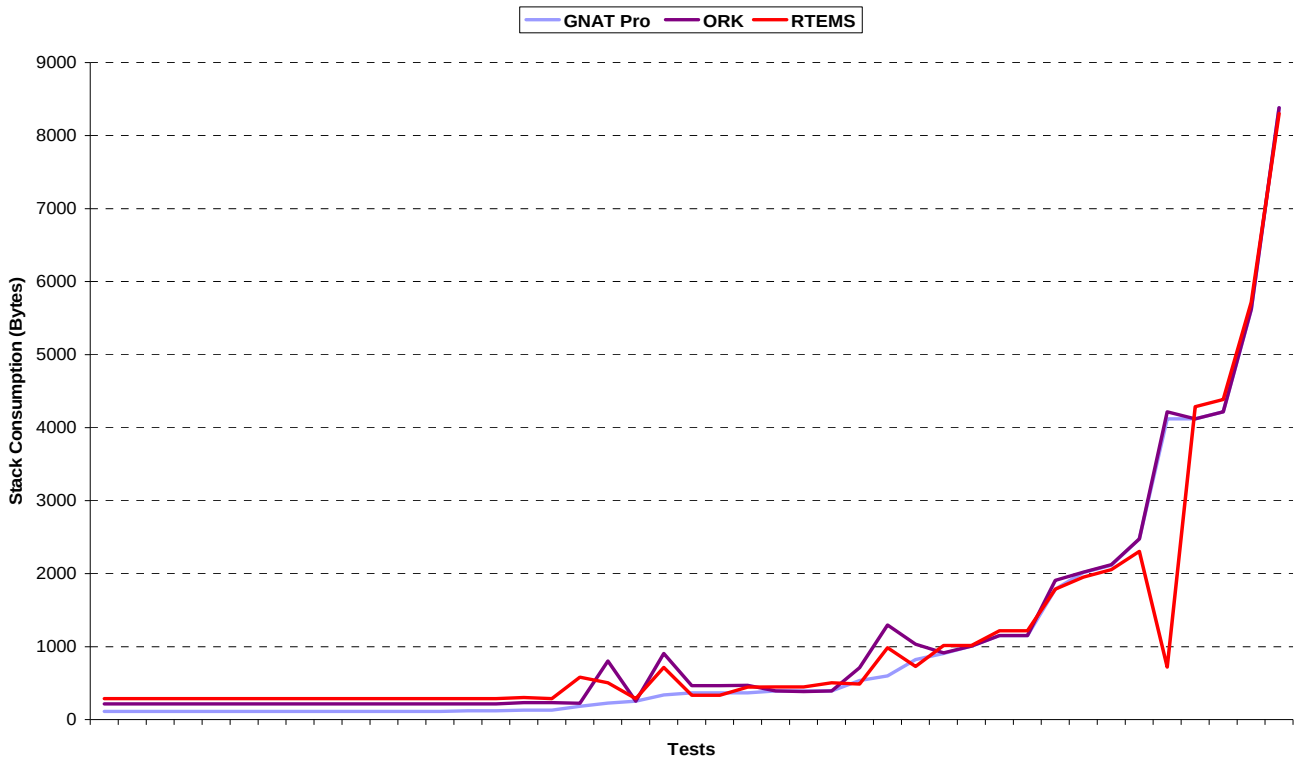

Figure 1: Stack consumption on GNAT Pro, ORK and RTEMS

Once one knows what is to be compared, it is best, in our experience, to use graphs to find interesting data points worth analysis. But making such graphs is not easy either. The execution times range from a few milliseconds up to several minutes and it is not meaningful to display absolute values. We prefer to take a baseline tool-chain A, order the test cases by increasing the value for A, and then display for another tool-chain B the ratio B/A between the result of A and the result of the reference tool-chain. Anything below 1 means that tool-chain B is faster; anything above that it is slower. For memory tests, presenting the ratio is not as useful as the values fit in a smaller interval.

An example of such graphs can be found in Figure 5.

3 The ERB Framework at Work

ERB has shown to be useful in three different contexts. First, it has allowed comparing over long periods of time the evolution of a given technology. Second, it has been used to compare the same tool-chain on different target platforms. And finally it will be used for the daily work on a given technology, making sure that no performance regressions appear.

3.1 Comparing various instances of GCC

The first use for a benchmark that proved useful is comparing different evolutions of the same technology over a long period of time.

