ADA USER JOURNAL

Volume 32
Number 1
March 2011

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication else-where.

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 this first editorial of 2011, I would like to start by briefly presenting to our readers some of the highlights this year will bring to the Ada community. First, I am happy to announce that the Ada User Journal will publish the Rationale for Ada 2012, under preparation by John Barnes. The Rationale will be a valuable asset for the Ada community and will be published in several parts, the first being planned for next September. Also, a brief note to the return of the International Real-Time Ada Workshop, which will take place next September, organized by our friends of the University of Cantabria, Spain. Note that the deadline is around the corner.

Last, but definitely not least, a note for all Ada practitioners to converge to Edinburgh, this June, for the Ada-Europe 2011 conference, which will join efforts with the Ada Conference UK, in an event called "The Ada Connection". I must draw your attention to the rich contents of this event (briefly presented in the forthcoming events section of the Journal), which, apart from the tutorials, exhibition and scientific and technical presentations will also provide a very rich networking environment. A particular highlight this year is the panel discussions which will be held, on the challenging topics of "DO178C and Object-Orientation for Critical Systems" and "Programming Languages Meet Multi-Core".

And it is in line with the latter, that the technical contents of this issue discuss parallel programming approaches. In the first paper, Brad Moore, of General Dynamics, Canada, discusses the results of applying different approaches to manage parallel work items using Paraffin, a set of Ada 2005 generics supporting the incorporation of parallel computation in Ada programs. Afterwards, the issue provides a paper, contributed by Tucker Taft, of SofCheck, USA, with the edited and consolidated trail of the ParaSail blog, detailing the (evolving) design of a new programming language, targeted to the development of parallel and high-integrity software. It is interesting to verify that although the time passing notion of a blog is difficult to map into the "static" and self-contained nature of a Journal paper, the result is a pleasure to read.

Continuing with the contributions, the Ada Gems section provides the series of gems on Reference Counting, from Emmanuel Briot, of AdaCore. And, as usual, you will find the valuable information of the News and Calendar sections, contributed by Marco Panunzio and Dirk Craeynest, their respective editors.

Luís Miguel Pinho
Porto
March 2011
Email: lmp@isep.ipp.pt

Quarterly News Digest

Marco Panunzio

University of Padua. Email: panunzio@math.unipd.it

Contents

Ada-related Organizations

New AdaIC website

From: Thomas Løcke <tl@ada-dk.org>
Date: Fri, 17 Dec 2010
Subject: New AdaIC website.
URL: http://ada-dk.org/?page=news&news_id=227

And it is gorgeous. It is probably the best looking Ada website I've seen.

Pleasant, clean and easy to navigate.

The website is a regular treasure trove of Ada information, and I especially enjoyed these pages:

- Ada Advantages

- Ada Learning Materials

- Introduction to Ada 2005

- Free Tools And Libraries

All this might also have been available on their old website, but never was it so visible and readily available.

[…]

And I spotted this on the reddit Ada subgroup, which is another "do not miss" Ada resource.

[visit http://www.adaic.com/ and http://www.reddit.com/r/ada/ —mp]

Ada 2012 Language Reference Manual (draft)

From: Thomas Løcke <tl@ada-dk.org>
Date: Tue, 08 Feb 2011
Subject: The Ada 2012 Reference Manual (Draft)
URL: http://ada-dk.org/?page=news&news_id=262

The ARA has made an early Ada 2012 Reference Manual Draft available to the Ada community. There's also a link to the proposed Amendment 2 to the Ada Standard.

All very nice. Of special interest to me is the new string encoding package, which looks to shape up very nicely. Also the Containers.Multiway_Trees is interesting.

Here's a comparison chart showing the differences between the various Ada standards.

I'm still a bit worried about Conditional expressions / Case expressions and In-out parameters for functions, but given the excellent track-record of Ada, I'm sure even those two will come out both looking and working quite well.

[the draft can be downloaded at

http://www.ada-auth.org/standards/ada12.html

the comparison chart can be downloaded at

http://www.adaic.org/advantages/ada-comp-chart/ —mp]

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. —mp]

Ada-Europe 2011 - Final Call for Industrial Presentations

From: Dirk Craeynest <dirk@cs.kuleuven.ac.be>
Date: Fri, 31 Dec 2010 08:10:45 +0000 UT
Subject: FINAL CfIP, Conf. Reliable Software Technologies, Ada-Europe 2011
Newsgroups: comp.lang.ada,fr.comp.lang.ada,comp.lang.misc

FINAL Call for Industrial Presentations

The Ada Connection

16th International Conference on

Reliable Software Technologies - Ada-Europe 2011

+

Ada Conference UK 2011

20 - 24 June 2011, Edinburgh, UK

http://www.ada-europe.org/conference2011

*** DEADLINE Saturday 8 JANUARY 2010 ***

The Ada Connection combines the 16th International Conference on Reliable Software Technologies - Ada-Europe 2011 - with Ada Conference UK 2011. It will take place in Edinburgh, Scotland's capital city and the UK's most popular conference destination.

In addition to the usual Call for Papers, the conference also seeks industrial presentations which deliver value and insight, but may not fit the selection process for regular papers.

Authors of industrial presentations are invited to submit a short overview (at least one page) of the proposed presentation by 8 January 2011. The Industrial Committee will review the proposals and make the selection. To submit your abstract, please visit:

http://conferences.ncl.ac.uk/adaconnection2011/industrial_contributions_form.php

The authors of selected presentations shall prepare a final short abstract and submit it by 16 May 2011, aiming at a 20-minute talk.

Accepted authors will also be invited to submit corresponding articles for publication in the Ada User Journal, which will host the proceedings of the Industrial Program of the Conference. For any further information please contact the Industrial Chair directly.

In addition to the award for best regular paper, Ada-Europe will also offer an honorary award for the best presentation, considering both regular and industrial presentations.

Schedule

----

08 January 2011: Submission of industrial presentation proposals

08 February 2011: Notification of acceptance to authors

16 May 2011: Industrial presentations required

20-24 June 2011: Conference

Industrial Committee

Jamie Ayre, AdaCore, France (Industrial Chair)

Guillem Bernat, Rapita Systems, UK

Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium

Hubert Keller, Forschungszentrum Karlsruhe GmbH, Germany

Ismael Lafoz, Airbus Military, Spain

Ahlan Marriott, White-Elephant GmbH, Switzerland

Paul Parkinson, Wind River, UK

Jean-Pierre Rosen, Adalog, France

Alok Srivastava, TASC Inc, USA

Rei Stråhle, Sweden

Rod White, MBDA, UK

Please circulate widely.

If you have a LinkedIn account, let your network know that you will be attending or are interested in thist event; see

http://events.linkedin.com/Ada-Connection-16th-International/pub/405820

Dirk.Craeynest@cs.kuleuven.be, Ada-Europe'2011 Publicity Chair

(V6.1)

14th Jornadas de Tiempo Real

From: Ada Spain news
Date: Tue, 01 Feb 2011 [fetched]
Subject: JTR 2011
URL: http://polaris.dit.upm.es/~str/jtr11/

The "Real-time systems" group at the Universidad Politécnica de Madrid organized the first edition of the "Jornadas de Tiempo Real" ("Real-time Days" —mp) in 1998, with the goal of contributing to establishing a common ground between the various Spanish research groups on real-time systems.

Since then, the Jornadas have been characterized by an open dialogue and collaboration, and as a meeting opportunity for those groups.

After thirteen years, the group at UPM returns to organize the "Jornadas de Tiempo Real".

This fourteenth edition shares the same goal: to facilitate the dissemination, analysis and constructive debate on the results of teaching and research experiences carried out by those groups.

The Jornadas will be held at the Escuela Técnica Superior de Ingenieros de Telecomunicación, located in the University Campus of Madrid, on the 3rd and 4th of February, 2011.

[Translated from Spanish. Find the program and download the papers or presentations (mostly in English) at

http://polaris.dit.upm.es/~str/jtr11/contribuciones.html —mp]

Ada in Debian and other distributions: FOSDEM video online

From: Dirk Craeynest <dirk@vana.cs.kuleuven.be>
Date: Sat, 12 Feb 2011 09:03:28 +0000 UTC
Subject: Ada at FOSDEM 2011 - video and slides available (update)
Newsgroups: comp.lang.ada

On Sunday February 6 at FOSDEM 2011, Ludovic Brenta and Miguel Telleria de Esteban gave a 1.5-hour talk about Ada. The video is online at:

http://www.youtube.com/watch?v=-3HvUH4fJPM&p=AC607424229CFC1A.

The slides for both parts of the talk ("Ada in Debian and Other Distributions" and "Ada Packaging Example: MAST") are available on Ada-Belgium's web page for this event:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/11/110205-fosdem.html

The Debian package for MAST has been uploaded as well:

http://ppa.launchpad.net/snapy/ppa/ubuntu/pool/main/m/mast/

MAST is a schedulability and time-completion analysis tool for real-time systems, developed at the CTR-lab of the University of Cantabria, Spain.

Enjoy!

The FOSDEM Team of Ada-Belgium

Videos from the High Assurance Software Symposium

From: Altran Praxis Press Center
Date: Fri, 21 Jan 2011
Subject: High Assurance Software Symposium and SPARK User Group Meeting
URL: http://www.altran-praxis.com/news/SPARKuserGroup_21_Jan_11.aspx

Altran Praxis and AdaCore are pleased to announce that the first videos from the October 2010 High Assurance Software Symposium and SPARK User Group meeting are now available on-line.

SPARK Ada in High SIL Active Life Support; Alex Deas, Deep Life.

Alex Deas describes how Deep Life have used SPARK to replace the core of a self-contained active life support system, used for diving, known as a rebreather. The rebreather maintains a diver's oxygen levels within a narrow range, as well as monitoring over 60 sensors and providing high integrity communications. Unsafe failure of the system can produce either low or high levels of oxygen, which will generally result in a fatal accident within a minute of the failure occurring. The application was assessed as IEC 61508 SIL 3, and was developed with SIL 4 rigour. A time-triggered architecture is used in which the microcontroller operates a task scheduler through which each computational module is executed with a predefined frequency and phase in relation to other tasks. Originally developed in MISRA C, the task scheduler has been converted to SPARK. This was so successful that all other functions are also being ported to SPARK, and SPARK will form the basis for future high-integrity systems within Deep Life.

Designing and Implementing a Verifiable High-Assurance Workstation; Alexander Senier, Secunet.

Alexander Senier presents a methodology and a system architecture for cost-efficient development of high-security systems. The approach, which uses SPARK for critical components, is being successfully used to build an interactive workstation for secure concurrent handling of multiple security domains, using the emerging system architecture called Multiple Independent Levels of Security (MILS). Reuse of general-purpose software and manageable verification effort make the proposed architecture both flexible and cost-efficient.

Autocoding – do we still need software design; Rod White, MBDA.

Over the past few years there has been a rapid proliferation of the use of autocoding across a wide range of software domains. With the heightened focus on code, that on the design and non-functional properties appears to have diminished. Rod White looks across the issues of autocoding and design and, from experience, examines the kind of problems that emerge and suggests how design might be used effectively in an autocoding environment.

More recordings from this highly successful event will be posted at the same location soon, so keep checking regularly for further updates.

The High Assurance Software Symposium and SPARK User Group meeting was hosted by Altran Praxis in Bath and brought together the SPARK community and clients interested in high assurance software.

[…]

Presentation of the results of project Couverture

From: Jamie Ayre <ayre@adacore.com>
Date: Wed, 19 Jan 2011 16:28:00 +0100

Subject: [ada-france] Le Projet Couverture
Newsletter: Ada-France newsletter

The Couverture project (http://www.open-do.org/projects/couverture/), by the Free Software group of the "Systematic" cluster, just finished. The project aimed to developing (with an open license) a set of software tools for the analysis of the structural coverage of an Ada program for use during its functional tests.

It also provides for the production of artifacts which permit the qualification of the developed tools according to the certification standards applicable to embedded safety-critical software.

If you want more information about the project and its results, you are cordially invited to participate to the event which will be held at the premises of AdaCore on January 27, 2011 at 17:30. Register at events@adacore.com

[translated from French —mp]

Ada-related Resources

On the book "Building Parallel, Embedded, and Real-Time Applications with Ada"

From: Marco <prenom_nomus@yahoo.com>
Date: Sat, 5 Feb 2011 04:57:20 -0800 PST
Subject: Questions on new John McCormick book
Newsgroups: comp.lang.ada

"Chapter 2 of the book "Building Parallel, Embedded, and Real-Time Applications with Ada" by McCormick, Singhoff, and Hugues includes a long section on low level programming with Ada. It covers both memory mapped and port I/O architectures. We develop polling and interrupt based device drivers. Everything is done using high level Ada abstractions for low level features. It includes a small amount of machine code insertion. We just turned in the final page proofs and expect to have hard copy by April."

1) Will this cover Ada 95 as well as 2005?

2) Do you cover some non-Intel x86 HW?

3) Will it be sold in the USA?

From: John McCormick <mccormick@cs.uni.edu>
Date: Sun, 6 Feb 2011 06:12:22 -0800 PST
Subject: Re: Questions on new John McCormick book
Newsgroups: comp.lang.ada

> 1) Will this cover Ada 95 as well as 2005?

Yes

> 2) Do you cover some non-Intel x86 HW?

Very little SPECIFIC hardware is covered. The only hardware specific topic is an example of inserting x86 machine code.

> 3) Will it be sold in the USA?

Yes

From: John McCormick <mccormick@cs.uni.edu>
Date: Wed, 9 Feb 2011 13:59:58 -0800 PST
Subject: Re: Questions on new John McCormick book
Newsgroups: comp.lang.ada

> […]

> John McCormick (on another thread) says they have only just handed the final proofs and hope to have hard copy in April. I will be interested to see what Amazon says now!

Indeed! If anyone finds a copy earlier they can let me know so I can get one :) I think that April is realistic. I do appreciate all of the excitement generated in this thread. Frank, Jérôme, and I hope our work justifies it. This semester is the second time students in my real-time embedded systems class have used the manuscript. The first group did a good job telling me where we needed to improve our explanations. Authors learn a lot from novices. The second group hasn't made any suggestions (yet) so maybe we got it right.

From: John McCormick <mccormick@cs.uni.edu>
Date: Thu, 10 Feb 2011 04:21:41 -0800 PST
Subject: Re: Questions on new John McCormick book
Newsgroups: comp.lang.ada

> […] Does the book cover any Ada 2012 innovations?

No.

Support for Ada in GitHub

From: R Tyler Croy <tyler@linux.com>
Date: 27 Nov 2010 08:17:15 GMT
Subject: GitHub added support for Ada
Newsgroups: comp.lang.ada

I posted this to /r/ada on reddit.com already but I thought you guys would enjoy this as well.

GitHub has finally added support recognizing Ada as a language meaning .adb and .ads files will now be syntax highlighted *and* you can find Ada projects using their "Explore" functionality: https://github.com/languages/Ada

[…]

Ada group on Identi.ca

From: R Tyler Croy <tyler@linux.com>
Date: Tue, 07 Dec 2010 19:53:15 GMT
Subject: Ada group on identica
Newsgroups: comp.lang.ada

I was bummed when I didn't find an Ada group on Identi.ca so I went ahead and created one:

http://identi.ca/group/ada

For the uninitiated, Identi.ca is a microblogging service not too dissimilar from Twitter, but far more heavily populated with hackers :)

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 09 Dec 2010 08:43:42 +0100
Subject: Re: Ada group on identica
Newsgroups: comp.lang.ada

[…]

There is http://identi.ca/adafrance/ ;-)

From: Maciej Sobczak <maciej@msobczak.com>
Date: Thu, 09 Dec 2010 00:58:14 -0800 PST
Subject: Re: Ada group on identica
Newsgroups: comp.lang.ada

[…]

Considering the small size of the Ada community, creating too many groups, forums, fan pages and what not like this will inevitably lead to dispersing the effort. I'm not sure if the community can afford this.

"Not too dissimilar from Twitter" begs the question: what is the added value of it?

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 09 Dec 2010 10:24:23 +0100
Subject: Re: Ada group on identica
Newsgroups: comp.lang.ada

[…]

The community is currently very small, compared to other languages. I think it is safe to say that we need all the visibility we can get. The Ada community generates some interesting things, but few people outside the Ada community is aware of this. I do not generate a lot of Ada code (yet!), but I do spend a bit of time every day trying to attract attention to the work done by people like you Maciej.

This is just another tool to help with that effort.

> "Not too dissimilar from Twitter" begs the question: what is the added value of it?

I think the identi.ca crowd is more technically oriented, so the audience is probably more inclined to check out a blurb about Ada.

Obviously, the more people who subscribe to the Ada group, the more alive and vibrant it will appear.

From: Marc A. Criley <mc@mckae.com>
Date: Fri, 10 Dec 2010 16:01:44 -0600
Subject: Re: Ada group on identica
Newsgroups: comp.lang.ada

> Considering a small size of the Ada community, creating too many groups, forums, fan pages and whatnot like this will inevitably lead to dispersing the

effort. I'm not sure if the community can afford this.

(I know your concern has diminished, but I wanted to make a point anyway :-)

I think effort (and people) are dispersed only when one totally abandons one "community" to join another. comp.lang.ada is probably the longest-serving, and probably strongest, Ada community on the Internet.

Sure, there's been turnover and participation waxes and wanes, but it does seem to be the home port, so to speak.

Taking Ada knowledge and advocacy to other venues in no way requires the dilution of the comp.lang.ada community. Overall it can increase the size of the community, some participants of which may end up here either casually or actively.

And instead of there being just a single home port for Ada, there can be a network of Ada havens that leverage one another.

For instance, I'm the creator and moderator of the Ada sub-reddit (http://www.reddit.com/r/ada). I regularly read comp.lang.ada for links to interesting articles and announcements of newly available products and releases, and I also monitor Thomas Locke's Ada Denmark site (http://ada-dk.org) for material. (Plus I subscribe to various mailing lists and just stumble across things from time to time.) And Reddit can be a source of material that I uncover, and those who monitor it can then pass that material on to others.

Each of us has our own set of readers and subscribers, and passing along information from one to another leverages network effects to get it to those who may not have the time to fish for information across the Web, or who are only dabbling in Ada to get a feel for it.

Our good buddy R. Tyler Croy recently wrote a blog post (http://unethicalblogger.com/posts/2010/12/ada_surely_you_jest_mr_pythonman) about his experiences learning Ada and posted it in the general programming forum on Reddit, where it did very well, garnering a lot of comments, discussion, and interest. I leveraged off his submission witha brief plug for the Ada sub-reddit, and in one day the number of subscriptions increased 10%, that was the largest one-day, or even one *week*, increase ever. It's hardly likely that there was just simply a bunch of Ada programmers on Reddit, who didn't know about the Ada sub-reddit, but who had now suddenly become aware that there was someplace associated with one of their hangouts to which they could join. No, I expect most of those new subscriptions were individuals who were intrigued by the discussion, and have opted to now keep on eye on what's up with Ada. Some of these may, like Tyler, become an active participant in the community.

The Ada group on identi.ca, then, is just another means of potentially expanding that community, providing another avenue for advocacy to those who might not have had any contact up till now with any of the established Ada points of presence on the Internet.

Ada links on Delicious

From: Georg Maubach <ada_resources@gmx.de>
Date: Tue, 4 Jan 2011 09:27:17 GMT
Subject: Ada Links on Delicious
Newsgroups: comp.lang.ada

Hi All,

over the time I collected the Ada resources I found and added them to my Delicious bookmarks. Today, I have opened them to the Ada community. You can use them be directing your browser to

http://www.delicious.com/tags/ada_resources

Any hints on other Ada links and resources not listed here are welcome and will be added to the list.

Literate Programs and Ada

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sat, 05 Feb 2011 19:11:21 +0100
Subject: Another place where to promote Ada (LP applied to Ada)
Newsgroups: comp.lang.ada

Hello,

Rosetta Code was previously (re-) introduced here, for the purpose of SPARK precisely.

Here is another similar place, oriented toward Literate Programming, and it proudly shows an Ada section, however currently with only two examples.

Furthermore, these two examples are so simple that they do not really explain anything; so this really needs to be fed.

If anyone is interested into promoting both Ada and LP, here is the place:

http://en.literateprograms.org/Category:Programming_language:Ada

Splitting a string with GNAT.String_Split

From: Thomas Løcke <tl@ada-dk.org>
Date: Fri, 24 Dec 2010
Subject: Using GNAT.String_Split to split a string
URL: http://ada-dk.org/?page=news&news_id=233

I've written a new article for our wiki: GNAT.String_Split Basic Usage Example.

[http://wiki.ada-dk.org/index.php/GNAT.String_Split_Basic_Usage_Example —mp]

Splitting a string into smaller parts based on one or more separators is not an uncommon task in programming, so when I stumbled on an Ada example for doing just that (thanks caracal!), I jumped at the opportunity to turn it into a short beginner article. And this really is a beginner's article, so everybody should be able to understand what's going on.

I hope.

As usual I've done my best to honor Ada by not letting too many bugs and errors slip by, but there's a good chance I might've failed somewhere.

If so, please let me know, or just fix it yourself. It is a wiki after all, so you have that power.

Ada-related Tools

Paraffin

From: Brad Moore <brad.moore@shaw.ca>
Date: Tue, 25 Jan 2011 09:04:59 -0700
Subject: Paraffin: Parallelism generics for Ada 2005
Newsgroups: comp.lang.ada

The initial release of paraffin is available at

http://paraffin.sourceforge.net/

It provides generics for adding parallelism to loops and recursive structures.

For iterative parallelism there are work-sharing, work-seeking, and work-stealing forms.

For recursive parallelism there are work-sharing and and work-seeking forms.

Also, the recursive parallelism can provide stack-safe recursion that avoids stack overflow.

The stack-safe recursion generics provide this capability even if used on a single processor.

Any feedback is greatly appreciated.

Magpie 0.10

From: Marc A. Criley <mc@mckae.com>
Date: Wed, 26 Jan 2011 19:12:41 -0600
Subject: Announce: Magpie 0.10 Utilities for multi-core execution
Newsgroups: comp.lang.ada

Hot on the heels of Brad Moore's Paraffin, and inspired by his article in Ada Letters (http://portal.acm.org/citation.cfm?id=1879078, membership required), comes the initial release of Magpie:

Magpie is a collection of generic procedures that distributes subranges of iterative application-defined processing across the processors of a multi-core CPU to achieve true concurrency and therefore increased application performance.

Magpie supports work sharing and work seeking processing for the GNAT compiler on Linux platforms. The platform restriction is due to processor-affinity not being a standard Ada feature until Ada 2012, so in the interim a GNAT-specific pragma is utilized (more info in the README).

Magpie is available for download at:

http://sourceforge.net/projects/magpie-mc

Break into groups, execute, and rejoin with your consensus.

From: Marc A. Criley <mc@mckae.com>
Date: Wed, 02 Feb 2011 20:02:12 -0600
Subject: Re: Announce: Magpie 0.10 Utilities for multi-core execution
Newsgroups: comp.lang.ada

> Brad, Marc, what is the difference between Magpie and Paraffin?

Magpie provides two generic functions and a generic procedure. They're designed to operate on a container of data that is indexed or keyed by values of a discrete type. E.g., an integer indexed array, or a vector.

The generic functions retrieve a value from the container, apply an application-supplied function to it, then combine ("reduce") each result with a running total, until all values have been processed, and the final value of the running total is returned as the generic function's value.

The generic procedure repeatedly invokes an application-supplied procedure, accompanying each invocation with an index value. There is no value returned, the supplied procedure is simply repeatedly invoked, presumably progressing through each element of the container, and it is responsible for managing its results.

The technique I got from Brad's article had to do with partitioning the range of index/keys amongst application-designated CPU cores.

One of the functions simply evenly splits the range across the number of cores and kicks off the execution of each task associated with a core.

The other function, and the procedure, initially split the range evenly, but if one of the core tasks finishes early, it advertises that it is available to take on more work. When one of the other tasks notices this, it stops, splits its remaining work, and both resume execution.

There's more detail about all this in the README, along with a couple examples, including a Magpie version of Jakob Sparre Andersen's Mandelbrot set generator.

Matreshka 0.0.6

From: Vadim Godunko <vgodunko@gmail.com>
Date: Fri, 4 Feb 2011 03:15:04 -0800 PST
Subject: Announce: Matreshka 0.0.6
Newsgroups: comp.lang.ada

We are pleased to announce Matreshka 0.0.6 release. Major improvements in this release is:

- FastCGI support;
- XML writer;
- access to command line arguments and environment variables;
- use of SSE2 instructions set on x86 platform when available;
- support for SHIFT-JIS encoding;
- update to Unicode 6.0.0.

Please visit http://adaforge.qtada.com/cgi-bin/tracker.fcgi/matreshka/downloader

to download source code and obtain additional information.

Matreshka is a set of reusable components to construct Ada applications. It includes components to handle strings of Unicode character, text codecs, message translator, regular expression engine, XML processor, and FastCGI module.

[see also "Matreshka v0.0.4 and v0.0.5" in AUJ 31-3 (Sep 2010), p.161 —mp]

Storage_Stream

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 09 Feb 2011 12:00:08 +0100
Subject: [Ann] package Storage_Stream, was: How do I write directly to a memory address?
Newsgroups: comp.lang.ada

Back to the OP, since many people have that need of putting various stuff at raw memory addresses, I'm happy to provide a stream that does just that: write data of any type at any memory address.

It is available from Adalog's component page:

http://www.adalog.fr/compo2.htm

GMGPL as usual, of course.

AVR-Ada and GCC 4.5.0

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Mon, 15 Nov 2010 15:27:00 +0000
Subject: AVR-Ada and GCC4.5.0 revisited.
Newsgroups: comp.lang.ada

Around August, I left this thread hanging as I ran out of play time…

Then, I was having difficulty getting a tool chain working under Linux - most of the information online including the article at

http://sourceforge.net/apps/mediawiki/avr-ada/index.php?title=Setup

and the script and links from that page, refer to GCC4.3.x.

Following these instructions to build GCC4.3.x with GCC 4.5.0 installed is just not going to work…

Modifying the script to use GCC 4.5 and newer packages, results in obscure errors about two non-existent files (which do exist, however a third file, tested but not reported, does not!) and my attempts to understand this failed…

(Gory details suppressed, but building mpfr apparently fails to find gmp.h)

Building the GCC4.5 tools from scratch, following the above article, but editing versions, I could build GCC with Ada support, and gnatbind, but not gnatprep, gnatlink, or gnatmake. Again, no idea why not…

HOWEVER…

In that thread, Tero Koskinen posted that he had the 4.5.0 files built for Fedora Core 13 at http://iki.fi/tero.koskinen/ avr-gnat/rpms

I was wary of trying these in case of incompatibilities with OpenSuse.

But after struggling with the source for long enough, I gave them a try.

They are not a complete set, (at least, for OpenSuse 11.3) I needed to add avr-binutils (avr-binutils-2.20-2.fc13.x86_64.rpm) and avr-libc (avr-libc-1.6.7-2.fc13.noarch.rpm), both from rpm.pbone.net. Thre is a newer avr-libc (1.70) for FC14, but I wanted to keep all the versions in sync for now.

On OpenSuse 11.3, I used the procedure:

(1) load the above and Tero's six rpms into a local directory

(2) added the local directory as a "plain rpm" repository

(3) YAST then allowed me to select and install the RPMs.

MUCH easier than previous attempts…

So far I have only built (not yet downloaded) the "led_on" example from the introductory tutorial at

http://sourceforge.net/apps/mediawiki/avr-ada/index.php?title=Tutorial

but the process went smoothly.

So… Thanks to Tero Koskinen, as well as to Warren and others for putting the tutorials together!

From: Rolf Ebert <rolf_ebert@users.sourceforge.net>
Date: Tue, 16 Nov 2010 14:15:32 -0800 PST
Subject: Re: AVR-Ada and GCC4.5.0 revisited.
Newsgroups: comp.lang.ada

[…]

Please note that gcc-4.5 for AVR has a serious bug (http://gcc.gnu.org/

bugzilla/show_bug.cgi?id=46192) that prevents you using the compiler for anything that is slightly more complex than the LED example (i.e. anything)

I'll probably use gcc-4.4 for the next AVR-Ada release.

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Wed, 17 Nov 2010 07:07:07 +0000
Subject: Re: AVR-Ada and GCC4.5.0 revisited.
Newsgroups: comp.lang.ada

[…]

Thanks for pointing this out. I've seen various suggestions that gcc-4.3.x is needed for AVR, but nothing explicit why - until now.

Pity.

I had just confirmed that (with minor changes to the Makefile) Warren's tutorial works on the Arduino Duemilanove.

Now it looks as if I need to find a native gcc-4.3.x for my system and start again.

From: Warren W. Gay <ve3wwg@gmail.com>
Date: Wed, 24 Nov 2010 21:26:10 +0000 UTC
Subject: Re: AVR-Ada and GCC4.5.0 revisited.
Newsgroups: comp.lang.ada

[…]

> I had just confirmed that (with minor changes to the Makefile) Warren's tutorial works on the Arduino Duemilanove. […]

Great to hear that Brian! Arduino has new life (for me anyway), when you can combine it with Ada instead of that "munged Arduino flavoured C++". ;-)

You can see my synth's atmega168 MIDI module here:

http://tinyurl.com/3xhlutv

Written in AVR-Ada, running with one main thread receiving MIDI messages and another processing "events" and scheduled events (through timeouts).

I have a VCA module just about ready for it as well, but no front module panel yet.

From: Warren <ve3wwg@gmail.com>
Date: Mon, 29 Nov 2010 15:10:50 +0000 UTC
Subject: Re: AVR-Ada and GCC4.5.0 revisited.
Newsgroups: comp.lang.ada

[…]

I guess you can't go directly to the picture without logging in.

Try the following thread link instead:

http://experimentalistsanonymous.com/board/index.php?topic=2445.0

[…]

From: Simon Wright <simon@pushface.org>
Date: Fri, 11 Feb 2011 18:16:27 +0000
Subject: Re: avr-ada ubuntu cross-compiler build failed
Newsgroups: comp.lang.ada

[…]

> My dreams of success were short-lived, though, when someone pointed out a catastrophic bug in GCC4.5.0 targeting the AVR… […]

Actually, this bug is also present in GCC version 4.6.0 20110203 (experimental) [trunk revision 157963] for x86_64-apple-darwin10.

