THE UNIVERSITY OF ARIZONA

TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

Icon Newsletter #22

Madge T. Griswold and Ralph E. Griswold

October 21, 1986

1. Newsletter Subscriptions

Whenever persons ask for information about Icon or order copies of the implementation, we add their names to our mailing list for the Icon Newsletter on the assumption that they may be interested in new developments. We presume some inquiries come only from passing curiosity, and we probably have lots of persons on our mailing list who no longer are interested in Icon. Since sending out Newsletters is both expensive and time-consuming, we would like to bring our subscription list up to date and remove persons who no longer want to receive the Newsletter. To this end, a subscription-verification form and questionnaire are included with this mailing.

In order to remain on our mailing list, you *must* return the form. In return, we will send you at least the next three Icon Newsletters without charge.

The questionnaire is designed to give us and our readers a better understanding of the interests of the 'Icon community'. It is not necessary for you to complete the questionnaire to remain on our mailing list — just the verification form will do — but we hope you will complete the questionnaire as well. A compilation of returned questionnaire responses will be published in a subsequent Newsletter.

2. Implementation News

MS-DOS

Version 6 of Icon for MS-DOS is now available for distribution. There are both large- and small-memory-model versions. The large-memory model version is recommended for applications involving large programs or for processing large amounts of data.

The Icon program library for Version 6 also is available on a separate diskette. See the order form at the end of this Newsletter.

Source code for the MS-DOS implementation is not available yet. We hope to have it ready in about a month. If you are interested in the source code and do not want to wait for the next Newsletter, drop us a note and we will let you know when it is available.

UNIX

Version 6 of Icon has been successfully installed on a wide range of computers running UNIX. These include the Amdahl 580, AT&T 3B2/5/15/20, HP 9000, PDP-11, Pyramid 90x, Ridge 32, Sun Workstation, and VAX-11. There also are PC implementations for PC/IX and SCO XENIX V.

The basic UNIX implementation is available on magnetic tape and cartridges. The PC implementations are distributed on 5-¼″ 2S/DD diskettes. See the order form at the end of this Newsletter.

Implementations in Progress

Several implementations for the Macintosh are in progress, and some success has been reported. To the best of our knowledge, no implementations are available for distribution at this time.

An implementation of Icon for the Amiga has been reported, but we have been unable to verify its existence.

An implementation for IBM 370 mainframes is in progress.

More information about these implementations will be published in future Icon Newsletters.

3. The Implementation Book

The Implementation of the Icon Programming Language, which describes Version 6, is in press and should be available in January, 1987. This book focuses on the Icon run-time system, with particular emphasis on data layout, the Icon virtual machine, expression evaluation, and storage management. The book, which is being published as the first in a series on computer science by Princeton University Press, consists of 336 pages, clothbound, at a price of $39.50.

4. Electronic Access

Program Material via FTP

As mentioned in the last Newsletter, Icon program material is available to persons who have access to ARPA Internet via an anonymous FTP to arizona.edu. The material presently available includes the UNIX distribution in *cpio* and *tar* formats and the MS-DOS implementation in *arc* format. To see what is available and how large the files are, get icon/README.

Electronic Bulletin Board at the University of Arizona

We have established an electronic bulletin board so that persons can dial up and download Icon program material. Initially the material available will be limited to executable binary files and the Icon program library for MS-DOS. Other material may be added later.

Both Kermit and Xmodem protocols are supported. There is also a limited message facility.

The telephone number of the bulletin board is (602) 621-2283. Before dialing, set your computer's telecommunication program to full duplex, 7 bits, even parity, 1 stop bit, at a baud rate of 2400, 1200, or 300. The initial answering baud rate is 1200. If you dial in at another rate, you will need to cycle the baud rate detection with break/carriage-return sequences until the answering modem recognizes your baud rate.

When you are properly connected, you should receive the following message:

```
xenix286!login:
```

Respond to this with guest followed by a carriage return and you will get a menu of the available services. From there, the operation of the bulletin board should be self-explanatory.

The bulletin board is scheduled to be in service from 5 pm MST to 8 am MST, as well as all day on weekends. It also will be in service during daytime hours when the computer on which it runs is not otherwise occupied, but service interruptions during daytime hours may occur without warning.

