The Icon Newsletter

No. 36 – July 1, 1991

For New Readers

If this is the first issue of the *Newsletter* that you've received, we'd like to welcome you to our readership, which now numbers more than 5,100.

If you're new to Icon, coming in at Issue 36 of the *Newsletter* may be a bit bewildering. If you find this issue interesting, you can get back issues at a nominal cost. See the order form at the end of this *Newsletter*.

If this is your first issue of the Newsletter and you didn't ask for it, you may wonder why you got it. Perhaps a friend told us you might be interested. Or maybe you inquired about a commercial version of Icon or SNOBOL4. Of course, if you're not interested, we'll be glad to remove your name from our mailing list.

If what's here looks interesting, but you know little or nothing about Icon, we'll be glad to send you a short technical report that describes the main features of Icon. Just ask for the "Icon Overview". You can reach us by telephone, postal mail, electronic mail, or fax as listed in the publication box on page 7.

The Icon Compiler

In previous Newsletters we've mentioned progress on Ken Walker's implementation of an optimizing compiler for Icon. This compiler is now running on a variety of UNIX platforms and work is under way on porting it to other operating systems.

Advantages of the Compiler

The main advantage of the Icon compiler is faster execution of Icon programs than with the interpreter. The increase in execution speed varies considerably from program to program. A factor of about three seems typical, but users have reported factors up to 10.

Another advantage of the Icon compiler is that it produces "stand-alone" executable files. These executable files do not need the separate run-time system that interpreter files do.

Compiled programs also generally require less memory during program execution than interpreted one, since an executable file produced by the compiler only includes the run-time routines the program needs, while the interpreter includes the entire run-time system (including the interpreter proper).

Since the Icon compiler generates C code, it's also relatively portable.

Disadvantages of the Compiler

The main disadvantage of the Icon compiler, compared to the interpreter, is the time required for compilation. With the interpreter, a program gets into execution quickly, since no real code is generated, no optimizations are done, and no linking of run-time routines is required. The compiler, on the other hand, does extensive analysis of the source program in order to optimize the executable code. For most programs, the Icon compiler itself is reasonably fast. But it produces C code, which then must be compiled and linked to form the executable file. This can be a time-consuming process.

The Icon compiler requires a substantial amount of memory — so much so that it probably will not run on MS-DOS platforms without extended memory.

The generation of C code by the Icon compiler is a disadvantage as well as an advantage. It's necessary to have a C compiler to use the Icon compiler. This is not a problem for most UNIX users, since C usually is bundled with the UNIX operating system. The need for a C compiler (and a robust one) may be a problem on some other operating systems and for some personal computer and workstation UNIX platforms.

Using the Icon Compiler

Using the Icon compiler is very similar to using the Icon interpreter. The Icon compiler supports almost all

of the features of that the interpreter does, although some esoteric features like string invocation have to be specified with a command-line option. Such features also may defeat compiler optimizations and adversely affect program running speed.

The Icon compiler supports linking, but by inclusion of the source code rather than pre-compiled modules.

The main features the compiler does not support at present are error conversion and large-integer arithmetic.

Getting the Icon Compiler

A preliminary version of the Icon compiler is available via FTP network transfer. It is available both in source form and in executable form for several UNIX platforms. See the article on page 4 about getting Icon material via FTP.

The compiler is only available via FTP at the present time. We plan to have a distribution on magnetic tape ready later this year.

Looking Ahead

As indicated above, the Icon compiler presently only runs on UNIX platforms. It should run on almost any UNIX platform on which the Icon interpreter runs, although building the compiler on a new platform takes a bit of work.

We expect porting the compiler to VMS to be relatively straightforward, and we plan to attempt that this fall. Porting to 370 mainframe platforms will be more difficult, but should be feasible.

The Icon compiler probably will not run on most personal computers because of the amount of memory it requires. However, it's possible to cross-compile, generating C code on some platform for use on another. We'll be exploring this and other possibilities.

As with any optimizing compiler, there are lots of things that could be done to produce faster code. We have a rather long list ...

Icon News Group

It's been a while since we've mentioned our electronic news group in this *Newsletter*. Since we have many new subscribers, we think it's time to point out that those of you who have Internet access can participate electronically in discussions and the exchange of information about Icon.

