Version 8.10 of Icon for OS/2

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Introduction

Version 8.10 of Icon runs under OS/2 2.0. It is a 32-bit application. A math co-processor is supported and used if present; otherwise software emulation is used.

This implementation of Icon is in the public domain and may be copied and used without restriction. The Icon Project makes no warranties of any kind as to the correctness of this material or its suitability for any application. The responsibility for the use of Icon lies entirely with the user.

The basic reference for Version 8 of Icon is the second edition of the book *The Icon Programming Language* [1]. This book is available from the Icon Project at The University of Arizona. It also can be ordered through any bookstore that handles special orders. The new features of Version 8.10 of Icon are described in an accompanying technical report [2]. Version 8.10 of Icon for OS/2 supports the graphic capabilities of X-Icon [3]. See that document for limitations and features specific to OS/2. See also the file README on the distribution diskette.

2. Installing OS/2 Icon

The distribution is contained in several files in LHarc (lzh) format. A copy of lh.exe is included for dearchiving. The distribution files are:

docs.lzh	documents
icon.lzh	executable binary files
lh.exe	dearchiving utility
readme	installation overview and recent notes
samples.lzh	sample Icon programs and data

There are four executable binary files for Version 8.10 of Icon:

icont.exe	text-mode translator
iconx.exe	text-mode interpreter
xicont.exe	PM translator
xiconx.exe	PM interpreter

These files should be located at a place on your PATH specification.

To install the .exe files, set your current directory to the desired place, place the appropriate distribution in a diskette drive, and dearchive the files there using lh.exe. For example, to dearchive the executable binary files using drive A, the following will do:

```
a:lh x a:icon.lzh
```

The same technique can be used for extracting the remaining files.

3. Running OS/2 Icon — Basic Information

As indicated in the preceding section, there are two versions of OS/2 Icon. The text-mode version (icont and iconx) provides a command-line interface and handles input and output from the console. The PM version (xicont and xiconx) emulates the console from a separate window. The PM version supports the X-Icon graphics facilities; the text-mode version does not. Depending on the circumstances, you may wish to run one version or the other.

Files containing Icon programs must have the extension .icn. Such files should be plain text files (without line numbers or other extraneous information). The icont translator produces an executable file. For example, an Icon program in the file prog.icn is translated by

```
icont prog.icn
```

The result is an executable file with the name prog.exe.

The extension .icn is optional on command lines. For example, it is sufficient to use

```
icont prog
```

Note: Although the Icon translators produce executable files, the interpreters iconx.exe and xiconx.exe must be on PATH.

4. Testing the Installation

There are a few programs on the distribution diskette that can be used for testing the installation and getting a feel for running Icon:

hello.icn — This program prints the Icon version number, time, and date. Run this test as

```
icont hello
hello
```

cross.icn — This program prints all the ways that two words intersect in a common character. The file cross.dat contains typical data. Run this test as

```
icont cross
cross <cross.dat
```

meander.icn — This program prints the ''meandering strings'' that contain all subsequences of a specified length from a given set of characters. Run this test as

```
icont meander
meander <meander.dat
```

roman.icn — This program converts Arabic numerals to Roman numerals. Run this test as

```
icont roman
roman
```

and provide some Arabic numbers from your console.

If these tests work, your installation is probably correct and you should have a running version of Icon.

5. More on Running Icon

For simple applications, the instructions for running Icon given in Section 3 may be adequate. The icont translator supports a variety of options that may be useful in special situations. There also are several aspects of execution that can be controlled with environment variables. These are listed here. If you are new to Icon, you may wish to skip this section on the first reading but come back to it if you find the need for more control over the translation and execution of Icon programs.

5.1 Arguments

Arguments can be passed to the Icon program by appending them to the command line. Such arguments are passed to the main procedure as a list of strings. For example,

```
prog text.dat log.dat
```

runs the icode file prog.icx, passing its main procedure a list of two strings, "text.dat" and "log.dat". These arguments might be the names of files that prog.icn reads from and writes to. For example, the main procedure might begin as follows:

```
procedure main(a)
   in := open(a[1]) | stop("cannot open input file")
   out := open(a[2],"w") | stop("cannot open output file")
                         .
                         .
                         .
```

5.2 More About the Translator

The icont translator can accept several Icon source files at one time. When several files are given, they are translated and combined into a single icode file whose name is derived from the name of the first file. For example,

```
icont prog1 prog2
```

translates the files prog1.icn and prog2.icn and produces one icode file, prog1.icx.

A name other than the default one for the icode file produced by icont can be specified by using the –o option, followed by the desired name. For example,

```
icont –o probe.icx prog
```

produces the icode file named probe.icx rather than prog.icx.