I've updated bugzilla

(http://gcc.gnu.org/bugzilla/show_bug.cgi?id=46192).

The problem is there at -O2, goes away at -O1 (and -O0).

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Fri, 11 Feb 2011 18:44:01 +0000
Subject: Re: avr-ada ubuntu cross-compiler build failed
Newsgroups: comp.lang.ada

[…]

Do you happen to know if this applies ("goes away at -O1") in GCC 4.5 too, or just 4.6?

(If so, I may stick with 4.5.0 and -O1 for the time being, until I run out of space)

From: Simon Wright <simon@pushface.org>
Date: Sat, 12 Feb 2011 13:27:37 +0000
Subject: Re: avr-ada ubuntu cross-compiler build failed
Newsgroups: comp.lang.ada

[…]

Yes, it does. (on x86_64-apple-darwin10).

Are you sure this is really a problem for you? It's that pragma Volatile isn't retained *through a renaming*. Do you have to rename?

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Sat, 12 Feb 2011 17:49:24 +0000
Subject: Re: avr-ada ubuntu cross-compiler build failed
Newsgroups: comp.lang.ada

[…]

>Are you sure this is really a problem for you? It's that pragma Volatile isn't retained *through a renaming*. Do you have to rename?

Good point, there are other ways around the problem.

However some of the example code uses renaming, and I can see why. Naming a register or bit according to it's intent, rather than its address, does add a lot to readability, essentially "for free" in terms of code generation.

From: Simon Wright <simon@pushface.org>
Subject: Re: avr-ada ubuntu cross-compiler build failed
Date: Sat, 12 Feb 2011 21:45:14 +0000
Organization: A noiseless patient Spider
Newsgroups: comp.lang.ada

[…]

It would be a lot of work, but inlined subprograms might do the job?

With GNAT, you can use high optimisation levels and -gnatn (-gnatN?) but I've found you need to be more selective (increases code size, often slows the executable). Use pragma Inline_Always (a GNAT special), not just Inline.

GNAT 2009 for the LEGO MINDSTORMS NXT — Linux port

From: Peter Bradley <pbradley@datsi.fi.upm.es>
Date: Thu, 18 Nov 2010 13:23:44 -0600
Subject: GNAT for LEGO Mindstorms using a Linux host
Newsgroups: comp.lang.ada

Dear Ada developers,

At the UPM we have been working with the LEGO Mindstorms trying to elaborate an environment to develop real-time applications in Ada for the Mindstorms robotics kit.

One of the things we have done so far is to port the GNAT cross compilation system to Linux and we want to share this work with the Ada community.

At the moment, the 2009 release has been ported but we are already working with the Ravenscar edition.

You can download the Linux x86 binaries at:

http://polaris.dit.upm.es/~str/proyectos/mindstorms

Regards.

[see also "AdaCore — GNAT GPL for the LEGO MINDSTORMS NXT - Ravenscar Edition" in AUJ 31-4 (Dec 2010), p.233 —mp]

GNAT AUX ported to Android

From: DragonLace website
Date: Sat, 29 Jan 2011 17:51:59
Subject: GNAT AUX ported to Android
URL: http://www.dragonlace.net/posts/GNAT_AUX_ported_to_Android/

This weekend, a GNAT AUX cross compiler was built for Android (ARM architecture).

While there are no regression test results available, we can at least demonstrate that a helloworld program written in Ada runs on the Android SDK emulator.

The standard GCC version of GNAT runs flawlessly on the ARM v7 port of Linux with the default sjlj exception handling, but to our knowledge it has never been ported to the Android operating system before.

The purpose for this cross compiler is that Android applications can be written entirely in Ada, and that they interface natively with the C libraries that control the phone/tables IO events. Starting with Android 2.3 (Gingerbread) it's possible to write applications in languages other than Java.

The intention was to make C and C++ programmers happy, but it opened the door for the rest of us.

Speaking of which, credit needs to be given to Mike Long for blazing the trail.

He blogged about getting a Fortran application running on his Android device, and even has a nice YouTube video to prove it. We'll try to produce similar results soon enough, but check out Fortran on Android. Stay tuned for more developments about Ada on Android.

[see also "Ada for Android" in AUJ 31-4 (Dec 2010), p.240 —mp]

Atmega and Attiny processors support for AVR-Ada

From: Tero Koskinen's blog
Date: Mon, 14 Feb 2011
Subject: Arduino Mega 2560 and Attiny13a/Attiny2313 support to AVR-Ada
URL: http://tero.stronglytyped.org/blosxom.cgi/2011/02/14#avradaimprovements

I recently got write access to the AVR-Ada repository and now I have pushed my changes there.

These changes improve support for Atmega2560, Attiny13a, and Attiny2313 processors. Attiny13a and Attiny2313 are pretty uninteresting, although common, AVR processors. I use them in my projects because they are cheap and that is why I also wanted better support for them.

On the other hand, Atmega2560 processor is used in the new Arduino Mega 2560 board.

This means that next release of AVR-Ada will support the new Mega board out of the box.

Some bits, like support for timers 3..5 and extra UARTS, are missing, but at the moment Atmega2560 should have about same features as Atmega328p supported.

Mathpaqs - November 28, 2010

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Mon, 29 Nov 2010 04:04:54 -0800 PST
Subject: Mathpaqs release 28-Nov-2010
Newsgroups: comp.lang.ada

Hello,

There is again a new release of mathpaqs…

The reason of a second release in such a short time is that I've realized that the first one was not really worth the name "release".

What's new:

- There is a GNAT project file
- Sparse matrix package is working (was in an incompatible Ada 83 mode!)
- An auxiliary portable graphics package (.ps output) was added, principally for the tests / demos
- New fractal demo was added
- Most packages, demos or tests are now working

Mathpaqs is a set of various mathematical packages in Ada including algebra, finite elements, sparse matrices, random variables, probability dependency models, unlimited integers.

URL: http://sf.net/projects/mathpaqs/

[see also "Mathpaqs - November 2010" in AUJ 31-4 (Dec 2010), p.229 —mp]

Ada 2005 Math Extensions

From: Simon Wright <simon@pushface.org>
Date: Thu, 23 Dec 2010 22:21:38 +0000
Subject: ANN: Ada 2005 Math Extensions 20101223
Newsgroups: comp.lang.ada

This release is now available at

http://sourceforge.net/projects/gnat-math-extn/files/20101223/

Changes:

The package is renamed to Ada_Numerics.Generic_Arrays.

An additional overloaded procedure Eigensystem returns the generalized eigenvalues and eigenvectors of a pair of non-symmetric real matrices.

NB, this is only supported for unconstrained Float and Long_Float a this time.

To do:

- Add support for constrained floats to this Eigensystem.

- Add generalized solutions for non-hermitian complex matrices.

…

Ada industrial control widget library v1.0

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 12 Feb 2011 15:22:12 +0100
Subject: ANN: Ada industrial control widget library v1.0 released
Newsgroups: comp.lang.ada

The library is intended for designing high-quality industrial control widgets for Ada applications.

The widgets are composed of transparent layers drawn by cairo. The widgets are fully scalable graphics. Time-controlled refresh policy is supported for real-time and heavy-duty applications. The library supports caching graphical operations and stream I/O for serialization and deserialization.

Ready-to-use gauge and meter widgets provided as samples as well as an editor widget for WYSIWYG design of complex dashboards. The software is based on GtkAda and cairoada, Ada bindings to GTK+ and cairo.

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

Comments and suggestions are welcome.

P.S. Thanks to Damien Carbonne for making cairoada binding available.

New release of the Embedded Web Server

From: Simon Wright <simon@pushface.org>
Date: Sun, 16 Jan 2011 12:31:24 +0000
Subject: ANN: EWS 20110115
Newsgroups: comp.lang.ada

This is a new release of my Embedded Web Server, to be found at

https://sourceforge.net/projects/embed-web-srvr/.

This release correctly supports HTTP/1.0 keep-alive, as requested by the "Connection: Keep-Alive" field.

[see also "EWS — Embedded Web Server" in AUJ 27-4 (Dec 2006), p.201 —mp]

Ada bindings to gtkdatabox

From: Per Sandberg <per.sandberg@bredband.net>
Date: Tue, 07 Dec 2010 21:33:52 +0100
Subject: [ANN] Ada bindimgs to gtkdatabox
Newsgroups: comp.lang.ada

There is now a binding to gtkdatabox located on:

https://github.com/persan/A-gtkdatabox

From: Per Sandberg <per.sandberg@bredband.net>
Date: Tue, 07 Dec 2010 22:14:47 +0100
Subject: Re: [ANN] Ada bindimgs to gtkdatabox
Newsgroups: comp.lang.ada

[…]

> Does it use time stamps on the X axis (i.e. oscilloscope), or is it just a XY

plot? It's basically an XY plot with float values.

Is it based on cairo?

Have not thought about it, I was looking for a widget that makes it possible to build a simple UI for a pc-oscilloscope.

> How does it handle the case when several data points must be rendered in one-pixel horizontal width?

I have not looked deeper into the scaling effects when several X-values should be rendered one pixel wide, but it seems like a mean value is drawn.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 7 Dec 2010 23:17:29 +0100
Subject: Re: [ANN] Ada bindimgs to gtkdatabox
Newsgroups: comp.lang.ada

[…]

I considered porting a C++ library that does the oscilloscope to Cairo Ada, but I am not so sure now.

[…]

Cairo looks a good choice. There are excellent Cairo Ada bindings by Damien Carbonne.

Considering the oscilloscope implementation there is a problem. I don't know how to get a BitBlt under Gtk or Cairo, which would be essential for good performance of scrolling.

[…]

> Have not looked deeper into the scaling effects when several X-values should be rendered one pixel wide, but it seems like a mean value is drawn.

I see, not a min-to-max vertical bar (the correct behavior).

Ada bindings for ØMQ 2.1.0

From: Per Sandberg <per.sandberg@bredband.net>
Date: Wed, 08 Dec 2010 06:30:37 +0100
Subject: [ANN] Ada Bindings for 0MQ 2.1.0
Newsgroups: comp.lang.ada

The Ada-bindings are now updated to reflect the 2.1.0 ØMQ specs and a lot of convenient functions are added.

The bindings are available from

https://github.com/persan/zeromq-Ada

[see also "Ada binding for ØMQ" in AUJ 31-2 (June 2010), p.86 —mp]

Ada binding to Linear Algebra PACKage

From: Nasser M. Abbasi <nma@12000.org>
Date: Tue, 14 Dec 2010 21:32:02 -0800
Subject: What is the status of Ada binding to lapack?
Newsgroups: comp.lang.ada

Searching I could only find a reference to

http://www.sigada.org/ada_95/bindings.html

With the following broken link to binding of Ada to Lapack

ftp://cs.nyu.edu/pub/gnat/contrib/lapack-ada/

How does one uses lapack from Ada without binding?

I understand that with the latest GNAT GPL 2010, these libraries (lapack and blas) are part of the distribution and are automatically linked with since they are used by some one the new Ada 2005 math functions (Annex G?) such as Solve().

But what if I want to make specific calls to other lapack functions myself from Ada?

ps. I found a page which says to have Ada binding to Blas,

http://topo.math.u-psud.fr/~sands/Programs/BLAS/

but did not try it.

From: John B. Matthews <jmatthews@wright.edu>
Date: Wed, 15 Dec 2010 07:00:46 -0500
Subject: Re: What is the status of Ada binding to lapack?
Newsgroups: comp.lang.ada

[…]

Simon J. Wright's Ada 2005 Math Extensions project may be if interest:

http://sourceforge.net/projects/gnat-math-extn/

[read also "Ada 2005 Math Extensions" in this issue of the AUJ —mp]

VTKAda 5 release 4

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Tue, 4 Jan 2011 02:31:19 -0800 PST
Subject: Announce : VTKAda version 5
Newsgroups: comp.lang.ada

Announce : VTKAda version 5 release 4 (work in progress)

VTKAda is an Ada-95(05) interface to VTK (Visualization Toolkit) and Qt4 graphics library

VTK version 5.6.1, Qt version 4.7.1 open source and qt4c.dll(libqt4c.so) built with Microsoft Visual Studio 2010 in Windows and GCC in Linux x86-64.

The package was tested with the GNAT GPL 2009 Ada compiler in Windows 32-bit and 64-bit and Linux x86-64 Kubuntu 9.10.

VTKAda is a powerful 2D, 3D rendering and imaging system and works inside Qt4 application.

You can get more information from the paper "Modern application development with Ada" on the website.

VTKAda and QtAda for Windows and Linux (Unix) are available from

http://users1.jabry.com/adastudio/index.html

[…]

[see also "VTKAda" in AUJ 31-3 (Sep 2010), p.158 —mp]

GtkAda contributions v2.9

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 10 Jan 2011 17:54:49 +0100
Subject: ANN: GtkAda contributions v2.9
Newsgroups: comp.lang.ada

A contribution to GtkAda, Ada bindings to GTK+, deals with the following issues:

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

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

Changes to the version 2.8:

1. Progress indication added to Gtk_Abstract_Directory_Record;
2. Style_Get added to Gtk.Widget.Styles returning GValue by name;

3. RGB convenience function was added to Gtk.Missed;
4. Gtk.Generic_Enum_Combo_Box provides combo box widget created from an enumeration type (contributed by Oliver Kellogg);
5. The documentation links to the AdaCore on-line GtkAda Reference Manual were fixed because the old reference manual is no longer available.

[see also "GtkAda Contributions v2.7 and v2.8" in AUJ 31-3 (Sep 2010), p.158 —mp]

Termination review requested for the Hibachi project

From: Tero Koskinen <tero.koskinen@iki.fi>
Subject: Hibachi project getting termination review
Date: Tue, 14 Dec 2010 07:24:09 +0200
Newsgroups: comp.lang.ada

[…]

It seems that Eclipse people are shutting down the Hibachi project since there has been no activity lately:

http://dev.eclipse.org/mhonarc/lists/hibachi-dev/msg00134.html

David M Williams wrote:

> The Tools PMC would like to request a termination review of the Hibachi project. For what ever reasons they have not been able to stay an active project working towards release, so if no others come forward to take over and revive the project, it should be formally terminated and archived. We hope this process will "flush out" any community interest that needs to get coordinated to keep the project going, and towards that end, it would be good to have a termination review scheduled to have a deadline.

[see also "Status of Hibachi" in AUJ 30-3 (September 2009), p.146 —mp]

SPARKRules 1.0

From: Phil Thornley <phil.jpthornley@gmail.com>
Date: Fri, 10 Dec 2010 07:20:41 -0800 PST
Subject: ANN: SPARKRules Version 1.0 now available
Newsgroups: comp.lang.ada

SPARKRules is designed to support libraries of user rules and review proofs. It avoids use of the verification directories for the permanent storage of user rules and review proofs.

Version 1.0 of SPARKRules is available from:

http://sparksure.com

The download includes documentation for the tool, a Windows executable, all the source code (which has also compiled and run on openSUSE) and, as an example of its use, reworkings of the proof rules in Tokeneer Version 2.

SPARKRules separates the storage of user rules from the context in which they are applied, so the documentation includes a discussion of the potential for unsound proofs to result from this and suggests an approach to avoid this problem.

Ada support in *BSD distributions

From: DragonLace website
Date: Sat, 01 Jan 2011 12:35:08
Subject: GPS packages built for FreeBSD and DragonFly
URL: http://www.dragonlace.net/posts/GPS_packages_built_for_FreeBSD_and_DragonFly/

There exist seven ports for FreeBSD and an analogous seven packages for Pkgsrc which serves DragonFlyBSD and NetBSD. These ports/packages are mature enough that the process to introduce them into their respective trees will begin soon.

We've used these ports/packages to build binaries for the following platforms:

- FreeBSD i386
- FreeBSD AMD64
- DragonFlyBSD i386
- DragonFlyBSD x86_64

The primary binaries available for these four platforms are:

- GNAT AUX Ada Compiler
- GNAT Programming Studio (GPS)
- Ada Web Server (AWS)

Supporting binaries available for these four platforms are:

- GNATPython (used for AWS regression testing)
- GTKAda (GTK binding required by GPS)
- XMLAda (General purpose XML library used by several projects)
- GPRBuild-AUX (used to build multiple-language projects)

All packages will build on NetBSD AMD64, but binaries are not available for this platform at the moment. GtkAda doesn't seem to build on NetBSD i386 and thus no GPS has ever been built on this platform. The issue is some kind of linker problem and more investigation is required. It is not known at this time if the problem will hold up the submission of the packages into the pkgsrc tree.

From: DragonLace website
Date: Sat, 22 Jan 2011 07:18:45
Subject: Going In The Ports Tree
URL: http://www.dragonlace.net/posts/Going_In_The_Ports_Tree/

All seven ports (GNAT-AUX, XML/Ada, GPRBuild, GTK/Ada, GPS, GNATPython, and AWS) were submitted to the FreeBSD Ports community on 9 January. The report that tracks the request is PR 153828 and they've just started the testing on the submission, so hopefully they'll be available to FreeBSD users soon.

Today, the same ports minus AWS were submitted to NetBSD's pkgsrc. Since DragonFlyBSD also uses pkgsrc, users with i386 and x86_64 processors of both operating systems will be able to build these from source. When the 2011Q1 bulk packages are built, they'll be able to just install the binaries from standard repositories. This will take about 3 months because the 2010Q4 snapshot is just getting built now.

Ever since Unwind Support was added to NetBSD, the Ada Web Server stopped building on NetBSD. It's linking to the system gcc library (too old) rather than the GNAT version of it. It's actually only a tool that's not build (awsres).

Ideally the authors of AWS will be able to suggest a fix, otherwise the three options are either exclude NetBSD from the platform list (in other words DragonFlyBSD is the only platform that can build it) or somehow patch the source to skip building the awsres tool. So for now, the AWS package hasn't been submitted.

The pending PRs on NetBSD are 44436 through 44441.

From: DragonLace website
Date: Sat, 12 Feb 2011 13:00:30
Subject: DragonLace Mail list and Pkgsrc
URL: http://www.dragonlace.net/posts/DragonLace_Mail_list_and_Pkgsrc/

The big news of the day is that NetBSD has imported all the packages into the pkgsrc repository. This means that NetBSD and DragonFlyBSD users can build GNAT, the GNAT Programming Studio, and others from source today if they update pkgsrc to the repository head. They will show up in a few months in the next quarterly release (2011Q1), and at that time maintainers for NetBSD and DragonFlyBSD will build the binary packages.

Our friends at ada.cx have graciously hosted our brand new mailing list.

For now, feel free to post about anything that DragonLace is involved with on our one list. Later we may differentiate and have separate lists for GNAT, Draco, and Android application building.

[http://lists.ada.cx/archives/dragonlace/ to register to the mailing list —mp]

[…]

AWS is now available via pkgsrc as well.

[GNAT AUX was originally the compiler for the AuroraUX project. See "Update on the AuroraUX project" in AUJ 31-2 (June 2010) p.96. While the project seems stagnating, the work on the compiler seems promising. —mp]

ZanyBlue Ada localization packages

From: Michael Rohan <michael@zanyblue.com>
Date: Mon, 22 Nov 2010 02:19:05 -0800 PST
Subject: ANN: ZanyBlue.Text, Localization Support for Ada
Newsgroups: comp.lang.ada

Hi Folks,

This is the first release of the ZanyBlue Ada packages. This release includes the Text package which support localization of Ada applications with Java style message formatting and .properties files. The released files are available at

http://sourceforge.net/projects/zanyblue/files/

There are two downloadable files:

1) The core release files: zanyblue-0.1.0b-r1663.tar.gz

2) Support files for testing (AUnit) and regneration of the built-in localizations (CLDR) zanyblue-0.1.0b-r1663-libs3rd.tar.gz

The primary dev environment is Unix but has been tested on Windows.

[…]

From: Michael Rohan <michael@zanyblue.com>
Date: Thu, 25 Nov 2010 23:00:44 -0800 PST
Subject: Re: ANN: ZanyBlue.Text, Localization Support for Ada
Newsgroups: comp.lang.ada

[…]

Noticed that the number of downloads for the libs3rd package exceeded the download for the actual ZanyBlue code: I've re-uploaded the code making it the default download. The libs3rd package contains the third-party dependencies for ZanyBlue needed only if you want to change the set of built-in localizations via the Unicode.org CLDR XML data or you don't have AUnit.

The regression tests, via a "make check" in the src directory attempt to first build to the AUnit library in the src/libs3rd directory. If AUnit is already installed simply run "make check" in the src/test directory rather than the src directory (the build of AUnit, via ZanyBlue, only works on Unix).

ssprep v1.5.6 and 1.5.7

From: Per Sandberg <per.sandberg@bredband.net>
Date: Mon, 17 Jan 2011 20:27:16 +0100
Subject: [ANN] ssprep-1.5.6
Newsgroups: comp.lang.ada

ssprep is a tool to generate initial project structures from a set of templates that is highly configurable and expandable.

sample:

```
$ssprep
  ssprep.simpleExecutableProject
  -Dproject=demo
```

will generate a full project tree with unit and regression tests.

```
$ssprep
  ssprep.simpleExecutableProject
  -Dproject=demo -dWITH_ASIS=True
```

will add a template for an ASIS program as well.

After the project generation its just

```
$make
$make test
$make install
```

or open the project in GPS

http://sourceforge.net/projects/ssprep/files/ssprep/1.5.6

From: Per Sandberg <per.sandberg@bredband.net>
Date: Tue, 18 Jan 2011 07:26:06 +0100
Subject: Re: [ANN] ssprep-1.5.6
Newsgroups: comp.lang.ada

URL should be:

http://sourceforge.net/projects/ssprep/files/ssprep/1.5.7

Found some last time glitches.

[…]

[see also "ssprep-1.5.3" in AUJ 31-4 (Dec 2010), p.231 —mp]

Ada-related Products

AdaCore — GNATemulator

From: AdaCore Press Center
Date: Tue, 01 Feb 2011
Subject: AdaCore Releases New GNATemulator Tool
URL: http://www.adacore.com/2011/02/01/gnatemulator/

AdaCore Releases New GNATemulator Tool for Efficient Embedded Software Testing Provides open source, integrated, lightweight target emulation

PARIS and NEW YORK, February 1, 2011 – AdaCore, a leading supplier of Ada development tools and support services, today announced the release of GNATemulator, an efficient and flexible emulator solution for testing embedded software applications. Based on the QEMU technology, a generic and open source machine emulator and virtualizer, the new GNATemulator tool allows software developers to compile code directly for their target architecture and run it on their host platform, through an approach that translates from the target object code to native instructions on the host. This avoids the inconvenience and cost of managing an actual board, while offering an efficient testing environment compatible with the final hardware.

There are two basic types of emulators. The first go far in replacing the final hardware during development for all sorts of verification activities, particularly those that require time accuracy. However, they tend to be extremely costly, and are often very slow. The second, which includes the GNATemulator, do not pretend to be complete time-accurate target board simulators, and thus cannot be used for all aspects of testing, but do provide a very efficient, cost-effective way of executing the target code very early and very broadly in the development and verification process. They offer a practical compromise between a native environment that is too far from the actual target, and the final hardware that is never available soon enough or in enough quantity.

“GNATemulator affords designers a lightweight, easy-to-use tool for target code execution during early development and verification processes, where greater agility and efficiency is important,” said Cyrille Comar, Managing Director at AdaCore. “As a streamlined, low-cost alternative to time-accurate target board simulators, GNATemulator is ideally suited for the broad range of testing scenarios for which full-feature emulators would be overkill.”

Combines Agile concepts with ease-of-use GNATemulator helps automate testing campaigns for embedded application code and thus allows developers to use continuous integration techniques made popular by the Agile community. Many versions of GNATemulator can be launched simultaneously, making it possible to parallelize testing. GNATemulator is smoothly integrated into the GNAT Pro toolset and can be used with other AdaCore tools, such as the GNAT debugger (a part of the GNAT Programming Studio) and GNATcoverage.

Cost effective

GNATemulator reduces hardware cost and maintenance: the actual target is needed only for integration testing, since unit and functional testing can be done directly on the emulator, which is installed on a standard desktop machine. Development teams can thus start producing and testing code for the target before acquiring the actual target hardware.

Improved productivity

GNATemulator can be installed directly on each developer's desktop machine, allowing tests to be written for the final target taking into account particulars, such as endianness and assembly code. It thus improves test development productivity. Optimized and efficient, GNATemulator runs on the host platform, which is usually more powerful than the target, and executes code faster than on the actual target.

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 safety, security, and reliability are critical. AdaCore's flagship product is the GNAT Pro development environment, which comes with expert on-line support and is available on more platforms than any other Ada technology. AdaCore has an extensive worldwide customer base; see http://www.adacore.com/home/company/customers/ for further information.

Ada and GNAT Pro continue to see growing usage in high-integrity and safety-certified applications, including commercial aircraft avionics, military systems, air traffic management/control, railway systems and medical devices, and in security-sensitive domains such as financial services.

AdaCore — GNATcoverage

From: AdaCore Press Center
Date: Tue, 01 Feb 2011
Subject: AdaCore Releases New GNATcoverage Tool
URL: http://www.adacore.com/2011/02/01/adacore-releases-new-gnatcoverage-tool/

AdaCore Releases First Non-Intrusive Coverage Tool to Fully Support All Levels of Safety Certification.

New GNATcoverage tool provides advanced program coverage analysis on both object code and source code.

PARIS and NEW YORK, February 1, 2011 – AdaCore, a leading supplier of Ada development tools and support services, today announced the release of GNATcoverage, a tool that analyzes and reports program coverage. Originally developed as part of the Couverture research project, GNATcoverage performs coverage analysis on both object code — instruction and branch coverage – and Ada and C language source code – statement, decision and Modified Condition/Decision Coverage (MC/DC). Unlike most current technologies, the tool works without requiring instrumentation of the executable. Instead, it runs directly on an instrumented version of GNATemulator, a lightweight and efficient emulator tool provided by AdaCore. GNATcoverage helps software developers assess the breadth of a testing campaign and provides precise answers to the needs of safety-certification processes, such as the DO-178 avionics standard.

"Until recently, the relationship between source coverage and object coverage had never been studied in detail. The research part of the 'Couverture' project proved mathematically the exact perimeter in which complex source coverage metrics, such as MC/DC, could be deduced from object coverage information," said Cyrille Comar, Managing Director of AdaCore. "It allowed us to build the first coverage tool, working on un-instrumented code and providing results sufficiently accurate for meeting the needs of the highest level of avionics certification. We are very proud of this achievement!"

Combines Agile concepts with ease-of-use

By automating testing and coverage analysis of embedded application code, GNATcoverage allows developers to use continuous integration techniques made popular by the Agile community. Many instances of the tool can be launched simultaneously, making it possible to parallelize coverage analysis. GNATcoverage can also be installed on individual developers' desktops, allowing them to verify locally and easily the adequacy of their testing strategy.

Analyzes the final code that will run on the embedded target GNATcoverage provides code coverage information directly on the embedded target application code. There is no instrumentation of the object code – the instrumentation is done directly at emulation level.

Aids in establishing certification requirements (DO-178B level A, EN 50128, IEC 61508, ECCS-E40B)

GNATcoverage provides source coverage information for all levels of safety certification (Statement Coverage, Decision Coverage and Modified Condition/Decision Coverage). Qualification material is available for DO-178 B up to level A. GNATcoverage can also provide object-level coverage metrics in term of binary instructions and which branches have been exercised.

[…]

Inspirel — YAMI4 v. 1.2.1 and 1.2.2

From: Inspirel Press Center
Date: Fri, 19 Nov 2010
Subject: YAMI4-1.2.1 released
URL: http://www.inspirel.com/news.html

The YAMI4-1.2.1 version of YAMI4, a messaging solution for distributed systems, is released.

This release is mainly a bugfix release, but also introduces performance improvements and more complete timeout management.

From: Inspirel Press Center
Date: Fri, 10 Dec 2010
Subject: YAMI4-1.2.2 released
URL: http://www.inspirel.com/news.html

The YAMI4-1.2.2 version of YAMI4, a messaging solution for distributed systems, is released.

This is a bugfix and performance improvement release.

From: Inspirel Press Center
Date: Tue, 16 Dec 2010
Subject: Wireshark plugin
URL: http://www.inspirel.com/news.html

The Wireshark plugin is made available for the YAMI4 protocol.

[see also "Inspirel — YAMI4 v. 1.1.0 and 1.2.0" in AUJ 31-4 (Dec 2010), p.233 and information about Wireshark at http://www.wireshark.org —mp]

R.R. Software — New Janus/Ada beta compiler available

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 23 Dec 2010 9:12:00 +0100
Subject: [Janus/Ada 95] New Janus/Ada beta compiler available
Mailing list: ada95@rrsoftware.com

Merry Christmas!

R.R. Software is happy to announce that a new beta test of Janus/Ada is available. The beta has a new, smarter installer, support for a few Ada 2005 features, partial support for stream attributes, and (as always) lots of bugs fixed.

Beta test keys have already gone out to supported customers (if you haven't already received it, please contact me ASAP). We're looking for a limited number of additional beta testers from customers who've let their support lapse; priority will be given to the most recent ones. Please contact me at randy@rrsoftware.com if you are interested.

References to Publications

Two interviews with Tucker Taft

From: Thomas Løcke <tl@ada-dk.org>
Date: Fri, 04 Feb 2011
Subject: Two Interviews With Tucker Taft
URL: http://ada-dk.org/?page=news&news_id=261

Tucker Taft should be a name familiar to all Ada programmers, as he was the lead designer of Ada 95 and heavily involved in the Ada 2005 process.

If you have a question about Ada, the chances are very high that Tucker is going to have the answer. An interesting man for sure.