The news group is like electronic mail, except that mail is automatically distributed to all subscribers to the group.

The news group's address is

icon-group@cs.arizona.edu

To subscribe (or unsubscribe), send your request to:

icon-group-request@cs.arizona.edu

(not to icon-group).

Please remember that mail to icon-group is redistributed to many persons. It's not the place to send personal mail or requests for specific information about Icon. Use

icon-project@cs.arizona.edu

for that.

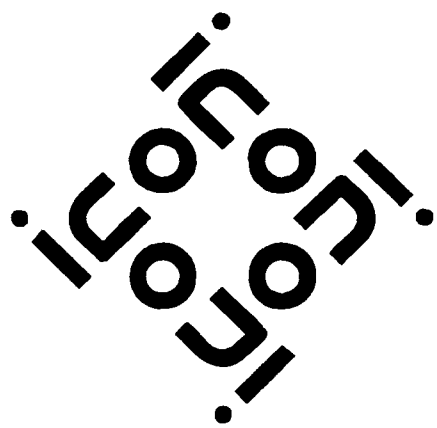
The Icon Analyst

The Analyst has completed its first year of publication and is starting into its second. The first six issues included 23 articles, ranging in length from one to five pages, plus programming tips and short notes.

We plan to keep the same format and publication schedule for the upcoming year — a 12-page newsletter every two months. Articles planned include ones on string synthesis, variant translators, result sequences, procedure libraries, understanding expression evaluation, writing efficient programs, getting started with ProIcon, program visualization, a series on the Icon optimizing compiler, and several more.

If you don't subscribe to the Analyst but are interested, we'll send you a free sample copy on request. If you're placing an order for other Icon material, just make a note on your order form. Or contact us as listed in the publication box on page 7.

Back issues of the Analyst are available while supplies last. See the order form at the end of this *Newsletter*.

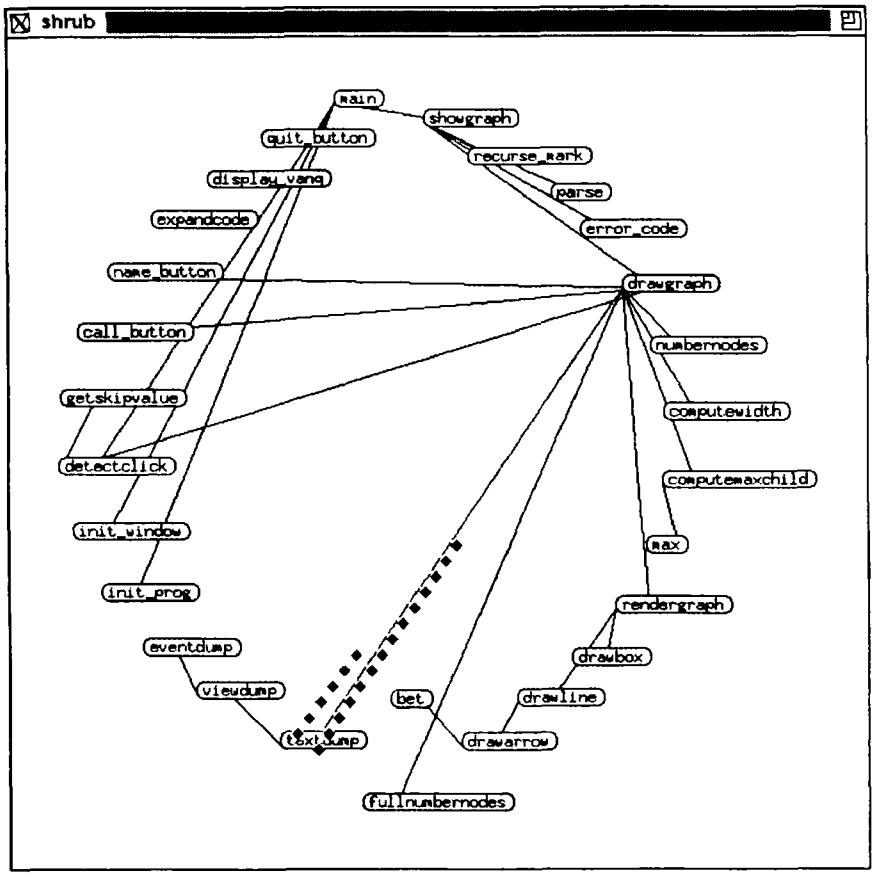


X-Window Facilities for Icon

Recent research in the Icon Project has focused on program visualization: the presentation of images depicting program behavior.