The ERB study provided an ideal context as four different tool-chains for the ERC32 target were based on different stages of the GCC technology: XGC actually is based on version 2.7, ORK is based on version 2.8, the version of gcc provided for the version of RTEMS we benchmarked was 2.95 and the most up-to-date GCC-based tool-chain was GNAT Pro 5.03a1, which uses GCC 3.2.3. This provides a 10 years span of technical evolution on the GCC technology.

We will focus in the following section on our finding regarding GNAT Pro, RTEMS and ORK. However, one have to keep in mind that those versions belong to different branches of the GCC technology. ORK for instance is a derivative of GNAT 3.13p and includes an Ada Compiler and an Ada run-time. The C compiler for RTEMS is a pure C compiler derived from regular FSF tree. We demonstrate how measurement methods provide interesting results on the evolution of the GCC technology of the time, but in from different points of view. This should not be perceived as a limitation: depending on one's own focus, it is possible to get information on the aspects that we will leave aside. The only limiting factor here is time, as understanding such differences requires an in-depth technical understanding of the tests and tool-chains and careful investigation.

Stack Consumption

The first area of interest for the ERB study was the stack consumption. Our first hypothesis here was that stack consumption was unlikely to change significantly between the variants of GCC student in the project.

Figure 1 presents the results of the stack measurements for the GNAT Pro, ORK and RTEMS tool-chains. In the first place, one can observe that, as expected, the three tool-

chains provide very similar results in most cases. These results are not surprising, not only because RTEMS, ORK and GNAT Pro are based on different stages of the GCC technology: as the calling convention for subprograms is mostly defined by the processor ABI it leaves little initiative to implementors. The compilers still have a number of possibilities to improve the handling of stack:

- They can inline the function call. This means that the code for this function is expanded within the code of the calling frame, without any stack frame of its own.
- They can use frameless functions instead of regular functions, putting in registers which usually go on the stack frame, like the return address of the function and the parameters.

A limited number of tests display evidence of significantly different behaviour in this respect.

The first category is representative of what can be obtained with the tests which do not need much stack. For these the measurement does not correspond to the real use of stack but more to the lower limit of what the instrumentation can measure. Those tests are located on the left hand side of Figure 1. The test featuring five successive additions of integers consumes, like most of the arithmetic tests, virtually no stack. It is interesting to study, because it shows the limits of what can be measured with the watermarking technology developed in ERB.

The accuracy of this method is of course limited by the space taken on the stack by the instrumentation routine itself. On RTEMS and ORK, it turns out that the measurement threshold is 208 and 288 bytes respectively, while it is slightly lower on GNAT Pro with 112 bytes. A detailed analysis of this test showed that the difference was due to a different layout of the stack frame of the instrumentation routine filling the stack. On RTEMS, the pattern area is located 100 bytes above the beginning of the frame; but it is located 116 bytes after the frame pointer on ORK, and only 12 bytes in GNAT Pro. The only thing we can say about all the tests whose measurements are equal to 112 bytes on GNAT Pro (208 on ORK and 288 on RTEMS) is that they use less stack than this amount.