Here are two fairly recent Tucker Taft interviews:

- TechWorld: The A-Z of programming languages: Ada
- Simple Talk Geek Of The Week: Tucker Taft

I found this quote to be pretty interesting:

> One of my pet projects is designing such an inherently parallel language, tentatively dubbed ParaSail, for Parallel Specification and Implementation Language. It will attempt to incorporate the lessons learned from a career spent working on Ada, while embracing annotations and implicit parallelism in all their glory, to support the development of efficient, safe, correct, and highly scalable parallel systems. Stay tuned…

[…]

[Find the interviews at: http://www.techworld.com.au/article/223388/a-z_programming_languages_ada/? and http://www.simple-talk.com/opinion/geek-of-the-week/tucker-taft-geek-of-the-week/

The blog on the creation of the ParaSail language is at http://parasail-programming-language.blogspot.com/

See also the paper "Designing ParaSail – Parallel Specification and Implementation Language", in this issue of the AUJ —mp]

Ada Inside

Ada in automotive software

From: Rolf Ebert <rolf_ebert@users.sourceforge.net>
Date: Thu, 16 Dec 2010 12:45:05 -0600 CST
Subject: safety critical automotive software
Newsgroups: comp.lang.ada, comp.lang.c.moderated

The upcoming ISO 26262 highly recommends "enforcement of strong typing" [1]. There is a corresponding footnote saying "The objective […] is to impose principles of strong typing where these are not inherent in the language".

Does anybody know if Ada has ever been used (in ECU series production) in an automotive application. Can you provide a reference?

How do you achieve the "principles of strong typing" using C?

[1] ISO DIS 26262-6, 2009

[…]

From: Midoan <midoan.ses@gmail.com>
Date: Sat, 18 Dec 2010 02:11:16 -0800 PST
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada

[…]

The MISRA C guidelines, if they are complied with, do impose stronger typing on C code (with its restrictions on type declarations and stricter conversions rules).

So it is possible to "impose principles of strong typing where these are not inherent in the language" for C.

Of course that sentence would not be there in the standard if the MISRA C guidelines did not exist.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 24 Dec 2010 13:11:24 -0600 CST
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada,comp.lang.c.moderated

[…]

> Does anybody know if Ada has ever been used (in ECU series production) in an automotive application.

Not to my knowledge.

> How do you achieve the "principles of strong typing" using C?

AFAIK, they don't care.

Then I guess that the recommendation does not really mandate or even mean *application* of strong typing in the software design. I.e. you could be free to deploy int8, unsigned16 instead of meaningful user-defined numeric data types of domain-specific ranges and defined behavior. So "principles of strong typing" in C could simply mean treating warnings about comparing signed with unsigned ints as errors.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Fri, 24 Dec 2010 13:13:25 -0600 CST
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada,comp.lang.c.moderated

[…]

> How do you achieve the "principles of strong typing" using C?

There are tools that enforce a stronger type model on C code than the compiler does. I believe PC-Lint, for example, can produce warnings about all implicit conversions and can treat typedefs as new types. I don't believe it allows you to define range constraints, or anything else requiring run-time checking, but if used aggressively it can bring a higher degree of discipline to C code than usual.

I'm sure there are other tools that can do similar things.

From: Walter Banks <walter@bytecraft.com>
Date: Fri, 7 Jan 2011 15:40:15 -0600 CST
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada, comp.lang.c.moderated

[…] As far as I know Ada has not ever been used. I am familiar with most of the current ECU's and almost all are using the same parts implemented in C with some of the ISO/IEC 18037 additions.

> How do you achieve the "principles of strong typing" using C?

Each of the automotive companies have internal design rules that are company mandated. I am generally impressed to the degree that automotive coding standards are maintained. MISRA is often looked at as a set of guidelines but most automotive companies use MISRA as one of many sources for their internal standards.

[…]

From: Gerd <GerdM.O@t-online.de>
Date: Sun, 9 Jan 2011 09:09:54 -0800 PST
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada

I have worked in the automotive area for many years.

I never had any project that should (nor even was allowed to) be done in Ada.

Everything here is C, C and C again (C++ upcoming).

Even the new AUTOSAR standard is C oriented.

The only Ada work in automotive range that I ever heard about, was a research project at BMW for assistance system. It was presented at Ada Germany some years ago. Look here:

http://www.ada-deutschland.de/aktuelles/Tagungsprogramm.html

"S4 Fahrbetrieb und -simulation: Dickmann (BMW): Softwareentwicklung in Ada95: Ein Erfahrungsbericht"

or look here:

http://www.automotive2006.de/programm/dickmanns.pdf

But - I don't know what has happened with it.

From: Peter Hermann <h@h.de>
Date: Mon, 10 Jan 2011 14:56:22 +0000 UTC
Subject: Re: safety critical automotive software
Newsgroups: comp.lang.ada

[…]
search for Dickmanns in

http://www.ihr.uni-stuttgart.de/forschung/ada/resources_on_ada/

for another report.

Indirect Information on Ada Usage

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

Job offer [Belgium]: C/Ada developer

As a C/Ada developer you will develop a dedicated GAIA gateway for interconnection to simulators and integrate the Airbus 4D-predictor replacing the current BADA model. You will implement the new DS-C ATN data link standard and the extended i4D CPDLC message, both as defined by the EUROCAE WG78 interim standard.

Profile

As an experienced C/Ada developer you have the following profile:

- an excellent knowledge of Ada 83-95 and C programming;
- experience with technical writing, testing and data modelling;
- a good knowledge of Unix, SQL, Java, ATC, Clearcase and Linux;

[…]

- you have at least 5 years of experience in development languages.

[…]

Job offer [United Kingdom]: Ada 95 Software Engineer

ADA 95 [sic —mp], Software Engineers – Embedded Development

A Software Engineer capable of design and development of embedded and test code for software-based Defence applications.

-The role will involve integration testing with the equipment containing the software under test and suitable skills in this area will be expected.

-The development languages will be mainly ADA 95 (AdaCore)

-The specification of the design of the embedded code uses MaDGe/MASCOT, and familiarity with this approach would be desirable. Knowledge of Rhapsody UML would be useful.

-Knowledge of DOORS and the Serena Dimensions configuration management system.

Job offer [Spain]: Senior Software Engineer

[…]

- Knowledge of software development life cycle.
- Knowledge of Ada, C, C++. Knowledge of Polyspace and C# are an advantage.

[…]

Experience: 3-5 years

[translated from Spanish —mp]

Job offer [Italy]: Software Architect

Your responsibilities include the definition and verification of design specifications and interfaces of software products for use in space applications; in particular for Guidance and Control systems.

You are required to:

- define the software architecture and the development and qualification process;
- supervise the development and qualification phases;
- define interfaces with the system and with software sub-contractors;
- ensure the coherence of the various software products integrated in the same system.

Profile

A degree in Computer/Automation Engineering or Physics is required.

The successful candidate has 5-10 years of experience.

Knowledge of Ada and CASE tools is an asset.

Required knowledge:

- Software architectures
- HW/SW interfaces
- Software development process and technologies
- Verification and Validation process
- Embedded/real-time software
- On-board software

[translated from Italian —mp]

Job offer [United States of America]: Embedded Software Engineer and Software Developer

Immediate need for (2) Embedded Software Engineers, and (1) Applications Software Development Engineer.

[…]

Responsibilities

- These positions are responsible for developing, implementing, simulating and verifying Embedded System Software for aircraft systems such as Engine Control, fly-by-wire Control, and Electrical Power Generation Control for military and commercial applications

Requirements

- All positions need a BS in Electrical engineering, computer engineering, or computer science. MS or PhD preferred with consideration for higher level degrees
- Embedded Software Specialists need 10+ years experience with strong embedded programming skills, preferably in C/C++ or Ada and experience with LDRA, VectorCAST or other verification tool suites
- Application Software Development Specialist need 10+ years experience with strong programming skills in C/C++ and Java, and strong working knowledge of application development techniques and technologies, including: Object-Oriented Design, Unified Modeling and Relational Database Design and Implementation

Ada in Context

OpenSSL library for Ada

From: Riccardo Bernardini <framefritti@gmail.com>
Date: Sun, 13 Feb 2011 01:43:04 -0800 PST
Subject: An easy question: any OpenSSL library for Ada?
Newsgroups: comp.lang.ada

The subject says everything: do you know about any Ada interface to OpenSSL (or other SSL libraries)? I tried a very fast Google "Ada SSL," but I did not see anything that convinced me…

From: Per Sandberg <per.sandberg@bredband.net>
Date: Thu, 17 Feb 2011 08:32:08 +0100
Subject: Re: An easy question: any OpenSSL library for Ada?
Newsgroups: comp.lang.ada

Have a look at

http://github.com/persan/A-openssl

Just an experiment to see "how hard it could be" to do a full binding to OpenSSL.

There are still a some hours of hack to get it into a state that could be called pre-Alpha.

[…]

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sun, 13 Feb 2011 03:34:23 -0800 PST
Subject: Re: An easy question: any OpenSSL library for Ada?
Newsgroups: comp.lang.ada

[…]

AXMPP includes binding to GNUTLS, see

http://adaforge.qtada.com/cgi-bin/tracker.fcgi/axmpp/wiki

From: Riccardo Bernardini <framefritti@gmail.com>
Date: Sun, 13 Feb 2011 12:12:07 -0800 PST
Subject: Re: An easy question: any OpenSSL library for Ada?
Newsgroups: comp.lang.ada

Thank you for your reply. Unfortunately, I need DTLS (Datagram TLS, i.e., more or less, TLS for UDP) and, from a first very fast look (I searched for "DTLS," "datagram" and "UDP" in the manual page http://www.gnu.org/software/gnutls/

manual/gnutls.html) it seems that GNUTLS does not support DTLS.

According to this thread

http://www.mail-archive.com/help-gnutls@gnu.org/msg01497.html

GNUTLS did not support DTLS 1.5 years ago and nobody was working on it…

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sun, 13 Feb 2011 14:48:14 -0800 PST
Subject: Re: An easy question: any OpenSSL library for Ada?
Newsgroups: comp.lang.ada

[…] PolyORB uses OpenSSL as one of security mechanism for its security service, you can use this binding as starting point for own one.

Queue container in Ada

From: Bryan Robinson <brobinson.eng@gmail.com>
Date: Tue, 8 Feb 2011 16:43:49 -0800 PST
Subject: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

I'm writing a small Ada program for a personal project. I usually write C++/STL for such projects due to my background, but because I wanted to get more practice with Ada, I decided to do the project with Ada. All has been well, except that I've been a bit surprised about generic containers.

I was looking for a simple queue data structure, but I noticed there is not a standard Queue container. I managed to make a Vector work like a Queue, but it feels a bit awkward: Append() for a "push" and First_Element() + Delete_First() for a "pop". I realize that I can write a wrapper interface and create my own queue, but queues are so common I can't believe we have Hashed_Maps and not a simple Queue container.

I'm just curious what the reasoning was behind not providing a Queue container? Is there some Ada philosophy behind this? Or is there something similar to the C++ Standard Template Library in Ada?

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Tue, 08 Feb 2011 17:57:09 -0700
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

> […] I realize that I can write a wrapper interface and create my own queue, but queues are so common I can't believe we have Hashed_Maps and not a simple Queue container.

I agree. And given that the Ada philosophy was usually to provide building blocks that the user can combine to create more complex things, it always surprised me that the library, when it finally came into existence, requires the implementor to have implementations of hash tables and an O(log N) searchable structure, but not to make them available to the user. (Of course, implementors have always had to have unlimited-precision math libraries, but not make them available.)

Personally, I'd base a queue on the lists package rather than the vectors.

You can find queues, as well as bounded data structures in the PragmAda Reusable Components:

http://pragmada.x10hosting.com/pragmarc.htm

From: Anh Vo <anhvofrcaus@gmail.com>
Date: Tue, 8 Feb 2011 17:26:48 -0800 PST
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

[…]

Or look at http://www.adaic.org/ada-resources/tools-libraries/ for Booch components and Charles (Charles Container Library). Although, Ada 95 is the target, Booch components should be compatible with Ada 2005.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 9 Feb 2011 09:33:30 +0100
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

[…]

> I'm just curious what the reasoning was behind not providing a Queue container? Is there some Ada philosophy behind this?

I think that queue is a too specific container to be included into standard library. Depending on the number of concurrent peers accessing the queue ends, whether the queue has fixed size, whether elements are copied or referenced (e.g. statically allocated elements moved from queue to queue) the implementation and interface may sufficiently vary.

From: Simon Wright <simon@pushface.org>
Date: Wed, 09 Feb 2011 10:56:05 +0000
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

[…] If you find a case where the Booch Components aren't compatible with Ada 2005 (read, modern GNATs in -gnat05 mode!) please report a bug at

http://sourceforge.net/tracker/?group_id=135616&atid=733923 .

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Wed, 09 Feb 2011 12:10:33 -0700
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

> […] Regarding concurrent peers, are the generic containers provided in the Ada.Containers library thread-safe?

No. You can safely access different container objects from different tasks at the same time, but not concurrent access to the same object from different tasks.

From: Simon Wright <simon@pushface.org>
Date: Wed, 09 Feb 2011 20:06:26 +0000
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

> […] Regarding concurrent peers, are the generic containers provided in the Ada.Containers library thread-safe?

No. Nor are the BCs. I believe that the Ada 2012 effort includes synchronized queues (though the last I heard there was a serious problem).

Personally I think Dmitry is right. Nothing to stop you using non-thread-safe containers as part of the implementation of something appropriate to your problem.

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada

Subject: Re: Generic Containers in Ada2005
Date: Wed, 9 Feb 2011 12:56:05 -0800 PST

> […] Depending on the number of concurrent peers accessing the queue ends, whether the queue has fixed size, whether elements are copied or referenced (e.g. statically allocated elements moved from queue to queue) the implementation and interface may sufficiently vary.

Exactly the same questions can be asked with regard to vectors.

Yet, we do have vectors in the standard library.

This means that the above concerns, even though perfectly valid, do not explain the absence of queue in the library. The authors might have done exactly the same as they did with vectors - give arbitrary (but reasonable) answers to these questions and provide the solution that fits the bill in the typical case.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 9 Feb 2011 22:22:16 +0100
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

[…] Not really. Vectors are used for general purpose programming, queues largely are for systems programming. You have less people who would demand queues and their requirements would contradict each other.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 9 Feb 2011 20:03:24 -0600
Subject: Re: Generic Containers in Ada2005

Newsgroups: comp.lang.ada

> […] I was looking for a simple queue data structure, but I noticed there is not a standard Queue container.

This is added by Ada 2012. See the latest draft of the Ada 2012 standard for details:

http://www.ada-auth.org/standards/12rm/html/RM-A-18-27.html

(and the next 5 clauses as well).

As to exactly when your favorite Ada compiler adds this container, you'd have to ask them. (I believe some versions of GNAT already have versions of this container.)

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 11 Feb 2011 02:08:30 -0500
Subject: Re: Generic Containers in Ada2005
Newsgroups: comp.lang.ada

[…]

There needs to be a Reset function for Peak_Use; it's not helpful to know that sometime in the past 5 years the queue held 500 objects!

On formal packages with "others => <>"

From: ytomino <aghia05@gmail.com>
Date: Sun, 13 Feb 2011 18:46:41 -0800 PST
Subject: formal package question
Newsgroups: comp.lang.ada

Hello,

Please look this code:

```
package formalpkg is
  generic
    type T is private;
    with procedure P (X : T) is <>;
  package F is
  end F;


  generic
    with package FA is
      new F (others => <>);
  package B is
  end B;

  procedure P1 (X : Character) is null;
  package F1 is new F (Character, P1);
        -- use P => P1
  package B1 is new B (F1); -- Error !!
  procedure P (X : Character) is null;
  package F2 is new F (Character);
        -- P => P

  package B2 is new B (F2); -- OK
end formalpkg;
```

```
--------
% gnatmake formalpkg.ads
gcc -c formalpkg.ads
formalpkg.ads:17:25: actual for "P" in
actual instance does not match
formal
gnatmake: "formalpkg.ads" compilation
error
--------
```

B1 was compile error, but B2 is ok. Why? Where is my code wrong?

I want to use instances of generic package with some different subprograms like B1…

(I use gcc-4.5.1)

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Mon, 14 Feb 2011 10:59:44 +0100
Subject: Re: formal package question
Newsgroups: comp.lang.ada

[…]

I get an error for B2, probably using a different GNAT.

Not really an answer, but can you leave out the "others" in ([others =>] <>) ?

```
package Formalpkg is
  generic
    type T is private;
    with procedure P (X : T) is <>;
  package F is
  end F;

  generic
    with package FA is new F (<>);
  package B is
  end B;

  procedure P1 (X : Character) is null;
  package F1 is
      new F (Character, P1);

  package B1 is new B (F1);
  procedure P (X : Character) is null;
  package F2 is new F (Character);
  package B2 is new B (F2);
end Formalpkg;
```

From: Adam Beneschan <adam@irvine.com>
Date: Mon, 14 Feb 2011 07:38:51 -0800 PST
Subject: Re: formal package question
Newsgroups: comp.lang.ada

[…]

Yes, it compiles for me if I say F(<>) but gets an error with F(others => <>).

There should be no difference between the two syntaxes, so this is an error in the compiler.

From: ytomino <aghia05@gmail.com>
Date: Mon, 14 Feb 2011 16:31:20 -0800 PST
Subject: Re: formal package question
Newsgroups: comp.lang.ada

I tried to remove "others =>", and got same result that it compiled correctly. umm…

This behavior of the compiler limits using formal parameter…

I want to write partial parameters and use defaults for remaining parameters.

But gcc(GNAT) can compile the example in

http://www.adaic.org/resources/add_content/standards/05rm/html/RM-12-7.html

I deduce, this behavior is applied only to generic package having formal package having formal subprograms ???

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 15 Feb 2011 00:17:01 -0800 PST
Subject: Re: formal package question
Newsgroups: comp.lang.ada

[…]

I reported this bug as http://gcc.gnu.org/PR47748 for you.

Check for input validity in Ada

From: KK6GM <mjsilva@scriptoriumdesigns.com>
Date: Fri, 4 Feb 2011 08:33:57 -0800 PST
Subject: What would be the Ada solution?
Newsgroups: comp.lang.ada

A colleague is working on some old code, written in C, that uses an "out of range" integer value to indicate no valid value. Thus, a default value (in this case 0x7FFFFFFF) means no value has been entered. All code that uses any values should check for this no-value and act accordingly, but of course not all the code actually does that, and there are odd cases where the no-value value gets processed as a valid value and then Bad Things Happen.

I'm curious what the Ada approach to this issue would be, the issue being to differentiate between valid and invalid values, and to catch (compile time or run time) any manipulation of an invalid value as if it were a valid value.

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Fri, 4 Feb 2011 08:38:13 -0800 PST
Subject: Re: What would be the Ada solution?
Newsgroups: comp.lang.ada

[…]

I guess something like this:

```
type T is range ...;
type Opt_T (Is_Valid : Boolean :=
False) is
  record
    case Is_Valid is
      when True =>
        Value : T;
      when False =>
        null;
    end case;
  end record;
```

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Fri, 04 Feb 2011 19:59:19 +0100
Subject: Re: What would be the Ada solution?
Newsgroups: comp.lang.ada

[…]

> I suggest you read these two "gems":

http://www.adacore.com/2010/03/22/gem-82/

http://www.adacore.com/2010/04/05/gem-83/

[Also available in the Ada Gems section in AUJ 31-3 (Sep 2010), p.217—mp]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de
Date: Fri, 4 Feb 2011 18:26:42 +0100
Subject: Re: What would be the Ada solution?
Newsgroups: comp.lang.ada

[…]

An Ada solution would be to declare the integer type of the valid range:

```
type ADC_16_Bit is range 0..2**16 - 1;
```

The compiler checks dynamically and, where possible, statically that the value is always valid.

When values read from the hardware include some bit patterns, that may indicate errors.

In such cases you can declare the full range of possible values and a subrange of the valid ones:

```
-- 2 octets as read from the station
type ADC_Word is mod 2**16;

Conversion_Error :
  constant ADC_Word := 16#FFFF#;
Short_Circuit_Error :
  constant ADC_Word := 16#EFFF#;
…
subtype Voltage is
  ADC_Word range 0..16#7FFF#;
  -- 0=-10V, 7FFF=+10V
```

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Fri, 04 Feb 2011 12:38:34 -0500
Subject: Re: What would be the Ada solution?
Newsgroups: comp.lang.ada

[…]

And another problem is that you probably have lots of unnecessary/redundant checks for the no-value case. That's "defensive programming", which Bertrand Meyer wisely said is a bad idea. But it's sort of necessary in C, because it's hard to keep track of which code actually needs the checks -- better to clutter the code with redundant junk than to miss some.

There are lots of ways to represent the "no valid value" value in Ada.

You suggested a magic number. Mark Lorenzen suggested a variant record.

Let's go with the magic number:

```
No_Value : constant := 2**31 - 1;
pragma Assert (No_Value =
                  16#7FFF_FFFF#);
```

You don't want to check for No_Value all over the place.

You want to check for it on input, and let the rest of your program operate on valid values, and be sure that No_Value doesn't sneak into the rest of the program.

You could do something like this:

```
type Optional_T is range -2**31 ..
No_Value with
   Default_Value => No_Value;
```

That means every object of subtype Optional_T will be initialized to No_Value, unless you explicitly initialize it to something else.

```
type T is new Optional_T with
   Static_Predicate => T /= No_Value;
```

That means every object of subtype T will be checked that's it's not No_Value, on assignment, parameter passing, etc. And T inherits the Default_Value.

You input values of type Optional_T. You then check whether they are valid, and convert them to type T. The rest of your program is written in terms of type T. If you try to pass an Optional_T to a parameter of type T, you get a compile time error.

You can convert:

```
procedure P(X : T);
Possibly_Invalid :
  Optional_T := Read_From_Input(...);
P (T (Possibly_Invalid));
  -- Might raise an exception
```

but the conversion will raise an exception if the value is No_Value.

If you declare an uninitialized variable of type T, you will get an exception, because the default value violates the predicate. If you're lucky, your compiler will warn about that at compile time.

Your objects (including parameters, record components, etc) are of type T or Optional_T, so it's easy to tell which ones are safe. Hopefully, there are few Optional_T objects, and few places where you convert, so you can be careful about those.

Depending on what you're trying to do, you might want to make one or both types private.

If the run-time checks are too slow (which is unlikely!), you can always suppress them, and get efficiency equal to the C code.

Default_Value and Static_Predicate are Ada 2012 features.

Both are implemented in the latest version of GNAT, but I don't know about the public version(s).

You can do more or less the same thing in Ada 2005, except that to get a default value, you have to wrap the thing in a record, and you can use range constraints instead of predicates.

You could probably do more-or-less the same thing in C, but I think it would take a fair amount of horsing around.

Predicates are my favorite feature of Ada 2012!

And user-defined integer types are one of my favorite features of Ada 83.

On the use of user-defined storage pools

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 22 Jan 2011 09:49:10 +0100
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

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

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

[…]

Storage pools are extremely useful for purposes, which are not directly related to memory allocation. If you take a look at the example I provided, you will notice that the storage pools there do not maintain a heap of their own. They rather take the memory from some backend pool.

Consider the doubly-link list implementation. It is storage pool based. The links are not in the element body. They are maintained by the pool. So you can have any type as the list element type, plain strings for example.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Sat, 22 Jan 2011 01:54:01 -0800 PST
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

We use storage pools on a helicopter flight simulator. All global objects are elements of one storage pool. Thus we can take snapshots of a current state by simply storing away the pool contents. Later, by restoring the pool contents, a previous situation, e.g. where a pilot did not perform correctly, can be restored and the pilot can try to do better.

From: Simon Wright <simon@pushface.org>
Date: Sat, 22 Jan 2011 15:13:40 +0000
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

The Booch Components (https://sourceforge.net/projects/booch95/) have always required users to provide storage pools (except for Bounded forms). I expect that most people who use(d) the BCs use(d) the one provided, which is a wrapper for the system storage pool.

However, while developing code on Windows for use in VxWorks, we found that the Windows allocator would zero-fill, while the VxWorks allocator did no initialization (which is perfectly legal, of course). The result was that uninitialized record components would appear to work on Windows and crash mysteriously on the target.

What we did to work round this (which was before GNAT offered pragma Initialize_Scalars and -gnatV) was to provide a specialised storage pool which fills the requested memory with 16#deadbeef#; so any uninitialized fields are (a) reasonably unlikely to correspond to reasonable values, (b) reasonably easy to spot in a store dump, (c) wrong on either OS.

From: Simon Wright <simon@pushface.org>
Date: Mon, 24 Jan 2011 16:09:58 +0000
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

> *Required* That's interesting […]

The rationale, such as it is, is pretty much lifted from the C++ BCs;

http://booch95.sourceforge.net/documentation.html#storage

I got a lot of complaints about the need for multiple levels of instantiation, lack of support for indefinite types, need for storage pools which I resisted on the grounds that the BCs were a translation. In 2005 Martin Krischik added support for indefinite types for some containers, and I have been toying with removing the need for storage pools (by providing BC.Indefinite_Unmanaged_Containers, see http://booch95.svn.sourceforge.net/viewvc/booch95/trunk/src/ about 2/3 of the way down).

Of course nowadays you'd use Ada.Containers, but there are still people using Ada95 out there…

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 24 Jan 2011 01:25:29 -0800 PST
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

At Eurocontrol, we use two custom storage pools besides the default one.

We have a "No-Dealloc" storage pool that raises an exception if the software attempts to deallocate from it. More importantly, this pool works in two modes: initially it accepts allocation and writing of new objects in the pool (but, of course, no deallocation), then it switches to read-only mode by calling mprotect(2) on all memory pages.

Any subsequent attempt to alter data in this pool will crash the program. We use this pool to parse a large database of immutable objects into memory; objects have lots of access values referencing one another, so we have to be able to alter references while parsing.

We switch to read-only mode when the parsing is complete.

The other custom storage pool we use is to detect memory leaks. We do that by preprocessing our sources when building in a special leak- detection mode.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Mon, 24 Jan 2011 05:43:09 -0800 PST
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

>> The other custom storage pool we use is to detect memory leaks. We do that by preprocessing our sources when building in a special leak-detection mode. […]

See the package GNAT.Debug_Pools and its documentation for more information.

From: Dirk Craeynest <dirk@vana.cs.kuleuven.be>
Date: Mon, 24 Jan 2011 16:48:24 +0000 UTC
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

For a paper co-authored by AdaCore and Eurocontrol about the introduction of that package, see:

http://www.cs.kuleuven.be/~dirk/papers/ae03cfmu-paper.pdf

It is titled "Exposing Memory Corruption and Finding Leaks: Advanced Mechanisms in Ada" and was presented at the Ada-Europe 2003 conference.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon, 24 Jan 2011 12:46:53 +0100
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

I use storage pools to implement persistent objects. I presented a paper on the subject at Ada Europe 2010. There is some demonstration source code at

http://www.jacob-sparre.dk/persistence/

My technique is depends on a well-behaved memory mapping implementation from your operating system. Especially memory layout randomization makes things more tricky than I like.

From: Timo Warns <Timo.Warns@Informatik.Uni-Oldenburg.DE>
Date: 24 Jan 2011 14:04:30 GMT
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

As an example, Ada Gem #77 (http://www.adacore.com/2010/01/11/gem-77/) shows how to use the GNAT debug storage pool for analyzing the memory usage of a program.

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 25 Jan 2011 01:14:30 +0100
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

> Because storage pools provide no means to check dereferences, GNAT offers a special type of storage pool, called a "Checked pool", with an additional abstract primitive operation called Dereference.

That's an interesting feature indeed, but not standard. How many Ada compiler vendors provides a similar feature?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 24 Jan 2011 19:36:27 -0600
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

Can't answer that, but we did consider adding such a feature to Ada 2012.

Eventually we decided on a more general purpose user-defined dereferencing capability instead.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 7 Feb 2011 16:18:51 -0600
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?

Newsgroups: comp.lang.ada

[…]

> You mean this is already part of the actual or next to come Ada 2012 ?

We expect it to be, although it isn't finished and it will have to be finished by the end of this month (as with all work on Ada 2012) in order to be included.

See AI05-0139-2. The "Reference" aspect combines with access discriminants and controlled types to give effectively a user-defined dereferencing mechanism.

The primary problem with user-defined dereferencing is controlling the lifetime of the returned access value -- we don't want it copied outside of the control of the underlying container abstraction. In addition, it is important that the container be able to get control twice: once when the dereference is created, and once when it is destroyed. The latter is needed in cases of persistence or other kinds of locking. (In the case of persistence, the object must be available in memory so long as a reference to it is valid, but it can be pushed back to the backing store once the references are all gone. Another example is a the tampering check of the Ada containers, which prevent the object of the reference from disappearing while the reference exists.)

It turns out that the mechanism needed already exists in the language, in the form of access discriminants. In addition, the object containing the access discriminant can be controlled, thus giving control upon destruction of the object (and the reference). So the only problem is the terrible syntax of such a dereference. The Reference aspect allows us to eliminate that.

Combined with the indexing aspect, you'll be able to write something like:

```
Foo(1).Bar := 10;
```

For a vector container Foo whose elements are a record type with a component Bar. And even:

```
Text_Map ("Ada").Count := 1;
```

For an indefinite map.

From: Warren W. Gay <ve3wwg@gmail.com>
Date: Wed, 26 Jan 2011 20:13:29 +0000 UTC
Subject: Re: User Defined Storage Pool : did you ever experiment with it ?
Newsgroups: comp.lang.ada

[…]

[A user-defined storage pool —mp] can be useful when you want to preallocate a maximum amount of allocatable memory to a class of (by usage) objects.

Emulator or simulators use this for safety. This avoids an errant emulation from allocating all of user memory and trashing the emulator's own environment by thrashing/swapping all available user memory.

Another useful application is permitting objects of a particular class (of use) to be allocated from a pool. Then at a clearly defined point in the code, *all* of those objects can be efficiently released by freeing the entire pool. No list traversal (or list maintenance) is required.

On loops and parallel execution

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 25 Jan 2011 16:40:38 +0100
Subject: Loops and parallel execution
Newsgroups: comp.lang.ada

A quick idea. Assume that some subprogram Op from package P is reentrant (and does not depend on global state). Then,

```
with P;
...
  for K in all First .. Last loop
     P.Op (K);
  end loop;
```

should have the effect of the following being permitted:

(a) to pick K from First .. Last in any order

(b) to execute P (J) in parallel with P (K) for J, K from First .. Last

The same would be allowed for sufficiently simple expressions:

```
  for K in all First .. Last loop
     L(K) := Standard."*" (K, 3);
  end loop;
```

Can this be borrowed from HPF [High-performance Fortran —mp] (IIUC)? Is pragma Pure (P) sufficient to signal reentrance?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 25 Jan 2011 17:37:03 +0100
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

[…]

No, it is not sufficient because it is wrong. P cannot be pure because all instances of P.Op must be synchronized at the end of the "loop." You need some frame, context relatively to which P might become pure. "Embedded task", thread, fiber, call it as you want. If you have that at the language level, then it becomes no matter whether the thing executed on such a context is pure or not. This is similar to proper Ada tasks, you can access shared data from a task as you wish. If you do this inconsistently that is your problem (erroneous execution).