BIX

The BYTE bulletin board BIX has a discussion group for Icon and SNOBOL4. Information about BIX can be found in any recent issue of BYTE. See, for example, page 271 of the October 1986 issue. As a BIX subscriber, you can enroll in the discussion group by

```
join other.langs/icon.snobol
```

5. From our Mail

In the last Newsletter we started publishing some questions of general interest from our correspondence, together with answers. Here are more.

My local bookstore says The Icon Programming Language *is out of print. How can I get a copy?*

We continue to get such reports from time to time. The Icon book is *not* out of print. Your problem is probably a misunderstanding or possibly the unwillingness of your bookstore to go to the trouble of ordering a book they do not have in stock. Of course, as with any book, it is possible for the publisher to run out of stock temporarily, although that has not happened yet with the Icon book. In any event, you can purchase the book from the Icon Project. See the

order form at the end of this Newsletter.

We are concerned about using Icon in a commercial environment. What guarantee do we have of support?

None, really. Icon is not a commercially marketed software system and we do not have the resources to provide customer support in any official way. We do, however, attempt to solve problems and correct bugs. We think we've done a good job of this so far. Furthermore, the source code for Icon is in the public domain, so users often can solve their problems themselves.

I thought the large-memory model was designed to overcome the segmentation problems on 8086-style processors. The Macintosh uses the Motorola 68000, which does not have segmented addressing. Can you clear up my misunderstanding?

The term 'large-memory model' may be misleading. The real problem has to do with the size of pointers and ints in C. If they are sized differently by a particular C compiler, regardless of the underlying architecture, there can be serious problems. This is especially true for a program written on the assumption that the sizes are the same, as Icon originally was. We think we now have the problem in hand.

Why is it so hard to implement Icon for the large-memory model? Can't you just

```
#define int long
```

and get a slow but workable implementation quickly?

We only wish it were so simple. Among the many problems are that run-time library routines commonly expect ints and return ints. Note also that C defines the difference of two pointers to be an integer, which may be either an int or a long, depending on the particular C compiler. Add to this the fact that some C compilers store longs and pointers with different byte orders. And so on. You have to live through the conversion of a system written on the assumption that ints and pointers are the same size to appreciate the full range of the problems.

Are you planning a compiler to replace the Icon interpreter?

No. This question continues to come up and often is a result of not understanding how Icon actually is implemented. True, there is an interpreter. However, Icon source programs are translated into machine code for an imaginary virtual machine. While this virtual machine code is interpreted, most virtual machine instructions are executed in compiled code. There is some overhead related to the interpretation, but the approach we have used is portable (no machine-dependent code generators are required) and flexible (we can change the virtual machine very easily). Furthermore, the interpreter approach allows programs to get into execution quickly without the delay of a loading phase. The main disadvantage of the interpreter approach, from our viewpoint, is the relative difficulty of adding user-coded C functions to extend Icon's built-in repertoire.

There has been a long-standing problem on VAX/VMS with the library function POW, *which returns 0.0 whenever the base is negative and the exponent is integral. DEC at first claimed this was a feature, but we persisted and they now have announced that the VMS C run-time library will correct the problem subsequent to Version 4.5.*

Congratulations! Several of our users have complained about this problem. It is encouraging to learn that your tenacity paid off.

6. Programming Corner

Archiving Programs

Our readers frequently ask for some examples of simple programs that do not require an extensive knowledge of Icon to understand. Here are two that are quite simple and lend themselves to extensions that provide good exercises for beginning Icon programmers.

Most computer systems have some kind of a facility for combining files within a common file that can be used for transporting a large number of small files from one place to another or just to keep track of them. Such archiving facilities have many features, but most of them are system-dependent.

Here are two simple, relatively portable, Icon programs for archiving files and de-archiving them. The idea is simply to concatenate the files to be archived to make one large file. A header precedes each file, giving its name. An rather arbitrary string, "!!!!!", is chosen to distinguish headers; it cannot occur in any file to be archived (but see the exercises).

The archiving program takes a list of file names to be archived from standard input:

```
procedure main()
   while name := read() do {
      input := open(name) | stop("cannot open \"",name,"\"")
      write("!!!!!",name)
      while write(read(input))
      close(input)
      }
end
```

Note the use of an alternative to produce an error message in case a named file cannot be opened. On most systems it is important to close a file once it is no longer needed, as shown, to release space used by the i/o system for reuse.