Naturally, we wanted to use Icon to develop our visualization tools. Such tools are experimental and problematical in nature, and Icon has proved to be an excellent prototyping language. And we admit to a certain bias. But Icon has no graphic capabilities.

Clint Jeffery decided to rectify this deficiency in Icon. The result is an experimental extension to Icon that provides access to many of the capabilities of X Windows.



You might ask "Why X Windows? Why not Microsoft Windows or the Macintosh?" Well, X has the capabilities we need, it is widely available, especially on UNIX platforms that we use for program development, and it's not much worse than most other graphics/window systems. (The negative tone here is intentional. All presently available graphic/window systems have problems and all are hard to program.)

The X facilities that we've added to Icon (we call the result X-Icon) are cast in terms of the low-level Xlib routines, not the higher-level X toolkits. The facilities don't match Xlib exactly; we've taken advantage of features of Icon such as generators and variable-length argument lists. X-Icon also automatically handles refreshing the contents of windows.

Although X-Icon does not support all the functionality of Xlib (which has hundreds of routines), X-Icon does provide what's needed for most graphic applications:

- *Text fonts:* write() and writes() can employ fonts with arbitrary typefaces and sizes, including proportional-width faces.
- *Raw keyboard input:* Keystrokes can be retrieved as they occur, instead of waiting for the user to press the enter key.
- *Mouse input:* The mouse interface allows programs to use menus, text selection, and other graphical screen entities.
- *Graphics:* Points, lines, arcs, smooth curves, and polygons can be freely intermixed along with text.
- *Colors:* Color can be used to enhance the presentation of both text and graphics. The color map can be changed dynamically if the hardware allows this.

Most window system interfaces use an event-driven model, in which events such as mouse clicks are handled at every instant by the program. In X-Icon, this event-driven model is optional — it's quite possible to write interesting and useful graphic applications in X-Icon using very ordinary-looking code.

X-Icon also does not enforce any particular look-and-feel. Individual programs or collections of programs, however, can provide a consistent look-and-feel.

It turns out to be quite easy to program many graphic applications in X-Icon. And, as is typical of Icon, programs are short. A program that you'd think should require 100 lines of code can be done in 100 lines, not 10 times that many, as is typical of C.

We're just beginning to use X-Icon. Some applications so far are:

- An "etch-a-sketch" program that allows two users to write on the same screen using their mice.
- A "colorizing" text browser.
- A bit map editor.
- Several tools for visualizing the execution of Icon

programs, including procedure activation trees, procedure call graphs, program histograms, and the internal structures of lists.

A snapshot of the visualization of the call graph of an Icon program is shown above. A "caterpillar" crawls along call paths between procedures, turning from black on call to red on return. The speed with which the caterpillar moves can be controlled by the person viewing the visualization. The user also can rearrange the procedure nodes to get a better layout than is provided automatically.

What's shown here doesn't really convey the full nature of the visualization — it's just a black-and-white snapshot of a color animation. We're working on videos, but that's a bit in the future yet.

X-Icon is still somewhat experimental and it's not yet available for distribution. We are considering including the X-Icon extensions as part of a combined release of the Icon compiler and interpreter for UNIX.

A technical report describing X-Icon is available free of charge with any order of Icon material of $15 or more. Just ask for the X-Icon report. Or you can purchase a copy of the report separately for $2, which includes the cost of mailing.

Getting Icon Material Via FTP

If you have access to FTP, that is by far the fastest and most reliable way to get Icon material.

FTP to cs.arizona.edu. When you are asked to log in, enter anonymous. When you are asked for a password, enter any non-empty string. Then

```
cd /icon
```

Downloading

Start by getting a guide to what's available:

```
get READ.ME
```

Look through READ.ME to see what to do next and for information about downloading.