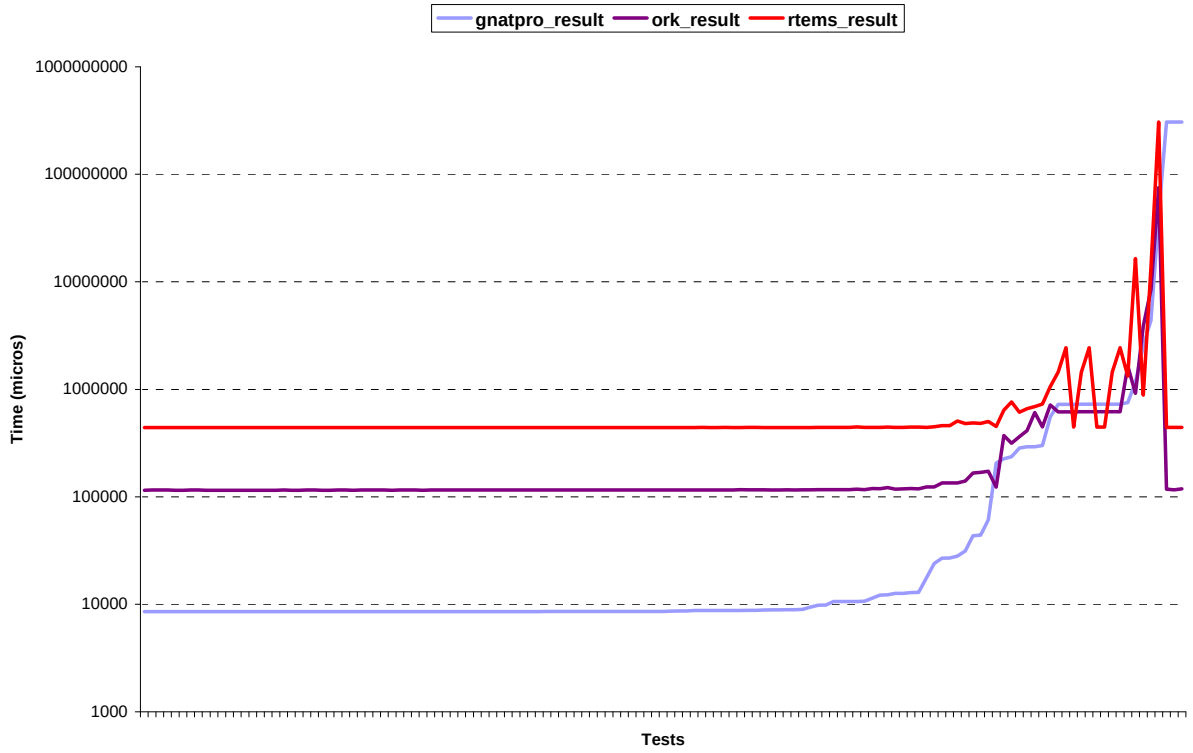

Figure 2: External Timing Measurement with GNAT Pro, ORK and RTEMS

Some other tests, on the other hand, are well above this threshold. This is the case, for example, of the tests dealing with various read and write accesses to protected objects from a number of tasks. We will describe later how implementations specifically targeting the Ravenscar profile can take advantage of this time-wise. In terms of stack consumption, though, the RTEMS task creation routine is able to support dynamic task creation. With GNAT Pro, on the other hand, tasks are statically allocated so task creation is a simpler process which indirectly uses less stack. Interestingly, the results of ORK and RTEMS in this area are strictly parallel.

The conclusion of this quick study of the stack consumption evolution of the GCC technology over time is that it is a stable area and is subject to little change over time. In the particular case of space applications, where memory is often very constrained, this is an interesting property.

Elaboration Times

Another parameter of interest is the amount of time spent on elaboration, which can be obtained by comparing the results of the external and semi-external methods. Figure 2 displays the results of testing with the external method on the RTEMS, ORK and GNAT Pro tool-chains. At first glance, the tests can be divided into two categories with all tool-chains:

The first category includes test cases where the duration of the test section is short with respect to the initialization and elaboration code. This is the case, for instance, for most of the arithmetic and iteration tests. Such tests are very simple, and the execution time of the main program is considerably shorter than the time required by elaboration.

The second category of tests includes test cases where the initialization code in the test is significant compared to the elaboration time. The test doing a tree sort of a 5000-node tree is an example of this: before the actual test section is started, the program initializes the tree by picking up 5000 random numbers, which takes a significant amount of time. When this section is suppressed, the external measurement goes from 226 ms down to 39.3 ms with GNAT Pro.

It is interesting to understand where these differences come from. As all the tool-chains are open-source, it is possible to look at their code and understand how elaboration code works on GNAT Pro and ORK, and initialization code works on RTEMS.

It turns out that RTEMS has the longest initialization time because it includes a wide range of features (such as a file system driver, a message queue and semaphore library) which need to be initialized. Such features have no counterparts in ORK and GNAT Pro. Of course, some of these "monitors" can safely be removed, depending on user needs and skills. However this requires a good understanding of the tool-chain.

The difference between GNAT Pro and ORK can also be easily explained: GNAT Pro features a dedicated Ravenscar runtime with minimal elaboration needs. This runtime is defined as a high-integrity run-time and includes only a carefully crafted subset of the Ada standard, sufficient to implement the Ravenscar profile and keep the certification process easy. The ORK run-time, on the other hand, is a full Ada run-time. As a consequence, it does not take advantage of the restrictions inherent to the Ravenscar profile. For instance, tasks are known statically at compile time under this profile, but ORK uses the general-purpose task creation routines anyway.

Tasking

The semi-external method provides a good tool for validating the theory that a runtime specifically designed to target the Ravenscar profile should improve performance. This method is actually used by the task-switching tests, which are presented in Figure 3. The first bar gives the results for a test featuring 6 tasks with the same priority performing 2000 yields. The second bar features the same test with increasing priorities, and the last one features the same test with decreasing priorities.

Figure 3 clearly shows that GNAT Pro is faster than ORK which in turn is faster than RTEMS. This difference can be attributed to several factors. One of particular interest is the use of Ravenscar specific optimizations, such as accessing protected objects by raising the priority of the active task rather than using explicit locking thanks to the FIFO_Within_Priorities policy or reorganizing the task queues so that the most common operations are done in constant time.