The de-archiving program is just a little more complicated:

```
procedure main()
   while line := read() do {
      line ? if ="!!!!!" then {
         close(\out)
         out := open(name := tab(0),"w") | stop("cannot open ",name)
         }
      else write(out,line)
      }
end
```

As lines are read in, they are examined for headers. For archives constructed by the program above, the first line is always a header. (What would happen if it were not?) When a header is found, the previous file is closed, the new file name is taken from the rest of the line, and the new file is opened. If the line is not a header, however, it is written to the current file.

There is one 'trick' used here that is, in fact, good idiomatic style in Icon — the file is closed only if it is not null-valued. This happens only when the first header is encountered. At this point, out is null-valued because no assignment has been made to it yet. Thus, \out fails and close is not called. Subsequently, out is assigned a (non-null) file value and \out succeeds. This idiom takes advantage of the fact that the initial values of variables in Icon are null. It saves special coding for the start-up case.

The two programs above are crude, but generally workable. There are all kinds of possibilities for improvements and extensions:

- Modify the archiving program so that the heading string can be specified by the user.
- Modify the de-archiving program to figure out the heading string.
- Modifying the archiving program so that the heading string contains creation date and time information.
- Modify the de-archiving program so that only specified files are extracted.
- Add the facility to list the contents of an archive file without extracting the files.
- Extend the archiving facility to handle arbitrary binary files.
- Combine all the archiving and de-archiving facilities in one program that prompts the user for commands.

A Simple Calculator

The following program, based on one written by Steve Wampler when his hand-held calculator broke, is a good illustration of the use of lists as stacks and of the motivation for 'string invocation' of operations:

```
procedure main()
   local stack, token, arg1, arg2

   stack := []
   while token := read() do
      if numeric(token) then push(stack,token)
      else if proc(token,2) then {
         arg2 := pop(stack) | {
            write(&errout,"*** empty stack ***")
            next
            }
         arg1 := pop(stack) | {
            write(&errout,"*** empty stack ***")
            next
            }
         push(stack,result := token(arg1,arg2)) |
             write(&errout,"*** operation failed ***")
         write(result)
      }
      else write(&errout,"*** invalid entry ***")
end
```

The calculator takes successive lines of input as numerical computations in reverse-polish notation. If the input token is numeric, it is pushed. Otherwise, there is a check to see if the token is a binary operator: proc(token, 2). This function, which is an extension to Version 5 of Icon, fails if token is not a string representing a binary operator. For example, "+" is such a string, but "$" is not. If the token represents a binary operator, two arguments are popped off the stack and the operator is applied to them. The result is pushed and the loop continues.

As exercises, consider the following:

- Why is the result of the computation assigned to result, pushed, and then written out separately?
- Consider the effect of different numeric types — such as integer, floating-point, and mixed-mode computations. Modify the calculator so that it performs only floating-point arithmetic.
- Rewrite the program to make it as short as possible and using the least number of identifiers.
- The program is designed to handle only binary operators. Modify it to handle unary ones also. Take care to consider operator symbols such as "–" that are both binary and unary.
- What happens if a token is the name of a function or procedure instead of an operator? Consider how this can be used to extend the usefulness of the calculator.
- There is no particular reason why the calculator should be limited to numeric data. Extend it to handle strings.
- Add a facility for commands that do not affect the stack but instead alter the mode of computation.

Acknowledgements

Generally we do not acknowledge work done by persons in our organization, although you will see their names on reports and papers from time to time. Many persons outside our organization make substantial contributions, however. These include numerous persons who have provided configuration information for different UNIX systems, persons who have made helpful suggestions about distribution, and others who have made suggestions for improvements to Icon and its documentation. Several dozen persons have helped in such ways recently. It is impractical to list everyone individually — and we surely would omit someone by accident. Nonetheless, these contributions are sincerely appreciated and have benefited many users of Icon.

There is one person who deserves special mention. Cheyenne Wills implemented Version 6 of Icon for MS-DOS, including solving many of the problems related to the so-called 'large-memory model'. This was a very difficult and time-consuming job. Those of you who get this implementation owe much to his efforts.