There are several subdirectories in /icon:

- compiler: program material related to the new optimizing compiler for Icon.
- contrib: user-contributed material.
- doc: documentation.
- interpr: program material related to the Icon interpreter.
- library: Icon program library material.
- misc: Odds and ends that don't fit any other category.
- newsgrp: Archived electronic mail from the Icon newsgroup.
- tools: Programs useful for processing program material.

The subdirectory compiler has two subdirectories: packages and source. The subdirectory packages contains pre-packaged versions of the Icon compiler for several UNIX platforms. These packages are in object-code format, so that they can be installed at user-specified locations. As the name implies, source contains source code for the Icon compiler. At present it's available only for UNIX platforms.

The documentation in docs generally comes in two forms: for printing on PostScript devices and for listing on monospaced devices, such as computer terminals and line printers.

In some cases, documents contain figures and diagrams that do not lend themselves to printing on monospaced devices. For such documents, only PostScript is available.

Uploading

You also can upload material and deposit it on our system. Just

```
cd /incoming
```

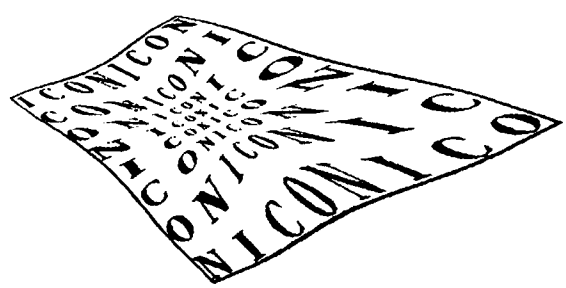
and put the files you want to upload.

If you upload files to our system, send electronic mail to icon-project@cs.arizona.edu so that we'll know the material is there and can copy it to another area before it's automatically deleted after five days.

File Transfer

File transfer via FTP usually goes well, but occasionally there are problems. These usually can be traced to transfer in the wrong mode. If you're transferring a "binary" file, such as an archived package, be sure to put FTP in image (binary) mode before the transfer.

Different versions of FTP vary in the command to do this. Try image or binary; you'll get a message that tells you if you've accomplished what you wanted. For text files, on the other hand, use the ASCII mode. Try ascii or text.



ICEBOL5

The *Fifth International Conference on Symbolic and Logical Computing* was held at Dakota State University on April 18-19, 1991.

The conference, organized by Eric Johnson, was held in the Karl E. Mundt Library, which has excellent facilities for small conferences. Digital Equipment Corporation provided financial assistance for the conference.

About 50 persons attended. As is typical for this conference, a wide range of interests and backgrounds was represented — from computing in the Humanities to programming language design and implementation.

The talks covered a similar range of interests. Several programming languages were featured, including Icon,

Prolog, Rexx, and SNOBOL4. Of the 20 papers, seven related directly to Icon:

- "Digitized Voice Management with Icon" by Jerry Nowlin and Rick Fonorow, AT&T Bell Labs and Iconic Software, Inc.
- "Making Reason out of Rhyme", Phillip Thomas, Borneo Literary and Historical Manuscript Project.
- "Database Tools for Navajo Lexicography", Kip Canfield, University of Maryland.
- "An Optimizing Compiler for Icon" by Ken Walker and Ralph Griswold, The University of Arizona.
- "X-Icon: An Icon Windows Interface" by Clint Jeffery and Ralph Griswold, The University of Arizona.
- "An Icon Program to Assist Writing for the Re-Abled", Marilyn Mantel-Guss, Goodwill Industries.
- "Icon String Scanning for Parsing Chemical Formulas and Equations" by Robert Freemen, Oklahoma State University.

Iconic Software, Inc. also demonstrated their voice management products, which have been developed using tools written in Icon.

Copies of the conference proceedings are $35, postpaid, and may be ordered from:

ICEBOL Proceedings
114 Beadle Hall
Dakota State University
Madison, South Dakota 57042-1799

Icon from ISI

Iconic Software, Inc. provides this information about one of their upcoming products:

ISIcon Does Modules

ISI's Icon implementation for UNIX systems will support module level scoping. The 386 version will be available before the end of the year. This new level of variable scoping falls between global and local, and should make Icon more attractive for large programming projects.

Modules were originally conceived as a means for solving the difficult problem of separate compilation in the ISI Icon Compiler, and while it is true that a module will be the "unit" that can be compiled separately, this extra scoping level is proving to be an intriguing and a valuable addition to the Icon language.

