The Icon Program Library*

Ralph E. Griswold

TR 90-7b

January 1, 1990; last revised March 8, 1990

Department of Computer Science

The University of Arizona

Tucson, Arizona 85721

*This work was supported by the National Science Foundation under Grant CCR-8713690.

The Icon Program Library

1. Introduction

The Icon program library consists of Icon programs and procedures as well as data. Icon Version 8 is required to run most of the library [1, 2].

In addition to the Icon program library proper, the library distribution contains an object-oriented version of Icon written in Icon. See [3] for instructions for unloading and using this program.

Section 6 briefly describes the contents of the library. More compete documentation is contained in comments in the program and procedure files. You may wish to print these files to have documentation handy.

The material in the Icon program library was contributed by Icon users. It is in the public domain and may be copied freely. The Icon Project packages and distributes the library as a service to Icon programmers. The Icon project makes no warranties of any kind as to the correctness of the material in the library or its suitability for any application. The responsibility for the use of the library lies entirely with the user.

2. Unloading the Library

The Icon program library consists of three parts: programs, collections of procedures, and data. Normally, these components should be placed in separate directories named progs, procs, and data. The method of unloading the distribution files varies from system to system; system-specific instructions are included separately.

The physical division of the library into progs, procs, and data is motivated by logical and organizational considerations, not operational ones. Other names can be used and all the material can be placed in one directory, for example. This may be necessary on some systems.

3. Link Search Paths

Many of the programs link procedures. For example, options is used by many programs for processing command-line options and is linked from ‘‘ucode’’ files obtained from translating options.icn.

Icon searches for ucode files first in the current directory and then in directories specified by the IPATH environment variable. IPATH consists of a sequence of blank-separated path names. The search is in the order of the names. For example, on a UNIX system running csh,

```
setenv IPATH "../procs /usr/icon/ilib"
```

results in a search for file names in link declarations first in the current directory, then in ../procs, and finally in /usr/icon/ilib.

The method of setting IPATH varies from system to system. Since the current directory always is searched first, if ucode files are placed in the same directory as the program files, IPATH need not be set. See the next section.

4. Installing the Library

Installing the Icon program library consists of two steps: (1) translating the procedure files to produce ucode files and (2) compiling the programs.

Ucode files are produced by translating the procedure files with the −c option to icont, as in

```
icont −c options
```

which translates options.icn. The result is two ucode files named options.u1 and options.u2. The .u1 files contains the procedure's code and the .u2 file contains global information about the procedure. It is these files that a link declaration such as

```
link options
```

needs.

A script for translating all the procedure files is provided with the most distributions. Once the procedure files have been translated, the ucode files can be moved to any place that is accessible from IPATH.

The programs are compiled using icont without the –c option, as in

```
icont deal
```

which compiles deal.icn, a program that produces randomly selected bridge hands. The result of compiling a program is an ''icode'' file whose name is system dependent. On some systems, the name is the same as the name of the program file with the .icn suffix removed (for example, deal). On other systems, the icode file has the suffix .icx in place of .icn (for example, deal.icx).

On systems that support the direct execution of icode files (UNIX, for example), an icode file can be run just by entering its name on the command line, as in

```
deal
```

On other systems (MS-DOS, for example), icode files must be run using the Icon executor, iconx, as in

```
iconx deal
```

(This also works on systems that support direct execution.) Note that the suffix (if any) need not be mentioned.

Many Icon programs take arguments and options from the command line. Options are identified by dashes. For example, in

```
deal –h 10
```

the –h 10 instructs deal to produce 10 hands.

Icode files can be moved to any location. Ucode files are needed only during compilation. They need not be accessible when icode files are run.

5. Usage Notes

It is important to read the documentation at the beginning of programs and procedures in the library. It includes information about special requirements, limitations, known bugs, and so forth.

Some of the programs in the Icon program library are quite large and may require more memory than is available on some personal computers.