Ordering Information

Distribution: Icon for personal computers is distributed on 2S/DD 5-¼" diskettes. Other implementations are distributed on magnetic tape and cartridges. The prices listed include media. Some other formats are available as special orders; if you need a format that is not listed below, ask us.

Delivery Charges: The charges listed below include parcel-post delivery in the United States, Canada, and Mexico. Shipment to other countries is made by air mail only, for which there are additional charges as follows: $5 per diskette package, $10 per tape or cartridge package, and $10 per documentation package. Commercial express delivery will be provided if authorization and an account number to charge is provided with the order.

Payment: Payment may be made by check or money order, but credit card orders cannot be accepted. Remittance *must* be in U.S. dollars, payable to The University of Arizona. There is a $10 service charge for a check drawn on a bank that does not have a branch in the United States. Payment should accompany the order. For organizations that cannot provide pre-payment, purchase orders will be accepted, but there is a $5 charge for processing such orders.

The UNIX Package: The UNIX package listed below includes source code, test programs, a library of Icon programs, and documentation in both printed and machine-readable form (but not the Icon book). The system can be configured to run on most computers with UNIX-based operating systems, including the VAX, Sun Workstation, IBM RT PC, AT&T 3B, and Pyramid. Since object and executable files depend on the target computer, they are not included on the tape.

The VMS Package: The VMS package contains the same material as the UNIX package, except that all material not needed for VMS is omitted. Object and executable files are included. It requires VMS Release 4.2 or higher.

The MS-DOS Packages: The MS-DOS implementation of Icon runs on computers with 8086/88/186/286-family processors. IBM hardware compatibility is not necessary. MS- or PC-DOS Version 2.0 or higher is required. There are both small- and large-memory-model implementations. The small-memory model uses 192 kilobytes of memory, while the large-memory model requires at least 256 kilobytes. The small-memory model is faster and more compact than the large-memory model, but it cannot handle programs that need large amounts of storage. It is inadvisable to run both the small- and large-memory-model implementations on the same computer because of file naming conflicts. Source code is not included; it will be released later. The Icon program library for MS-DOS is the same for the small- and large- memory models and is distributed on a separate diskette.

The PC/IX Packages: The PC/IX implementation of Icon is a small-memory-model one that has only 64 kilobytes for data. Source code and the Icon program library are distributed separately. The source code diskettes contain configuration information and test suites for other UNIX installations, such as the IBM RT PC. All PC/IX diskettes are written in *dump/restore* format, which is the same format as *backup/restore* on the RT PC under AIX.

The XENIX Packages: The XENIX implementation of Icon is for SCO XENIX V. There are both small- and large-memory model implementations. It is inadvisable to run both the small- and large-memory-model implementations on the same computer because of file naming conflicts. The Icon program library is distributed separately. Source code for XENIX is not yet available. All XENIX diskettes are written in *tar* format.

The Documentation Package: This package contains a copy of the Icon book mentioned above, a description of recent additions to the language, a compilation of material from the programming corners of previous Icon Newsletters, and other material related to programming in Icon.

Order Form

Ship to:	Return this form with payment to:
Name _____	Icon Project
Address _____	Department of Computer Science
_____	The University of Arizona
_____	Tucson, AZ 85721
Telephone _____	

UNIX Icon (9-track magnetic tape) options: ☐ tar ☐ cpio ☐ 1600 bpi ☐ 6250 bpi	$25	_____
UNIX Icon (3M DC 300 XL/P cartridge) options: ☐ tar ☐ cpio	$40	_____
VAX/VMS Icon (9-track magnetic tape) options: ☐ 1600 bpi ☐ 6250 bpi	$25	_____
MS-DOS Icon, small-memory model (diskette)	$15	_____
MS-DOS Icon, large-memory model (diskette)	$15	_____
MS-DOS Icon Program Library (diskette)	$15	_____
PC/IX Icon (diskette)	$15	_____
PC/IX Icon source code (4 diskettes)	$35	_____
PC/IX Icon Program Library (diskette)	$15	_____
XENIX Icon, small-memory model (diskette)	$15	_____
XENIX Icon, large-memory model (diskette)	$15	_____
XENIX Icon Program Library (diskette)	$15	_____
Documentation Package	$28	_____
Other charges (see above)		_____
Total		_____