Prior to modules, global variables were known "everywhere", and this could cause name conflicts and associated program malfunctions, especially when an undeclared local variable had the same name as the global variable.

Modules help solve this problem, since names can now be known among associated procedures that make up the module, and nowhere else. It is now possible to protect Icon code in ucode libraries so that an undeclared local variable will not unintentionally reference a global variable used in the library.

The beauty in ISIcon modules is that they are completely upward compatible with Version 8.0. Anyone writing the standard "one-shot" Icon program won't have to worry about modules. They shouldn't even notice them. However, any programming effort that requires the use of ucode libraries, or coordination among several software developers will find module-level scoping a valuable asset.

Modules have the form

```
module identifier
  declarations
end
```

The declarations include import, export, and local. Modules can contain procedure and record declarations, which can be either local (by default) or exported. An import declaration is of the form

```
import import-list
```

Items in an import list are separated by commas and are either identifiers or identifiers followed by parentheses to indicate an imported procedure. These are variables and procedures that must be supplied by something outside the module. Built-in functions can be imported but it is not necessary. An example is:

```
import var1, proc1(), proc2()
```

There are two kinds of export declarations. One is a list of variables following the word export, and the other is either a procedure or record declaration following the word export. These items are added to the program's global name space, but are read-only outside the module. (Read-write variables can be declared global outside the module and imported). Examples are:

```
export x, y
export procedure foo()
  ...
end
```

Local identifiers have a global lifetime (because modules have a global lifetime), but are not visible outside the module. Local identifiers include those declared with local, and procedures and records declared without export. Examples are:

```
local a, b
procedure bar()
  ...
end
```

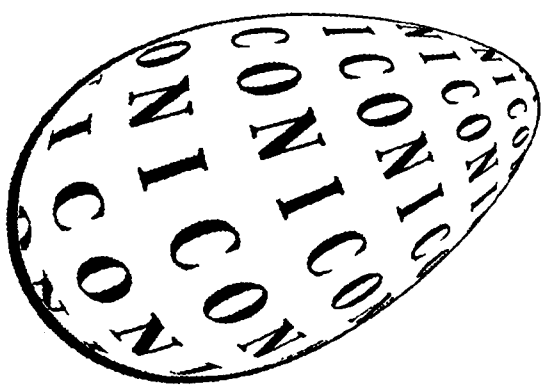
For more information on the availability of ISIcon, contact Iconic Software, Inc. by electronic mail at

```
uunet!isidev!isi
```

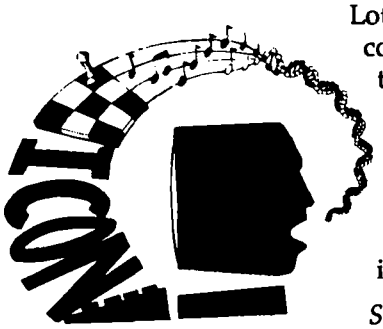
or write to:

Iconic Software, Inc.
318 North Center Street
Plano, IL 60545

IsI



Programming Corner



Lots of interesting things come across our "electronic desk". Here's an exchange that dates back to 1987 that we've just unearthed. It's a bit on the bizarre side, but you may find it worth a chuckle.

Steve Wampler: I suspect you've seen this kind of thing before, but it's the first time a student has come up with it. After talking about lists, stacks, and queues, I gave as part of an assignment the problem of writing a file out with the lines displayed last-to-first. One student wrote:

```
procedure main()
  if line := read() then {
    main()
    write(line)
    }
end
```

Oh well ...

Ken Walker: This can of course be written more compactly:

```
procedure main()
  write(read(), main())
  return
end
```

By the way, didn't something like this come up before?

Dave Gudeman: How about a more Icon-like solution:

```
procedure main()
  write(read(), main() | "")
end
```

Steve Wampler: Compare these two programs:

```
procedure main()
  (write(read(), main()))
  return
end

procedure main()
  (write(read()), main())
  return
end
```

(The returns aren't really needed, but one of them would have to be modified slightly to work properly — destroying the similarity.)

