The ModulaTor

Oberon-2 and Modula-2 Technical Publication

Ubaye's First Independent Modula-2 & Oberon-2 Journal! Nr. 9,

Oct-1996

Juice

Michael Franz (franz@uci.edu) and Thomas Kistler (kistler@ics.uci.edu), University of California, Irvine, U.S.A., 17th June 1996

Introducing Juice

Juice is a new technology for distributing executable content across the World Wide Web. In this respect, it is similar to Java from Sun Microsystems. However, as we will try to explain in the following, Juice differs from Java in several important aspects that allows it to outperform Java in many "downloadable Applets" applications. The advantages of Juice become even more pronounced when the "Applets" are large. Java, on the other hand, has an advantage in the area for which it was originally invented: embedded applications in consumer electronics and household appliances. Juice is intended to be a complement to Java, giving users a choice: Java or Juice.

A Little History To Start With

Juice isn't simply a Java-lookalike; that we have created in quick reaction to Sun Microsystems' Java announcement. Quite to the contrary, work on Juice has preceded Java by several years, but it all happened in an academic environment with more limited resources than available at Sun. The only part of Juice that has been influenced by Java is the food-related name.

Juice grew out of a research project headed by Niklaus Wirth, the inventor of the Pascal and Modula programming languages. His aim was to invent a language that was at once simple and safe to use. The new language should also support the object-

oriented programming paradigm, be efficiently compilable, and applicable to a wide range of programming tasks from systems programming to teaching. A language with these properties was defined by Wirth in 1988 and given the name Oberon http://caesar.ics.uci.edu/oberon/intro.html.

At first sight, Oberon and Java seem to be as different as two languages can possibly be. Oberon looks a lot like Pascal (not very surprising, since they were both invented by the same person), and Java looks a lot like C (thereby appearing "familiar" to many programmers). A closer comparison of Oberon and Java, however, yields surprisingly many similarities: both languages

* are strongly typed while permitting the definition of extended types that are backward-compatible with their ancestors,

* prohibit pointer arithmetic and mandate the presence of a garbage collector in their run-time environment, and

* provide the notion of modular packages that are compiled separately and dynamically linked at run-time

Juice is to the Oberon language what the Java Virtual Machine (http://java.sun.com/java.sun.com/whitePaper.Platform/JavaPlatform.doc2.html#6978) is to the Java language: the means of distributing portable software. Work on what has now become Juice started in 1989 and contributed to Michael Franz's Ph.D. degree in 1994 (http://www.ics.uci.edu/~franz). The title of his 1994 dissertation, almost a year and a half before the announcement of Java, was "Code-Generation On-The-Fly: A Key to Portable Software".

What Is Juice and How Does It Differ From Java

Juice encompasses three key components

- an architecture-neutral software distribution format,

- a compiler that translates from Oberon into the Juice portable format, and

- a "plug-in" that enables the Netscape browser (http://home.netscape.com) to run Juice applets.

To the technical layman, the most notable difference between comparable Juice and Java applets is probably that the Juice version of any applet is likely to run a lot faster than its Java counterpart. Rather than being interpreted, as Java applets normally are, Juice always compiles each applet into the native code of the target machine before it begins execution. Once that an applet begins executing, it runs at the full speed of compiled code. Juice's on-the-fly compilation process is not only exceptionally fast, but

it also generates object code that is comparable in quality to commercial C compilers. In contrast, even the latest just-in-time compilers for Java that have been announced by major vendors are not yet trying to compete with regular compilers.

Further, the portable Juice representation is denser than Java byte-codes. Users with slow connections to the Internet save time by downloading Juice applets rather than Java applets of the same functionality. In fact, the time saved by the faster downloading normally more than compensates for the on-the-fly compilation phase.

How then, do a few academic researchers manage to come up with a better on-the-fly compiler than a whole industry of software companies? Well, the key to Juice's effectiveness isn't really found in the compiler technology embedded within the Netscape plug-in, but in the Juice software distribution format itself: Behind the scenes, Juice and Java differ by far more than one might expect.

Java's distribution format is a sequence of so-called byte-codes: the inventors of Java thought up an "ideal" processor designed specifically for running Java programs and use the instruction set of this ideal processor as the "intermediate language" in which Java programs are shipped. Hence, on the programmer's side, a Java compiler translates the source into a byte-code sequence, and on the side of the eventual consumer, a byte-code interpreter simulates the ideal Java processor. It is entirely conceivable that the ideal Java processor can actually be implemented in hardware, and Sun has actually already announced such processors.

Juice's distribution format, on the other hand, is far more complex. It is based on a tree-shaped program representation as is typically used transiently within optimizing compilers. Rather than containing a linear code-sequence that can be interpreted byte-by-byte, a Juice-encoded applet contains a compressed tree that describes the actions of the original program. The tree preserves the control-flow structure of the original program, which makes it much easier to perform code optimization while the tree is translated into the native instruction set of the target machine.

As an example of what kinds of optimizations may be beneficial, consider a modern processor that has several functional units. These require a certain instruction-mix in order to operate at top speed. By re-ordering certain mutually independent instructions, a better instruction-mix may be achieved. A tree-based encoding maintains the notion of a basic block and makes it relatively easy to decide if two instructions are mutually independent. This is not so simple with a linear instruction stream, such as Java byte-codes, in which any instruction can potentially be the target of a branch.