Without going into the details of the RTEMS run-time implementation, one can guess that RTEMS pays the price for implementing non-restricted threading features. In particular, dynamic task creation and task termination are fully supported. The richer semantics they provide has a cost in terms of code which appears in our findings.

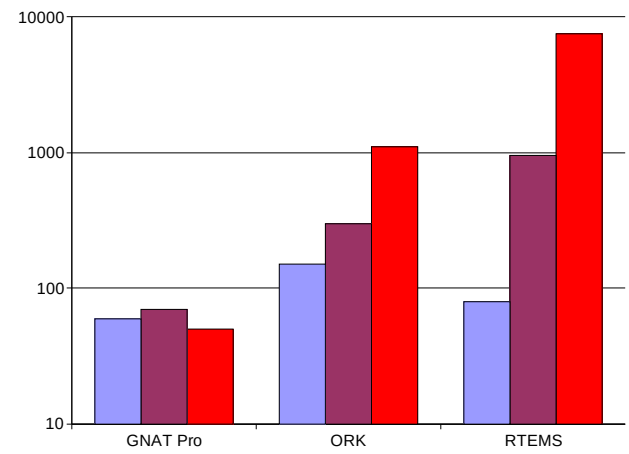

Figure 3: Task-Switching times with GNAT Pro, ORK and RTEMS

So far, the differences that we have found were mostly accountable to the runtime. It is a reasonable assumption to expect code generation improvements in more recent versions of the GCC technology. ERB, and in particular the dual-loop measurements, provide a way to validate this hypothesis.

Compiler And Code Generation

Figure 4 displays a comparison between ORK and GNAT Pro on all the tests where the dual-loop method makes sense, using GNAT Pro as the baseline. The results can be divided into three categories. For the first 2 tests, ORK is much faster than GNAT Pro. In the next 133, the two tool-chains provide reasonably close results: the difference is less than 50%. In the last 120 tests, GNAT Pro is from 50% to 522% faster. This general trend mostly indicates code generation enhancements with more recent versions of GCC.

One can notice in particular that tests where GNAT Pro is at least twice as fast as ORK include most tests in the arithmetic, data structure, high level algorithm and tasking chapters. The explanations that we have provided for better

tasking with GNAT Pro can be applied to these tests as well.

The results of the arithmetic tests show that 32-bits integer arithmetic is faster with GNAT Pro than ORK thanks to back-end improvement between GCC 2.8.1 and 3.4. The mod, rem, Rotate_Left and division operations are, for instance, all faster.

