The Icon Newsletter

No. 54 – December 1, 1997

Contents

The Curse of DOS

Most of the software related to Icon has been developed on UNIX platforms. UNIX has a very liberal (perhaps too liberal) policy on file names. Upper-and lowercase letters are distinct and only two characters are excluded: the null character and the slash. File names also can be very long.

From the time we started distributing software for other platforms, we had to deal with the so-called 8-3 problem visited on us by DOS — a maximum of 8 characters for the file name followed by an optional period and a maximum of 3 characters for an "extension". In addition, only a few "special characters" are allowed.

In some cases, this wasn't much of a problem. In the source code for Icon, we had to struggle to find even semi-mnemonic names for files, but once done, only new files, added infrequently, caused problems.

The Icon program library has been more of a problem, since additions are frequent and many come from outside the Icon project with no regard for the limitations on file names imposed by DOS.

Where our material was developed on UNIX platforms and only distributed for UNIX platforms, we didn't worry about the lengths of file names and the characters in them. File names like icon_unix_sun_os.tar.gz came about naturally.

The addition of so-called long file names to recent versions of Microsoft Windows didn't help. For one thing, DOS is still there and will be. For another, long file names are problematical for distribution.

So why are we griping *now*? It's because we're preparing a CD-ROM to be packaged with the book *Graphics Programming in Icon*. That CD-ROM needs to be usable on different platforms — DOS, Windows, UNIX, and others. To accomplish this cleanly, file names must satisfy the DOS restrictions.

That's already been dealt with for the Icon source code and program library. But many UNIX files and other files on our FTP and Web sites *are* problems. On our FTP site, we used names like ipd282.ps.gz so that the content and form would be clear from the name alone. On our Web site, which will be included on the CD-ROM, we used the suffix .html for pages without thinking about the possible consequences.

So .html had to be changed to .htm, not only in file names but also the many references to them in files.

In some cases, there are 3-character extensions that are widely recognized, like .tgz in place of

.tar.gz. In other cases, like .tar.Z, there are not. We've switched to .taz for the latter, but .ps.gz stumped us. For that, we've changed to ZIP format and explanatory documentation.

It's highly unlikely that when DOS was first written that anyone anticipated where PC computing would go and the many unfortunate effects of the limitations in the design of DOS that would ensue. At the time, it no doubt seemed prudent to use short file names to save space. Nonetheless, design that does not provide for growth and adaptation to changing hardware and user expectations can have horrendous consequences. In the case of DOS, the consequences weren't just visited on DOS users: It's been a plague for "innocent bystanders" who never used DOS and never wanted to.

We learned a long time ago that design mistakes often are difficult or impossible to correct. What seems trivial and unimportant at the time may come back to haunt. We would like to think that the current generation of more experienced and knowledgeable program designers are more careful than some in the past. But we're hardly confident of this. And mistakes sometimes take a long time to surface: Witness the "Year-2000 Problem".

Unfortunately, the pessimist often is right.

New Program Library Release

Version 9.3.1 of the Icon program library now is available, replacing Version 9.3, which was released over a year ago.

We've used the numbering 9.3.1 to indicate that the library corresponds to Version 9.3 of Icon, which is still current. This is not, however, a minor revision of the library; it's a new version.

There are many changes, improvements, and additions in 9.3.1. Notable are scrolling text-list vidgets, which allow the user to browse and select lines from lists. Text-list vidgets are supported in an improved version of VIB, the visual interface builder for Icon. There also are new sections of the library for material related to program monitoring and visualization under MT Icon.

The new library is available on our Web and FTP sites.

Status of the Graphics Book

The book *Graphics Programming in Icon* is nearing completion. The manuscript itself is complete. Work remains to be done on the cover, color plates, and the CD-ROM.

We hope the book will be available by March 1998.

Unicon

In recent mail, Shamim Mohamed told us about additions he'd made to Icon to provide UNIX interfaces. He put it this way:

> A long time ago I was thinking "Wouldn't it be nice to use Icon instead of Perl for sysadmin sorts of things?" My level of frustration with Perl kept rising, until one day I made some time and implemented UNIX interfaces to Icon.

Shamim has made the results of his work, called Unicon, available on the Web:

http://www.crl.com/~spm/unicon/

The author can be contacted at

spm@crl.com

Chinese Icon Book

Professor Zhang Weiguo of Renmin University of China in the People's Republic of China has written a book on Icon in Chinese. The title translates into English as "The Course of the Icon Programming Language".