The point is, if you are going to somehow derive concurrency stuff from a sequentially written program using pragmas and a "mind reading" compiler, I doubt that could go anywhere. If you want to add light-weight embedded in code tasking constructs a-la Occam, that might go, but I don't think that they could be much useful. You need to map them onto OS services in order to gain something, because normally there is no direct access to the cores.

That is not light-weight. Have you some certain OS in mind?

This thing you wanted in present Ada:

```
  task type Worker (
     Do_Me : not null access
     procedure (K : Integer)) is
    entry Op (K : Integer);
  end Worker;
  task body Worker is
    I : Integer;
  begin
    accept Op (K : Integer) do
      I := K;
    end Op;
    Do_Me (I);
  end Worker;

  procedure Print (K : Integer) is
  begin
    Put_Line (Integer'Image (K));
  end Print;
…
  declare
    Threads : array (1..20) of
            Worker (Print'Access);
  begin
    for K in Threads'Range loop
      Threads (K).Op (K);
    end loop;
  end;
```

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 25 Jan 2011 18:36:29 +0100
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

> […] No, it is not sufficient because it is wrong. P cannot be pure because all instances of P.Op must be synchronized at the end of the "loop."

I was thinking of relatively small things, starting from this observation:

If a compiler detects two independent paths inside a loop, it will generate sequences of instructions that, while written sequentially, will be executed in parallel by the processor because the processor can do the corresponding distribution across the many register words. Generalize that.

The Paraffin library just posted by Brad Moore looks like it will resolve all other issues nicely! :-)

[read the related news in this issue of the AUJ —mp]

On an AS-IF basis, I thought of a tasking profile much < Ravenscar:

Other than exiting with a value, each P.Op (K) has no need for communication. Each P.Op (K) is associated with a result object, like this

```
compiler_type Result_Object is
   limited record
   Ready : Boolean_Slot;  -- a CAS
                             bit?
   Result : Value_Type;
end record;
```

The idea of employing Pure or something similar was to have the compiler prevent each P.Op (K) from meddling in P.Op (J)'s affairs, J /= K, as with protected objects, again on an AS-IF basis, as much as possible.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 25 Jan 2011 22:32:57 +0100
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

[…]

> If a compiler detects two independent paths inside a loop,

I don't see any application for this. Can you remember the last time you wrote such loop? I cannot.

The Occam's par-statement could be a better candidate, but I don't see how this could be useful under a modern general-purpose OS with their "vertical" parallelism, when each task is assigned to one core. The thing you propose is "horizontal" parallelism, when a task/process would run on all cores simultaneously. Inmos' Occam ran under no true OS, and the processor architecture was well suited for such ad-hoc parallelism. Modern processors are very different from T805 and I doubt that they would allow an efficient implementation of this.

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Tue, 25 Jan 2011 23:07:01 +0100
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

[…]

In fact, I have seen such a loop recently; it computes a Mandelbrot set twice as fast. (I am confident that the lessons learned in finding this loop have found applications in other loops that manipulate larger amounts of numeric data.) The author has found a way to split Ada's Complex type into its constituent parts (two FPT objects) such that the program is a lot more efficient. (One would wish that types like Complex would be treated specially by the compiler.)

> […] Modern processors are very different from T805 and I doubt that they would allow an efficient implementation of this.

I have recently seen small boards carrying one processor each that could be connected to one another on all sides, IIRC. What matters then is, I think, the efficiency of

(a) distribution of small computation, and

(b) the delivery of results at some nodes.

Is it therefore so unthinkable to have something like a transputer these days?

[…]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 26 Jan 2011 10:04:07 +0100
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

On Tue, 25 Jan 2011 23:07:01 +0100, Georg Bauhaus wrote:

> […] I have recently seen small boards carrying one processor each that could be connected to one another on all sides, IIRC […]

The Parix OS (actually a monitor) did that. E.g. if you called, say, "printf" in a node which didn't have a direct link to the server (the server was an MS-DOS PC or a Solaris workstation), the output would be routed to the node connected to the server and from there to the server which printed the output.

> Is it therefore so unthinkable to have something like a transputer these days?

I saw them too. BTW, they are in some sense a step back comparing to the level Inmos arrived before its fall. They introduced a programmable TP-link switch, so that you could reconnect the network of transputers on the fly.

But the problem is. I really see no use for the par-statement or alike. The main argument against par is that using threads causes to much overhead. If the argument stands, I mean if you don't have very long code alternatives running in parallel for seconds, then using a mesh of processors would make things only worse. The overhead to distribute the code and data over the mesh of processors is much bigger than doing this on a machine with shared memory (multi-core). There certainly exist examples of long independent code alternatives, but I would say that most of them are constructed or marginal.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Wed, 26 Jan 2011 06:29:02 -0500
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

[…]

I've often wondered what it would take to support OpenMP (or something like it) in Ada. The advantage with such an approach is that OpenMP is well documented and widely used and understood. Right now the OpenMP standard only supports C (and C++?) and Fortran. Why not Ada?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 26 Jan 2011 15:57:12 -0600
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

> […] Is pragma Pure (P) sufficient to signal reentrance?

I've thought about such an idea. But it requires restrictions well beyond those enforced by pragma Pure. For instance, Pure packages can write dereferences of pointers to keep global state. Moreover, there can't be any "cheating", which is common in pragma Pure packages.

So there would need to be a new kind of categorization for this. I was hoping that we could using the proposed global in/global out categorizations to do the job, but those got dropped from Ada 2012.

Also, I think that "no communication" is impractical in most real applications. But it is sufficient if the communication is tightly limited (via atomic and protected objects, and/or synchronized interfaces - you'll need to access global data, just safely). That's another reason why "checked global in/global out" is needed.

Finally, like Dmitry, I'm skeptical about fine-grained parallelism buying much. Unless there is specific architectural support (something that doesn't exist in commonly used processors -- and especially in commonly used target OSes/RTOSes), the management overhead will kill any savings on "small" expressions. Thread creation is not cheap! The "win" is on larger tasks - which means that subprograms - and separately compiled subprograms - have to be involved in some way.

My main interest in this technology is to make it much easier to create programs that use threads but don't deadlock, livelock, or have dangerous use of globals. That seems to require restrictions on what you can do, and definitely requires some form of compile-time checking to enforce those restrictions. If done usefully, that could be a giant win, as you could use sequential reasoning for the majority of your programming and debugging, and still get parallelism when useful.

From: Tom Moran <tmoran@acm.org>
Date: Thu, 27 Jan 2011 23:01:54 +0000 UTC
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

> Finally, like Dmitry, I'm skeptical about fine-grained parallelism buying much. Unless there is specific architectural support (something that doesn't exist in commonly used

processors -- and especially in commonly used target OSes/RTOSes), the management overhead will kill any savings on "small"

What about the SIMD (vector) instructions in Intel CPUs? Or is that better done by simply calling their optimized, CPU capability detecting, libraries?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 28 Jan 2011 18:23:13 -0600
Subject: Re: Loops and parallel execution
Newsgroups: comp.lang.ada

[…]

That's a code generation problem; I don't believe that there is much if any value to the programmer cluttering their code with parallel operations in that point.

To expand on that a bit: code generation for a CISC machine is primarily a pattern matching problem. That is, the intermediate code is a list of very simple pseudo instructions, and the code generator needs to map those to more complex machine instructions (along with simple ones when the pattern matching fails). Matching SIMD instructions is a more complex problem than the simple matcher used in Janus/Ada (to take the example I'm most familiar with), but it is fundamentally the same problem. In this case, I would probably apply a loop unrolling optimization, then a series of pattern matching operations to create the SIMD instructions.

We already do something like this for aggregates in Janus/Ada. An aggregate assignment like:

```
My_Str :=  (others => Ch)
```

can get turned into the Intel STOSB (I think that's the right opcode) instruction (plus a bit of setup code); which is a lot simpler than the loop that would be otherwise generated.

In either case, you'll automatically get the benefit of the advanced instructions when they can be used, and no code changes are needed. Of course, if your code doesn't match the pattern, the advanced instructions wouldn't be used, but it's unlikely that adding a "parallel" direction to the loop would somehow change that.

I'd be surprised if GCC doesn't already do something like this. (This particular problem hasn't been on my radar, in part because I didn't even have a machine that supported most of those instructions until last year.)

Encapsulation of Ada.DirectIO

From: Bryan Robinson <brobinson.eng@gmail.com>
Date: Tue, 16 Nov 2010 20:44:49 -0800 PST
Subject: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

I'm trying to port some code to Ada for dealing with Big5-encoded files. I realize that I might be able to use Ada.Wide_Text_IO, but I'm trying to learn Ada and understand the language better. I'm still working on wrapping my head around types and packages. My original code in C++ opens a file as binary and parses it byte by byte and breaking it into Big5 characters depending on the byte codes. I thought I'd try to do something similar by encapsulating Ada.Direct_IO into a package. I'm not having much luck, however.

Spec file:

```
with Ada.Direct_IO;
package Big5_Text_IO is
  type File_Type is limited private;
  procedure Close(
       File : in out File_Type );
private
  package Byte_IO is
       new Ada.Direct_IO(Character);
  type File_Type
       is new Byte_IO.File_Type;
end Big5_Text_IO;
```

Body file:

```
package body Big5_Text_IO is
  procedure Close(
       File : in out File_Type ) is
  begin
      Byte_IO.Close(File);
  end Close;
end Big5_Text_IO;
```

Test driver:

```
with Big5_Text_IO;
with Ada.Text_IO;
procedure Big5_Test is
  Input_File : Big5_Text_IO.File_Type;
begin
  Ada.Text_IO.Put_Line("OK?");
end Big5_Test;
```

If I leave out the Close method and remove the body file, I can build the test driver with no issues. Otherwise, I get the following from GNAT:

```
gcc -c big5_text_io.adb
big5_text_io.adb:6:23: expected private
type "Ada.Direct_Io.File_Type"
from instance at big5_text_io.ads:11
```

I would greatly appreciate any tips in how I can better design my package so that it can encapsulate Ada.Direct_IO or some other method of binary I/O. I looked at the GNAT source and I'm hoping I won't have to emulate what they have done…its a bit over my head at this point.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 16 Nov 2010 21:20:05 -0800 PST
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

You're close. Try changing

```
Byte_IO.Close (File);
```

to

```
Byte_IO.Close (
  Byte_IO.File_Type (File));
```

When you declare a derived type "type T2 is new T1", then T2 and T1 are not the same type, so you can't use an object of type T2 where something of type T1 is expected. But you can use a type conversion.

Note: I'm at home so I can't try this easily. I seem to recall that there were some issues using this paradigm with limited types (including an incompatibility with earlier versions of the language), but I don't recall the details and it's hard for me to look them up right now. If it turns out the type conversion doesn't work, then you might have to make File_Type a record in the private part:

```
type File_Type is record
  F : Byte_IO.File_Type;
end record;
```

and then use File.F whenever you want to use a Byte_IO operation, e.g.:

```
Byte_IO.Close (File.F);
```

Hope this helps,

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Wed, 17 Nov 2010 07:25:18 -0500
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

Direct_IO allows random access. There is nothing wrong with that, of course, but if your intention is to read the file sequentially you might prefer using Sequential_IO.

Something I wonder about (I don't have the answer) is if it necessary to use a representation clause to force the size of the objects being read to be 8 bits. I'm a little unclear if the standard requires Character to be stored in a file in 8 bit units. That is, the language might treat the type Character rather more abstractly than you want. Again I'm not sure of this and I'd love to get some insights from others myself.

Thus you might want to do something like

```
package Big5_Text_IO is
  …
  type Byte is mod 2**8;
  for Byte'Size use 8;
  …
```

```
private
  package Byte_IO is
      new Ada.Direct_IO(Byte);
  …
end Big5_Text_IO;
```

This approach has the advantage of creating a separate type to represent the raw data from the file. Thus

```
C : Character;
B : Big5_Text_IO.Byte;
C := B;  -- Compile error.
            You haven't decoded B yet.
```

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 17 Nov 2010 19:16:07 -0600
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

Surely not. Not all machines have 8-bits as any sort of native type. For instance, the Unisys U2200 (a 36-bit machine, with 9-bit bytes) used Character'Size = 9. (It was great fun for the cross-compiler.)

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 17 Nov 2010 23:39:13 -0800 PST
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

Huh. How then is Character defined there? According to RM A.1(35), Character has 256 positions, so Character'Size should be still 8. Of course stand-alone objects would have X'Size = 9.

Note that Natural'Size = Integer'Size - 1.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 18 Nov 2010 12:38:44 -0600
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

I suppose you are right, but Ada 95 Type'Size has no important meaning.

(It's a terrible definition, IMHO.) What matters is what AdaCore calls Type'Object_Size, and that is what I was referring to. (Typically, specifying Type'Size will have some effect on Type'Object_Size, but exactly what that is will vary depending on the target.)

You could, I suppose, have packed characters into 8-bits in an array, but the code to access them would have been unspeakably bad. And there would have been no reason to do so anyway, since files and streams are automatically converted when crossing into that machine's domain.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Wed, 17 Nov 2010 21:21:34 -0500
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

> Surely not. Not all machines have 8-bits as any sort of native type. So in that case if you absolutely wanted to read 8 bit units from a file (because the file is in some externally defined binary format that uses 8 bit units) it would be necessary to do something like:

```
type My_Byte is mod 2**8;
for My_Byte'Size use 8;   -- This is
                             important.

package My_Byte_IO is
   new Ada.Sequential_IO(My_Byte);
```

… and then convert from My_Byte to Character only as appropriate during the file decoding process. True?

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 18 Nov 2010 08:36:56 -0800 PST
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

I have a longish rant that touches on this, but the short answer is that I don't think there's an Ada answer to your question. If you had a file defined as using 8-bit units, and that file got put onto a system that uses 36-bit words, you'd need to know just what the OS is going to do with it, and how the particular Ada implementation will deal with files on that OS. It may be that the native "read from file" service on that OS will put each byte into a 9-bit byte and zero the high bit. I don't see how to avoid implementation-dependent code in a case like this.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Thu, 18 Nov 2010 13:21:57 -0500
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

I understand what you are saying but it is less than satisfying. :)

I'm thinking about the very common case when one is trying to read a file that has a format defined by some third party. For example the specification of the format might say, "The first octet of the header defines the message type and can be one of the following values… The type field is followed by a 24 bit length field in big endian form. The body of the message follows the length field, and finally a 32 bit CRC follows the message body."

I want to write an Ada program that can read in a file like this and process it. Are you saying that it's impossible to write such a program in a portable manner?

What I've been doing is as I showed earlier… define my own "Byte" type with Byte'Size set to 8 and then instantiate Sequential_IO. My program then interprets the individual bytes as necessary. It seems to work with GNAT.

I suppose that C has the same issue, really. The C standard does not promise that char is exactly 8 bits. If it isn't I'm not sure what happens when you do

```
int Ch;
while ((Ch = fgetc(infile)) != EOF ) { … }
```

I guess that's the same point you are making. Maybe the C standard talks about this issue. Checking…

I just took a quick look at C99's description of fgetc and it says, "the fgetc function returns the next character from the input stream." That seems to beg the question, doesn't it?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 18 Nov 2010 12:36:00 -0600
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

Adam is right. What we did is the same thing as the C compiler.

When you imported files from the "real world" into that system, it added a zero bit to every byte. So there normally would not be any such thing as a file with 8-bit characters.The same thing happened to sockets.

The real trouble came when you did the reverse, as it then *dropped* the high bit if it thought that the files were text files. That would be bad news for truly binary files.

Ada can only define things portably that it has control over. If you read and write files on the same machine, then the results should be portable. Once you start communicating to other machines, there can be translation layers that mess things up.

The good new is that it is pretty rare that you will have to deal with any machines that have other than 8-bit bytes these days. So I wouldn't worry about those issues unless you happen to be working with Unisys. ;-)

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 18 Nov 2010 11:48:49 -0800 PST
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

> I'm thinking about the very common case when one is trying to read a file that has a format defined by some third party. […]

The problem is that this *definition* is not sufficient to tell you what an OS will stick in your memory buffer if you ask to read from such a file. You need additional OS-dependent information.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 18 Nov 2010 21:15:35 +0100
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

I don't think so. I presume that the context of the definition above is the OS. It is a reasonable presumption because files from another OS (file system) cannot be read at all unless they are converted into the format supported by the OS's file system. Another presumption is that the definition describes memory layout, rather than the media layout. The latter is inaccessible anyway. So a stream of octets is what a file reading OS service delivers, when Ada would call it. There of course is no guaranty that Direct_IO would use this service and not some other service.

> You need additional OS-dependent information.

The Ada compiler vendor will likely document the services used for Direct_IO, but there is no way to verify that using the representation clauses and/or assertions.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 18 Nov 2010 08:31:20 -0800 PST
Subject: Re: Encapsulating Ada.Direct_IO
Newsgroups: comp.lang.ada

[…]

What I'm leading up to, though, is that I think Peter's question is too simple. We're all spoiled in having to deal exclusively, or almost exclusively, with machines with 8-bit byte addressability and files that are unstructured sequences of 8-bit bytes. But there are other systems out there. There are machines in use that are not byte-addressable---Analog Devices 21060 series comes to mind, which uses 32-bit words whose bytes are not individually addressable. When reading from a file on that system, do you want each word to hold one byte, or four?

Even in VAX/VMS, which does run on a machine with 8-bit byte addressability, the OS is able to create files that have more structure than just being sequences of bytes. What would it mean to instantiate Direct_IO(Character) on a file like that? I don't think the answer is trivial. How did the Pick operating system treat files conceptually? How would Direct_IO work on one of that system's files?

Ada's designers have tried to design a language that could work on any of those systems, and therefore I think the standard does not and cannot answer Peter's question. In fact, I'm not entirely sure that his question is meaningful on platforms that don't use 8-bit bytes and/ or use files with some structure. (It might have to be rephrased.)

In any event, I think that details like this are left up to the implementation. And if you were trying to do something like this on a U2200, there is no Ada answer to the question, because you would have to know more about the particular OS's file system and how the Ada implementation interacts with it.

Stack overflow with arithmetic operators and large arrays

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Fri, 3 Dec 2010 22:32:59 -0800 PST
Subject: Large arrays passed to arithmetic operators overflows GNAT stack
Newsgroups: comp.lang.ada

For what it's worth, passing large numerical arrays to the arithmetic operators in GNAT can cause the stack to overflow (segfault) even when the arrays are allocated from the heap. I suppose that indicates that they are being copied rather than being passed by reference.

For example, on OS X which has a default stack size of 8192 KB, the following program segfaults when N is greater than about 1_048_138 (about 8 MB per Long_Float array) but runs OK when it is somewhat less than that number.

```
with
   Ada.Numerics.Long_Real_Arrays;
use
   Ada.Numerics.Long_Real_Arrays;
procedure array_test is
   type Real_Vector_Access
         is access Real_Vector;
   N : Integer := 1_048_130;
   t_Ptr : Real_Vector_Access :=
         new Real_Vector(0 .. N);
    t : Real_Vector renames t_Ptr.all;
   t_Diff_Ptr : Real_Vector_Access :=
         new Real_Vector(0 .. N - 1);
    t_Diff : Real_Vector
         renames t_Diff_Ptr.all;
begin
   for i in t'range loop
      t(i) := 1.0;
   end loop;
   t_Diff := t(1 .. N) - t(0 .. N - 1);
end array_test;
```

The quick fix (in my case) is to increase the stack size from the shell:

```
ulimit -s 65532
```

or whatever is appropriate for your machine--65532 is the hard limit set by OS X.

Another way to expand the stack without invoking shell commands, as noted by Björn Persson on this thread

http://groups.google.com/group/comp.lang.ada/browse_thread/thread/ae395e5c1=1de7bc9/bda8d61bd3a66ee9?hl=3Den&q=3DJerry+stack&lnk=3Dnl&

is to call getrlimit and/or setrlimit from within the program, linking to these POSIX C routines.

Another fix would be to write operator procedures that specifically take pointers as arguments; that would possibly be the only fix for arrays that still overflow the stack when it is maxed out.

As usual, taking the dumb-guy approach, this seems like an unnecessary nuisance.

From: Vinzent Hoefler
Date: Sat, 04 Dec 2010 10:19:47 +0100
Subject: Re: Large arrays passed to arithmetic operators overflows GNAT stack
Newsgroups: comp.lang.ada

[…]

It's probably not the operator call that crashes, but the creation of the temporary arrays t(…). Couple of days ago, I stumbled upon a similar problem with aggegrate assignments:

- Do array initialization in a loop.
- If it's done with an aggregate assignment, GNAT raises Storage_Error during elaboration.

> Another fix would be to write operator procedures that specifically take pointers as arguments; that would possibly be the only fix for arrays that still overflow the stack when it is maxed out.

This wouldn't help here, I suppose.

From: Jeffrey Creem <jeff@thecreems.com>
Date: Sat, 04 Dec 2010 08:27:30 -0500
Subject: Re: Large arrays passed to arithmetic operators overflows GNAT stack
Newsgroups: comp.lang.ada

[…]

In general, GNAT (and most other compilers) don't copy arguments like large arrays on the parameters to subprograms. The problem here is almost certainly in the return value associated with the subtraction. I am not sure I'd say it is impossible but I don't think I have seen any Ada implementation that can avoid the creation of the temporary based on the internally defined local variable that is generally required to construct code the returns an array.

Even if certain cases could be found by the optimizer and made to work, there are all sorts of cases where users would end up being surprised when temporaries had to be created.

In cases where array slices, assignment and the math involved 'destroy' the future values as the current values are being read, the compiler would have no choice (given the semantics of the operator) but to introduce some temporaries and it would be quite a feat for a compiler to figure out how much of a mini-slice temp it would need.

I really think it is a shame that Generic_Real_Arrays was defined the way that it was as the pretty code one gets when being able to use infix notation is nice but the overhead involved in the resulting copies renders the operators useless for all but the smallest of array vectors.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Sat, 04 Dec 2010 09:17:08 -0500
Subject: Re: Large arrays passed to arithmetic operators overflows GNAT stack
Newsgroups: comp.lang.ada

[…]

```
> with
    Ada.Numerics.Long_Real_Arrays;
  use
    Ada.Numerics.Long_Real_Arrays;
  procedure array_test is
     type Real_Vector_Access
        is access Real_Vector;
     N : Integer := 1_048_130;
     t_Ptr : Real_Vector_Access :=
        new Real_Vector(0 .. N);
     t : Real_Vector renames t_Ptr.all;
```

I don't think you need this renaming. If you do t_Ptr'Range or t_Ptr(i) the compiler will forward those operations directly to the object.

```
>    t_Diff_Ptr : Real_Vector_Access :=
        new Real_Vector(0 .. N - 1);
     t_Diff : Real_Vector
        renames t_Diff_Ptr.all;
  begin
    for i in t'range loop
       t(i) := 1.0;
    end loop;
    t_Diff := t(1 .. N) - t(0 .. N - 1);
```

I believe a temporary object (on the stack) needs to be created here to hold the result of the difference before it gets assigned to t_Diff. So despite the fact that the result is ultimately stored on the heap it ends up passing through the stack on its way. If I'm correct it's not the subtraction operator itself that is causing the problem but rather what is happening with the result.

In short "-" is returning the array by value and not an access to the array. The compiler has to figure out what to do with that value before assigning it to t_Diff and it is using a stack temporary.

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Sat, 4 Dec 2010 17:22:04 -0800 PST
Subject: Re: Large arrays passed to arithmetic operators overflows GNAT stack
Newsgroups: comp.lang.ada

[…]

That is true but the renaming is still convenient for other reasons.

For example, the lines

```
t     : Real_Vector_Access :=
   new Real_Vector(0 .. N);
t_Diff : Real_Vector_Access :=
   new Real_Vector(0 .. N - 1);
...
[Line 14] t_Diff := t(1 .. N);
```

causes the complaints

```
expected type "Real_Vector_Access"
defined at line 14
found type
"Ada.Numerics.Generic_Real_Arrays.R
eal_Vector" from
instance at a-nlrear.ads:18
```

And passing t to a vector cosine (e.g.) function that expects to see a Real_Vector will also fail. Of course, with two like pointers t and x, assignment t := x assigns the pointers, not the array contents, and with three like pointers t, x, y, t := x - y fails to find an applicable operator function.

This clever renaming trick was mentioned by Brian Drummond in this thread:

http://groups.google.com/group/comp.lang.ada/browse_thread/thread/ae395e5c11de7bc9/bda8d61bd3a66ee9?hl=en&q=Jerry+stack&lnk=nl&

Task safeness of packages

From: Riccardo Bernardini <framefritti@gmail.com>
Date: Wed, 2 Feb 2011 12:51:04 -0800 PST
Subject: About task-safeness
Newsgroups: comp.lang.ada

Dear all,

I have a question (better, two questions) about packages and concurrence.

We have a software, a fairly complex one, that makes use of tasks. (Just to give you the context, it is a network communication software that can have several connections open at once and every connection is handled by a task.) Data structure that are designed to be shared among different tasks are implemented as protected objects, but it came to my mind that an innocent-looking package (that maybe provides some general-purpose functions) could have some "internal state" represented by some variable global to the package. (For example, a package defining some type of object could keep the number of allocated objects, so it can give to each object a unique ID.) If such a package was used by two different tasks, and the counter was not protected, obscure bugs can arise. This type of structure maybe is not very recommended, but it happens… :-(

Of course, one could do a review of all the packages to check for this type of problems, but since an Ada compiler has the good habit of protecting you from yourself, I searched for a way to have the compiler to check the task-safety of the packages used by tasks.

My first tentative was to ask that all the packages with-ed by a package that defines a task should be Pure (a Pure package cannot have any global variables). Unfortunately, I soon discovered that asking for Pure-ity is a too strong requirement: all the ancestors must be Pure and no un-Pure package can be used. Although such constraints make perfectly sense, they prevent you from using several standard (and useful) packages such as Unbounded_Strings, all (?) the Containers hierarchy and GNAT.Sockets (which turns out handy in a networking program…:-). (To be honest, my action of Pure-fication was not useless; while making my packages Pure, I caught a global counter in a package…) So, my first question is:

- Can you suggest a way to have the compiler to check for some task-safety of packages? Even a technique for a non-totally exhaustive check could be useful.

The thoughts above triggered in me another question. Consider, for example, the Ordered_Maps package. That package is not Pure (it cannot be, since it would prevent the use of named access types), so how can I be granted that the package does not have some "hidden" and unprotected state? Please note that I am *not* asking for an *object* of type Ordered_Map to be task-safe, if I need I can wrap it in a protected object; I am asking for the *package* to be task-safe. Note that if, say, Ordered_Maps has some hidden status, two task can modify the status at the same time by accessing to two Maps, even if the Maps have been wrapped inside two different protected objects. So, my second question is

- Am I granted (maybe by some obscure paragraph of our beloved RM ;-) that the standard packages are task-safe? (I would be surprised if they weren't, but it is nice to be sure…)

[…]

From: Vinzent Hoefler
Date: Wed, 02 Feb 2011 22:01:20 +0100
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

> […] - Am I granted (maybe by some obscure paragraph of our beloved RM ;-) that the standard packages are task-safe? […]

Well, ARM 05, A(3/2) says:

The implementation shall ensure that each language-defined subprogram is reentrant in the sense that concurrent calls on the same subprogram perform as specified, so long as all parameters that could be passed by reference denote non-overlapping objects.
Apart from that, you probably have to trust the programmer - or some tool.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 02 Feb 2011 23:38:21 +0100
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

Or you can use AdaControl, have a look at rule Global_References.

This can check all global variables accessed from more than one task (and it does so by following the call graph).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 2 Feb 2011 20:44:34 -0600
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

Note that "protection" may simply be declaring the object `Atomic`. Presuming the compiler supports that, there isn't a problem with multiple tasks accessing the same counter. (The main problem with small objects is compilers that get too helpful and optimize the unoptimizable.)

…

> […] how can I be granted that the package does not have some "hidden" and unprotected state?

Hidden and truly unprotected would violate the RM, as noted by others. But hidden and Atomic is fine (the Janus/Ada containers will use counters this way to detect dangling cursors).

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 03 Feb 2011 09:53:03 +0100
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

Randy, could you be more explicit about your suggested use of Atomic? As I understand it, even if a counter variable N is Atomic, two tasks concurrently executing an assignment of the form

```
N := N + 1;
```

can interleave their actions so that N is increased by only 1, not by 2 as intended.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Thu, 03 Feb 2011 12:07:52 +0100
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

When I first read about atomic counting, I thought so, too. But then, I noticed a new(?) Implementation Advice in the LRM that says, IIRC, that compilers should make atomic processor ops available, including decrement. A neighboring paragraph is about intrinsic operations.

From: Cristoph Grein <christoph.grein@eurocopter.com>
Date: Thu, 3 Feb 2011 03:22:53 -0800 PST
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

You mean C.1(11-12):

(11) It is recommended that intrinsic subprograms be provided for convenient access to any machine operations that provide special capabilities or efficiency and that are not otherwise available through the language constructs. Examples of such instructions include:

(12) Atomic read-modify-write operations — e.g., test and set, compare and swap, decrement and test, enqueue/dequeue.

But `N := N + 1`; is not such an operation even if N is atomic. An implementation would have to provide something like a CAS operation.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Thu, 03 Feb 2011 11:13:25 -0700
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

[…]

Those are examples of such operations, which will differ from machine to machine. If a machine offers an atomic increment operation, a compiler implementing Annex C should provide a subprogram to use it.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 3 Feb 2011 18:33:00 -0600
Subject: Re: About task-safeness
Newsgroups: comp.lang.ada

> […] even if a counter variable N is Atomic, two tasks concurrently executing an assignment of the form
>
> N := N + 1;
>
> can interleave their actions so that N is increased by only 1, not by 2 as intended.

