Version 9 of Icon for UNIX

Ralph E. Griswold and Gregg M. Townsend

Department of Computer Science, The University of Arizona

1. Introduction

Version 9 of Icon runs on a wide variety of UNIX systems.

The basic reference for Icon is the second edition of the book *The Icon Programming Language* [1]. This book, which describes Version 8, 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 9 of Icon are described in accompanying technical reports [2, 3].

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.

2. Running UNIX Icon

Basic Information

Files containing Icon programs must have the suffix .icn. The Icon translator, icont, produces an ‘‘icode’’ file that is executable. An Icon program in the file prog.icn is translated by

```
icont prog.icn
```

The .icn suffix is optional, so

```
icont prog
```

can be used instead.

The result is an icode file with the name prog (with no suffix). This file can be run by

```
prog
```

Alternatively, icont can be instructed to execute the icode file after translation by appending a –x to the command line, as in

```
icont prog –x
```

If icont is run with the –x option, the file icode prog is left and can be run subsequently as described above.

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.

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 prog
```

produces the icode file named probe rather than prog.

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. These ucode files can be used in a subsequent icont command by using the .u1 name. This saves translation time subsequently. 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.

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 effect of –t is to produce diagnostic messages for procedure calls, returns, and resumptions and for co-expression activation.

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

The Icon translator supports several other options. There also is an optimizing compiler for Icon. See the manual page.

Libraries

Libraries are directories containing ucode files built by icont –c. The standard Icon Program Library contains many useful procedures and is practically essential for constructing graphics programs of any complexity.

Libraries are made available to icont through the IPATH environment variable. This space-separated list is searched to satisfy link declarations.

For more information about libraries in general and the Icon Program Library in particular, see [4].

Program 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, passing its main procedure a list of two strings, "text.dat" and "log.dat". The program also can be translated and run with these arguments with a single command line by putting the arguments after the –x:

```
icont prog –x text.dat log.dat
```

These arguments might be the names of files that prog.icn reads. 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")
                        :
                        :
```

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 (500000): 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 (500000): 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.

QLSIZE (5000): This variable determines the size, in bytes, of the region used by the garbage collector for pointers to strings.

3. Other Facilities

Icon for UNIX supports a number of other facilities.

Variant translators provide a way to use Icon's translator to build preprocessors that change the form of Icon programs. See [5].

On some platforms, C functions can be loaded dynamically to extend Icon's computational repertoire. See [6].

4. Contacting the Icon Project

Icon Project
Department of Computer Science
The University of Arizona
P.O. Box 210077
Tucson, AZ 85721-0077
U.S.A.

(520) 621-6613 (voice)
(520) 621-4246 (fax)

icon-project@cs.arizona.edu

Acknowledgements

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

Clint Jeffery collaborated with the authors in the development of Version 9.

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 9.1 of the Icon Programming Language*, The Univ. of Arizona Icon Project Document IPD267, 1995.
3. G. M. Townsend, R. E. Griswold and C. L. Jeffery, *Graphics Facilities for the Icon Programming Language; Version 9.1*, The Univ. of Arizona Icon Project Document IPD268, 1995.
4. R. E. Griswold and G. M. Townsend, *The Icon Program Library; Version 9.1*, The Univ. of Arizona Icon Project Document IPD269, 1995.

5. R. E. Griswold, *Variant Translators for Version 9 of Icon*, The Univ. of Arizona Icon Project Document IPD245, 1994.

6. R. E. Griswold and G. M. Townsend, *Calling C Functions from Version 9 of Icon*, The Univ. of Arizona Icon Project Document IPD240, 1995.