The book is in the hands of the publisher and should be available in two to three months.

The author can be contacted at

wgzhang@ht.rol.cd.net

Downloading Icon Material

Most implementations of Icon are available for downloading via anonymous FTP:

ftp.cs.arizona.edu (cd /icon)

From Our Mail

I want to try out some of the programs from the Icon Analyst *that do dynamic program analysis. What do I need?*

If you are referring to programs that monitor events in other programs, you need MT Icon, the multi-thread version of Icon. Prepackaged versions are available on our Web and FTP sites for Sun and DEC Alpha workstations and for MS-DOS. For other platforms, you'll need to build Icon with

```
#define EventMon
```

in src/h/define.h.

You've mentioned an implementation of Icon in Java, but I've not seen anything about its being available. What's up?

We could give a short and factual but unhelpful answer like "We have no release date yet" or "We're still working on it", but here's the whole scoop.

The implementation of Icon in Java is complete except for Icon's graphics facilities and a few features like storage-region information that make no sense in the Java implementation. The performance of the implementation is, however, unsatisfactory. This problem must be overcome before a release.

The situation is complicated by the fact the Todd Proebsting, the leader of the project, has left the faculty of our department to join Microsoft Research. He retains an active interest in the project, but it's uncertain how much time he'll have to work on it. Nonetheless, work is progressing, if slowly.

If anything changes in this regard, we'll announce it on our Web site and in this *Newsletter*.

I want to prototype an application in Icon and then automatically translate it to C when I have the application the way I want it. Can you give me guidelines to ensure that this will work?

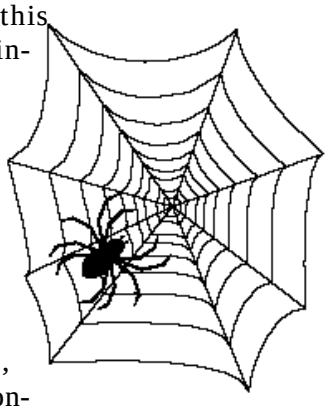
Many persons have successfully prototyped applications in Icon and then converted them to C. It's rarely possible, however, to translate a program from one language to another mechanically. You should not plan on being able to translate from Icon to C automatically. Instead expect to do a partial translation (the Icon program library has some "helper" utilities for this) and then do the rest by hand. Or better, do a hand translation. It may not be that hard if you are careful in the way you program in Icon. Things to avoid are:

- complicated string scanning
- co-expressions
- goal-directed evaluation except in the context of iteration (every … do …)

If you use sets, tables, or lists as stacks or queues, be prepared to provide C support routines to handle these things in the translated program.

Avoid Icon idioms and clever tricks. Most of all, keep you code clean, clear, and well-organized.

Updated Icon Web Site



As the lead article in this issue of the *Newsletter* indicates, we've made substantial changes to our Web site. We not only "fixed" file name problems, but we've also made substantial improvements to the content and structure of our site. For example, there are links to other Icon-related sites, and all the material on our FTP site now is available on our Web site.

Come visit!

Icon Mirror Site

Tony Sprinzl is maintaining mirrors of our Web and FTP sites at Goodie Domain Service at the University of Technology at Vienna, Austria. The addresses are:

http://gd.tuwien.ac.at/languages/icon/
ftp://gd.tuwien.ac.at/languages/icon/

Using Icon to Spin the Web

Editors' Note: The following article was contributed by Clint Jeffery.

The World Wide Web is a popular application area for text-oriented languages such as Icon. In addition to writing one-shot Icon programs to generate HTML versions of existing documents to put them on the Web, you can use Icon to generate Web pages on the fly. Programs that generate Web pages as their output are called CGI scripts. This article provides an overview of writing CGI scripts in Icon using cgi.icn, an Icon program library module written by Joe Van Meter and Clinton Jeffery. Some knowledge of HTML is required in order to put this module to work.

CGI Basics

CGI stands for Common Gateway Interface. CGI scripts are programs invoked by Web servers to assist in processing Web input and output. The standard reference on CGI is available on the Web at

http://hoohoo.ncsa.uiuc.edu/cgi/

The most frequent use of CGI scripts is to process input from *forms* that appear in Web pages. The typical scenario is: a user reads a Web page that asks for information; the user types information into the page and presses a Submit button; the Web server invokes a CGI script and passes it a copy of the information submitted by the user; the CGI script generates a Web page to inform the user of the action taken on as a result of the input.

