Version 8.10 of Icon for MS-DOS 386/486 Platforms

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Overview

This implementation of Icon runs on MS-DOS 386/486 PCs in 32-bit protected mode. It was built using the Intel 386/486 C Code Builder Kit and the Intel DOS extender. See config.doc on the distribution diskette for current information about system compatibility.

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 math co-processor if one is present; otherwise it uses software emulation.

This implementation uses the 386/486 small memory model, which supports segments up to 4GB. There are no 64KB memory limitations.

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 the Icon programming language is a book [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. 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 [2]. Features that have been added to Icon since the book was written are described [3]. These technical reports, together with this document 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 Icon

Two executable binary files are needed to run Icon:

icont.exe	translator
iconx.exe	executor

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

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

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

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

```
a:lharc x a:icon.lzh
```

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

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 on the command line. For example, it is sufficient to use

```
icont prog
```

and

```
iconx prog
```

iconx will find an icode file if it is in the current working directory or at a place on your PATH specification.

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 3 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(args)
   in := open(args[1]) | stop("cannot open input file")
   out := open(args[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 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.

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.

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 initial region in which strings are stored.

HEAPSIZE (512000). This variable determines the size, in bytes, of the initial 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 8.10 of Icon, with the following exceptions and additions:

- Pipes are not supported. A file cannot be opened with the "p" option.
- The extended function repertoire for MS-DOS [4] is not supported.
- 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 (voice)
(602) 621-4246 (fax)

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

Acknowledgements

Many individuals contributed to the design and implementation of Icon. Clint Jeffery and Gregg Townsend collaborated with the author in the development of Version 8.10.

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, *An Overview of Version 8 of the Icon Programming Language*, The Univ. of Arizona Tech. Rep. 90-6, 1990.
3. 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.
4. R. E. Griswold, *Version 8.10 of Icon for MS-DOS*, The Univ. of Arizona Icon Project Document IPD221, 1993.