On a saner note, Steve Wampler also contributes the following procedure:

```
# sleep (restlessly) for n seconds
procedure sleep(n)
  local start
  start := &time
  while &time < start + n * 1000
  return
end
```

He comments "The nice thing about it is that it can 'sleep' for fractions of a second — sleep(0.5) sleeps for half a second". Note that this depends on adequate clock resolution.

The Icon Newsletter

Madge T. Griswold and Ralph E. Griswold
Editors

The Icon Newsletter is published three times a year, at no cost to subscribers. To subscribe, contact

Icon Project
Department of Computer Science
Gould-Simpson Building
The University of Arizona
Tucson, Arizona 85721
U.S.A.

(602) 621-8448

fax: (602) 621-4246

Electronic mail may be sent to:

icon-project@cs.arizona.edu

or

...{uunet,allegra,noao}!arizona!icon-project

THE UNIVERSITY OF
ARIZONA
TUCSON ARIZONA

and

The Bright Forest Company
Tucson Arizona

From Our Mail

You have been so kind to continue to keep me on your Icon Newsletter list that I feel I should reciprocate with some old-time stories when the occasion presents itself. "Saddle stitching" on page 7 of No. 34 is such an occasion.

When a car turns, the wheels on the inside of the turn should be turned more sharply than those on the outside since they must follow a smaller turning circle. This is relatively unimportant if the tires are steel as on a wagon, since the slippage is more or less not noticed. But the different amount of turn for the inside and the outside wheels is very important if the tires are rubber and resist slipping.

Near the turn of the century two Yankee brothers named Sheldon invented and patented a lever system for wagons that provided this differential turning. It was just at the beginning of the Automobile Revolution and all automobile manufacturers had to pay them for their patent. They were devoted to inventing and one of the pair decided to invent a machine that would do saddle stitching, then (and perhaps now) done only by hand. He retired to a barn for a year and came out with his invention only to discover that in the meantime stapling had been invented and there was no market for his machine.

All this is told by the other brother, or a descendant, in a stapled pamphlet called "Unscrewing the Inscrutable", which is somewhere on my shelves. The story is my personal reminder that successful inventions and innovations must be more than clever; they must be needed. I have been involved in several clever enterprises that did not meet this last criteria.

I started this letter with the intention of amusing you with an old time story. I see now that it could be read as a comment on Icon. I do not so mean it. Has my sub-conscious taken over my word processor?

Yours,

Eric A. Weiss.

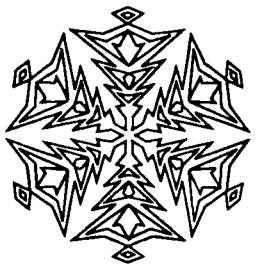
Amazing! Thank you for passing this along to us.

Please send me source for the Icon programming language to my e-mail address at the head of this message. If it's too big for one message, please break it up into smaller pieces.

Electronic mail is fine for messages and for sending small files, but it's impractical for sending large and complex programs. The source for Icon is too large to send by electronic mail. Just the code alone, not allowing for all the scripts and programs that are needed to compile and test it, amounts to about 1.4MB and consists of about 4,800 lines. It's also organized in a hierarchy that's not easy to package for flat file transfer.

Finally, we're willing to make Icon freely available for interested persons to pick up from our RBBS or by FTP. We also provide physical copies for a nominal charge. But we don't have the resources to initiate file transfers or handle special requests.

SNOBOL4 Corner



Catspaw, Inc., has enhanced its SPITBOL-386 product to be compatible with the new 32-bit DOS Protected Mode Interface (DPMI) standard now becoming popular.

Formerly, SPITBOL-386 has included a DOS Extender from PharLap Software that permitted 32-bit SPITBOL to use all available memory on 80386 and 80486 MS-DOS systems. This DOS Extender was compatible with native DOS and with expanded memory managers implementing the Virtual Control Program Interface (VCPI). It was not compatible with Microsoft Windows Enhanced mode, and licensing restrictions prevented the generation and free distribution of EXE files.

SPITBOL-386's new DPMI-compliant DOS Extender allows it to run both on native MS-DOS systems and under environments such as Windows Enhanced mode that support DPMI. Using this DOS Extender, SPITBOL-386 now can generate stand-alone, royalty-free executable files. In addition, a built-in virtual memory manager enlarges SPITBOL's workspace to 4 gigabytes, limited only by free disk space.