Writing Scripts with cgi.icn

cgi.icn provides a standard program organization as well as procedures that process CGI input and help generate HTML output. In order to use cgi.icn you have to have permission to execute programs on a Web server. The details on how to do this vary from platform to platform and Web server to Web server. An example for the UNIX Apache Web server is given in Reference 1.

Input Processing

CGI scripts process input data supplied by the Web browser that invoked the script and write a new Web page, in HTML, to standard output. cgi.icn includes a main() procedure and performs input processing prior to starting the script's execution, so when linking cgi you must name your main procedure cgimain().

CGI input is provided to the Icon program in a single Icon global variable, a table named cgi. The keys of this table are exactly the names given in HTML INPUT tags on the Web page that invoked the script. For example, an HTML form might include a line such as

```
<INPUT TYPE="text" NAME="PHONE" SIZE=15>
```

defining a field named PHONE in which the user may type up to 15 digits. In the Icon program, cgi["PHONE"] contains this user input.

Output Processing

cgi.icn includes a large number of helper procedures that convert Icon values into strings,

The Icon Newsletter

Ralph E. Griswold, Madge T. Griswold,
and Gregg M. Townsend

Editors

The Icon Newsletter is published three times a year and is available on the World Wide Web. To receive printed copies, contact:

Icon Project
Department of Computer Science
The University of Arizona
P.O. Box 210077
Tucson, Arizona 85721-0077
U.S.A.

voice: (520) 621-6613

fax: (520) 621-4246

e-mail: icon–project@cs.arizona.edu

THE UNIVERSITY OF ARIZONA.
TUCSON ARIZONA

and

Bright Forest Publishers
Tucson Arizona

wrapped with appropriate HTML tags that format them attractively. An example is cgiSelect(name, values), which writes an HTML SELECT tag for a field named *name*, which generates a list of radio buttons with labels given in the second parameter. An Icon programmer might write

```
cgiSelect("GENDER", ["female", "male"])
```

to generate the HTML

```
<SELECT NAME="GENDER">
<OPTION SELECTED>female
<OPTION>male
</SELECT>
```

Summary

With cgi.icn writing CGI scripts is easy. CGI is an interesting application area for Icon, because it involves heavy synthesis (rather than the more customary analysis) of strings. The source code for cgi.icn is available from the Web page given below. We welcome improvements to this library.

Reference

1. Clinton Jeffery and Jonathan Vallejo. *Writing CGI Scripts in Icon*. Technical Report CS-97-6, The University of Texas at San Antonio, July 1997.

http://www.cs.utsa.edu/research/icon/cgi.html

An Introduction to Wi

Editors' Note: The following article was contributed by Clint Jeffery.

Wi is a program for editing, compiling, and running Icon programs under Microsoft Windows. For the most part, Wi replaces the use of command-lines to compile and run Icon programs; it also contains a source code editor and project *make* facility. This article gives an overview of Wi; interested readers can consult IPD271 for more information [1]. (A note about pronunciation: Wi is pronounced like "Wye", or when you are frustrated, "Why".)

Why Wi?

Windows Icon includes wicont.exe and nticont.exe, which provide Icon's standard command-line interface for compiling programs. Wi, however, is still essential. First, Wi is more appealing and easier to learn for novice users than are icont and iconx. Second, Wi makes Windows Icon usable under Windows 3.1, where users are unable to invoke wicont.exe and wiconx.exe from MS-DOS prompt command-lines due to technical limitations of the operating system. Third, Wi demonstrates the use of native Windows interface facilities: Wi is written in about 700 lines of Icon.

Running Wi

Wi is launched by double-clicking on Windows Icon within the Windows Icon program group that is created during installation. You can also run Wi from the command line (on Windows 95 and NT) and supply the name of a file to edit. See Figure 1.

