Version 8 of Icon for MS-DOS/386

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Overview

This implementation of Icon runs on MS-DOS 386 PCs in ''true'' 32-bit protected mode. It was built using the Intel 386/486 C Code Builder Kit and the Intel DOS extender.

It uses memory *above* the 1MB of conventional memory. It runs comfortably on a 2MB 386 PC. It is doubtful if it will run on a 1MB 386 PC. It uses a real 287 or 387 if it is present; otherwise it simulates the device.

This implementation uses the 386 small memory model, which supports segments up to 4GB. There are no 64KB memory limitations. It uses Icon's expandable-regions form of memory management, in which the sizes of the block and string regions are adjusted automatically as needed.

This implementation supports co-expressions, keyboard functions, the system() function, and large-integer arithmetic. The –x option to obtain automatic execution following linking is not yet supported. It also does not support Icon's extended function repertoire for MS-DOS.

The basic reference for the Icon programming language is the book

> *The Icon Programming Language*, second edition, Ralph E. Griswold and Madge T. Griswold, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990. 367 pages. ISBN 0-13-447889-4.

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.

Note that the first edition of this book, published in 1983, describes an older version of Icon and does not contain information about many of the features of Version 8.

A brief overview of Icon is contained in technical report TR 90-6 [1] (tr90-6.doc on the distribution diskette). Features that have been added to Icon since the book was written are described in TR 90-1 [2] (tr90-1.doc on the distribution diskette). These technical reports, together with this document (ipd135.doc on this diskette), provide enough information to write and run simple Icon programs, but persons who intend to use Icon extensively will need the book.

2. Installing MS-DOS/386 Icon

The distribution diskettes contain the two executable files needed to run Icon:

icont.exe	Icon translator and linker
iconx.exe	Icon executor

These files should be copied to a place on your PATH specification.

3. Running MS-DOS/386 Icon — Basic Information

Files containing Icon programs must have the extension .icn. Such files should be plain text files (without line numbers or other extraneous information). The command processor icont produces an ''icode'' file that can be executed by iconx. For example, an Icon program in the file prog.icn is translated and linked by

```
icont prog.icn
```

The result is an icode file with the name prog.icx. This file can be run by

```
iconx prog.icx
```

The extensions .icn and .icx are optional. For example, it is sufficient to use

```
icont prog
```

and

```
iconx prog
```

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
iconx 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
iconx 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
iconx meander <meander.dat
```

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

```
icont roman
iconx 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 4 may be adequate. The icont command processor 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,

```
iconx 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 Translating and Linking

icont 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 and links 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 the Icon linker 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, only translation is performed and intermediate ''ucode'' files with the extensions .u1 and .u2 are kept. For example,

```
icont –c prog1
```

leaves prog1.u1 and prog1.u2, instead of linking them to produce prog1.icx. (The ucode files are deleted unless the –c option is used.) These ucode files can be used in a subsequent icont command by using the .u1 name. This avoids having to translate the .icn file again. For example,

```
icont prog2 prog1.u1
```

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

```
icont prog2 prog1.u
```

Ucode files also can be added to a program when it is linked by using the link declaration in an Icon source program as described in [2].

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 and linker 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 program 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. The warnings are issued during the linking phase.

Icon has several tables related to the translation and linking of programs. These tables are large enough for most programs, but translation or linking is terminated with an error message, indicating the offending table, if there is not enough room. If this happens, larger table sizes can be specified by using the –S option. This option has the form –S[cfgilnrstCFL]*n*, where the letter following the S specifies the table and *n* is the number of storage units to allocate for the table.

c	constant table	100
f	field table	100
g	global symbol table	200
i	identifier table	500
l	local symbol table	100
n	line number table	1000

r	record table	100
s	string space	20000
t	tree space	15000
C	code buffer	20000
F	file names	10
L	labels	500

The units depend on the table involved, but the default values can be used as guides for appropriate settings of –S options without knowing the units. For example,

```
icont –Sc200 –Sg600 prog
```

translates and links prog.icn with twice the constant table space and three times the global symbol table space that ordinarily would be provided.

5.3 Environment Variables

When an Icon program 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. This may be necessary if your computer does not have enough memory to run programs that require an unusually large amount of data. Particular care should be taken when changing default sizes: unreasonable values may cause Icon to malfunction.

The following environment variables can be set to affect 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 (256000). This variable determines the size, in bytes, of the region in which strings are stored.

HEAPSIZE (512000). This variable determines the size, in bytes, of the region in which Icon allocates lists, tables, and other objects.

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 MS-DOS/386 Icon

MS-DOS/386 Icon supports all the features of Version 8 of Icon, with the following exceptions and additions:

- Pipes are not supported. A file cannot be opened with the "p" option.
- There are two additional options for open: "t" and "u". The "t" option, which is the default, indicates that all carriage-return/line-feed sequences are translated into line-feed characters on both input and output. The "u" option indicates that the file is to be untranslated. Examples are:

```
untranfile := open("test.fil","ru")
tranfile := open("test.new","wt")
```

  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 MS-DOS 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.

- The option –x to icont to obtain automatic execution after linking is not supported.

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

icon-project@cs.arizona.edu (Internet)
… {uunet, allegra, noao}!arizona!icon-project (uucp)

Acknowledgements

The design and development of the Icon programming language was supported, in part, by the National Science Foundation under grants MCS75-01397, MCS79-03890, MCS81-01916, DCR-8320138, DCR-8401831, and DCR-8502015.

Many individuals contributed to the design and implementation of Icon. The principal ones are Cary Coutant, Dave Gudeman, Dave Hanson, Tim Korb, Bill Mitchell, Kelvin Nilsen, Janalee O'Bagy, Gregg Townsend, Ken Walker, and Steve Wampler.

Bob Goldberg implemented Version 7.5 of Icon for MS-DOS/386. Mark Emmer adapted it to Version 8 and supplied the executable files.

References

1. R. E. Griswold, *An Overview of Version 8 of the Icon Programming Language*, The Univ. of Arizona Tech. Rep. 90-6, 1990.
2. R. E. Griswold, *Version 8 of Icon*, The Univ. of Arizona Tech. Rep. 90-1, 1990.