Good point. All of the examples I've used have been where one task set the value to some constant (like True or False), and other read it. That's safe, while doing both may not be.

Practically(*), however, this will be generated atomically on an x86 (at least on Janus/Ada): it will generate an

```
Inc [N]
```

instruction, which I don't think can be interrupted between the read and the write. The net effect is that the operation will be undivided on a single processor. (I don't really know how multicore systems might affect this; there may be additional locks needed here.)

(*) Note that Ada language rules might make generating this atomically impossible, such as if an overflow check is needed. In that case, separate read and write instructions might be needed. That's not a problem in the Janus/Ada runtime (where checks are suppressed always, mostly for size and speed reasons), but it might be in your code.

And of course, this is very target dependent. `N := N + A`; is much harder to generate atomically, and for expressions involving atomic array components might be impossible.

P.S. None of this actually depends on "Atomic" in the current Janus/Ada compiler, as it isn't actually supported. But this optimization is done for all objects. In addition, Janus/Ada currently treats all writes as volatile. Most likely, the next version of Janus/Ada will have a full implementation of Atomic (it's been fully implemented in the front-end for years - Isaac did it as part of an ACATS review contract back in the mid-90s. But I never implemented it in the optimizer and back-end - and that needs to be fixed.)

How to send an email with an Ada program

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Fri, 17 Dec 2010 15:23:21 -0800 PST
Subject: How to have Ada program send an email
Newsgroups: comp.lang.ada

How can I get my Ada (GNAT) program to send me an email? It's a long-running simulation and I'd like to know when it finishes.

I'm on OS X so I suppose I could write an Applescript to cause Mail.app to do the job, then execute the Applescript as a command line within the Ada program, but is there an easier or more direct way?

From: Pascal Obry <pascal@obry.net>
Date: Sat, 18 Dec 2010 00:30:30 +0100
Subject: Re: How to have Ada program send an email
Newsgroups: comp.lang.ada

[…]

One solution is to use AWS's SMTP API.

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Sat, 18 Dec 2010 11:06:20 -0800 PST
Subject: Re: How to have Ada program send an email
Newsgroups: comp.lang.ada

[…]

AWS - took a minute to realize that the Linux versions of GNAT GPL 2010 were suitable for Mac OS X too! […]

Conference Calendar

Dirk Craeynest

K.U.Leuven. Email: Dirk.Craeynest@cs.kuleuven.be

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 *Conferences and events 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.

2011

☺ April 10-13	6th **European Conference on Computer Systems** (EuroSys'2011), Salzburg, Austria. Topics include: all areas of operating systems and distributed systems, including systems aspects of Dependable computing and storage, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, etc. Includes workshops on: Systems for Future Multi-Core Architectures (SFMA'2011), Workshop on Rigorous Systems Engineering (WRiSE'2011), 5th EuroSys Doctoral Workshop (EuroDW'2011), Rigorous Embedded Design (RED'2011), etc.
April 12-15	2nd **International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering** (PARENG'2011), Ajaccio, Corsica, France.
April 18-20	3rd **NASA Formal Methods Symposium** (NFM'2011), Pasadena, California, USA. Topics include: Theorem proving, Model checking, Static analysis, Dynamic analysis, Model-based development, Application experiences, etc.
☺ April 25-29	5th **Latin-American Symposium on Dependable Computing** (LADC'2011), São José dos Campos, São Paulo, Brazil. Topics include: Dependability of software (frameworks and software architectures for dependability, model driven dependability engineering, testing, verification, software certification, ...); Dependability of maintenance; Dependability and human issues; Security; Safety (safety-critical applications and systems, ...); etc.
April 27-29	18th Annual **IEEE International Conference and Workshops on the Engineering of Computer Based Systems** (ECBS'2011), Las Vegas, Nevada, USA. Theme: "Engineering Next Generation Systems". Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Integration Engineering; Model-Based System Development; Modelling and Analysis of Complex Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc.
Apri 27-29	16th Annual **IEEE International Conference on the Engineering of Complex Computer Systems** (ICECCS'2011), Las Vegas, Nevada, USA. Topics include: Verification and validation, Model-driven development, Reverse engineering and refactoring, Design by contract, Agile methods, Safety-critical & fault-tolerant architectures, Real-time and embedded systems, Tools and tool integration, Industrial case studies, etc.
May 10-12	11th Annual **International Conference on New Technologies of Distributed Systems** (NOTERE'2011), Paris, France. Topics include: recent technology advances and latest research results in the design, implementation, deployment, and evaluation of distributed system, applications, algorithms and architectures; such as on: Distributed middleware (implementations and applications); Distributed real-time embedded middleware and applications; Domain specific languages for distributed systems; Modeling, formal and semi-formal methods, and tools for distributed systems; Reliability and scalability of distributed systems; etc.
May 16-19	23rd Annual **Systems and Software Technology Conference** (SSTC'2011), Salt Lake City, Utah, USA.

☺ May 16-20 25th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2011), Anchorage, Alaska, USA. Topics include: all areas of parallel and distributed processing, such as: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, middleware, libraries, parallel programming paradigms, programming environments and tools, etc.

☺ May 20 12th **International Workshop on Parallel and Distributed Scientific and Engineering Computing** (PDSEC-11). Topics include: parallel and distributed computing techniques and codes; practical experiences using various parallel and distributed systems; task parallelism; scheduling; compiler issues for scientific and engineering computing; scientific and engineering computing on parallel computers, multicores, GPUs, FPGAs, ...; etc.

☺ May 20 **Workshop on Multithreaded Architectures and Applications** (MTAAP'2011). Topics include: programming frameworks in the form of languages and libraries, compilers, analysis and debugging tools to increase the programming productivity.

☺ May 17-20 **DAta Systems In Aerospace** (DASIA'2011), Marrakesh, Morocco.

☺ May 21-28 33rd **International Conference on Software Engineering** (ICSE'2011), Waikiki, Honolulu, Hawaii, USA. Theme: "Software by Design". Topics include: Engineering of distributed/parallel software systems; Engineering of embedded and real-time software; Engineering secure software; Patterns and frameworks; Programming languages; Reverse engineering and maintenance; Software architecture and design; Software components and reuse; Software dependability, safety and reliability; Software economics and metrics; Software tools and development environments; Theory and formal methods; etc.

☺ May 21 4th **International Workshop on Multicore Software Engineering** (IWMSE'2011). Topics include: Modeling techniques for multicore software; Software components and composition; Programming models and their impact on multicore software engineering; Testing and debugging parallel applications; Software reengineering for parallelism; Development environments and tools for multicore software; Experience reports from research projects or industrial projects; etc.

May 21-22 8th **International Working Conference on Mining Software Repositories** (MSR'2011). Topics include: Mining of repositories across multiple projects; Characterization, classification, and prediction of software defects based on analysis of software repositories; Search techniques to assist developers in finding suitable components and code fragments for reuse, and software search engines; Analysis of change patterns and trends to assist in future development; Case studies on extracting data from repositories of large long-lived and/or industrial projects; etc.

May 21-28 2nd **Student COntest on softwaRe Engineering** (SCORE'2011).

May 22-24 24th IEEE-CS **Conference on Software Engineering Education and Training** (CSEET'2011), Waikiki, Honolulu, Hawaii, USA. Topics include: Technology Transfer, Student projects and internships, Industry-academia collaboration models, Software engineering professionalism, Education & training for "real-world" Software Engineering practices, Training models in industry, Systems and Software Engineering, etc.

May 24-27 22nd IEEE **International Symposium on Rapid System Prototyping** (RSP'2011), Karlsruhe, Germany. Topics include: Real Time embedded system challenges; Specification and Language Models for hardware/software systems; Very large scale system engineering; Embedded System verification/validation; Critical Embedded Systems design; Reliability and failure analysis; Emerging Technologies and Applications; Industrial Designs in Automotive, Medical and Avionics domains; etc.

☺ May 26-28 9th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2011), Busan, Korea. Topics include: all aspects of parallel and distributed computing and networking, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, High-performance scientific and engineering computing, etc.

June 01-03	11th **International Conference on Computational Science** (ICCS'2011), Tsukuba, Japan. Topics include: recent developments in methods and modelling of complex systems for diverse areas of science, advanced software tools, etc.
June 06-09	6th **International Federated Conferences on Distributed Computing Techniques** (DisCoTec'2011), Reykjavik, Iceland. Includes the COORDINATION, DAIS, and FMOODS & FORTE conferences.
June 05-17	36th Annual **USENIX Technical Conference** (USENIX ATC'2011), Portland, Oregon, USA. Topics include: Distributed and parallel systems; Embedded systems; Reliability, availability, and scalability; Security, privacy, and trust; etc.
June 20-23	2011 **International Conference for Computational Science and its Applications** (ICCSA'2011), Santander, Spain. Topics include: Parallel and Distributed Computing, Security Engineering, Risk Analysis, Reliability Engineering, Software Engineering, etc. Deadline for early registration: April 4, 2011.
♦ June 20-24	16th **International Conference on Reliable Software Technologies - Ada-Europe'2011**, Edinburgh, UK. Organized together with the Ada Conference UK 2011, under the common name of "The Ada Connection". Sponsored by Ada-Europe, in cooperation with ACM SIGAda.
June 20-24	17th **International Symposium on Formal Methods** (FM'2011), Limerick, Ireland. Theme: "Formal Methods Come of Age". Topics include: advances and maturity in formal methods research, education, and deployment via tool support and industrial best practice, and their role in a variety of industries, domains, and in certification and assurance; in particular experience with practical applications of formal methods in industrial and research settings, experimental validation of tools and methods as well as construction and evolution of formal methods tools.
June 20-24	9th **Working IEEE/IFIP Conference on Software Architecture** (WICSA'2011), Boulder, Colorado, USA. Topics include: Software architecture and software qualities; Architecture description languages and model driven architecture; Software architecture discovery and recovery; Software architects' roles and responsibilities; Training, education, and certification of software architects; Industrial experiments and case studies; etc.
June 20-24	11th **International Conference on Application of Concurrency to System Design** (ACSD'2011), Kanazawa, Japan. Topics include: (Industrial) case studies of general interest, gaming applications, consumer electronics and multimedia, automotive systems, (bio-)medical applications, internet and grid computing, ...; Synthesis and control of concurrent systems, (compositional) modelling and design, (modular) synthesis and analysis, distributed simulation and implementation, ...; etc.
June 27-29	16th Annual **Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2011), Darmstadt, Germany.
☺ June 28-30	49th **International Conference Objects, Models, Components, Patterns** (TOOLS Europe'2011), Zurich, Switzerland. Topics include: Applications to safety- and security-related software; Distributed and concurrent object systems; Domain specific languages and language design; Experience reports, including efforts at standardisation; Language implementation techniques, compilers, run-time systems; Multicore programming, models and patterns; Object technology, including programming techniques, languages, tools; Practical applications of program verification and analysis; Real-time object-oriented programming and design; Tools and frameworks for supporting model-driven development; Trusted and reliable components; etc.
☺ June 30	**Workshop on Entwicklung zuverlässiger Software-Systeme**, Stuttgart, Germany. Topics include (in German): Multi-Core-Architekturen und Migration; Echtzeitsysteme; Sprachen versus Bibliotheken; Risikobetrachtungen, Normen und SIL; Nachweis der Erfüllung der Sicherheits-Anforderungen; Bewertung von realisierten Systemen; etc.
June 30 – July 02	5th **International Conference on Complex, Intelligent and Software Intensive Systems** (CISIS'2011), Seoul, Korea.
	☺ June 30 — **International Workshop on Multi-Core Computing Systems** (MuCoCoS'2011). Topics include: multi-core embedded systems; programming languages and models; algorithms for multi-core computing systems; applications for multi-core systems;

performance modeling and evaluation of multi-core systems; design space exploration; tool-support for multi-core systems; compilers, runtime and operating systems; etc.

July 03-06 4th **International Conference on Software Language Engineering** (SLE'2011), Braga, Portugal. Topics include: Formalisms used in designing and specifying languages and tools that analyze such language descriptions; Language implementation techniques; Program and model transformation tools; Language evolution; Approaches to elicitation, specification, or verification of requirements for software languages; Design challenges in SLE; Applications of languages including innovative domain-specific languages or "little" languages; etc. Deadline for submissions: April 8, 2011 (papers).

July 05-07 15th **International Conference on System Design Languages of the SDL Forum Society** (SDL'2011), Toulouse, France. Topics include: Industrial application reports (industrial usage and experience reports, tool engineering and frameworks, domain-specific applicability, such as aerospace, automotive, control, ...); Evolution of development tools and languages (domain-specific profiles and extensions, modular language design, semantics and evaluation, methodology for application, standardization activities); Modeling in multi-core and parallel applications; Education and Promotion of System Design Languages; etc. Deadline for submissions: May 15, 2011 (posters, exhibits).

July 13-14 11th **International Conference on Quality Software** (QSIC'2011), Madrid, Spain. Topics include: Software quality (review, inspection and walkthrough, reliability, safety and security, ...); Evaluation of software products and components (static and dynamic analysis, validation and verification); Economics of software quality; Formal methods (program analysis, model construction, ...); Applications (component-based systems, distributed systems, embedded systems, enterprise applications, information systems, safety critical systems, ...); etc.

July 14-20 23rd **International Conference on Computer Aided Verification** (CAV'2011), Snowbird, Utah, USA. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Verification methods for parallel and concurrent hardware/software systems, Applications and case studies, Verification in industrial practice, etc.

☺ July 25-29 25th **European Conference on Object-Oriented Programming** (ECOOP'2011), Lancaster, UK. Topics include: all areas of object technology and related software development technologies, such as Analysis and design methods and patterns; Distributed, concurrent, real-time systems; Language design and implementation; Modularity, components, services; Software development environments and tools; Type systems, formal methods; Compatibility, software evolution; etc. Deadline for submissions: April 15, 2011 (worskhop papers, doctoral symposium, PhD workshop), May 23, 2011 (studentships).

> ☺ July 26 6th **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2011). Topics include: efficient implementation and compilation of OO languages in various application domains ranging from embedded and real-time systems to desktop systems. Deadline for paper submissions: April 15, 2011.

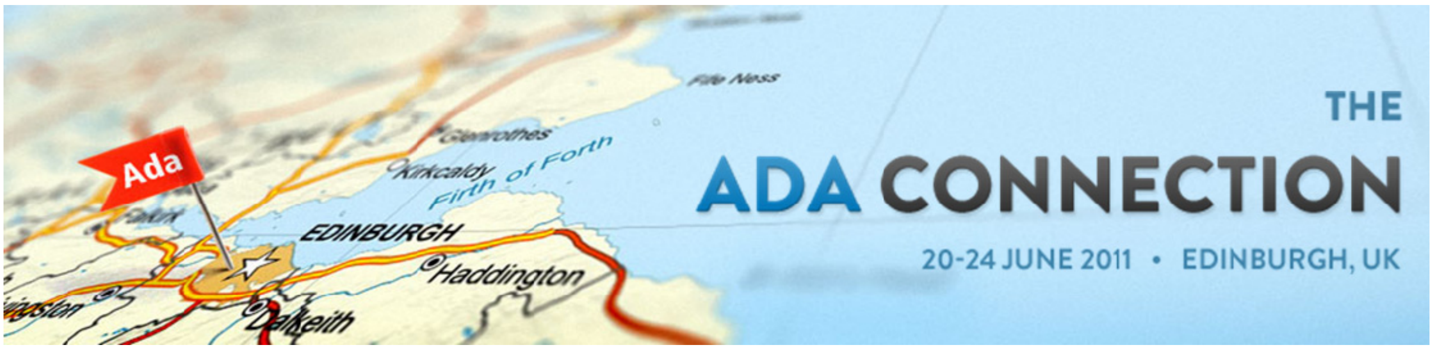
August 15-18 6th IEEE **International Conference on Global Software Engineering** (ICGSE'2011), Helsinki, Finland. Topics include: Strategic issues in distributed development (cost-benefit-risk analysis, ...); Methods and tools for distributed software development (requirements engineering, design, coding, verification, testing and maintenance, development governance); Empirical studies and lessons learnt from distributed development; etc.

☺ August 29-30 16th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2011), Trento, Italy. Topics include: Design, specification, code generation and testing based on formal methods; Verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability; Tools for the development of formal design descriptions; Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; Impact of the adoption of formal methods on the development process and associated costs; Application of formal methods in standardization and industrial forums; etc.

Aug 29 – Sep 02 15th IEEE **International Enterprise Computing Conference** (EDOC'2011), Helsinki, Finland. Topics include: the full range of engineering technologies and methods contributing to intra- and inter-enterprise distributed application systems; industry specific solutions, e.g. for aerospace, automotive, finance, logistics, medicine and telecommunications; etc.

☺ Aug 30 – Sep 02	**International Conference on Parallel Computing 2011** (ParCo'2011), Gent, Belgium. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular Applications of multicores, GPU-based applications, Parallel programming languages, compilers and environments, Best practices of parallel computing, etc.
September 05-09	8th **Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering** (ESEC/FSE'2011), Szeged, Hungary. Topics include: Case studies and experience reports; Engineering of distributed/parallel software systems; Engineering of embedded and real-time software; Engineering secure software; Reverse engineering and maintenance; Software architecture and design; Software components and reuse; Software dependability, safety and reliability; Software tools and development environments; Theory and formal methods; etc. Deadline for applications: April 3, 2011 (industry track, technical briefings, doctoral symposium); June 12, 2011 (new ideas track, tool demonstrations).
September 12-14	16th **European Symposium on Research in Computer Security** (ESORICS'2011), Leuven, Belgium. Topics include: Accountability, Information Hiding, Information Flow Control, Integrity, Formal Security Methods, Language-Based Security, Risk Analysis and Management, Security Verification, Software Security, etc.
September 13-16	5th **European Conference on Software Architecture** (ECSA'2011), Essen, Germany. Topics include: software tools and environments for architecture-centric software engineering; component-based models, middleware, component-based deployment; technology of components and component-based frameworks; industrial applications, case studies, best practices and experience reports; architecture description languages and metamodels; etc. Deadline for submissions: April 3, 2011 (abstracts), April 10, 2011 (papers).
☺ Sep 13-16	40th **International Conference on Parallel Processing** (ICPP'2011), Taipei, Taiwan. Topics include: all aspects of parallel and distributed computing, such as Compilers, Programming Models and Languages, Multi-core and Parallel Systems etc.
☺ Sep 19-23	11th **International Conference on Parallel Computing Technologies** (PaCT'2011), Kazan, Russia. Topics include: all aspects of the applications of parallel computer systems; methods and tools for parallel solution of large-scale problems; languages, environment and software tools supporting parallel processing; etc.
♦ Sep 14-16	15th **International Real-Time Ada Workshop** (IRTAW'2011), Liébana, Cantabria, Spain. Deadline for position paper submissions: May 15, 2011.
September 22-23	5th **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2011), Banff, Alberta, Canada. Topics include: Generative programming, metaprogramming; Product-line architectures; Analysis of language support for generative programming; Semantics, type-systems of generative programs; Case Studies and Demonstration Cases; etc. Deadline for submissions: June 15, 2011 (industry experience reports, short papers, posters).
Sep 25 – Oct 01	27th IEEE **International Conference on Software Maintenance** (ICSM'2011), Williamsburg, VA, USA. Topics include: reverse engineering and re-engineering; static and dynamic analysis; software migration and renovation; maintenance and evolution process; mining software repositories; empirical studies in software maintenance and evolution; testing, only in relation to maintenance (e.g., regression testing); etc. Deadline for submissions: April 8, 2011 (research abstracts), April 15, 2011 (research papers), June 10, 2011 (doctoral symposium, tutorials), June 20, 2011 (industry abstracts, early research achievements abstracts), June 25, 2011 (industry presentations, early research achievements papers, tool demonstrations).
☺ Sep 26-30	CBSoft2011 - 15th **Brazilian Symposium on Programming Languages** (SBLP'2011), Sao Paulo, Brazil. Topics include: the fundamental principles and innovations in the design and implementation of programming languages and systems; such as: Programming paradigms and styles, including object-oriented, real-time, multithreaded, parallel, and distributed programming; Program analysis and verification, including type systems, static analysis and abstract interpretation; Programming language design and implementation, including new programming models, programming language environments, compilation and interpretation techniques; etc. Deadline for submissions: April 22, 2011 (abstracts), April 29, 2011 (full papers).

Sep 29-30	10th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SoMeT'2011), Saint Petersburg, Russia. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Software developments techniques and legacy systems; Automatic software generation versus reuse, and legacy systems; Software evolution techniques; Agile Software and Lean Methods; Formal methods for software design; Software maintenance; Software security tools and techniques; Formal techniques for software representation, software testing and validation; Software reliability, and software diagnosis systems; Model Driven Development (DVD), code centric to model centric software engineering; etc.
☺ Oct 04-07	30th IEEE **International Symposium on Reliable Distributed Systems** (SRDS'2011), Madrid, Spain. Topics include: distributed systems design, development and evaluation, particularly with emphasis on reliability, availability, safety, security, trust and real time; high-confidence systems; distributed objects and middleware systems; formal methods and foundations for dependable distributed computing; analytical or experimental evaluations of dependable distributed systems; etc. Deadline for submissions: April 3, 2011 (full papers).
☺ Oct 20-22	12th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2011), Gwangju, Korea. Topics include: all areas of parallel and distributed computing; Reliability, and fault-tolerance; Formal methods and programming languages; Software tools and environments; Parallelizing compilers; Component-based and OO Technology; Parallel/distributed algorithms; Task mapping and job scheduling; etc. Deadline for submissions: April 5, 2011 (workshops), May 20, 2011 (submissions).
October 25-28	13th **International Conference on Formal Engineering Methods** (ICFEM'2011), Durham, UK. Topics include: Abstraction and refinement; Formal specification and modelling; Software verification; Program analysis; Tool development and integration; Software safety, security and reliability; Experiments involving verified systems; Applications of formal methods; etc. Deadline for submissions: April 7, 2011 (papers).
Nov 06-10	ACM **SIGAda Annual International Conference on Ada and Related Technologies** (SIGAda'2011), Denver, Colorado, USA. Deadline for submissions: June 30, 2011.
November 14-18	9th **International Conference on Software Engineering and Formal Methods** (SEFM'2011), Montevideo, Uruguay. Topics include: programming languages, program analysis and type theory; formal methods for real-time, hybrid and embedded systems; formal methods for safety-critical, fault-tolerant and secure systems; light-weight and scalable formal methods; tool integration; applications of formal methods, industrial case studies and technology transfer; etc. Deadline for submissions: April 23, 2011 (abstracts), April 30, 2011 (papers).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!
December 18-21	18th IEEE **International Conference on High Performance Computing** (HiPC'2011), Bengaluru, Bangalore, India. Topics include: Parallel and Distributed Algorithms, Parallel Languages and Programming Environments, Scheduling, Fault-Tolerant Algorithms and Systems, Scientific/Engineering/Commercial Applications, Compiler Technologies for High-Performance Computing, Software Support, etc. Deadline for submissions: May 16, 2011 (papers, workshops), Sep 16, 2011 (student symposium). Deadline for early registration: November 14, 2011.



16th International Conference on Reliable Software Technologies - Ada-Europe 2011

Ada Conference UK 2011

John McIntyre Conference Centre, Edinburgh, UK
20 - 24 June 2011
http://www.ada-europe.org/conference2011

The Ada Connection 2011 sees a union of two Ada events that have both been very successful in their own right. The **Ada-Europe** series of conferences has become established as an international forum for providers, practitioners and researchers in all aspects of reliable software technologies. The **Ada Conference UK** has been running in its current form since 2006 as a series of biennial one-day events, to highlight the increased relevance of Ada in safety- and security-critical systems. By combining these events, the **Ada Connection** will provide a unique opportunity for interaction and collaboration between academics and industrial practitioners.

Edinburgh is a first-class city and conference venue. The John McIntyre Conference Centre is located within the University of Edinburgh campus, approximately 20 minutes walk from the city centre. The Conference Centre is adjacent to Holyrood Park, whose centrepiece, Arthur's Seat, offers a half-hour 'climb' from which excellent views can be gained of the city of Edinburgh and beyond.

Monday	*Tuesday*		*Wednesday*		*Thursday*		*Friday*
	Opening/Welcome		Ada 2012 update		GAP update, Educators' session, AdaWay Prize		
Tutorials	Keynote - Peter Ladkin		Keynote - Pippa Moore		Keynote - Jeff O'Leary		*Tutorials*
Tutorials	Technical Papers: Multi-Core		Industrial Presentations: Modelling and Complexity		Panel: DO178C and Object-Orientation for Critical Systems		*Tutorials*
Tutorials	Technical Papers: Verification	Sponsor talks	Industrial Presentations: Real-time and Longevity	Exhibitor talks	Technical Papers: Architecture and Modelling	Industrial Presentations: Innovation and New Markets	*Tutorials*
Tutorials	Panel: Programming Languages Meet Multi-Core	Sponsor & Exhibitor talks	Industrial Presentations: Transitioning and Debugging	Exhibitor talks	Technical Papers: Education and Mixed Criticality		*Tutorials*

In cooperation with ACM SIGAda

The conference programme includes tutorials, technical papers, industrial papers, vendor presentations and a comprehensive vendor exhibition, together with two panel discussion sessions and three keynote talks. These are summarised below, but for full details and registration see the conference website:

http://www.ada-europe.org/conference2011

Tutorials

Six half-day tutorials are presented on Monday, and three full-day tutorials on Friday.

Half Day Tutorials

T1: **Experimenting with ParaSail Parallel Specification and Implementation Language** *by S. Tucker Taft, SofCheck, Inc (USA)*

T2: **Designing and Checking Coding Standards for Ada** *by Jean-Pierre Rosen, Adalog (France)*

T3: **Programming Distributed Systems with YAMI4** *by Maciej Sobczak, Inspirel (Poland)*

T4: **Why and How to Measure Non-Functional Properties On Target** *by Ian Broster, Rapita Systems Ltd (UK)*

T5: **Revamping the Software Technical Review Process** *by William Bail, The MITRE Corporation (USA)*

T6: **Use of Object-Oriented Technologies in High-Reliability Systems** *by Jean-Pierre Rosen, Adalog (France)*

Full Day Tutorials

T7: **MAST: Predicting Response Times in Event-Driven Real-Time Systems** *by Michael G. Harbour, Universidad de Cantabria (Spain)*

T8: **SPARK: The Libre Language and Toolset for High-Assurance Software** *by Roderick Chapman, Altran Praxis (UK)*

T9: **Distributed Programming Techniques in Ada** *by Thomas Quinot, AdaCore (France)*

For full tutorial details, please see the conference web pages.

Keynotes

Three keynote talks are provided by eminent speakers: **Peter Bernard Ladkin** (Causalis Ltd) on the *Future of Software Safety Standards*, **Pippa Moore** (UK CAA) on the topic of *Hippocrates and DO-178B*, and **Jeff O'Leary** (USA FAA) on *Assuring Software Reliability While Using Web Services and Commercial Products*.

Panels

Panel 1: Programming Languages Meet Multi-Core

The advent of multi-core is shaking the very foundations of programming languages for concurrency, resource sharing, synchronisation, etc. We are asking some language designers to discuss solutions to this challenge.
Confirmed participants include: **Erhard Plödereder** (moderator – University of Stuttgart), **Alan Burns** (University of York), **Tucker Taft** (SofCheck, Inc), **Kevin Hammond** (University of St Andrews).

Panel 2: DO178C and Object-Orientation for Critical Systems

The high-integrity systems industry faces the challenge of reaping the benefit of object-orientation in their rigid and demanding development process. Domain experts will debate pros and cons, risks and opportunities, and ways to introduce elements of object-orientation into safety-critical system development.
Confirmed participants include: **Cyrille Comar** (AdaCore), **Jean-Pierre Rosen** (Adalog), **Dewi Daniels** (Verocel).

Vendor Exhibition

The exhibition will open on Tuesday morning, and run until the last session on Wednesday. It will take place in the conference venue; coffee breaks and lunch on Wednesday will be served in the exhibition space.
Companies who have already booked their exhibition space at the conference include, in alphabetical order: **AdaCore, Altran Praxis, Causalis Limited, Ellidiss Software, Green Hills Software, LDRA, Objektum Solutions, Rapita Systems, Resource Engineering, TTE Systems, Vector Software** and **Wind River.**
Exhibitors will also deliver a technical presentation in one of the tracks on Tuesday and Wednesday afternoon.

Social Programme

Welcome Reception - Tuesday 21st June

The Ada Connection offers a Welcome Reception with a difference, combining an informal opportunity to meet our exhibitors with an educational tasting experience. Each exhibition stand will have a different bottle of single malt Scotch whisky available, specially chosen to span the aroma and flavour spectrum; you are invited to visit and sample an appropriate number of these fine and rare "malts". Wine and soft drinks will, of course, also be available.

Conference Banquet Dinner - Wednesday 22nd June

The Ada Connection Gala Dinner will be held in true Celtic style at **The Signet Library** situated on the Royal Mile, considered to be one of the finest examples of Georgian architecture in the country.
Drinks will be served in the Lower Library, which is a working law library by day, enjoying an outlook onto Parliament Square and the Royal Mile. Once settled in, we will enjoy a pre-dinner presentation by **Professor Les Hatton** – a noted contributor to safer software engineering. Dinner will then be served in the Upper Library, with magnificent architectural features, including a magnificent stained glass window celebrating Queen Victoria's Jubilee.

15TH INTERNATIONAL REAL-TIME ADA WORKSHOP (IRTAW-15)

September 14-16, 2011 – Liébana (Cantabria), Spain

http://www.artist-embedded.org/artist/IRTAW-15.html

CALL FOR PAPERS

Since the late Eighties the **International Real-Time Ada Workshop** series has provide d a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent IRTAW meetings have significantly contributed to the Ada 2005 standard and to the proposals for Ada 2012, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, and in light of the current revision process that will lead to the new Ada 2012 standard, the goals of **IRTAW-15** will be to:

- review the current status of the Ada 2012 Issues that are related with the support of real-time systems;
- examine experiences in using Ada for the development of real-time systems and applications, especially – but not exclusively – those using concrete implementation of the new Ada 2005 real-time features;
- report on or illustrate implementation approaches for the real-time features of Ada 2012;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other extra-functional concerns;
- examine and develop paradigms for using Ada for real-time distributed systems, with special emphasis on robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;
- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects;
- consider the language vulnerabilities of the Ravenscar and full language definitions.