The library has evolved over a period of time. Some programs were written to run under earlier versions of Icon and do not take advantage of all the features of Version 8.

The procedure getopt, used to process command-line options in the previous version of the library, has been replaced by the procedure options, which is somewhat easier to use. If you presently use getopt in other programs, you may wish to convert to options.

6. Library Contents

As mentioned earlier, detailed documentation about programs and procedures is contained in their files. A brief catalog of the contents of the Icon program library follows.

6.1 Programs

animal:	Play the familiar ‘‘animal’’ game.
calc:	Calculate Icon values.
colm:	Arrange data items in columns.
concord:	Produce a concordance.
cross:	Arrange words in intersecting crossword fashion.
csgen:	Generate sentences from a context-sensitive grammar.
deal:	Display randomly generated bridge hands.
delam:	Delaminate file into several files according to field specifications.
delamc:	Delaminate file into several files according to tabs.
diffn:	Show differences among several files.
diffword:	List the distinct words in a file.
edscript:	Produce script for the ed editor.
empg:	Produce program to measure Icon expressions.
farb:	Produce a ‘‘Farberism’’.
fileprnt:	Display representations of characters in a file.
filter:	Filter file.
format:	Format text.
gcomp:	Produce the complement of a UNIX file specification.
grpsort:	Sort groups of lines.
hufftab:	Compute state transitions for Huffman decoding.
ilnkxref:	Produce link cross-reference of Icon program.
ipp:	Preprocess Icon programs.
iprint:	Print Icon program.
ipsort:	Sort procedures in Icon program.
ipsplit:	Split Icon program into separate procedure files.
ipxref:	Produce cross reference for Icon program.
itab:	Entab Icon program.
iundecl:	Find undeclared Icon identifiers.
iwriter:	Produce Icon expressions that write lines of file.
krieg:	Play game of kriegspiel.
kross:	Show all intersecting characters in two strings.
kwic:	Produce index of keywords in context.
labels:	Format mailing labels.
lam:	Laminate several files into one file.
latexidx:	Process LaTeX .idx file.
linden:	Generate strings in 0L-system.
lisp:	Interpret Lisp program.
loadmap:	Produce load map of UNIX object file.
miu:	Generate strings in MIU system.

memsum: Summarize memory usage of Icon program.

monkeys: Generate random text.

pack: Package a group of files in a single file (see unpack).

parens: Generate random parenthesis-balanced strings.

parse: Parse infix expressions (see also parsex).

parsex: Parse arithmetic expressions (see also parse).

press: Compress or uncompress file.

proto: Compile all Icon syntactic forms.

queens: Generate solutions to the n-queens problem (see also vnq).

recgen: Produce recognizer for context-free language.

roffcmds: List commands and macros in roff text.

rsg: Generate random sentences from grammar.

ruler: Write character ruler.

shuffile: Shuffle lines in a file.

solit: Play solitaire.

tablc: Tabulate characters in a file.

tablw: Tabulate words in a file.

textcnt: Tabulate properties of a text file.

trim: Trim lines in a file.

turing: Simulate a Turing machine.

unique: Filter out identical adjacent lines of a file.

unpack: Unpackage a group of files (see pack).

vnq: Display solutions to the n-queens problem interactively on an ANSI-standard terminal (see also queens).

zipsort: Sort labels by ZIP code.

6.2 Procedures

allof: Perform iterative conjunction.

bincvt: Convert binary data.

bold: Enbolden and underscore text.

codeobj: Encode and decode Icon values as strings (see also object).

collate: Collate and decollate strings.

colmize: Arrange data in columns.

complex: Perform complex arithmetic.

currency: Format in American currency.

dif: Generate differences.

escape: Interpret Icon literal escapes.

filename: Parse file name.

fullimag: Produce full image of Icon value (see also image and ximage).

gcd: Compute greatest common divisor.