If the –c option is given to icont, the translator stops before producing an icode file and intermediate ''ucode'' files with the extensions .u1 and .u2 are left for future use (normally they are deleted). For example,

```
icont –c prog1
```

leaves prog1.u1 and prog1.u2, instead of producing prog1.icx. These ucode files can be used in a subsequent icont command by using the .u1 name. This saves translation time for subsequent uses. For example,

```
icont prog2 prog1.u1
```

translates prog2.icn and combines the result with the ucode files from a previous translation of prog1.icn. Note that only the .u1 name is given; the .u2 name is implied. The extension can be abbreviated to .u, as in

```
icont prog2 prog1.u
```

Ucode files also can be added to a program using the link declaration.

Icon source programs may be read from standard input. The argument – signifies the use of standard input as a source file. In this case, the ucode files are named stdin.u1 and stdin.u2 and the icode file is named stdin.icx.

The informative messages from the translator can be suppressed by using the –s option. Normally, both informative messages and error messages are sent to standard error output.

The –t option causes &trace to have an initial value of –1 when the icode file is executed. Normally, &trace has an initial value of 0.

The option –u causes warning messages to be issued for undeclared identifiers in the program.

5.3 Environment Variables

When an icode file is executed, several environment variables are examined to determine execution parameters. The values assigned to these variables should be numbers.

Environment variables are particularly useful in adjusting Icon's storage requirements. Particular care should be taken when changing default values: unreasonable values may cause Icon to malfunction.

The following environment variables can be set to adjust Icon's execution parameters. Their default values are listed in parentheses after the environment variable name:

TRACE (undefined). This variable initializes the value of &trace. If this variable has a value, it overrides the translation-time –t option.

NOERRBUF (undefined). If this variable is set, &errout is not buffered.

STRSIZE (65000). This variable determines the size, in bytes, of the initial region in which strings are stored. If additional string regions are needed, they may be smaller.

BLKSIZE (65000). This variable determines the size, in bytes, of the initial region in which Icon allocates lists, tables, and other objects. If additional block regions are needed, they may be smaller.

COEXPSIZE (2000). This variable determines the size, in 32-bit words, of each co-expression block.

MSTKSIZE (10000). This variable determines the size, in words, of the main interpreter stack.

6. Features of OS/2 Icon

OS/2 Icon supports all the features of Version 8.10 of Icon, with the following exceptions and additions:

- The –x option works for icont.exe but not for xicont.exe.
- For files opened in the translate mode, the position produced by seek() may not reflect the actual byte position because of the translation of carriage-return/line-feed sequences to line-feed characters.
- Path specifications can be entered using either a / or a \. Examples are:

```
A:\ICON\TEST.ICN
A:/ICON/TEST.ICN
```

- The following OS/2 device names can be used as file names:

console	CON
printer	PRN LST LPT LPT1
auxiliary port	AUX COM RDR PUN
null	NUL NULL

For example,

```
prompt := open("CON", "w")
```

causes strings written to prompt to be displayed on the console. Use of a null file name means no file is created.

As mentioned above, xiconx supports the graphic facilities of X-Icon. A few features are handled differently under OS/2 than under other implementations of X-Icon, and there are a few features that OS/2 does not support. See Appendix C of [3] for more information on this.

7. Reporting Problems

Problems with Icon should be noted on a trouble report form (included with the distribution) and sent to

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

(602) 621-8448 (voice)
(602) 621-4246 (fax)

icon-project@cs.arizona.edu (Internet)
… uunet!arizona!icon-project (uucp)

8. Registering Copies of Icon

If you received your copy of Icon directly from the Icon Project, it has been registered in your name and you will automatically receive the Icon Newsletter. This newsletter contains information about new implementations, updates, programming techniques, and information of general interest about Icon.

If you received your copy of Icon from another source, please fill out the registration form that is included in the distribution and send it to the Icon Project at the address listed above. This will assure that you receive the Icon Newsletter and information about updates.

Acknowledgements

The design and implementation of the Icon programming language was supported, in part, by grants from the National Science Foundation.

Clint Jeffery and Gregg Townsend collaborated with the author in the development of Version 8.10 of Icon. The OS/2 implementation was done by Darren Merrill.

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, 1990.
2. R. E. Griswold, C. L. Jeffery and G. M. Townsend, *Version 8.10 of the Icon Programming Language*, The Univ. of Arizona Icon Project Document IPD212, 1993.
3. C. L. Jeffery and G. M. Townsend, *X-Icon: An Icon Windows Interface; Version 8.10*, The Univ. of Arizona Tech. Rep. 93-9, 1993.