Participation at **IRTAW-15** is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will, however, be given to those submitting papers.

Format

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of Ada Letters (ACM Press) (to be confirmed). Selected papers will also appear in the Ada User Journal.

Submission

Please submit position papers, in PDF format, to the Program Chair by e-mail: **aldeam@unican.es**

Important Dates

Receipt of Position Paper: 15 May 2011	Final Copy of Paper: 31 July 2011
Notification of Acceptance: 15 June 2011	Workshop Date: 14-16 September 2011

Introduction

"The Ada Way" is an annual student programming contest organized by Ada-Europe, the international organization that promotes the knowledge and use of Ada in European academia, research and industry. A Steering Committee formed by representatives of promoting institutions oversees the organization of the contest. The Steering Committee is currently comprised of: Dirk Craeynest and Ahlan Marriott (Ada-Europe), Ricky Sward (ACM SIGAda), Jamie Ayre and Matteo Bordin (AdaCore), Jean-Pierre Fauche (Atego), Ian Broster (Rapita), Rod White (MBDA).

This initiative aims to attract students and educators to Ada in a form that is both fun and instructive. For this reason the contest is a yearly programming competition among student teams, whereby each team must have a university affiliation and be endorsed by an educator. The ideal, but not exclusive, context for participation is as part of an organized teaching/course activity in which the theme and requirements of the contest are endorsed and supported by the educator. See the "Participation Requirements" section for details.

The contest opens in September with the announcement of the theme, and allows submissions until the end of April the following year. See below for the 2010-11 edition theme and the Submissions section for the submission requirements.

Students and educators who may consider participating and want more information on "The Ada Way" in general and its 2010-11 edition in particular are invited to make contact with the Steering Committee at *board@ada-europe.org*.

Project Theme for Academic Year 2010-11: Software simulator of a football (soccer) match

The following specification intentionally leaves some room for interpretation and extension: participants are encouraged to use their intelligent creativity to firm up the derivative specification they want to work against.

The software system shall support at least the following features:

- Users must be able to play a single game; support for playing a series of matches, with fixtures and associated rules, is optional and can be omitted
- The chosen variant of the game shall be configurable in all relevant parameters, allowing for any of 5-a-side, 7-a-side, and the canonical 11-a-side formats
- The members of the squads will feature individually configurable characteristics for, at least, technical and tactical skills, speed, physical parameters including fatigue; some of those parameters shall be dynamic and evolve with the match according to some programmed logic
- Each squad shall have a (software) manager able to configure the initial players line up, the initial tactic and to issue commands for tactic changes and substitutions, all subject to the rules of the game as in the corresponding standard
- Each squad shall play according to the tactic commanded by the manager; deviations shall be permitted in so far as they result from programmable characteristics of the players
- Each match shall have one independent (software) referee and two to three subordinate (software) assistants who control the game and ensure that the applicable rules are followed; the behavior and the performance of the referee and assistants need not exhibit the physical limitations of actual humans.

The software system shall include at least:

- A software core, whether centralized or distributed, implementing all of the logic of the simulation
- One read-only graphical panel (window) for the display of the football field, the players, the ball, the referee and assistants; as for the (simulated) human figures on the pitch it shall be sufficient to represent them as moving numbered dots on the display without resorting to sophisticated graphical rendering, as in a view of a subbuteo table seen from the top
- Two distinct read-write graphical panels (windows) for the user to influence the otherwise independent action of the team managers; the panel shall display the current parameters for each player; the refresh rate of such display shall be user-configurable
- One read-only graphical panel (window) for the display of a user-configurable selection of statistics; the refresh rate of such display shall be user-configurable.

The software core shall be programmed in Ada. The software design shall permit the principal algoritms to be modified and replaced at will: in other words, the software system shall be as modular, configurable and scalable as possible. These qualities will contribute to the evaluation.

The graphical panels can be programmed in any language that the participating teams will consider fit for purpose. The graphical beauty of such panels will however be only a minor factor in the evaluation. What shall matter instead is that the interaction and the flow of data and control between the software core and the graphical panels is governed by good architectural principles and shows sufficient accuracy and performance.

To be considered for evaluation, the system shall run out of the box. The target platform may be freely chosen between Linux, Windows and MacOS. Portability across them will however be a competitive advantage.

Participation requirements

Participating teams shall be composed by a minimum of 2 and a maximum of 7 members. Each team shall have a codename and a logo. Team work may be performed as part of an organized teaching/course activity or as a volunteer project. Either way, each team must be recognised and endorsed by an academic educator.

Team members must be full-time students: they must provide evidence of their status when submitting their project. The contest is open to undergraduate and Master students. Teams may but need not include a mix of undergraduate and graduate students. Team members may belong to distinct institutions.

Submission

The software system shall be delivered in source (as a single compressed archive), accompanied by:

1. A software specification document (in PDF), which describes the principal design decisions and argues their quality, and presents the points of extension and modification in the system; the specification shall clearly single out all places at which the team made arbitrary interpretation of the specification or added or extended requirements
2. A user manual describing the compilation and installation procedures, the configuration options and the allowable use of the system (in PDF)
3. The team codename, logo and composition: name, email contact, evidence of enrollment as full-time students (in a single PDF)
4. The written endorsement to the submission by an academic or otherwise senior instructor in whose class the project was launched (in PDF).

The submission shall be made as a single compressed archive of all items listed above at the URL that will appear on this page in due time.

All sources shall be released for the good of the general public, to become reference material for educational and promotional purposes. To this end the use of GPL (GNU General Public License) is recommended, though we are not prescriptive of a specific scheme, so long as the general intent of free dissemination is preserved.

Submissions shall be accepted during the whole month of April 2011, at the Ada Way website, *http://www.ada-europe.org/AdaWay*.

Evaluation and Prize

The evaluation criteria will include:

- Coverage of requirements
- Syntatic, semantic, programmatic and design correctness
- Clarity and readability of the code
- Quality of design
- Ingenuity and cuteness of the solution
- Time and space efficiency of the solution.

The evaluation will be performed by a team of distinguished Ada experts comprised of: John Barnes (UK), Tucker Taft (US), Joyce Tokar (US), Pascal Leroy (F), Ed Schonberg (US).

The winning submission shall be announced on 31 May 2011 by a post on the site and by an email communication to all participating teams.

The prize will consist of: a framed award; one free registration and up to 3 reduced student fees for representatives of the winning team to attend to the Ada-Europe 2011 Conference; accommodation and airfare for the team representatives (with ceiling at EUR 3,000); an exhibition slot in the conference program; visibility in electronic and printed media including:

- Ada User Journal: *http://www.ada-europe.org/journal.html*
- Ada Letters: *http://www.sigada.org/ada_letters/*

For up-to-date information on Ada-Europe's student programming contest, please go to the official web site of "The Ada Way", *http://www.ada-europe.org/AdaWay,*

Sponsors

This year's competition is sponsored by Ada-Europe, AdaCore, and Atego.

A comparison of work-sharing, work-seeking, and work-stealing parallelism strategies using Paraffin with Ada 2005

B. J. Moore

General Dynamics Canada, 1020 68th Ave. N.E., Calgary, Alberta, Canada; Tel: 001.403.730-1367; email: brad.moore@gdcanada.com

Abstract

A computing trend today is towards the increased use of multicore computers. Achieving optimal performance on multicore architectures involves taking advantage of parallelism opportunities when possible. Iteration and recursion are examples of constructs where parallelism opportunities can often be applied.

Paraffin is an open source set of portable generics written in Ada 2005 that may be used to inject parallelism into loops and recursive algorithms. This paper explores Paraffin's capabilities to compare various parallelism approaches including work-sharing, work-seeking, and work-stealing. The paper suggests when one approach might be a better choice over another.

Keywords: Ada 2005, Paraffin, parallel loops, parallel recursion, work-sharing, work-seeking, work-stealing.

1 Introduction

While mainstream languages such as C and C++ do not currently provide direct support parallel loops and other parallelism constructs, there are language extensions to these languages [1, 5] that provide such capabilities. One problem with non-standard language extensions is that they are inherently non-portable.

A portable solution for parallelism does exist however, if one considers the Ada programming language [4]. Ada was designed from the ground up to support concurrency and multi-tasking. While Ada does not currently provide direct support for parallel loops and parallel recursion, the language does provide the building blocks needed to provide these parallelism constructs in the form of reusable generic subprograms, without having to resort to non-standard language extensions.

One such example of a parallelism library is Paraffin [3]. Paraffin is an open source set of Ada 2005 generics that may be used to inject parallelism into loops and recursive algorithms. Paraffin is 100% Ada code that does not have known dependencies on any particular Ada 2005 compiler, nor does it have any known dependencies on OS specific features. Thus, it should be possible in theory to use paraffin on any target that has an Ada 2005 compiler.

The Paraffin set of generics subprograms provide three parallelism approaches; Work-Sharing, Work-Seeking, and Work-Stealing.

Work-Sharing is the simplest approach that attempts to evenly distribute the work between a set of worker tasks at the start of a parallelism opportunity. Once the work has been divided amongst the tasks, each task proceeds with its share of the work until completion. When all workers have completed their tasks, the parallelism is complete, and the program continues executing sequentially until the next parallelism opportunity. This simpler approach generally is ideal when the amount of work can be evenly divided amongst the workers, though scheduling and differing processor loads tend to make it unlikely that workers will actually complete all their work at the same time, and better performance may actually result from selecting a different parallelism approach.

Work-Seeking starts off just like work-sharing except that when a worker completes its work, it will seek more work from other busy workers. Busy workers regularly check an atomic boolean flag indicating that a worker is seeking work. If the flag is set, then the busy worker reports to the generic the current progress of its tasking, and then the generic makes an offer to the idle worker through a protected object to split the remaining work assigned to the busy worker.

If the idle worker has not already accepted an offer from another busy worker that may have responded to the seeking work flag request, then the idle worker accepts the offer and both worker tasks proceed, with their half of the remaining work that was originally allotted to the busy worker. Work-Seeking proceeds in this manner until all work has been completed, where the program then continues sequential execution until the next parallelism opportunity.

The Work-Seeking hand-off of work between a busy worker and an idle worker is a cooperative hand-off. Work-Seeking generally makes sense when the work cannot be divided evenly between the workers, or when the workers will not likely complete the work at the same time.

The other parallelism variant is called Work-Stealing. This approach is similar to work-seeking, except that the goal is to have the idle worker undertake most of the effort involved with acquiring more work, when a worker completes its current assignment.

An idle worker will search randomly through the set of workers to identify a busy worker that has work that can be stolen. The idle worker then steals the work by manipulating the busy workers remaining work variables, and then the busy worker and the idle worker continue after having split the remaining work of the busy worker.

Unlike Work-Seeking, this approach is not cooperative. The busy worker does not make an offer the idle worker.

Work-Sharing, Work-Seeking, and Work-Stealing generics may be applied to iterative loops using Paraffin, while only Work-Sharing and Work-Seeking may currently be applied to recursive algorithms.

This paper examines the results of applying these parallelism strategies to various problems, and then suggests possible explanations for the results, and suggests which approach is the better approach to the problem if there is a clear winner. All tests were conducted on an AMD Athlon II 64 four core processor running Linux. Similar results have been previously reported for running the generics on an Intel Atom 333 1.6 Ghz CPU under Windows on an ASUS notebook computer [2]. The previously reported results did not include work-stealing, as the work-stealing generics were not available in Paraffin at that time.

2 Iterative Parallelism

The following set of problems involve exercising Paraffin's iterative parallelism generics.

2.1 Problem 1 - Parallel sum of integers

The first problem involves calculating the sum of integers between 1 and 400_000_000. The graph in figure 1 does not show the work-stealing results because the work-stealing times were not in the same ballpark as the work-sharing and work-seeking results. The work-stealing results are shown in figure 2.

The results in figure 1 show that the work-sharing and work-seeking generics perform comparably well. It is interesting to note that running the generic code with a single worker performs comparably to the sequential form of the loop without the generics. This is encouraging as it shows that the overhead of the generic processing can be quite minimal.

Looking at the results more closely, we see that the work-sharing generics generally performed marginally better than the work-seeking. This one would expect, as the load is evenly distributed amongst the workers, and the distributed overhead of the work-seeking does not provide enough benefits to make much of a difference. There was one case where work-seeking outperformed work-sharing, which was when the number of workers was one greater than the number of cores. This has more to do with the fact that the work-sharing performance dropped, not that the work-seeking was any faster. Work-sharing had the best overall time when the number of workers equalled the number of cores.

An interesting point that will be reinforced by subsequent test results is that using more workers than there are cores typically leads to a degradation in performance for work-sharing, whereas work-seeking forms tend to hold the performance more closely to the performance associated with running as many workers as there are cores, although once the number of workers reaches over and above the number of cores, the changes to the execution times profile is fairly flat and not that noticeable.

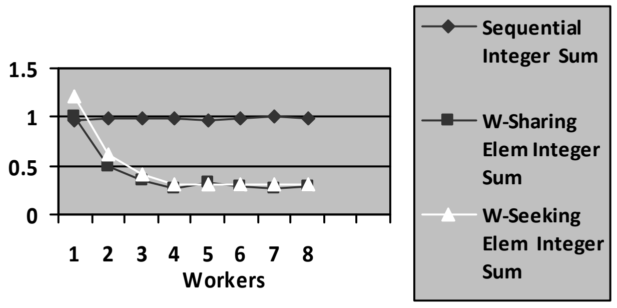


Figure 1 Sum of Integers Loop

Looking at the work-stealing results in Figure 2 for the same test, we see that the work-stealing approach cannot be recommended at all for this problem. All results were significantly worse than the sequential version of the code.

The explanation for this is that work-stealing involves saving the state of the iterator as an atomic reference type. This is needed as the work-stealer needs to have visibility into each workers progress within its range of loop iterations.

It is assumed that writing atomic updates through an indirect type is expensive. If the amount of iteration is significant relative to the actual processing performed during each iteration, then the work-stealing approach is likely going to perform poorly. On the other hand, as we will see later, if the amount of iteration is not significant compared to the processing performed during each iteration, then work-stealing can outperform the other two strategies.

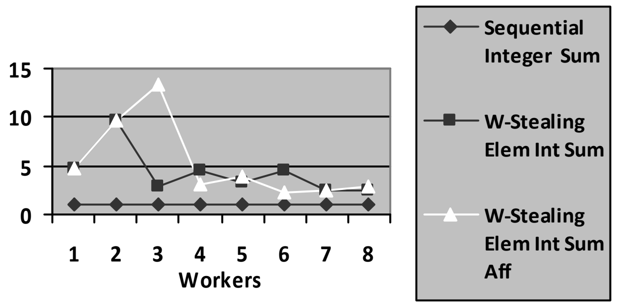


Figure 2 Integer Sum Loop – Work Stealing

The work-sharing and work-seeking tests were performed without setting the affinity of the worker tasks. The work-stealing tests were executed with and without setting the worker affinities to see if any mode of operation might generate favourable results. For this particular problem, all work-stealing results were unfavourable.

2.2 Problem 2 - Sum of tagged type Integers

The next problem is similar to the previous problem except that instead of producing a sum of integers, the desire is to produce a sum of tagged type objects which happen to have an integer component. As with the previous problem, there are 400_000_000 iterations.

The execution times shown in Figure 3a are slightly longer due to the manipulations of a composite tagged type requiring more processing. The results of this test however are very similar to the previous test. Work-sharing appears to have a very slight favourable performance over work-seeking, while work-stealing is again a non-starter in Figure 3b.

It is interesting again to note that work-seeking outperformed work-sharing when the number of workers was one or two higher than the number of cores. In fact, the best overall time occurred for work-seeking with 6 workers. In all other cases, work-sharing had the edge.

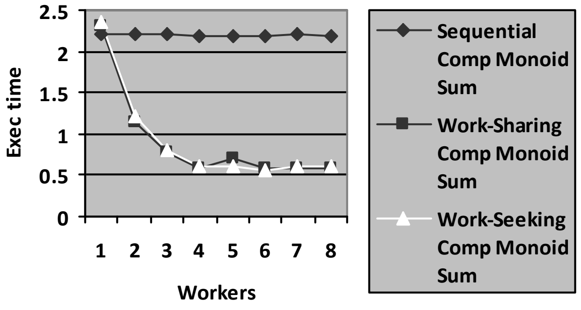


Figure 3a Monoid Integer Sum

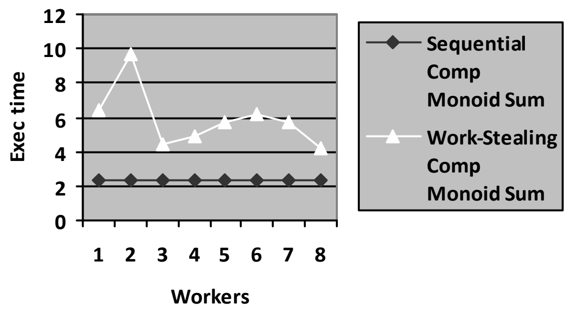


Figure 3b Monoid Integer Sum

2.3 Parallel sum of float

This test is similar to the first test except that instead of producing a sum of integers, the problem is to produce a sum of floating point values. In this case, as shown in Figure 4, work-seeking consistently produced marginally better results than work sharing. All the work-stealing results however were worse than the sequential version, which again is likely due to the higher number of iterations relative to the processing for each iteration.

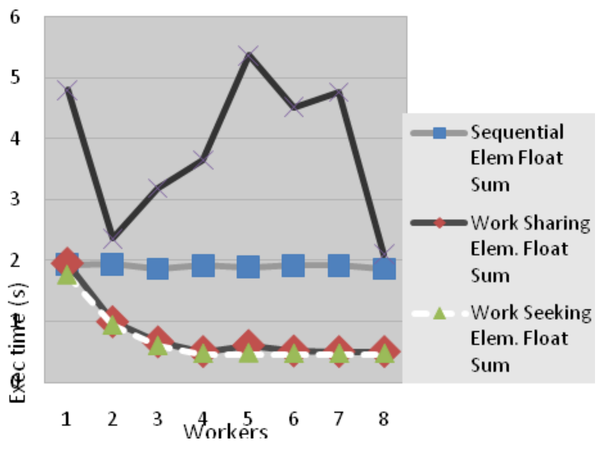


Figure 4 Float Sum

Another interesting point is that the execution time for a single work-sharing worker was equivalent to the sequential execution time, and the single worker work-seeking execution time was actually faster than the sequential time. While it is difficult to explain how this could be possible, it may be that the parallel code triggered the clock speed of the processor to run at a slightly higher rate. Selecting a worker count equal to the number of cores (four) is recommended here. Marginal improvements to both work-seeking and work-sharing occur for selecting worker counts greater than 4, the number of cores.

2.4 Iteration without results

The previous tests all involve generating a final result, called a reduction. Sometimes parallelism does not involve a need to produce a final result. Paraffin provides generics specifically for this purpose. For parallel iteration, this may involve loops that iterate through a container or array, possibly modifying each element in the container or array, without having to generate a final result.

This test involves iterating through a packed bit array of 100_000_000 Booleans and toggling the state of each element in the array. For this test, we finally see (Figure 5)

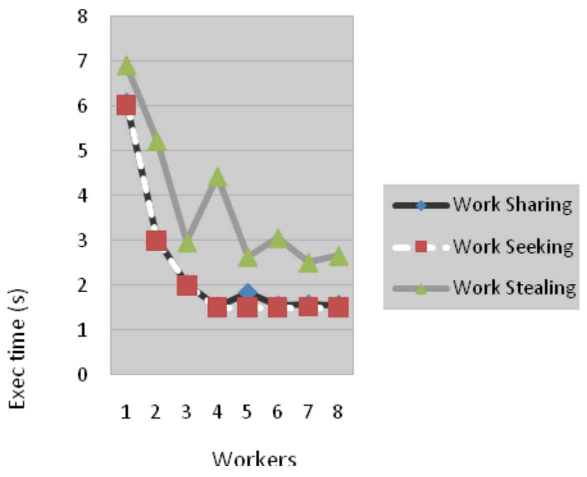


Figure 5 Iteration only

some respectable work-stealing results, though work-stealing still cannot be recommended because the execution times are noticeably behind the other results. In this test, the work-seeking consistently outperformed work-sharing although only by a very small margin, except for the usual case of one over the number of cores which consistently seems to have an adverse affect on the work-sharing approach, and the advantage of work-seeking was more noticeable.

2.5 Unbalanced iteration

This is a test that is designed to show how work-seeking and work-stealing should outperform work-sharing. Specifically, each iteration through the loop adds more processing to the iteration. Workers that are assigned higher ranges of iteration should take considerably longer to complete their work assignment than workers that are assigned lower ranges of iterations. As workers with lower iterations complete earlier, having them seek or steal more work to lighten the remaining work load should be beneficial.

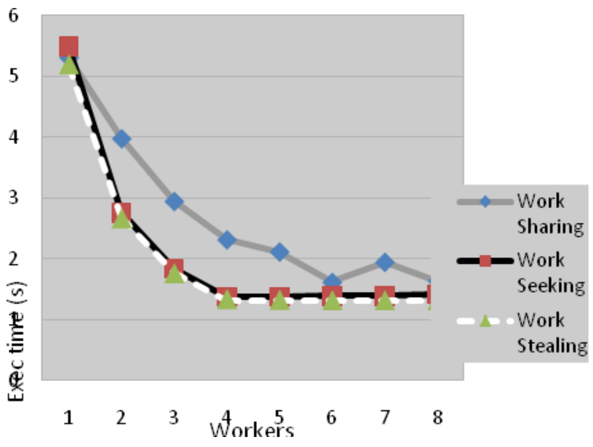


Figure 6 Unbalanced Iteration

For this test, the number of iterations is significantly less than the previous tests (60_000 instead of 400_000_000). The results (Figure 6) were surprising because for the first time, the work-stealing generic produced respectable results and not only were they respectable, but they were consistently better than work-seeking with the best times overall, although the performance differences between these approaches are marginal. As expected, the work-sharing generics did not fare so well on this test.

2.6 Matrix Solving

This problem involves applying elementary linear algebra in parallel to solve matrices. In particular, the gauss-jordan algorithm [11] is applied in parallel. An interesting point about this example is that the Paraffin generics are used to add parallelism to an algorithm that at the top level isn't a loop. Also, worth mentioning is that the algorithm requires sequential processing at lower levels of the iterative processing. To achieve this, Paraffin's synchronous barriers were used to ensure that the parallelism did not interfere with the sequential portions of the algorithm. Due to the nature of this algorithm, it only makes sense to apply work-sharing.

The test involves solving an 800x800 matrix of floating point values. The results (Figure 7) show the importance of selecting an optimal worker count.

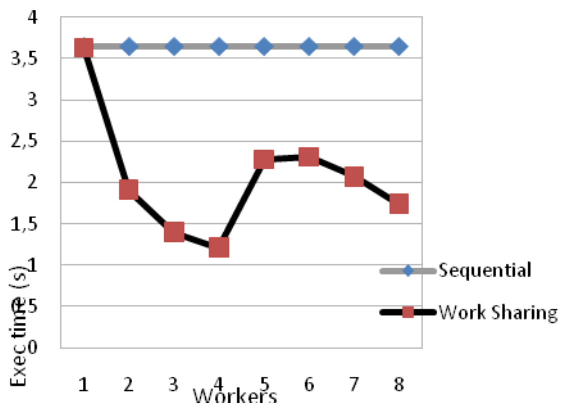


Figure 7 Parallel matrix solving

2.7 Parallel Bernoulli Numbers

This problem involves generating the Bernoulli number for 2000 by applying the Akiyama-Tanigawa algorithm in parallel [9]. This problem is quite different than previous problems because it is operating on the GNU Multiple Precision Arithmetic Library (GMP) [10] data types which have infinite precision representing fractions as rational values.

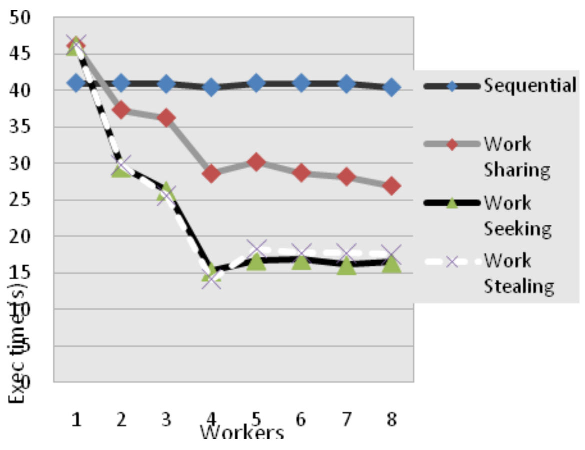


Figure 8 Parallel Bernoulli Numbers

Note that an algorithm for determining Bernoulli numbers was one that Ada Lovelace describes for use with Charles Babbage's Difference Engine, the first computer [8]. In her case the algorithm used, the method of finite differences, was not amenable to a parallel approach. This can be easily reasoned because determining subsequent Bernoulli numbers on the Difference Engine involved turning a crank. Thus, the result for a subsequent Bernoulli number depends on the result for the previous Bernoulli number. Such a dependence on other results from previous iterations typically means the algorithm must be run in sequential order ruling out parallelism.

The results (Figure 8) show that work-seeking and work-stealing have the best performance. The work-stealing generic had the best overall time when the number of workers matched the number of cores, otherwise work-seeking mostly had a slight edge over work-stealing.

3 Recursive Parallelism

The following tests exercise the recursive generics of Paraffin. The recursive generics currently only support work-sharing and work-seeking modes of parallelism, so the test results will only compare these strategies.

3.1 Recursive Iteration through a Red-Black Tree

This problem involves iterating through the nodes of a binary red-black tree containing 100_000 nodes. A red-black tree is a balanced tree, but there will always be more nodes on one side of the tree than the other, so it makes sense that a work-seeking approach would perform better (Figure 9) because the processing loads are unbalanced.

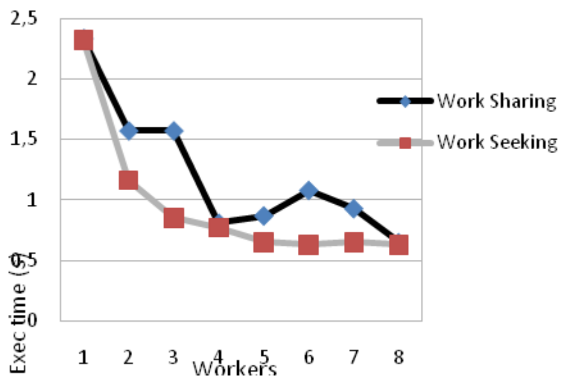


Figure 9 Parallel Recursion through tree

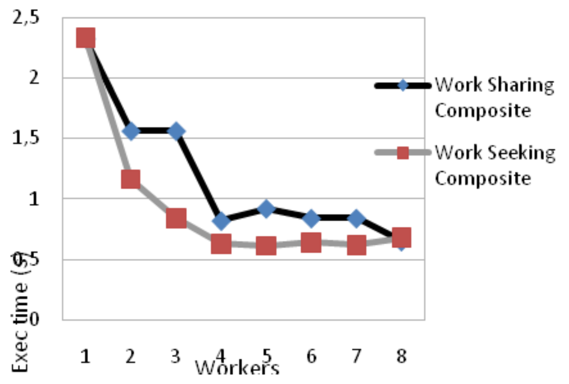


Figure 10 Parallel Reduction through tree

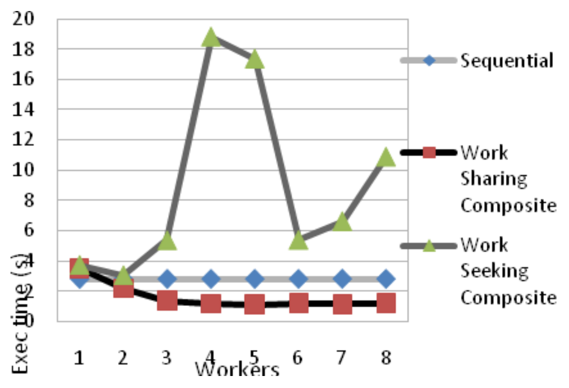


Figure 11 Parallel Recursive Fibonacci

3.2 Recursive Reduction of tree nodes

This test involves calculating the sum of all 100000 nodes in a red-black binary tree container. Each node contains an integer value that is examined by the recursive generics to produce the final result. The results (Figure 10) are similar to the previous recursive iteration example. Work-Seeking clearly performs better than work-sharing for this test.

3.3 Recursive Fibonacci

This test involves applying the paraffin recursive generics to generate Fibonacci numbers in parallel.

Work seeking has generally produced good results in all the other previous tests. It is surprising to see the results here (Figure 11) showing that work-seeking fails to produce a timely result for any number of workers, whereas work-sharing clearly is producing desirable performance results.

3.4 Recursive Integration

This test involves finding the area under a curve for a mathematical function, using a recursive trapezoidal integration algorithm [13] adapted to Ada 2005 [12]. In this case, the test involves integrating the square root function from 1.0000 to 1.00001. The results shown are for an Intel Dual Core Atom processor with four threads running on a Windows notebook.

This problem is interesting because it involves both recursion and iterative loops. The outer level of processing is the recursive part where if the iterative part for a high number of iterations significantly differs from a small number of iterations, then the overall range is split in half and the algorithm is rerun with two workers tackling the smaller sub-range.

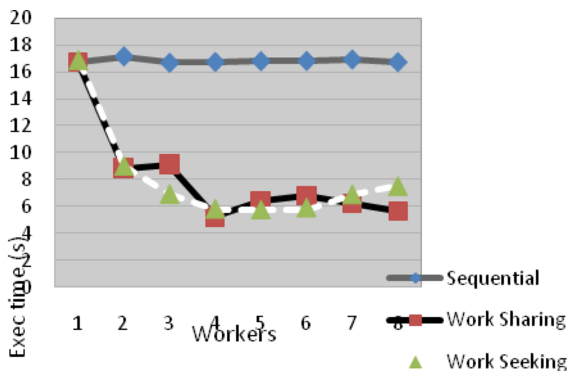


Figure 12 Parallel Recursive Integration

Once parallelism has been applied at the higher level part of an algorithm, it has been found that attempting to apply further parallelism to nested parts of the code typically results in poor results. The added parallelism code provides no benefits, but adds overhead to the processing.