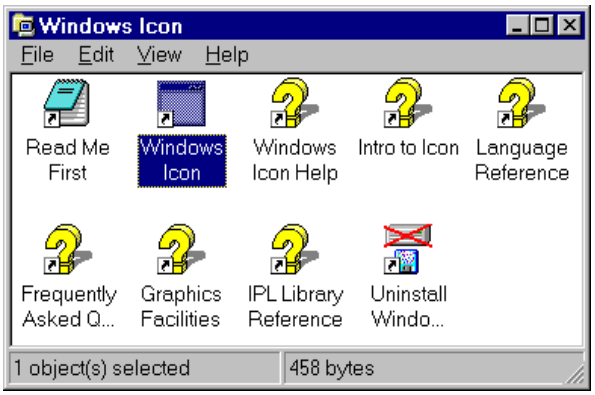


Figure 1. Launching Wi

Editing, Compiling, and Running Programs

You start with an empty window, editing a file

called noname.icn. Editing your program occurs within the main Wi window, which is shown below. The editor provides a standard Windows interface and supports cut / copy / paste, find / replace, and one level of undo. Besides loading and saving files and editing operations, Wi's menus allow you to compile and run programs, to launch VIB for editing your program's visual interface (if it has one), to customize screen appearance, and to invoke the extensive on-line help. See Figure 2.

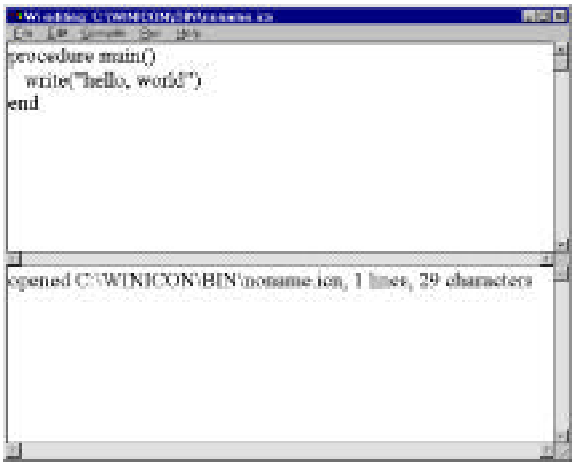


Figure 2: Wi's Editing Interface

The top area shows program source code, while the bottom portion shows messages such as compilation errors. You can change the font and the number of lines used to show messages from the Edit menu.

When you are finished editing your program, you can save it, compile it, or just make (save, compile and link an executable) and run your program with menu options. The Arguments command in the Run... menu lets you specify any command-line arguments that the program should be given when it is executed.

Error Handling

Compilation and run-time errors result in messages in which the editor highlights the line at which the error was detected. See Figure 3.

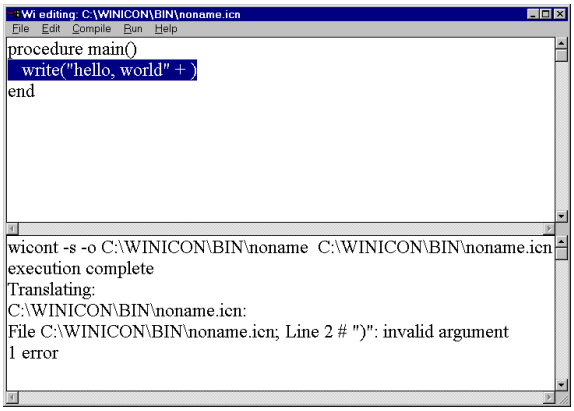


Figure 3: Reporting Errors in Wi

On-Line Help

Extensive on-line help is easily accessed from within Wi, including tutorial and overview articles contributed by Ralph Griswold, Dave Hanson, and John Shipman. You can invoke keyword searches not only on built-in language features, but also the graphics facilities and the entire Icon program library.

Summary

The Wi program goes a long way toward making Windows Icon easy to learn and use. I expect it to evolve further towards integration with other parts of the Icon environment. It would be good to merge Wi and VIB, Icon's visual interface builder, for example. In the meantime, however, Windows Icon in general and Wi in particular have gained a lot of functionality in recent months and it may be worth your time to keep up to date on the latest version. The status on Windows Icon and related projects can be obtained on the Web at http://www.cs.utsa.edu/research/icon/

Acknowledgments

Bob Goldberg produced the first program launcher for Windows Icon, providing the impetus for work in this area. Steve Schiavo developed a second program launcher that inspired many of the features found in Wi.

Reference

1. Clinton L. Jeffery. *Version 9 of Icon for Microsoft Windows.* Icon Project Document 271. The University of Texas at San Antonio and The University of Arizona. October 11, 1997.

http://www.cs.utsa.edu/research/icon/ipd271.htm

Icon on the Web

Icon is on the World Wide Web at

http://www.cs.arizona.edu/icon/