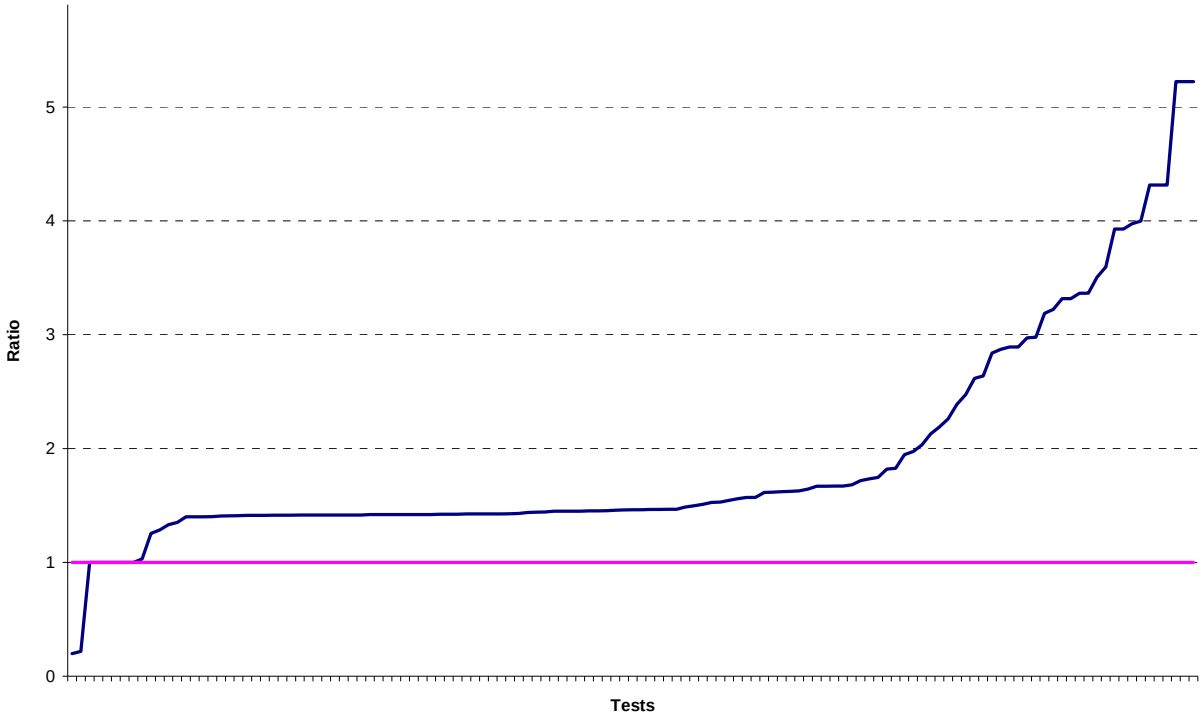

Figure 4: Dual-loop measurements for ORK and GNAT Pro (GNAT Pro base 1)

It also proves that the general layout of data structures is more efficient with a recent GCC back-end. The test featuring the assignment of a packed record of three integers is more efficient with GNAT Pro than ORK because it uses half-word store operations while ORK uses bytes.

Alignment issues are, however, particularly delicate and the test featuring a packed record of three floats is among the two tests where ORK generates better code. It appears that in this particular case, the GNAT Pro compiler uses an alignment of 1 byte and therefore needs to copy 12 elements, while the ORK compiler uses an alignment of 4 bytes and only needs three copies. Interestingly, when compensating for the inefficient default alignment selected by the compiler using a "for <type>'Alignment use 4;" clause to the code, GNAT Pro generates better code than ORK again.

To summarize, code generation appears to be considerably better with later generation backend. However, we have been able to identify and fix a number of cases where code generation evolution went in the wrong direction. Such regressions can be worrisome and we will see in section 3.3 how ERB can be used to address this issue.

3.2 Comparing GNAT Pro on different platforms

Another interesting usage of ERB can be found in the work AdaCore was contracted to do by a hardware manufacturer. The task involved performing a study whose goal was to help select the best hardware configuration for a final customer developing and maintaining a long-lived application with GNAT Pro. Here it was therefore a matter of comparing the performances of the same technology on different platforms.

The "best hardware configuration" is defined in this context in terms of quality of the GNAT Pro port on this particular configuration, minimization of the compilation time and performance of the resulting application. Of particular interest to the manufacturer were practical results showing if a server with many relatively slow CPUs could compete with a server with few very fast CPUs.

Concerning compilation time, ERB does not provide the necessary harness to conduct performance analysis of the tool-chain usage itself. Even if it had such a capability, it is probable that the results would not have been relevant since it would have computed the performance of the toolset when compiling many small or medium tests whereas we were interested here in the performance of the toolset when building few very big applications.

The situation was different for the part of the study interested in the performance of the generated code. The reasoning of the previous paragraph would have pushed us to compare the performance of the application itself on the different target platforms if it was possible. This would have required a full port and qualification of the end-user