SPITBOL-386 now includes both VCPI- and DPMI-compliant versions, allowing users to select the systems appropriate for their operating system environments.

Contact Catspaw, Inc. for pricing and upgrade information:

Catspaw, Inc.
P.O. Box 1123
Salida, CO 81201

voice: 719-539-3884
fax: 719-539-4716

Downloading Icon Material

Most implementations of Icon are available for downloading electronically:

BBS: (602) 621-2283

FTP: cs.arizona.edu (cd /icon)

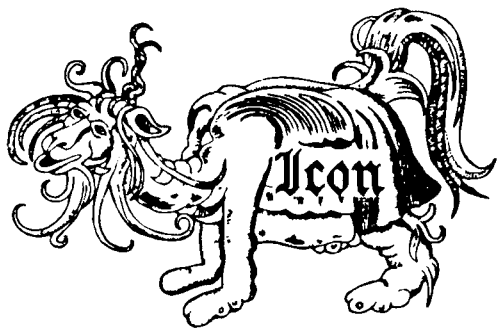
Ordering Icon Material

What's Available

There are implementations of Icon for several personal computers, as well as for CMS, MVS, UNIX, and VMS. Source code for all implementations is available. Program material is accompanied by installation instructions and users' manuals in printed and machine-readable form.

There also is a program library that contains a large collection of Icon programs and procedures, as well as an object-oriented version of Icon that is written in Icon.

In addition to users manuals that are included with program material, there are three books and two newsletters about Icon.



Icon Program Material

The current version of Icon is 8. All the program material here is for Version 8.

All program material is in the public domain except the MS-DOS/386 implementation of Icon, which is a commercial product that carries a standard software license.

Personal Computers: Executables, source code, and the Icon program library for personal computers are provided in separate packages. Each package contains documentation in printed and machine-readable form. *Note:* Icon for personal computers requires at least 640KB of RAM; it requires more on some systems.

CMS and MVS: The CMS and MVS packages contain executables, source code, test programs, the Icon program library, and documentation in printed and machine-readable form.

UNIX: The UNIX package contains source code but not executables, test programs, related software, the Icon program library, and documentation in printed and machine-readable form. UNIX Icon can be configured for most UNIX systems. *Note:* executables for Xenix and the UNIX PC are available separately.

VMS: The VMS package contains object code, executables, source code, test programs, the Icon program library, and documentation in printed and machine-readable form.

Porting: Icon source code for porting to other computers is distributed on MS-DOS format diskettes. There are two versions, one with a flat file system and one with a hierarchical file system.

Update Subscriptions: Updates to the Icon source code and the Icon program library are available by subscription.

Source-code updates are distributed on MS-DOS diskettes in ARC format for hierarchical file systems, and are suitable for compilation under MS-DOS or for porting to new computers. Each update usually provides a completely new copy of the source. A source-code subscription provides five updates. Updates are issued about three times a year.

Icon program library updates are distributed on MS-DOS diskettes in plain ASCII format. A library subscription provides four updates. Updates are issued about three times a year.

Documentation

In addition to the installation guides and users' manuals included with the program packages, there are three books on Icon. One contains a complete description of the language, the second describes the implementation of Icon in detail, and the third is an introductory text designed primarily for programmers in the Humanities.

There are two newsletters. *The Icon Newsletter* contains news articles, reports from readers, information of topical interest, and so forth. It is free, and is sent automatically to anyone who places an order for Icon material. There is a nominal charge for back issues of the *Newsletter*.

The Icon Analyst contains material of a more technical nature, including in-depth articles on programming in Icon. There is a subscription charge for the Analyst.

Payment

Payment should accompany orders and be made by check, money order, or credit card (Visa or MasterCard). The minimum credit card order is $15. Remittance *must* be in U.S. dollars, payable to The University of Arizona, and drawn on a bank with a branch in the United States. Organizations that are unable to pre-pay orders may send purchase orders, subject to approval, but there is a $5 charge for processing such orders.

Prices

The prices quoted here are good until August 1, 1991. After that, prices are subject to change without further notice. Contact the Icon Project for more current pricing information.