gener:	Generate various strings.
hexcvt:	Convert hexadecimal numbers.
image:	Produce image of Icon value.
isort	Sort with customization.
largint:	Perform arbitrary-precision integer arithmetic.
lmap:	Map list elements.
mapbit:	Map string into its bit representation.
math:	Perform mathematical computations.
morse:	Convert string to Morse code.
ngrams:	Tabulate n-grams in a text file.
numbers:	Convert numbers to various forms.
object:	Encode and decode Icon values as strings (see also codeobj).
options:	Process command-line options.
patterns:	Perform SNOBOL4-style pattern matching.
patword:	Produce letter pattern for a word.
pdae:	Perform programmer-defined argument evaluation.
pdco:	Perform programmer-defined control operations.
permute:	Perform permutations, combinations, and other character rearrangements.
phoname:	Generate possible words from telephone numbers.
printcol:	Print columnar data.
printf:	Format in C printf style.
radcon:	Convert radix.
rational:	Perform rational arithmetic.
segment:	Segment string.
seqimage:	Produce string image of Icon result sequence.
shquote:	Quote words for shells.
shuffle:	Shuffle string or list.
snapshot:	Show state of Icon string scanning.
strings:	Perform operations on strings.
structs:	Perform operations on structures.
tuple:	Simulate n-tuples.
usage:	Provide interface operations.
wildcard:	Match UNIX wild-card patterns.
wrap:	Wrap text lines.
ximage:	Produce image of Icon value (see also fullimag and image).

6.3 Data

∗.csg:	Input to csgen.
∗.krs:	Input to kross.
∗.lbl:	Input to label.

∗.lin:	Input to linden.
∗.rsg:	Input to rsg.
∗.tur:	Input to turing.
∗.txt:	Sample text.
∗.wrd:	Word lists.
farber.sen:	Farberisms.
palin.sen:	Palindromic sentences.

7. Future Library Releases

There are many contributions to the Icon program library that have not yet been distributed. This material includes:

- Programs that are operating-system specific.
- Complex packages.
- Programs that require specific data files.
- Programs that need more documentation.
- Recent arrivals.

The Icon program library will be updated as this material is put into a form suitable for distribution.

8. Note to Contributors

Material for the Icon program library always is welcome. It must be prepared in the style exemplified by the material in this release. Adequate documentation is essential; it must be in the format used for present library — we do not have the resources to rewrite or reformat contributed documentation. Test data also must be provided — at least enough so that we can determine that the contributed program material is basically functional. In cases where test data is impractical because of the nature of the contribution, instructions for testing should be provided.

Program material can be submitted by electronic mail at one of the addresses given in the next section or on magnetic media. Printed listings are not acceptable.

Contributions to the Icon program library must be free of any restrictions. The decision to include contributed material in the Icon program library rests entirely with the Icon Project. The Icon Project reserves the right to modify submissions to conform to library standards, to correct errors, and to make improvements. Contributors wil be consulted in the case of substantial changes.

9. Bugs

If you find a bug in the Icon program library or can suggest an improvement, please let us know:

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)

Acknowledgements

The following persons contributed material to this release of the Icon program library:

Paul Abrahams	Anthony Hewitt	Gregg M. Townsend
Robert J. Alexander	Thomas R. Hicks	Kenneth Walker
Allan J. Anderson	Tim Korb	Stephen B. Wampler
David S. Cargo	William P. Malloy	Kurt A. Welgehausen
nary A. Coutant	William H. Mitchell	Robert C. Wieland
Ward Cunningham	Jerry Nowlin	Cheyenne Wills
Michael Glass	Randal L. Schwartz	George D. Yee
Ralph E. Griswold	David Slate	David Yost

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.
2. R. E. Griswold, *Version 8 of Icon*, The Univ. of Arizona Tech. Rep. 90-1, 1990.
3. C. L. Jeffery, *Programming in Idol — An Object Primer*, The Univ. of Arizona Tech. Rep. 90-10, 1990.