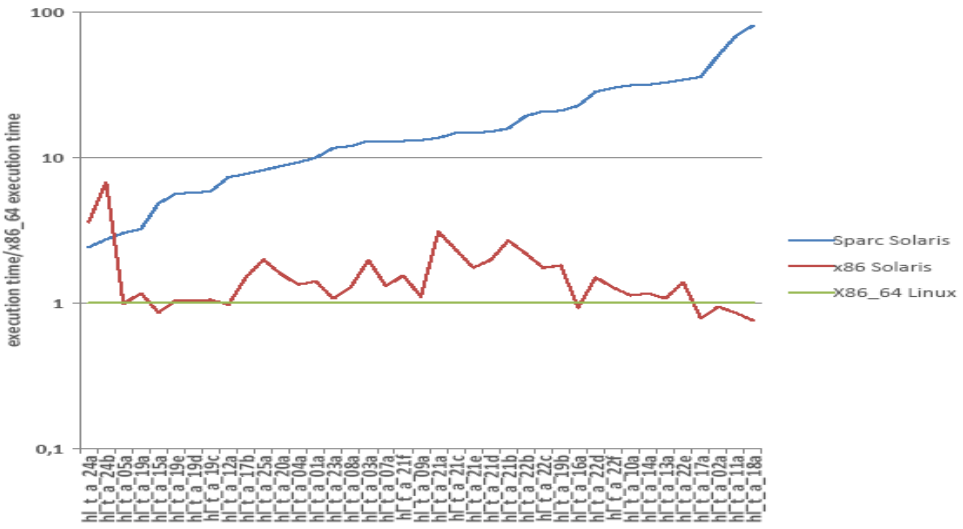

Figure 5: High-Level Algorithms Execution Times

applications on each target. Even though the porting effort is very much simplified by the usage of a language such as Ada, designed to help code portability, and by the use of the same version of the GNAT Pro toolset on all platforms, such an option was not on the table for several reasons: given the size and complexity of the application, the porting effort remained prohibitive in the context of this study. Furthermore, code porting is not the only element to be taken into account: availability and access to the environment of the data and databases is also critical and often difficult to simulate outside of the end-user setup. Even had we overcome all the above issues, code needs to have specific characteristics and needs to be manually instrumented in order to benefit from accurate measurements such as those given by the dual loop method, as described in the previous chapter.

ERB proved useful in quickly giving a general idea of the performance trend on the various platforms. Having many tests, already properly instrumented and known to be non-deficient for performance comparison, is a big asset. Having many tools available to make the analysis of the produced timing results easier was an even more important factor of suitability for using ERB in this context.

For example, Figure 5 shows the comparative results on a logarithmic scale of the chapter implementing high-level algorithms on the three platforms being tested. The green platform is used as a reference and we see immediately that the blue platform is significantly slower (more than 10 times slower on 2/3 of the results). The first two tests are worth additional analysis since they are the only tests that show at the same time the smallest difference between the blue and green platforms while showing the highest difference between the green and red. More detailed analysis will show that the specific test cases have a high number of active tasks and thus automatically take advantage of the higher number of processors on the blue platform without compensating completely for the slowness of the CPUs.

The comparison of the green and red platform is also of interest since they share the same hardware and only vary by their Operating System. Comparable performance could be naively expected for all tests. So many differences for almost all tests requires some explanation especially since it would be difficult to blame OS itself when these tests do not use any OS services except implicitly through the Ada runtime for actions such as for file IO and thread management. A more detailed analysis of the most important performance differences shows that there are significant differences between GNAT Pro ports on these two systems. Many small differences favouring the green platform can be attributed to a difference of ABI: the green platform using a recent x86 64bit ABI while the red one uses the standard x86 32 bit ABI. Most of the other differences can be attributed to a different exception handling mechanism, as is made obvious by the specific results on the exception chapter shown in Figure 6: the mechanism used by the red platform (based on setjmp/longjmp) is extremely efficient for exception propagation but incurs a significant distributed overhead for each handled sequence of statement. This explains why the red platform is much faster for the tests concentrating on exception propagation and slower in most other cases.

The other rare cases where the red platform was faster can be traced to specific deficiencies of the 64bit code generator which shows the powerfulness of such a test suite for code generation tuning.

3.3 Regression Testing

This section describes the last user case for a benchmarking technology: using it for non regression on a daily basis. The goal is to make sure that the bug fixes and new features regularly introduced in the compiler do not cause

performance regressions like the one discussed above. Unlike the previous user cases, this is still work in progress at AdaCore and the method presented in this section is likely to be adjusted according to future findings.

Non-regression testing is well-known good practice for software development: a base is created and progressively

Those variations are due to machine load changes , network activity or any other system event that may perturb the measurement beyond what the dual loop method can handle seamlessly. In order to avoid this, one could either run the benchmark only on simulated platforms like the ERC32 with TSIM [7], or use another metric to identify significant