Ordering Instructions

Media: The following symbols are used to indicate different types of media:

- 9-track magnetic tape
- DC 300 XL/P cartridge
- 360K 5.25" diskette
- 400K 3.5" diskette
- 800K 3.5" diskette

All cartridges are written in raw mode. All diskettes are written in MS-DOS format except for the Amiga, the Atari ST, and the Macintosh.

CMS and MVS tapes are available only at 1600 bpi. When ordering UNIX or VMS tapes, specify 1600 or 6250 bpi (1600 bpi is the default). When ordering diskettes that are available in more than one size, specify the size (5.25" is the default).

Shipping Charges: The prices listed include handling and shipping by parcel post in the United States, Canada, and Mexico. Shipment to other countries is made by air mail only, for which there are additional charges as noted in brackets following the price. For example, the notation $15 [$5] means the item costs $15 and there is a $5 shipping charge to countries other than the United States, Canada, and Mexico. UPS and express delivery are available at cost upon request.

Ordering Codes: When filling out the order form, use the codes given in the second column of the list to the right (for example, AME, ATE, ...).

Icon Executables				
Amiga	AME		$15	[$5]
Atari ST	ATE		$15	[$5]
MS-DOS	DE	(2) or	$20	[$5]
MS-DOS/386	DE-386	or	$25	[$5]
Macintosh (MPW)	ME		$15	[$5]
OS/2	OE	or	$15	[$5]
UNIX PC	UE	or	$15	[$5]
Xenix	XE	(2) or	$15	[$5]
Xenix/386	XE-386	or	$15	[$5]
Icon Source				
Amiga	AMS		$15	[$5]
Atari ST	ATS		$15	[$5]
MS-DOS and OS/2	DS	(2) or	$20	[$5]
Macintosh (MPW)	MS		$15	[$5]
Porting (flat, ASCII)	PFS	(5) or (2)	$40	[$8]
Porting (hier., ARC)	PHS	(2) or	$20	[$5]
Source Updates (5)	SU	(2) or	$50	[$15]
Icon Program Library				
Amiga	AML		$15	[$5]
Atari ST	ATL		$15	[$5]
MS-DOS and OS/2	DL	or	$15	[$5]
Macintosh (MPW)	ML		$15	[$5]
Porting (ASCII)	PL	(2) or	$15	[$5]
UNIX (cpio)	UL	(2) or	$15	[$5]
Library Updates (4)	LU	or	$30	[$12]
Complete Systems				
CMS	CT		$30	[$10]
MVS	MT		$30	[$10]
UNIX (cpio)	UT-C		$30	[$10]
UNIX (cpio)	UC-C		$45	[$10]
UNIX (cpio)	UD	(9) or (4)	$40	[$8]
UNIX (tar)	UT-T		$30	[$10]
UNIX (tar)	UC-T		$45	[$10]
VMS	VT		$30	[$10]
Books				
The Icon Programming Language (2nd ed.)		LB	$34	[$13]
The Implementation of Icon + update		IB	$50	[$14]
Icon Programming for Humanists + disk		HB	$30	[$10]
Newsletters				
The Icon Newsletter (all back issues, 1-35)		INC	$15	[$5]
The Icon Newsletter (single issues, each)		INS	$1	[$2*]
The Icon Analyst (1 year, 6 issues)		IA	$25	[$10]
The Icon Analyst (single issues, each)		IAS	$5	[$2*]

* Per order, regardless of the number of issues purchased.

Order Form

Icon Project • Department of Computer Science
Gould-Simpson Building • The University of Arizona • Tucson AZ 85721 U.S.A.

Ordering information: (602) 621-8448 • Fax: (602) 621-4246

name _____

address _____

_____

city _____ state _____ zipcode _____

(country) _____ telephone _____

☐ check if this is a new address

qty.	code	description	price	shipping*	total

subtotal	
sales tax (Arizona residents)	
extra shipping charges*	
purchase-order processing	
other charges	
total	

Make checks payable to The University of Arizona

The sales tax for residents of the city of Tucson is 7%.
It is 5% for all other residents of Arizona.

Payment ☐ Visa ☐ MasterCard ☐ check or money order

I hereby authorize the billing of the above order to my credit card: ($15 minimum)

card number

exp. date

name on card (please print) _____

signature _____

*Shipping charges apply only to addresses outside the United States, Canada, and Mexico