Similarly, adding parallelism only to nested parts of an algorithm, while being called iteratively by a higher level sequential section of code may also lead to poor results, if the overhead of creating worker threads outweighs the speed gains by the parallelism.

Thus for this test, parallelism was only applied at the top level of the algorithm, which was the recursive section. We see in Figure 12 that work seeking appears to give more consistent results, but both strategies generate favourable results. It is interesting to note in this case that increasing worker beyond 8 generally lead to deteriorating work-seeking performance, whereas work-sharing performance seemed to level off.

4 Conclusions

The results show that for iterative parallelism, work-seeking is a reliable choice that generates consistently good results. Work-stealing should not be used if the number of iterations in the loop is high, and the amount of processing during each iteration is minimal.

For these situations work-stealing may generate results considerably worse than the sequential version of the code. On the other hand, there are cases where work-stealing can outperform work-sharing and work-seeking, though, in these cases the work-seeking results are typically very close to the best work-stealing results.

Similarly, there are cases where work-sharing can outperform work-seeking and work-stealing, but in these cases, work-seeking typically generates results very close the work-sharing times. For recursive parallelism, it can be difficult to suggest a general approach, as it can be difficult to determine which strategy will perform the best.

If the work-load is unbalanced, a work-seeking approach might be worth trying; otherwise the work sharing approach might be the best choice. It may be necessary to try both forms to see which one best fits the problem.

One other disadvantage of work-stealing that should be mentioned is that it can only be applied to while loops, since the technique involves modifying the loop termination values which isn't possible with for loops in Ada. Work-sharing and work-seeking do not have this limitation, and can be used with for loops or while loops.

References

[1] Frigo M., Halpern P., Leiserson C., and Lewin-Berlin S., “Reducers and Other Cilk++ Hyperobjects”, *ACM SPAA ’09* (2009).

[2] Moore B., “Parallelism generics for Ada 2005 and beyond”, SIGAda'10 Proceedings of the ACM SIGAda annual conference on SIGAda.

[3] Moore B., Paraffin, http://paraffin.sourceforge.net/ (Feb 2011)

[4] Taft, S.T., Duff, R. A., Bruckardt, R.L. And Plödereder, E. Eds (2000). Consolidated Ada Reference Manual. LNCS 2219, Springer-Verlag.

[5] Barney Blaise, Lawrence Livermore National Laboratory, https://computing.llnl.gov/tutorials/openMP/#WorkSharing (Sept 2010)

[6] Fox, G., Williams, R., Messina G., Parallel Computing Works!, ISBN 1-55860-253-4 Morgan Kaufmann Publishers, Inc., 1994.

[7] Walker J., Red Black Trees, http://www.eternallyconfuzzled.com/tuts/datastructures/jsw_tut_rbtree.aspx (Aug 2010)

[8] Menabrea, L.F., Lovelace A., Sketch of the Analytical Engine Invented by Charles Babbage, *Bibliothèque Universelle de Genève*, October, 1842, No. 82.

[9] Kaneko M., The Akiyama-Tanigawa algorithm for Bernoulli Numbers, Journal of Integer Sequences, Vol 3 (2000), Article 00.2.9.

[10] Free Software Foundation, GMP, http://gmplib.org, Sept 2010.

[11] Anton, H., Rorres. C, Elementary Linear Algebra, Drexel University (2010).

[12] Barnes, J., Programming in Ada 2005, Addison Wesley, 2006, p202-203.

[13] Sanchit K., Integration and Area Under Curve Using Limit of Sums, http://www.dreamincode.net/code/snippet325.htm, Feb 2011.

Designing ParaSail – Parallel Specification and Implementation Language

S. Tucker Taft

SofCheck, Inc, 11 Cypress Drive, Burlington, MA 01803 USA.; Tel: +1 781 850 7068; email: stt@sofcheck.com

Abstract

This (edited) blog follows the trials and tribulations of designing a new programming language designed to allow productive development of parallel, high-integrity (safety-critical, high-security) software systems. The language is named "ParaSail" for Parallel, Specification and Implementation Language.

Keywords: parallel programming language, high-integrity software, formal methods.

Introduction

This article is an extract from a blog started in September 2009 describing the process of designing ParaSail, a new programming language for high-integrity, parallel programming. If you wish to read the full blog, please visit [1]. We have left out large parts of the blog, which in its entirety would run to about 35 pages. We have included editorial comments in *"[...]"* both to describe what we are leaving out, and to describe changes since a given blog entry was written. We have also fixed the examples to use the latest ParaSail syntax, to avoid creating confusion. Note that the online blog entries have not been edited, so if you read those, beware that some of the syntactic details have been changed over time.

September 2009

Why design a new programming language?

So why would anyone want to design a new programming language? For some of us who have the bug, it is the ultimate design project. Imagine actually creating the language in which you can express yourself. But there is another reason. I have been in the software business for over 40 years, and despite everything that might have been said to the contrary, I still believe that a well-designed programming language can result in more productive programmers building higher quality software. In the particular area of high-integrity software, including both safety-critical software and high-security software, there is all the more reason to use the very best programming language you can, because the problems you are trying to solve and the level of quality required is at the very limits of what can be accomplished.

This new language is meant to address the goals of producing inherently safe and secure software, while taking advantage of the wider availability of true parallel processing in the form of multi-core chips. It is intended to promote a formal approach to software, where the program text includes pre- and postconditions, liberal use of assertions and invariants, etc., with tool-supported proof of correctness with respect to the formal annotations.

The language is tentatively named **ParaSail**, for Parallel Specification and Implementation Language. I would have spelled it "ParaSAIL" but for the danger of confusion with the original Stanford AI Language, "SAIL" [2], and its more modern follow-on "MAINSAIL" (for Machine Independent SAIL). I don't mind making the connection with SAIL, as it was a very interesting language in its day, and MAINSAIL remains worth a look today. ParaSail is a completely new language, but it steals liberally from other programming languages, including the ML series (SML, CAML, OCAML, etc.), the Algol/Pascal family (Algol, Pascal, Ada, Modula, Eiffel, Oberon, etc.), the C family (C, C++, Java, C#), and the region-based languages (especially Cyclone). Perhaps one significant deviation from the excellent baseline established by SAIL, ML, Eiffel, Java, etc. is that ParaSail is intended to avoid "fine-granule" garbage collection in favor of stack and region-based storage management.

Why blog about designing a programming language?

So why did I decide to start this blog? Designing a new language is a long process, but it is hard to find someone who is willing to sit down and discuss it. Those who like to design languages generally have their own strong biases, and the discussions tend to be more distracting than satisfying, because there are so many good answers to any interesting language design question. Those who don't have any interest in designing a new language tend to lack the patience to even talk about it, as they believe we already have more than enough programming languages, and all we need are better tools, better processes, and better training to be more productive and to achieve higher quality.

So writing a blog seems like a nice way to record the process, to perhaps get some feedback (though that may be optimistic), and to hopefully make progress by being forced to actually get the ideas down onto "paper."

That's probably enough meta-discussion for now. In the next post I plan to dive into the technical issues.

ParaSail language themes and philosophy

So what will make ParaSail an interesting programming language? What is the philosophy behind the language? ParaSail tries to minimize implicit operations, implicit parameters, implicit dynamic binding (virtual function calls), implicit initializations, implicit conversions, etc. This is both in the name of clarity for the human reader, and in the name of formal testability and verifiability. ParaSail uses a small number of concepts to represent all of the various composition mechanisms such as records, packages, classes, modules, templates, capsules, structures, etc. Arrays and more general containers are treated uniformly.

On the other hand, ParaSail allows many things to proceed in parallel by default, effectively inserting implicit parallelism everywhere. Parameter evaluation is logically performed in parallel. The language disallows uses that would make the result depend on the order or concurrency of parameter evaluation. The iterations of a **for** loop are by default executed in parallel. Explicit ordering must be specified if it is required by the algorithm. Even sequential statements are essentially converted into a data-flow based DAG which is then evaluated in parallel in so far as possible. In all cases, the language disallows code that could result in race conditions due to inadequately synchronized access to shared data, either by using per-thread data, structured safe synchronization, or a handoff semantics (similar to that of linear types, distributed languages like Hermes [3], or the UVM virtual memory system). …

Modules in ParaSail

…ParaSail has only two kinds of modules: interface modules and class modules. A class module implements one or more interface modules. Perhaps a class ought to be called an implementation module, but that is an awfully long word, and in many ways a class module matches what languages like Simula and C++ and Java and C# and Eiffel call a class. In general ParaSail tries to use familiar terminology in familiar ways. Of course there is danger of confusion, but hopefully the benefits of familiarity outweigh the dangers of confusion. *[We have subtly altered the vocabulary since this was written. We now refer to the interface* of *a module, and the class that* defines *the module. So there is really only one kind of module, but like a Modula module or an Ada package, it has a part that declares the external interface for the module, and a part that defines the internals of the module. The "class" part is optional. It need not be provided if the module is* ***abstract****, or if there are no operations in the interface, merely components.]*

Both interface and class modules are parameterized, essentially generic templates, as used in Ada, C++, Java, etc. Because of their wide-spread adoption, ParaSail uses "<...>" for generic parameters. While we are talking about notation, ParaSail uses "(...)" for normal function/procedure parameters, uses "[...]" for selecting from an array or similar container. ParaSail uses words for bracketing control flow (e.g. "if ... then ... else ... end if;"). ParaSail uses "{...}" for annotations such as constraints, assertions, pre/post-conditions, etc. This is intended to be familiar from Hoare logic, which uses the notation "{P} S {Q}" to represent a statement S with precondition P and postcondition Q.

Interface modules have the following basic syntax:

```
interface name < generic_parameters >
is
   interface_items
end interface name;
```

The interface_items comprise a sequence of function, procedure, and nested interface specifications. In addition constants and variables may be declared, but these are considered equivalent to corresponding functions. … In addition, operators may be declared. Operators are essentially functions with special syntax for calling them.

As subtly implied above, an interface defines a type, and within the interface module, the interface's name represents that type. A type is essentially an interface that has been bound to a set of actual generic parameters. There are also subtypes in ParaSail, which are types with some additional constraints (such as a range limitation), specified inside "{...}".

Generic parameters are themselves either types derived from specified interfaces, or static constants used to parameterize the implementation of the interface, and/or select among different implementations of the same interface. A generic array interface, with its generic parameter list, might look like this:

```
interface Array < Element_Type is Assignable<>;
             Index_Type is Discrete<>>
is
  function First(Arr : Array) -> Index_Type;
  function Last(Arr : Array) -> Index_Type;
  function Element(Arr : ref Array;
    Index : Index_Type {Index in First(Arr)..Last(Arr)})
      -> ref Element_Type;
  function Create_Array(First, Last : Index_Type)
      -> Result:Array
        {First(Result) = First; Last(Result) = Last};
  ...
end interface Array;
```

Note that there is no implicit parameter in ParaSail (identified as `this` or `self` in many object-oriented languages). All parameters are explicit….

More on ParaSail interfaces

Continuing with the above example… To make indexing into an Array look more familiar, we might want to use the "[]" operator instead of the function Element:

```
operator "[]"(Arr : ref Array;
    Index : Index_Type {Index in First(Arr)..Last(Arr)})
      -> ref Element_Type;
```

Now we could use this interface roughly as follows:

```
var Names: Array<String, Student_ID> :=
  Create_Array(First => First_Student_ID,
        Last => Last_Student_ID);
Names[ID_Of_Mike] := "Mike";
```

This kind of thing should look pretty familiar.

A few things to note: There is no separate notion of a constructor. A function that returns an object of the type defined by the enclosing interface is effectively a constructor. A function that returns a **ref** is not a constructor, but may be used on the left-hand side of an assignment, provided the **ref** parameter(s) to the function can be used on the left-hand side. Parameters are by default read-only, but **ref var** parameters can be updated. The result of a function is specified after a "->". If the function has multiple results, they are enclosed in parentheses in the same way as the (input) parameters:

```
function Two_Outputs(X : Input_Type) ->
    (Y : Output_Type1; Z : Output_Type2);
```

Once we start talking about statements, we will see that in ParaSail, as in C++ and Java, declarations and executable statements may be interspersed. All declarations start with a reserved word, such as **var**, **const**, **interface**, **type**, etc. so they are easy to distinguish from assignment statements, procedure calls, etc.

Next time we will discuss how a class implements an interface.

How a ParaSail class implements its interface

In ParaSail, a class implements an interface. Unless an interface is marked as **abstract**, there is required to be a class with the same name as the interface, which provides the default implementation of the interface. There may be multiple implementations of an interface. Some may be specializations of the default. Specializations specify certain restrictions on the generic parameters, along with a preference level, which allows the compiler to choose the appropriate implementation automatically if not specified explicitly.

Here is a class implementing the interface Array shown in an earlier post:

```
class Array < Element_Type is Assignable<>;
          Index_Type is Discrete<>>
is
  const First : Index_Type;
  const Last : Index_Type;
  function Length(Arr : Array) -> Integer is
    return Last-First+1;
  end function Length;
  var Data : Basic_Array<Element_Type>;
exports
  function First(Arr : Array) -> Index_Type is
    return Arr.First;
  end function First;
  ...
  operator "[]"(Arr : ref Array;
    Index : Index_Type {Index in First(Arr)..Last(Arr)})
      -> ref Element_Type is
    return Data[Index-First];
  end operator "[]";
  function Create_Array(First, Last : Index_Type)
    -> Result:Array
      {First(Result) = First and Last(Result) = Last}
  is
    return (First => First, Last => Last,
      Data => Create(Last-First+1));
  end function Create_Array;
  ...
end class Array;
```

A class must include implementations for each operation in its interface, plus any number of local declarations as necessary to implement the exported operations. The exported declarations inside a class follow the word **exports**. Declarations preceding **exports** are local to the class. Even though the interface associated with the class implies what operations are exported, we make the distinction explicit in the class itself, since changing the specification of an exported declaration has a much larger implication than does changing that of a local declaration. Also, by segregating the exported declarations at the end, we make it easier to find them. Finally, because a class might implement interfaces other than its own, the exported declarations allow it to fulfill the set of operations of other interfaces. C uses "extern" vs. "static" to make a similar distinction. Java uses "public" vs. "private". …

Resolving names in ParaSail

In ParaSail, the names exported by an interface can in most cases be used without any explicit qualification of the interface from which they came. Hence, continuing with our Array interface example, we can generally write "First(Arr)" without having to specify which "First" function we mean, presuming we know the type of Arr. If qualification is necessary, the "::" notation of C++ is used, such as "Array::First(Arr)". ParaSail reserves "." for selecting a component of a composite object, such as "Obj.Field", or as a way of indicating an operation is defined within the module defining the type of the object, such as "Arr.First()", which is equivalent to "<type_of_Arr>::First(Arr)".

…

ParaSail extension, inheritance, and polymorphism

ParaSail fully supports object-oriented programming, including interface and class extension/inheritance/reuse, and both static (compile-time) and dynamic (run-time) polymorphism.

Interface extension is the most straightforward. One interface can be defined to extend another, meaning that it inherits all of the operations and generic parameters of some existing interface, and optionally adds more of each:

```
interface Extensible_Array extends Array is
  operator "[]"(Arr : ref Extensible_Array;
    Index : Index_Type {Index >= First(Arr)})
      -> ref Element_Type {Last(Arr) >= Index};
end interface Extensible_Array;
```

Here we have essentially the same operation, but now the array automatically grows (at only one end) if the indexing operation is applied to it with an index that is greater than the prior value of Last(Arr). Note that the generic parameters for Extensible_Array are all inherited as is from Array.

The operations that are not redeclared in the new interface are inherited from Array, but with each occurrence of Array replaced systematically with Extensible_Array.

Unless Extensible_Array is declared as an **abstract** interface, there must be a corresponding Extensible_Array **class** that provides its default implementation. The Extensible_Array class might be defined as an extension of the Array interface (and indirectly its associated class), but it need not be. *[We now require that a class **extends** another module if and only if its interface also **extends** that module. The reserved word **implements** is used (rather than **extends**) when an interface inherits from another module's interface but does* not *inherit the code from the other module's class. An interface may specify any number of other modules after the reserved word **implements** but only one after **extends**.]* … Here is what it would look like if it were an extension of the Array interface.

```
class Extensible_Array extends Parent: Array
is
  ...
exports
  operator "[]"(...) -> ... is ... end operator "[]";
  function Create_Array(...) -> Extensible_Array is ...
  end function Create_Array;
end class Extensible_Array;
```

Local operations and objects could be defined as needed, and exported operations could be redefined as appropriate. An object of the interface being extended (the underlying interface) is created as a local variable of the class (this variable is the *underlying object*). A name can be given to the underlying object -- we use "Parent" above.

Note that the class is extending an *interface*, rather than directly another class, and in fact any implementation of the interface can later be optionally specified when using Extensible_Array. Effectively the actual class to use as the implementation of the underlying interface becomes another optional generic parameter. If unspecified, it defaults to the default implementation of the underlying interface.

For those operations of the interface that are not provided explicitly in the class, an inherited operation is provided, based on the corresponding operation of the underlying interface. The implementation of this inherited operation invokes the corresponding operation of the Array interface, passing it the underlying Array object of each operand of type Extended_Array. …

Although an interface may *[implement]* any number of other interfaces, a class can only extend one interface. Of course it may have local variables of many different interfaces, but only one of them is the underlying object, and only inherited operations corresponding to operations on the underlying interface will be automatically provided.

One important thing to note about how ParaSail implementation inheritance works. It is a completely *black box*. Each implicitly provided inherited operation calls the corresponding operation of the underlying interface, passing it the underlying objects of any operands of the type being defined. The underlying interface operation operates only on the underlying object(s), having no knowledge that it was called "on behalf" of some extended class. If the underlying operation calls other operations, they too are operating only on the underlying object(s). There is no "redispatch" on these internal calls, so the extended class can treat these underlying operations as black boxes, and not worry that if it explicitly defines some operations while inheriting others, that that might somehow interact badly with how the underlying operations are implemented.

Which brings us to *polymorphism*. Static, compile-time polymorphism has already been illustrated, through the use of generic parameters, and the name resolution rules. We also see it here in that the particular class implementing the underlying interface for an extended class can also vary depending on the particular instantiation of the extended class. Although we didn't mention the syntax for that, it makes sense for it to use the **extends** keyword as well at the point when we are providing the actual generic parameters:

```
var XArr : Extended_Array
  extends Sparse_Array := Create_Array(...);
```

Now what about dynamic, run-time polymorphism? When do we actually need it? A classic example is that of a heterogeneous tree structure, where all of the nodes in the tree are extensions of some basic Tree_Node type, but each is specialized to carry one particular kind of information or another. Static polymorphism is great for creating general purpose algorithms and data structures, but they are inevitably homogeneous to some degree. The algorithm manipulates numbers all of the same precision, or the data structure holds elements all of the same type. With dynamic polymorphism, heterogeneity rules.

In ParaSail, dynamic polymorphism is indicated by appending a "+" to a type name *[(e.g. `Tree_Node+`)]*. What this means is that the corresponding object, parameter, or result might be of a type coming from any extension of the given type's interface, with the proviso that the generic parameters inherited from this original interface have the same bindings for the two types. That is important because, without that, the operations shared between the two types would not necessarily take the same types of parameters. …

October 2009

ParaSail pass-by-reference …

ParaSail has no global variables

ParaSail has no global variables. So how does that work? Global variables have long been recognized as trouble makers in the software world, and they get much worse in a language with lots of parallelism. But can we eliminate global variables completely? ParaSail tries to do that by eliminating any notion of statically allocated variables. In C or C++, global variables are those marked static or extern, or in a language with modules/packages like Ada, global variables are those declared at the module level. Although their declaration might be hidden, such variables still represent global, variable state, and still create trouble.

In ParaSail, interface modules are generic types, and class modules implement interfaces. That is, after providing generic parameters to an interface, you end up with a type. Any variable declared in a class is what is often called an *instance variable*, that is, there is one per object of the type defined by the class. There is nothing corresponding to what are called *class variables*. In other words, the variables declared in a class will always be effectively fields/components of some enclosing object. You can of course also have local variables inside the body of a procedure or function, but these are never statically allocated; they are stack-resident variables only (automatic in C parlance).

…

ParaSail's implicit parallelism

As mentioned in the original overview of ParaSail, implicit parallelism is at the heart of ParaSail (and of its name!). Every procedure/function call with multiple parameters involves implicit parallelism, in that all of the parameters are evaluated in parallel. Handoff semantics is used for writable parameters, meaning that a (non-concurrent) object that is writable by one parameter evaluation, isn't available to any other parameter evaluation. Operations on concurrent objects are not ordered.

Furthermore, in a sequence of statements, the only default limitation on parallelism is that imposed by operations on non-concurrent objects. Parallelism can be inhibited by using ";;" instead of merely ";" to separate two statements. This forces sequencing for operations on concurrent objects, which would otherwise not have any specified order. By contrast, "||" can be used to indicate that parallelism is expected, and it is an error if there are conflicting operations on non-concurrent objects in the statements on either side of the "||". That is, two statements separated by "||" are executed effectively in parallel, just as though there were embedded within two different parameter evaluations to a single procedure or function call. *[We have changed the precedence for these operators since this blog entry, and added another operator "**then**" (equivalent to ";;") with lowest precedence, "||" higher, and ";" and ";;" at the highest precedence (bind most tightly).]* For example:

```
   X := F(A, B);
   Y := G(B, C);
then
   Z := H(R, S);
|| Z2 := Q(R, T);
```

So in the above, the assignments to X and Y are ordered only if that would be required by the shared use of B (at least one of them has B as a writable parameter). The assignments to Z and Z2 occur in parallel, and it would be an error if the shared use of R is unsafe (e.g. because R is writable by one or both, and is not a concurrent object). The assignments to X and Y will complete before beginning the assignments to Z and Z2 (though of course, a "very intelligent" compiler might still find parts that can be executed in parallel because they can't possibly affect one another). In all cases the evaluations of the parameters to a single procedure/function call happen in parallel. …

Implicit parallelism also shows up in ParaSail loops. The iterations of a **for** loop in ParaSail are executed as though each iteration were separated from the next by a ";", but with no particular ordering on the iterations. A particular ordering may be specified by using the **forward** or **reverse** keyword, which effectively puts a ";;" between the iterations. Alternatively, the **concurrent** keyword may be used to indicate that the iterations should be effectively separated by "||", in which case it is an error if there are any conflicts due to operations on a non-concurrent variable. …

Using "=?" in ParaSail to implement "==", "!=", "<=", ">=", etc.

Here is a simple ParaSail feature. It is often the case that a user-defined type will have its own definition for equality and comparison. However, it is always a bit of an issue to make sure that "==" and "!=" are complements, and similarly for "<" and ">=", ">" and "<=". ParaSail skirts this issue by requiring the user to define only one operator, "=?", which returns a value from the enumeration #less, #equal, #greater, #unordered. The comparison and equality operators are defined in terms of "=?" (which is pronounced "compare"), in the natural way…

November 2009

ParaSail region-based storage management ...

ParaSail concurrent interfaces

A ParaSail interface can be marked as **concurrent**, meaning that an object of a type produced by instantiating such a concurrent interface can be accessed concurrently by multiple parallel threads. We use the term *concurrent type* for a type produced by instantiating a concurrent interface, and *concurrent object* for an object of a concurrent type.

Both *lock-based* and *lock-free* synchronization techniques can be used to coordinate access to a concurrent object. In addition, *operation queues* are provided to hold operation requests that cannot be serviced until the concurrent object attains a desired state, as specified by the *dequeue condition*. When a thread initiates an operation request, if it can't be performed immediately because the condition is False, the thread is blocked until the request gets dequeued and the associated operation is completed, or the operation request is canceled for some reason, such as a timeout. …

With a lock-based concurrent interface, one of the operands of a procedure or function of the interface … can be marked indicating it should be **locked** upon call. When calling the operation, the operand's lock will be acquired automatically upon entry to the operation, and the lock will be released upon return. Alternatively, one operand of a concurrent interface operation may be marked **queued**, meaning that the *dequeue condition* must be true before the operation will be performed, and when it is performed it will be locked automatically as well.

Within the concurrent class implementing a lock-based concurrent interface, the components of an operand marked **locked** or **queued** may be referenced directly, knowing that the access is single-threaded at that point. If there are other (non-locked/queued) operands of the associated concurrent type, their non-concurrent components may *not* be referenced by the operation. An operation with a **queued** operand has the further assurance that at the point when the operation starts, the dequeue condition is True. The dequeue condition for a given operation may depend on the values of the non-concurrent parameters, plus the internal state of the (concurrent) operand marked **queued**.

Within the concurrent class implementing a *lock-free* concurrent interface, all data components must themselves be concurrent objects, since no locking is performed on entry to the operations of the class. Typically these components will be of low-level concurrent types, such as a single memory cell that supports atomic load, atomic store, and atomic compare-and-swap (CAS), or a somewhat higher-level concurrent type that supports multi-word-compare-and-swap (MCAS). The ParaSail standard library provides a small number of such lower-level concurrent types to support implementing lock-free concurrent interfaces. The higher level **queued** operations are implemented using a lower-level race-free and lock-free concurrent type similar to a private semaphore, which can be used directly if desired.

Here is an example of a concurrent interface and corresponding concurrent class for implementing a simple bounded buffer:

```
concurrent interface Bounded_Buffer
  <Element_Type is Assignable<>;
   Index_Type is Integer<>> is
    function Create_Buffer(
      Max_In_Buffer : Index_Type {Max_In_Buffer > 0})
      -> Result : Bounded_Buffer;
      // Create buffer of given capacity
    procedure Put(Buffer : queued Bounded_Buffer;
      Element : Element_Type);
      // Add element to bounded buffer;
      // remain queued until there is room
      // in the buffer.
    function Get(Buffer : queued Bounded_Buffer)
      -> Element_Type;
      // Retrieve next element from bounded buffer;
      // remain queued until there is an element.
end interface Bounded_Buffer;

concurrent class Bounded_Buffer is
    const Max_In_Buffer : Index_Type
      {Max_In_Buffer > 0};
    var Data : Array<Element_Type, Index_Type> :=
      Create_Array(1, Max_In_Buffer);
    var Next_Add : Index_Type
      {Next_Add in 1..Max_In_Buffer} := 1;
    var Num_In_Buffer : Index_Type
      {Num_In_Buffer in 0..Max_In_Buffer} := 0;
  exports
    function Create_Buffer(
      Max_In_Buffer : Index_Type {Max_In_Buffer > 0})
      -> Bounded_Buffer is
        return (Max_In_Buffer => Max_In_Buffer);
    end function Create_Buffer;
    procedure Put(Buffer : queued Bounded_Buffer;
      Element : Element_Type) is
      queued until
        Buffer.Num_In_Buffer < Buffer.Max_In_Buffer then
        Buffer.Data[Buffer.Next_Add] := Element;
        // Advance to next element, cyclically.
        Buffer.Next_Add :=
         (Buffer.Next_Add mod Buffer.Max_In_Buffer) + 1;
        Buffer.Num_In_Buffer += 1;
    end procedure Put;
    function Get(Buffer : queued Bounded_Buffer)
      -> Element_Type is
      queued until Buffer.Num_In_Buffer > 0 then
         return Buffer.Data[
           ((Buffer.Next_Add - Buffer.Num_In_Buffer - 1)
             mod Buffer.Max_In_Buffer) + 1];
    end function Get;
end class Bounded_Buffer;
```

This concurrent interface has one constructor, plus two queued operations, Put and Get. The dequeue conditions are provided as part of the implementation of an operation with a queued operand. Here the dequeue conditions are Buffer.Num_In_Buffer < Buffer.Max_In_Buffer for Put, and Buffer.Num_In_Buffer > 0 for Get. These dequeue conditions ensure that when the operations are performed, they can proceed without an error. …

December 2009

ParaSail end-of-scope operator

When an object goes out of scope in ParaSail it may require some amount of cleanup. For example, if the object provided access to some external resource, then when the object goes away, some release action on the external resource might be required. This is provided by the "end" operator. …

ParaSail module details …

ParaSail character, string, and numeric literals

… The *[five]* basic kinds of literals in ParaSail, and their corresponding universal types, are as follows:

kind of literal	example	universal type
string literal	"this is a string literal"	Univ_String
character literal	'c'	Univ_Character
integer literal	42	Univ_Integer
real literal	3.14159	Univ_Real
enum literal	#blue	Univ_Enumeration

The universal types can be used at run-time, but they are primarily intended for use with literals and in annotations. Univ_String corresponds to UTF-32, which is a sequence of 32-bit characters based on the ISO-10646/Unicode standard [4]. Univ_Character corresponds to a single 32-bit ISO-10646/Unicode character (actually, only 31 bits are used). Univ_Integer is an "infinite" precision signed integer type. Univ_Real is an "infinite" precision signed rational type, with signed zeroes and signed infinities. *[Univ_Enumeration is described in a later blog entry.]*

The universal numeric types have the normal four arithmetic operators, "+", "-", "*", "/". …

By providing conversions to and from a universal type, a "normal" type can support the use of the corresponding literal. These special conversion operations are declared as follows (these provide for integer literals):

```
operator "from_univ"(Univ : Univ_Integer)
  -> My_Integer_Type;
operator "to_univ"(Int : My_Integer_Type)
  -> Univ_Integer;
```

If an interface provides the operator "from_univ" converting from a given universal type to the type defined by the interface, then the corresponding literal is effectively overloaded on that type. The complementary operator "to_univ" is optional, but is useful in annotations to connect operations on a user-defined type back to the predefined operators on the universal types.

Annotations may be provided on the conversion operators to indicate the range of values that the conversion operators accept. So for a 32-bit integer type we might see the following:

```
interface Integer_32<> is
   operator "from_univ"
    (Univ : Univ_Integer {Univ in -2**31 .. +2**31-1})
     -> Integer_32;
   operator "to_univ"(Int : Integer_32)
     -> Result: Univ_Integer {Result in -2**31 .. +2**31-1};
   ...
end interface Integer_32;
```

With these annotations it would be an error to write an integer literal in a context expecting an Integer_32 if it were outside the specified range.