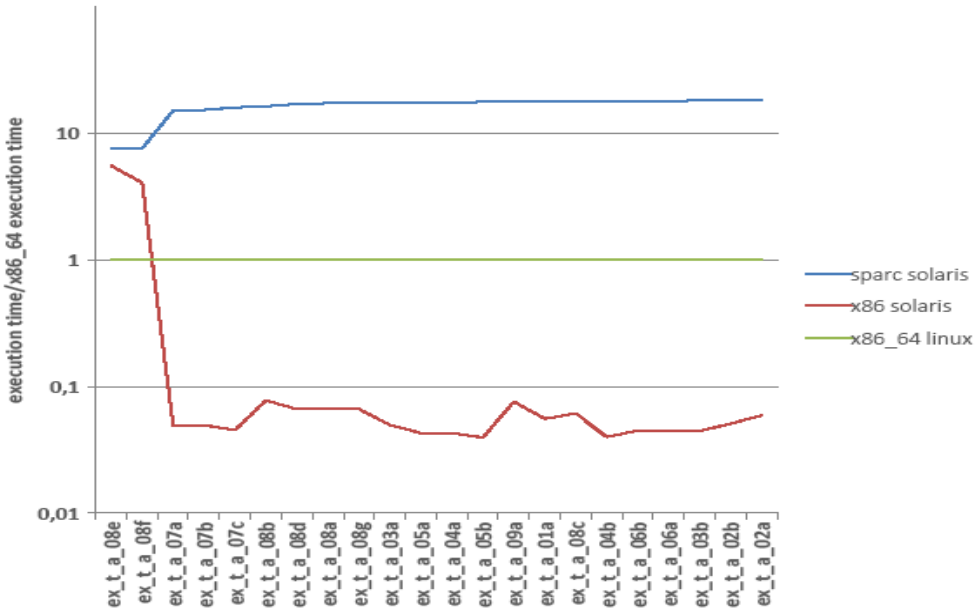

Figure 6: Comparison of exception tests

updated with tests exercising all previously fixed problems. This whole test base is executed on a regular basis, every night at AdaCore, to make sure that recent changes do not cause old problems to reappear. However, the current regression test suite leaves performance apart. Each test provides an output, which is compared to an expected output to make sure there is no regression. The execution time information is neither measured nor stored, nor is the memory consumption. It is interesting to have other means of checking the impact of changes on these parameters. We are therefore discussing performance regression testing.

AdaCore is planning to use ERB for this purpose and leverage on the porting effort to non ERC32 targets. The basic idea is to run the ERB benchmark every night with the freshly built version of the compiler and compare the results with respect to a baseline to identify tests that have changed significantly. In particular, one could compare the results of a given day with the results of the day before and issue a report with all the performance regressions imputable to the changes introduced in those 24 hours.

This raises a number of issues, though. This would be a valid approach for stack measurements and footprint measurements, as they are completely deterministic, but would be much harder for execution time measurements. On native platforms, the latter are subject to small variations that make strict equality comparison pointless.

changes to test results.

On the one hand, running the ERB test suite only on a simulated platform is not satisfactory, because AdaCore supports only a limited number of such platforms, all of them using restricted runtimes such as the Zero Footprint run-time or the Ravenscar run-time. It would mean that all the ports on which such run-times do not figure could not be properly tested. This approach is too restrictive for complete performance testing on a tool-chain supporting many platforms and several run-times.

On the other hand, finding alternative metrics addressing those issues is delicate. For instance, one could consider that a test whose timing result is in a 10% range around the result of the day before should not be reported as a regression. That seems a simple way of dealing with small unrelated variations but this is not reliable enough: a situation where successive changes would cause several 5% drifts would not be caught even though it creates a significant difference after a while.

To avoid this, one could reconsider the baseline used for the comparison; instead of using the compiler built on the previous day, one could use the previous stable release as a baseline. After all, the goal of such benchmarking is to make sure that the next release will be as good, or better, than the previous ones. It is not to detect random variations between development versions. All results would therefore

be expressed as a fraction of the results obtained with the reference release of GNAT Pro.

Now that the baseline is determined, we need to determine the threshold which will be used to identify regressing tests. A number of issues have been identified and we propose here a method to solve them.

To do this, we propose to dynamically adapt the threshold used by a test to the history of results. At the beginning, the threshold would be determined by past results and then be adjusted according to daily results.

Let T1… TN be the tests inside the ERB test-suite and G1, G2 … GM the set of GNAT Pro compilers which will form the initial comparison base. G1 … GM would be, for instance, all daily development versions built in the previous year. We first make the assumption that the evolution of the results obtained with this subset of compilers is representative of the future evolution of results. This assumption is probably valid in the short term. The method we propose also makes it valid in the long term as new data is fed to the model.

During an initialization phase, tests T1 … TN will be run against all G1… GM compilers, producing a huge two-dimensional matrix of results that can be noted R(1,1) … R(N,M). For each test TJ we will compute over R(J,1) … R(J,M) the 10th percentile of the results and the 90^{th} percentile of the results. Any result above and beyond these limits will be considered as significantly different and be reported to developers.

Such a method can be used dynamically. Once we have the results for the past, it is possible to dynamically adapt the database, adding the results vector R(1,M+1)…R(N, M+1) every day and computing the thresholds again . However, we want regressions to remain apparent. If they are fed back to the base, the regressions will introduce a bias and may, ultimately, no longer be reported. In order to avoid this, reported regressions are not fed back to the result base unless it is traced as a necessary loss of performance related to a new feature, rather than a *bona-fide* regression.

