Version 8 of Icon for the Atari ST

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Introduction

Icon for the Atari ST is designed to run on both the 520 and 1040 models, but the limited amount of memory on the 520 makes its use there problematical for Version 8, which is larger than previous versions. Icon for the the Atari ST is designed to run under a command-line processor, such as ASH. This document assumes the use of ASH, although other command-line processors can be used.

Version 8 of Icon for the Atari ST is distributed on a single-sided diskette, which includes executable binary files, ASH, a few test programs, and documentation in machine-readable form. Printed documentation is included with diskettes distributed by the Icon Project at the University of Arizona.

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. Documentation

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. 365 pages. ISBN 0-13-447889-4. $29.95.

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 or by telephone directly from Prentice-Hall: (201) 767-9520.

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 (ipd136.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.

3. Installing Icon on the Atari ST

Two executable binary files are needed to run Icon:

icont.prg	translator and linker
iconx.prg	executor

These files should be located at a place on the PATH specification for your command-line processor. Because of the way path searching is done, it is advisable to place these files in the root directory of one of your devices.

Most of the distribution files are packaged in ARC format. A copy of ARC is included on the distribution diskette. The distribution files are:

arc.prg	archiving utility
icon.arc	Icon executable binaries [214KB]
samples.arc	Icon programs and data [2KB]
docs.arc	documents [139KB]
readme	installation overview and recent notes
ash.prg	ASH
ash.hlp	ASH help file
ash.ini	ASH initialization file

The figures in brackets give the approximate amount of disk space needed when the files are extracted from their archives.

First copy arc.prg to a place on your path. To install the .prg files, set your current directory to the desired place and dearchive the files using arc on the distribution diskette*. For example, if the distribution diskette is in drive a:, the following will do:

```
arc x a:icon.arc
```

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

4. Running Icon on the Atari ST — 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 runs icont and ilink to produce 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
```

If input or output is redirected, i/o redirection must appear at the beginning of the arguments for iconx, as in

```
iconx <prog.dat prog
```

5. 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
```

*If you are not familiar with the capabilities of arc, you can get a brief summary by

```
arc h
```

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.dat cross`

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.dat meander`

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.

6. 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.

6.1 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")
                         :
```

6.2 The Command Processor

The command processor 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.

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].

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.

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 prog.icn
        ilink –o probe.icx prog.u1
```

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

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.

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

The option –u to icont 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
l	line number table	1000
r	record table	100
s	string space	20000
t	tree space	15000
C	code buffer	15000
F	file names	10
L	labels	500

The options must be specified both for icont and ilink and must have the same values for both.

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.icn
```

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

6.3 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 (65000). This variable determines the size, in bytes, of the region in which strings are stored.

HEAPSIZE (65000). This variable determines the size, in bytes, of the region in which Icon allocates lists, tables, and other objects.

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.

The maximum region size is 65000. Specifying a larger size may cause program malfunction or unexpected results.

7. Memory Utilization

Icon requires a significant amount of memory. It may not run or it may run slowly if enough memory is not available.

The executor, iconx, reserves 200KB of RAM for its use. If that amount is not available, it aborts with a setblock failure. Note that using a RAM disk may be a problem in this regard.

For some programs, it may be necessary to reduce the default region sizes. For example, under ASH

```
set STRSIZE=40000
```

sets the size of the string region to 40KB.

8. Features of Icon for the Atari ST

Icon for the Atari ST supports all the features of Version 8 of Icon, with the following exceptions and additions:

- The -x option to icont for automatic execution is not supported.
- Large-integer arithmetic is not supported.
- 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 the file is to be translated into UNIX* format. 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.

*UNIX is a trademark of AT&T Bell Laboratories.

- The following Atari ST device names can be used as file names:

console	CON
printer	PRN
auxiliary device	AUX
null	NUL

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. These special names are associated with the devices above even if device designations or filename extensions are added to them. For example, A:CON.XXX refers to the console and is not the name of a disk file.

9. Bugs

The known bugs in all implementations of Icon are listed in [2].

There are two known bugs in Icon for the Atari ST:

- The -e on the command line to iconx to redirect error output does not work properly.
- The system() function may not work properly.

10. 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) 621-4049

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

11. Registering Copies of Icon

If you received your copy of Version 8 of Icon directly from the Icon Project, it has been registered in your name and you will receive the Icon Newsletter without charge. This Newsletter contains information about new implementations, updates, programming techniques, and information of general interest about Icon.

If you received your copy of Version 8 of Icon from another source, please fill out the registration form that is included in the documents in the distribution) and send it to the Icon Project at the address listed above. This will entitle you to a free subscription to the Icon Newsletter and assure that you receive information about updates.

Acknowledgements

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

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.

Owen Fonorow and Jerry Nowlin did the original implementation of Icon for the Atari ST. Charles Richmond made significant contributions to subsequent versions.

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.