Further, our tree-based encoding avoids many of the security issues that are difficult to solve in Java, or in any virtual-machine representation for that matter. By the very definition of our tree-encoding scheme, it is impossible to encode a tree that violates scoping rules of our source language. Such an "illegal" program cannot even be constructed by hand. Java's byte-code instructions, on the other hand, are at a much

lower semantic level. Although it is possible to verify that a given byte-code sequence doesn't perform any illegal action, this requires data-flow analysis and a partial repetition of the compiler's integrity checks. The tree-based Juice distribution format in effect makes this particular data-flow analysis unnecessary. It also allows for highly efficient load-time integrity checking, as every node in the encoded tree is fully typed.

How to Obtain Juice

The easiest way to learn more about Juice is by simply having a look at it. The Juice-homepage which contains an introduction to Juice, the plug-in download-instructions, and demo applets can be found on the Internet at: http://caesar.ics.uci.edu/juice

In order to see the demos, you need to install the Netscape Juice plug-in. This plug-in is available for the following platforms:

* Apple PowerPC Macintosh

* PC-compatible running Windows 95

Conclusion

We believe that Juice is a better technology than Java for bringing executable content to the World Wide Web. Juice allows substantial amounts of software to be encoded space-efficiently, and to be compiled on-the-fly into high-quality code. The Juice architecture-neutral distribution format has a variety of properties that make it inherently better suited for applications that require the speed of optimized object code.

Java, on the other hand, is the better choice in the domain for which it was originally created: embedded applications in consumer electronics and household appliances. Aided by a hardware implementation of the Java virtual machine, this creates a standard run-time architecture for embedded software that requires a small memory footprint.

[ed. note: published with permission of the author.]

The ModulaTor Forum

AlphaOberon News: Oberon-text to LaTex Converter

by Guenter Dotzel, ModulaWare

An Oberon to LaTex converter called Edit2LaTex is now available for the Oberon System V4. Source code of course. Developed by Joerg Derungs.

Edit2LaTex can handle Oberon texts, graphics- and table-elements. Its OO-architecture makes extensions to support custom hypertext-elements very easy. Skeleton texts are included. They show how the paragraph controls (parcs) of Oberon texts serve to determine the LaTex document structure.

Edit2LaTex is very easy to use, but unfortunately, the documentation is in German only (Edit2LaTex_Docu.Text).

Edit2LaTex was added to ModulaWare's AlphaOberon distribution.

With one further step, using the LaTex2Html converter, you can generate your HTML files (e.g. for the world wide web) automatically from the Oberon wysiwyg-text/graphics editor.

Native XDS-x86 v2.20 for Windows NT and Windows 95 Release

by xTech, Ltd.

The native code XDS-x86 v2.20 Oberon-2 and ISO Modula-2 Compiler for Windows NT and Windows 95 is now available.

xTech Development System (XDS) is the family name of a professional Modula-2 and Oberon-2 programming system.

XDS is available for most popular platforms, including PC/MS-DOS, OS/2, Linux, Windows NT, Windows 95, Unix workstations (Sun, HP, DEC, MIPS), PowerMac, etc. XDS provides a uniform programming environment for all mentioned platforms and allows to design and implement truly portable software.

The XDS Modula-2 compiler implements the ISO standard of Modula-2. The ISO Modula-2 library set is accessible from XDS Modula-2 and Oberon-2 programs.

All XDS implementations share the same platform-independent front-end for both source languages. There are basically two back-ends. The output code can be either native code for the target platform (Native XDS) or text in the ANSI C language. The ANSI C code generator is available for all above mentioned platforms. Native code compilers are currently available for the PC under WNT/Win95, OS/2, and Linux.

XDS includes the standard ISO and PIM libraries along with a set of utility libraries and an interface to the ANSI C library set.

Native XDS-x86 for Windows NT and Windows 95 produces highly optimized 32-bit

code and comes with two major enhancements: IDE and Win32 API support. A significant subset of the Win32 API is accessible from XDS Modula-2 and Oberon-2.

XDS includes online help and manuals in PostScript format: isolib.ps, o2rep.ps, xc.ps, xdslib.ps.

Furthermore a TopSpeed Modula-2 like library implementation written in XDS Modula-2 is included with source code: fio.def, fio.mod, fio.txt, io.def, io.mod, io.txt, lib.def, lib.mod, lib.txt, mathlib.def, mathlib.mod, shtheap.def, shtheap.mod, shtheap.txt, str.def, str.mod, str.txt, syserr.def, volume.def, volume.mod, xfilepos.sym. It is accessible from XDS Modula-2 and Oberon-2.

For more information see http://www.modulaware.com/xdsprod.htm.

Book recommendation:

George Roche: "Free markets, free man: Frederic Bastiat, 1801-1850", The Hillsdale College Press and The Foundation for Economic Education, 1993. ISBN 0-916308-73-1. 181 pages.
This book is now difficult to find. Try Laissez Faire Books or Amazon.Com

Milton Friedman: "Capitalism and Freedom". University of Chicago Press, Chicago and London, 1962. 202 pages.
Winner of Nobel Price in Economics. This book is about the relation between economic and political freedom, the problems associated with centralised governments, their control of money, fiscal policy, government controlled education, monopolies, redistribution of income, social welfare, and last but not least about alleviation of poverty.

IMPRESSUM: The ModulaTor is an unrefereed journal. Technical papers are to be taken as working papers and personal rather than organizational statements. Items are printed at the discretion of the Editor based upon his judgement on the interest and relevancy to the readership. Letters, announcements, and other items of professional interest are selected on the same basis. Office of publication: The Editor of The ModulaTor is Guenter Dotzel; he can be reached by tel/fax: +33 492.81.30.99 or by mailto:gd@modulaware.com

Home Site_index Contact Legal Buy_products OpenVMS_compiler Alpha_Oberon_System XDS_family DOS_compiler ModulaTor Bibliography Oberon[-2]_links Modula-2_links

Webdesign by www.otolo.com/webworx, 02-Dec-1998