The dimensions of the problem need to be checked for feasibility. If AdaCore implements this for a number P of platforms, the amount of data to be stored is the following is D = T x M x N x S, where S is the size in bytes of the output of a test. If T is 3 (for instance Linux, Windows and ERC32), M is 365, N is 266 and S is 20, the "knowledge base" would be in the order of magnitude of 5.8 Mbytes and the daily increase of data would be about 15 Kbytes. On a recent GNU/Linux machine, running ERB takes roughly one hour and a half. This means that the reference base needs around 22 days to be fully built, and therefore must be run on a dedicated machine.

A similar situation occurs with tests that have volatile results. This can happen, for instance, for tests that depend on the operating system load, either because they involve tasking or because they last a long time. The system we propose should be able to handle such tests quite seamlessly because regressions would be reported only if the results fall in a statistically exceptional area. This is the main advantage of this method: regression is not detected through a binary decision mechanism, but rather by an adaptable system that detects exceptional results by comparing them with history.

Conclusion

This paper first discussed the various challenges of creating a reliable benchmarking framework for compilers in a Ravenscar context. We then described very different situations where this benchmarking suite has been useful. Having access to such a tool is a good way to answer specific performance oriented questions. It is much more difficult, on the other hand, to answer general questions such as which of two completely different technologies is the best one.

In any event, any careful analysis requires a very good understanding of the underlying technology, which is why such a tool is of particular interest for compiler vendors. ERB will therefore be made available to any interested party on AdaCore libre site [2] with full sources and documentation, under the terms of the General Public Licence [1] so that everyone can find answers to their own questions.

References

[1] Free Software Foundation (1991), *The General Public Licence, version 2*
Available at http://www.gnu.org/licenses/gpl.txt

[2] The AdaCore libre site
Available at http://libre.adacore.com/

[3] R. Berrendonner and J. Guitton (2005) *The ESA Ravenscar Benchmark*, Springer-Verlag, LNCS 3555, *Proceedings of Reliable Software Technology-Ada Europe 2005*.

[4] Performance Issues Working Group (PIWG), *The PIWG benchmark*
http://unicoi.kennesaw.edu/ase/support/cardcatx/piwg.htm

[5] High Order Language Control Facility (2003), *Ada Compiler Evaluation system Reader's Guide for Version 2.1*
Available at http://www.adaic.org/compilers/aces/aces-intro.html

[6] AdaCore (2007), *ESA Ravenscar Benchmark Final Report*, ESA Contract No. 16962/02/NL/LvH/bj

[7] Jiri Gaisler (2003), *TSIM Simulator User's Manual*, Gaisler Research
Available at http://www.gaisler.com

[8] J. A. de la Puente, J. Ruiz and J. Zamorano (2000), *An Open Ravenscar Real-Time Kernel for GNAT*, Springer-Verlag, LNCS 18455, *Proceedings of Reliable Software Technology-Ada Europe 2000*.

[9] Ravenscar profile for high-integrity systems
Available at http://www.ada-auth.org

Ada-Europe 2006 Sponsors

AdaCore
Contact: Zépur Blot
8 Rue de Milan, F-75009 Paris, France
Tel: +33-1-49-70-67-16 Fax: +33-1-49-70-05-52
Email: sales@adacore.com URL: www.adacore.com

Aonix
Contact: Jacques Brygier
66/68, Avenue Pierre Brossolette, 92247 Malakoff, France
Tel: +33-1-41-48-10-10 Fax: +33-1-41-48-10-20
Email : info@aonix.fr URL : www.aonix.com

Green Hills Software Ltd
Contact: Christopher Smith
Dolphin House, St Peter Street, Winchester, Hampshire, SO23 8BW, UK
Tel: +44-1962-829820 Fax: +44-1962-890300
Email : URL : www.ghs.com

I-Logix
Contact: Martin Stacey
1 Cornbrash Park, Bumpers Way, Chippenham, Wiltshire, SN14 6RA, UK
Tel: +44-1249-467-600 Fax: +44-1249-467-610
Email : info_euro@ilogix.com URL : www.ilogix.com

Praxis High Integrity Systems Ltd
Contact: Rod Chapman
20 Manvers Street, Bath, BA1 1PX, UK
Tel: +44-1225-466-991 Fax: +44-1225-469-006
Email : sparkinfo@praxis-his.com URL : www.sparkada.com

Ellidiss Software
TNI Europe Limited
Contact: Pam Flood
Triad House, Mountbatten Court, Worrall Street, Congleton, CW12 1DT, UK
Tel: +44-1260-29-14-49 Fax: +44-1260-29-14-49
Email: info@tni-europe.com URL: www.ellidiss.com