ParaSail Universal types in Annotations

As explained in the prior entry, ParaSail has *[five]* universal types, *[Univ_Enumeration]*, Univ_Integer, Univ_Real, Univ_Character, and Univ_String, with corresponding literals. The universal numeric types have the usual arithmetic operators, and the universal character and string types have the usual operations for concatenation and indexing. In some ways values of the universal types may be thought of as abstract mathematical objects, while the values of "normal" types are the typical concrete representations of such mathematical objects, with attendant limitations in range or precision. To connect a normal, concrete type to a corresponding universal type, the type defines the operator "from_univ" to convert from the universal type to the concrete type, and "to_univ" to convert back to the universal type.

In the prior entry, we learned that by defining "from_univ" a type can use the corresponding literal notation…. So what about the "to_univ" operator? If "to_univ" is provided, then ParaSail allows the use of a special convert-to-universal notation "[[...]]", which would typically be used in annotations, such as:

```
{ [[Left]] + [[Right]] in First .. Last }
```

This is equivalent to:

```
{ "to_univ"(Left) + "to_univ"(Right) in First .. Last }
```

This notation is intended to be a nod to the double bracket notation used in *denotational semantics* [5] to represent the *denotation* or abstract *meaning* of a program expression. Using this notation we can give a more complete definition of the semantics of the Integer interface:

```
interface Integer<First, Last : Univ_Integer> is
   operator "from_univ"
     (Univ : Univ_Integer {Univ in First .. Last})
     -> Integer;
   operator "to_univ"(Int : Integer)
     -> Result : Univ_Integer {Result in First .. Last};
   operator "+"(Left, Right : Integer
       {[[Left]] + [[Right]] in First .. Last})
     -> Result : Integer
       {[[Result]] == [[Left]] + [[Right]]};
   operator "-"(Left, Right : Integer
       {[[Left]] - [[Right]] in First .. Last})
     -> Result : Integer
       {[[Result]] == [[Left]] - [[Right]]};
   ...
end interface Integer;
```

Here we are defining the pre- and postconditions for the various operators by expressing them in terms of corresponding universal values, analogous to the way that denotational semantics defines the meaning of a program by defining a transformation from the concrete program notations to corresponding abstract mathematical objects.

January 2010

ParaSail thread versus task

In ParaSail, there is a pervasive, implicit parallelism. Because of that, it really doesn't make sense to talk about specific threads of control as there are potentially hundreds or thousands of them, appearing and disappearing all of the time. Hence any notion of thread identity is probably inappropriate. On the other hand, there might still be a reason to identify what might be called a *logical* thread of control, which would perhaps be reasonable to call a *task*, namely a logically separable unit of work. Here it might make sense to have a well defined task identity, and an ability to set a priority, or a deadline, or some sort of resource limits (such as maximum space, or maximum number of simultaneous threads, etc.). …

February 2010

Physical Units in ParaSail …

April 2010

ParaSail BNF …

May 2010

ParaSail enumeration types

We haven't talked about enumeration types in ParaSail yet. One challenge is how to define an enumeration type by using the normal syntax for instantiating a module, which is the normal way a type is defined in ParaSail. Generally languages resort to having special syntax for defining an enumeration type... But that seems somewhat unsatisfying, given that we can fit essentially all other kinds of type definitions into the model of instantiating a module. Another challenge with enumeration types is the representation of their literal values. In many languages, enumeration literals look like other names, and are considered equivalent to a named constant, or in some cases a parameterless function or constructor. Overloading of enumeration literals (that is using the same name for a literal in two different enumeration types declared in the same scope) may or may not be supported.

In ParaSail we propose the following model: We define a special syntax for enumerals (enumeration literals), of the form #name (e.g. #true, #false, #red, #green). We define a universal type for enumerals, Univ_Enumeration. We allow a type to provide conversion routines from/to Univ_Enumeration, identified by operator "from_univ" and operator "to_univ". If a type has a from_univ operator that converts from Univ_Enumeration, then that type is effectively an *enumeration* type, analogous to the notion that a type that has a from_univ that converts from Univ_Integer is essentially an *integer* type. As with other types, double-bracket notation applied to a value of an enumeration type, e.g. [[enumval]], is equivalent to a call on to_univ(enumval), and produces a result of Univ_Enumeration type. An additional special Boolean "in" operator which takes one Univ_Enumeration parameter determines which enumerals are values of the type. The notation "X in T" is equivalent to a call on the operator T::"in"(X). The precondition on from_univ would generally be {univ in Enum_Type}.

Here is an example:

```
interface Enum<Enumerals :
   Vector<Univ_Enumeration>> is
  operator "in"(Univ_Enumeration) -> Boolean<>;
  operator "from_univ"
    (Univ : Univ_Enumeration {Univ in Enum}) -> Enum;
  operator "to_univ"(Val : Enum)
    -> Result: Univ_Enumeration { Result in Enum };
  ...
end interface Enum;
...
type Color is Enum<[#red,#green,#blue]>;
var X : Color := #red;  // implicit call on from_univ
```

Here we presume the class associated with the Enum interface implements "in", "from_univ", and "to_univ" by using the Enumerals vector to create a mapping from Univ_Enumeration to the appropriate value of the Enum type, presuming the enumerals map to sequential integers starting at zero. A more complex interface for creating enumeration types might take such a mapping directly, allowing the associated values to be other than the sequential integers starting at zero. The built-in type Boolean is presumed to be essentially Enum<[#false,#true]>.

So why this model? One is that it fits nicely with the way that other types with literals are defined in ParaSail, by using from_univ/to_univ. Secondly, it naturally allows overloading of enumeration literals, in the same way that the types of numeric literals are determined by context. Finally, it makes a strong lexical distinction between enumeration literals and other names. This could be considered a bug, but on balance we believe it is useful for literals to stand out from, and not conflict with, normal names of objects, types, operations, and modules. Note that this model is reminiscent of what is done in Lisp with the quote operator applied to names, e.g. 'foo. It also borrows from the notation in Scheme of #f and #t for the literals representing false and true.

ParaSail without pointers?

We are working on defining the semantics for pointers in ParaSail. As described in an earlier entry, we plan to use region-based storage management rather than more fine-grained garbage collection. However, there is the question of how pointers are associated with their region, and more generally how pointers are handled in the type system. We are considering a somewhat radical approach: effectively *eliminate* most pointers, replacing them with *generalized container indexing* and *expandable objects*. …

By *generalized container indexing*, we mean using an abstract notion of index into an abstract notion of container. The most basic kind of container is an array, with the index being an integer value or an enumeration value. Another kind of container is a hash table, with the index (the key) being something like a string value. …

By *expandable objects*, we mean treating a declared but not initialized object (including a component of another object) as a kind of stub into which a value can be stored. Even an initialized object could be overwritten via an assignment which might change the size of the object. Finally, an initialized object could be set back to its stub state, where the object doesn't really exist anymore….

Expandable objects eliminate a common use of pointers as a level of indirection to support objects of variable or unknown size. We propose to provide qualifiers **optional** and **mutable** on objects to indicate that they are expandable. The qualifier **optional** means that the object starts out **null**, but may be assigned a non-**null** value. The qualifier **mutable** means that the object may be assigned multiple values of different sizes during its lifetime. If both are given, then the object starts out **null**, can be assigned non-**null** values of varying size during its lifetime, and can be assigned back to **null**. …

Handling concurrent events in ParaSail …

Talking about ParaSail …

June 2010

Generalized For loops in ParaSail

As we start to dive into the lower-level control structures for ParaSail, one clearly very important structure is the **for** loop. We have already suggested the basic structure:

```
for I in 1..10 [forward | reverse | concurrent] loop
```

However, it would be nice if the **for** loop could also be used for more general looping structures. A number of languages have adopted a for loop where there is an initialization, a "next step", and a termination/continuation condition….

The above considerations lead us to the following possible approach to a generalized for loop in ParaSail…:

```
for T => Root then T.Left || T.Right
  while T != null loop
    … loop body
end loop
```

This would represent a "loop" which on each iteration splits into two threads, one processing T.Left and the other T.Right. The iteration stops when all of the threads have hit a test for T != null which returns false. …

If we were to specify **concurrent loop** then it would presumably imply that the daughter threads are to be created immediately once the test was found to be true, not waiting for the loop body of the parent thread to finish. This would effectively create a thread for every node in the tree, with the loop body executing concurrently for each node.

An intentional race condition in a ParaSail concurrent loop

An intriguing question, given the presence of the concurrent loop construct in ParaSail, is what would happen if there were a loop **exit** or a **return** statement in the body of a **concurrent** loop. Earlier we have said that only **forward** and **reverse** loops would allow an **exit** statement. But perhaps we should allow an **exit** or a **return** from within a concurrent loop as a way of terminating a concurrent computation, aborting all threads besides the one exiting or returning. This would essentially be an intentional race condition, where the first thread to **exit** or **return** wins the race. This would seem to be a relatively common paradigm when doing a parallel search, where you want to stop as soon as the item of interest is found in the data structure being walked, for example, and there is no need for the other threads to continue working.

…

For example, using the generalized for loop from the previous posting:

```
var Winner : optional Node := null;
for T => Root then T.Left || T.Right
  while T != null concurrent loop
    if T.Key == Desired_Key then
       exit loop with Winner => T;
    end if;
end loop;
```

At this point Winner is either still **null** if no node in the tree has the Desired_Key, or is equal to some node whose Key equals the Desired_Key. If there is more than one such node, it would be non-deterministic which such node Winner designates.

Clearly this kind of brute force search of the tree would not be needed if the tree were organized based on Key values. This example presumes the Key is not controlling the structure of the tree. Also note that if this were a function whose whole purpose was to find the node of interest, the **exit loop with** ... statement could presumably be replaced by simply **return** T and we wouldn't need the local variable Winner at all.

Flex-based lexical scanner for ParaSail …

(A)Yacc-based parser for ParaSail …

Some additional examples of ParaSail …

July 2010

N Queens Problem in ParaSail

Here is a (parallel) solution to the "N Queens" problem in ParaSail, where we try to place N queens on an NxN chess board such that none of them can take each other. This takes the idea of using the "**continue**" statement as a kind of implicit recursion to its natural conclusion. This presumes you can turn a "normal" data structure like Vector<> into a concurrent data structure by using the keyword "**concurrent**," which presumably means that locking is used on all operations to support concurrency. It is debatable whether this use of a "**continue**" statement to effectively start the next iteration of a loop almost like a recursive call is easier or harder to understand than true recursion. *[This example has been updated and corrected.]*

```
interface N_Queens <N : Univ_Integer := 8> is
  // Place N queens on an NxN checkerboard so that
  // none of them can "take" each other.
  type Chess_Unit is new Integer<-N*2 .. N*2>;
  type Row is Chess_Unit {Row in 1..N};
  type Column is Chess_Unit {Column in 1..N};
  type Solution is Array<optional Column, Indexed_By => Row>;

  function Place_Queens() -> Vector<Solution>
   {for all Sol of Place_Queens => for all Col of Sol => Col not null};
end interface N_Queens;

class N_Queens is
  type Sum_Range is Chess_Unit {Sum_Range in 2..2*N};
  type Diff_Range is Chess_Unit {Diff_Range in (1-N) .. (N-1)};
  type Sum is Set<Sum_Range>;
  type Diff is Set<Diff_Range>;
 exports
```

```
   function Place_Queens() -> Vector<Solution>
     {for all Sol of Place_Queens => for all Col of Sol => Col not null}
   is
     var Solutions : concurrent Vector<Solution> := [];
    *Outer_Loop*
     for (C : Column := 1; Trial : Solution := [.. => null];
       Diag_Sum : Sum := []; Diag_Diff : Diff := []) loop
         // Iterate over the columns
         for R in Row concurrent loop
             // Iterate over the rows
             if Trial[R] is null and then
               (R+C) not in Diag_Sum and then
               (R-C) not in Diag_Diff then
                 // Found a Row/Column combination that is
                 // not on any diagonal already occupied.
                 if C < N then
                     // Keep going since haven't reached Nth column.
                     continue loop Outer_Loop with (C => C+1,
                       Trial      => Trial | [R => C],
                       Diag_Sum   => Diag_Sum | (R+C),
                       Diag_Diff  => Diag_Diff | (R-C));
                 else
                     // All done, remember result with last queen placed
                     Solutions |= (Trial | [R => C]);
                 end if;
             end if;
         end loop;
     end loop Outer_Loop;
     return Solutions;
   end function Place_Queens;
end class N_Queens;
```

Updated (A)Yacc grammar for ParaSail …

Pointer-free primitives for ParaSail …

August 2010

Eliminating the need for the Visitor Pattern in ParaSail …

Ad hoc interface matching in ParaSail …

Initial implementation model for ParaSail types …

No exceptions in ParaSail, but exitable multi-thread constructs

We have been mulling over the idea of exceptions in ParaSail, and have pretty firmly concluded that they aren't worth the trouble. In a highly parallel language, with lots of threads, exception propagation across threads becomes a significant issue, and that is a nasty area in general. Also, exceptions can introduce many additional paths into a program, making thorough testing that much harder. And the whole business of declaring what exceptions might be propagated, and then deciding what to do if some other exception is propagated can create numerous maintenance headaches.

There is a feature in ParaSail as currently designed which provides some of the same capabilities of exceptions, but is particularly suited to parallel programming. This is the "**exit with**" statement, which allows a construct to be exited with one or more values specified as results, and at the same time terminating any other threads currently executing within the construct. For example, here is a loop implementing a parallel search of a tree with the first thread finding the desired node exiting and killing off all of the other threads as part of the "**exit ... with**" statement:

```
const Result : optional Tree_Id;
for T => Root then T.Left || T.Right
  while T not null concurrent loop
    if T.Value == Desired_Value then
        // Found desired node, exit with its identifier
        exit loop with (Result => T.Id);
    end if;
end loop with (Result => null);
```

This declares a Result object of type Tree_Id. It then walks the tree in parallel, starting at Root and continuing with T.Left and T.Right concurrently. It continues until it reaches "null" on each branch, or some node is found with its Value component matching the Desired_Value. The value of Identifier at the end indicates the identifier of the node having the desired Value, or null to indicate that no node was found. The presence of **optional** in the declaration for Result indicates that its value might be null.

Supporting this kind of intentional "race" seems important in parallel programming, as many problems are amenable to a divide and conquer approach, but it is important that as soon as a solution is found, no further time is wasted searching other parts of the solution space. The "**end ... with**" phrase allows the specification of one or more results if the construct ends normally, as opposed to via an "**exit ... with**" (in this case, ending normally means all threads reach a null branch in the walk of the tree without finding the desired node). Effectively the "**exit ... with**" skips over the "**end ... with**" phrase.

So how does this all relate to exceptions? Well given the "**exit ... with**" capability, one can establish two or more threads, one which monitors for a failure condition, and the others which do the needed computation. The thread monitoring for a failure condition performs an "**exit ... with**" if it detects a failure, with the result indicating the nature of the failure, and as a side-effect killing off any remaining computation threads. If the normal computation succeeds, then an "**exit ... with**" giving the final result will kill off the monitoring thread. Note that the "**exit ... with**" statements must occur textually within the construct being exited, so it is visible whether such a premature exit can occur, unlike an exception which can arise deep within a call tree and be propagated out many levels.

As an example of the kind of failure condition which might be amenable to this kind of monitoring, imagine a resource manager object, which provides up to some fixed maximum of some kind of resource (e.g. storage) to code within a block. This resource manager (which is presumably of a concurrent type) could be passed down to operations called within the block for their use. Meanwhile, a separate monitoring thread would be created immediately within the block which would call an operation on the resource manager which would suspend the thread until the resource runs out, at which point it would be awakened with an appropriate indication of the resource exhaustion, and any other information that might be helpful in later diagnosis. On return from this Wait_For_Exhaustion operation, the monitoring thread would do an "**exit block with** (Result => Failure, ...)" or equivalent, to indicate that the computation required more resources than were provided. The code following the block would then be able to take appropriate action.

September 2010 …

[Please continue reading the blog on the web if interested. We hope the above extract has captured the essence of the ParaSail language design process.]

Conclusions about ParaSail

So what have we accomplished? The design of ParaSail is now largely complete. As of this writing, a grammar exists and a parser based on it is working; a simplified ParaSail Virtual Machine interpreter has been implemented, and a prototype compiler is under develoment. ParaSail has been presented at several conferences and other venues. So far it has been quite well received. In June at the upcoming AdaEurope Ada Connection conference in Edinburgh, there will be a workshop/tutorial allowing experimentation with the language and its prototype implementation. But of course, a programming language is only of significant value if it is *used* to build software.

Our view is that to succeed in the “multicore” world, we need new languages that make parallel programming as productive as sequential programming, without adding further complexity to the already challenging job of writing correct and secure software. By making parallel expression evaluation the default, by making it easy to insert even more race-free parallelism explicitly, and by integrating compile-time checked preconditions, postconditions, and various other kinds of assertions into a unified language from day one, we believe ParaSail can be one of those new languages that carry us into the multicore era.

References

[1] S. T. Taft (2011), *Designing ParaSail, a new programming language*, http://parasail-programming-language.blogspot.com .

[2] Stanford Artificial Intelligence Laboratory (1976), *SAIL*, http://pdp-10.trailing-edge.com/decuslib20-01/01/decus/20-0002/sail.man.html .

[3] R. E. Strom, *et al* (1991), *Hermes: A Language for Distributed Computing*, Prentice-Hall, Series in Innovative Technology, ISBN 0-13-389537-8.

[4] Unicode Consortium (2011), *Unicode 6.0.0*, http://www.unicode.org/versions/Unicode6.0.0/.

[5] L. Allison (1987), *A Practical Introduction to Denotational Semantics,* Cambridge University Press.

Ada Gems

The following contributions are taken from the AdaCore Gem of the Week series. The full collection of gems, discussion and related files, can be found at http://www.adacore.com/category/developers-center/gems/.

Gem #97: Reference Counting in Ada – Part 1

Emmanuel Briot, AdaCore

Date: 17 January 2011

Abstract: This series of three Gems describes a possible implementation for automatic memory management, via the use of reference counting. Part 1 explains how controlled types can be used to achieve automatic reference counting, and addresses some aspects of correct handling of reference counts. Part 2 analyzes a few issues related to tasking. Finally, Part 3 describes the use of weak reference counting.

Let's get started…

Memory management is typically a complex issue to address when creating an application, and even more so when creating a library to be reused by third-party applications. It is necessary to document which part of the code allocates memory and which part is supposed to free that memory. As we have seen in a previous Gem, a number of tools exist for detecting memory leaks (gnatmem, GNATCOLL.Memory or valgrind). But of course, it would be more convenient if the memory were automatically managed.

Some languages include an automatic garbage collector. The Ada Reference Manual has an implementation permission allowing a conformant compiler to provide one, although none of the mainstream compilers do so. Ada's design allows implementations to use the stack in many situations where other languages use the heap; this reduces the need for a garbage collector.

An alternative implementation for getting automatic memory management is to use reference counting: every time some object is allocated, a counter is associated with it. This counter records how many references to that object exist. When that counter goes down to zero, it means the object is no longer referenced in the application and can therefore be safely deallocated.

The rest of this Gem will show how to implement such a mechanism in Ada. As we will see, there are a number of minor but delicate issues involved, so implementing such types is not as trivial as it first seems. The GNAT Components Collection (GNATcoll) now includes a reusable generic package that simplifies this, and we will discuss this briefly at the end of this Gem.

As stated above, we need to associate a counter with the objects of all types we want to monitor. The simplest is to create a tagged type hierarchy where the root type defines the counter:

```
   type Refcounted is abstract tagged private;
   procedure Free (Self : in out Refcounted) is null;
private
   type Refcounted is abstract tagged record
      Refcount : Integer := 0;
   end record;
```

This approach is mostly suitable when building a reusable library for reference-counted types, such as GNATcoll. If you just want to do this once or twice in your application, you can simply add a new Refcount field to your record type (which doesn't need to be tagged).

Next, we need to determine when to increment and decrement this counter. In some languages this counter needs to be manually modified by the application whenever a new reference is created, or when one is destroyed. This is, for instance, how the Python interpreter is written (in C). But we can do better in Ada, by taking advantage of controlled types. The compiler calls special primitive operations each time a value of such a type is created, copied, or destroyed.

If we wrap a component of a simple access type in a type derived from Ada.Finalization.Controlled, we can then have the compiler automatically increment or decrement the reference count of the designated entity each time a reference is established or removed. We thus create a smart pointer: a pointer that manages the life cycle of the block of memory it points to.

```
   type Refcounted_Access is
         access all Refcounted'Class;
   type Ref is tagged private;
   procedure Set (Self : in out Ref;
                  Data : Refcounted'Class);
   function Get (Self : Ref) return Refcounted_Access;
   procedure Finalize (P : in out Ref);
   procedure Adjust   (P : in out Ref);
private
   type Ref is new Ada.Finalization.Controlled with record
      Data : Refcounted_Access;
   end record;
```

Let's first see how a user would use the type. Note that Get returns an access to the data. This might be dangerous, since the caller might want to free the data (which should remain under control of Ref). In practice, the gain in efficiency is worth it, since it avoids making a copy of a Refcounted'Class object. This is also essential if we want to allow the user to easily modify the designated entity. The user is ultimately responsible for ensuring that the lifetime of the returned value is compatible with the lifetime of the corresponding smart pointer.

```
declare
  type My_Data is new Refcounted with record
    Field1 : ...;
  end record;
  R1 : Ref;
begin
  Set (R1, My_Data'(Refcounted with Field1 => ...));
  --  R1 holds a reference to the data
```

```
 declare
    R2 : Ref;
 begin
    R2 := R1;
    --  R2 also holds a reference to the data
    -- (thus 2 references)
    ...
    --  We now exit the block. R2 is finalized,
    --  thus only 1 ref left
 end;
 Put_Line (Get (R1).Field1);  --  For instance
 --  We now leave R1's scope, thus refcount is 0,
 --  and the data is freed.
end;
```

Now let's look at the details of the implementation. First consider the two subprograms for setting and getting the designated entity. Note that the default value for the reference count is zero in the Refcounted type. The implementation of Set is slightly tricky: it needs to decrement the reference count of the previously designated entity, and increment the reference count for the new data. Instead of calling Adjust and Finalize explicitly (which is not a recommended practice when it can be avoided), we use an aggregate and let the compiler generate the calls for us.

```
procedure Set (Self : in out Ref;
                          Data : Refcounted'Class) is
 D : constant Refcounted_Access :=
      new Refcounted'Class'(Data);
begin
 if Self.Data /= null then
     Finalize (Self); -- decrement old reference count
 end if;
 Self.Data := D;
 Adjust (Self);  -- increment reference count (set to 1)
end Set;

function Get (P : Ref) return Refcounted_Access is
begin
  return P.Data;
end Get;
```

In GNATCOLL.Refcount, we provide a version of Set that receives an existing access to Refcount'Class, and takes responsibility for freeing it when it is no longer needed. The implementation is very similar to the above (although we need to be careful that we do not Finalize the old data if it happens to be the same as the new, since otherwise we might end up freeing the memory).

Adjust is called every time a new reference is created. Nothing special here:

```
overriding procedure Adjust (P : in out Ref) is
begin
  if P.Data /= null then
     P.Data.Refcount := P.Data.Refcount + 1;
  end if;
end Adjust;
```

The implementation of Finalize is slightly more complicated: the Ada reference manual indicates that a Finalize procedure should always be idempotent. An Ada compiler is free to call Finalize multiple times on the same object, in particular when exceptions occur. This means we must be careful not to decrement the reference counter every time Finalize is called, since a given object only owns one reference. Hence the following implementation:

```
overriding procedure Finalize (P : in out Ref) is
  Data : Refcounted_Access := P.Data;
begin
  --  Idempotence: the next call to
  -- Finalize will have no effect
  P.Data := null;
  if Data /= null then
    Data.Refcount := Data.Refcount - 1;
    if Data.Refcount = 0 then
      Free (Data.all);  --  Call to user-defined primitive
      Unchecked_Free (Data);
    end if;
  end if;
end Finalize;
```

That's it for the basic implementation. The next Gem in this series will discuss issues of task safety associated with reference-counted types.

Gem #99: Reference Counting in Ada – Part 2: Task Safety

Emmanuel Briot, AdaCore

Date: 14 February 2011

Let's get started…

In Part 1, we described a reference-counted type that automatically frees memory when the last reference to it disappears. But this type is not task safe: when we decrement the counter, it might happen that two tasks see it as 0, and thus both will try to free the data. Likewise, the increment of the counter in Adjust is not an atomic operation, so it is possible that we will be missing some references.

In some applications this restriction is not a big issue (for instance, if there are no tasks, or if the types are only ever used from a single task). However, let's try to improve the situation.

The traditional solution is to use a lock while we are manipulating the counter. We could, for instance, use a protected type for this. However, this means that a nontasking application using our reference-counted types would have to initialize the whole tasking run-time, which could impact execution somewhat, since part of the code goes through slower code paths.

GNAT provides a global lock that we can reuse for that, and that does not require the full tasking run-time. We could use that lock in a function that changes the value of the counter atomically. We need to return the new value from that function: changing the value atomically solves the problem we highlighted for Adjust, but not the one we showed for Finalize, where two tasks could see the value as 0 if they read it separately.

```
function Atomic_Add
  (Ptr : access Integer; Inc : Integer) return Integer
is
  Result : Integer;
begin
  GNAT.Task_Lock.Lock;
```

```
   Ptr.all := Ptr.all + Value;
   Result := Ptr.all;
   GNAT.Task_Lock.Unlock;
   return Result;
end Atomic_Add;
```

On some systems there is actually a more efficient way to do this, by using an intrinsic function: this is a function provided by the compiler, generally implemented directly in assembly language using low-level capabilities of the target machine. We need special handling to check whether this facility is available, but if it is, we no longer need a lock. The GNATCOLL.Refcount package takes full advantage of this.

```
function Atomic_Add
  (Ptr : access Integer; Inc : Integer) return Integer
is
   function Intrinsic_Sync_Add_And_Fetch
     (Ptr   : access Interfaces.Integer_32;
      Value : Interfaces.Integer_32) return
                              Interfaces.Integer_32;
   pragma Import
     (Intrinsic, Intrinsic_Sync_Add_And_Fetch,
      "__sync_add_and_fetch_4");

begin
   return Intrinsic_Sync_Add_And_Fetch (Ptr, Value);
end Atomic_Add;
```

(Note: In actual practice, it would be necessary to declare the access parameter of function Atomic_Add with type Interfaces.Integer_32, for type compatibility with the intrinsic.)

Once we have this Atomic_Add function we need to modify our reference-counted type implementation. The first change is to declare the Refcount field as aliased, in the definition of Refcounted. We then revise the code as follows:

```
overriding procedure Adjust (P : in out Ref) is
   Dummy : Integer;
begin
   if P.Data /= null then
      Dummy := Atomic_Add (P.Data.Refcount'Access, 1);
   end if;
end Adjust;

overriding procedure Finalize (P : in out Ref) is
   Data : Refcounted_Access := P.Data;
begin
   P.Data := null;
   if Data /= null
      and then Atomic_Add (Data.Refcount'Access, -1) = 0
   then
      Free (Data.all);
      Unchecked_Free (Data);
   end if;
end Finalize;
```

The last Gem in this series will talk about a different kind of reference, generally known as a weak reference.

Gem #99: Reference Counting in Ada – Part 3: Weak References

Emmanuel Briot, AdaCore

Date: 28 February 2011

Let's get started…

As we mentioned in the first two parts of this Gem series, GNATCOLL now includes a package that provides support for memory management using reference counting, including taking advantage of the efficient synchronized add-and-fetch intrinsic function on systems where it is available.

There is one thing that reference-counted types cannot handle as well as a full-scale garbage collector: cycles. If A references B which references A, neither of them will ever get freed. A garbage collector is often able to detect such cycles and deallocate all the objects as appropriate, but such a case cannot be handled automatically through reference counting. However, there's a variant approach that can handle such cases with only minor changes in the code.

Let's take an example: you are retrieving values from some container (a database for instance), and want to have a local cache to speed things up. The code would likely be organized as follows:

- Get a reference-counted value from the container. Its counter is 1.
- Put it in the cache for later use. The counter is now 2, since the cache itself owns a reference.
- When you are done using the value in your algorithm, you release the reference you had. Its counter goes down to 1 (the cache still owns the reference).

Because of the cache, the value is never freed from memory. This is not good, since memory usage will only keep increasing.

GNATCOLL provides a solution for this issue, through the use of weak references. This is a standard industry term for a special kind of reference: you have a type that points to the same object as a true reference-counted type would, but that type does not hold a reference. Thus, it does not prevent the counter from reaching 0, and the object from being freed.

When the deallocation occurs, the internal data of the weak reference is reset. Thus, if you retrieve the data stored in the weak reference, you get null, not an erroneous access to some freed memory (which might sooner or later result in a Storage_Error).

If we set up the cache so that it uses weak references, the code becomes:

- Get a reference-counted value from the container. Its counter is 1.
- Put it in the cache, through a weak reference. The counter is still 1.
- When you are done using the value, the counter goes down to 0, and the memory is freed.
- At this point, the cache still contains the weak reference, but the latter uses just a little memory.

Using slightly more complex code, it is possible, in fact, to remove the entry for the cache altogether when the value is freed, thus really releasing all memory to the system. Though GNATCOLL does provide a capability for using weak references, a future package will provide easier handling of such caches.

One way to implement weak references is by adding an extra pointer in type Refcount. GNATCOLL chooses to make this optional: if you want to systematically have that extra pointer in your data structure, you can use weak references. Otherwise, you still have access to the code we described in the first part of this Gem series.

We will not go into the details of the implementation for a weak reference. Interested parties can look at the code in GNATCOLL.Refcount.Weakref, which is relatively small.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

Ada-France
attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

Ada-Sweden
attn. Rei Stråhle
Rimbogatan 18
SE-753 24 Uppsala
Sweden
Phone: +46 73 253 7998
Email: rei@ada-sweden.org
URL: www.ada-sweden.org

Ada Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch