The Icon Program Library; Version 8.1

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Introduction

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

In addition to the Icon program library proper, the library distribution contains Idol, an object-oriented version of Icon written in Icon. See [2] for instructions using this program.

Section 6 lists 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: complete programs, collections of procedures, and data. Normally, these components should be placed in separate directories named progs, procs, and data.

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.

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 file 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.

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 Complete Programs

adlcheck	Check address list data
adlcount	Count address list entries
adlfiltr	Filter address list entries
adllist	List address list entries

adlsort	Sort address list entries
animal	Play the familiar ‘‘animal’’ game
bj	Play blackjack
calc	Calculate Icon values
colm	Arrange data items in columns
concord	Produce a concordance
countlst	Count items in a list
cross	Arrange words in intersecting crossword fashion
csgen	Generate sentences from a context-sensitive grammar
cstrings	Print strings in C programs
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
detex	Strip LaTeX commands from text files
diffn	Show differences among several files
diffu	Show differences among several files
diffword	List the distinct words in a file
duplproc	Find duplicate procedures
edscript	Produce script for the ed editor
empg	Produce program to measure Icon expressions
empg	Produce expression measurement program
farb	Produce a ‘‘Farberism’’
farb2	Produce a ‘‘Farberism’’ using seek into data base
filecnvt	Convert line terminators on text files
fileprnt	Display representations of characters in a file
filter	Filter file
findstr	Find character strings embedded in a file
fixpath	Change strings in a binary file
format	Format text
fset	Perform set operations on file specifications
gcomp	Produce the complement of a UNIX file specification
genqueen	Generate solutions to the n-queens problem
grpsort	Sort groups of lines
hcal4unx	Give Jewish/civil calendar dates (UNIX version)
hebcalen	Give Jewish/civil calendar dates (MS-DOS version)
hufftab	Compute state transitions for Huffman decoding
ibrow	Browse in Icon program library
icalc	Perform computations in infix form
icontent	List procedures and records in an Icon program
icvt	Convert between ASCII and EBCDIC forms in Icon programs

idxtext	Index text base
ihelp	Get on-line help for Icon
iidecode	Decode files in UNIX uuencode format
iiencode	Encode files in UNIX uuencode format
ilnkxref	Produce link cross-reference of Icon program
ilump	Lump linked Icon source files
interpe	Interpret Icon expressions
interpp	Interpret Icon programs
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
itags	Create tag file for Icon program
iundecl	Find undeclared Icon identifiers
iversion	Show icode version
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
memsum	Summarize memory usage of Icon program
miu	Generate strings in the MIU system
monkeys	Generate random text
mtf3	Map tar file
nocr	Remove carriage-returns
pack	Package a group of files in a single file (see unpack)
parens	Generate random parenthesis-balanced strings
pargen	Produce parser
parse	Parse infix expressions (see also parsex)
parsex	Parse arithmetic expressions (see also parse)
patchu	Patch program in UNIX patch style
post	Post news
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
reply	Reply to news or mail
repro	Reproduce program
roffcmds	List commands and macros in roff text
rsg	Generate random sentences from grammar
ruler	Write character ruler
shuffile	Shuffle lines in a file
sing	Sing ‘‘The Twelves Days of Christmas’’
snake	Play the snake game
solit	Play solitaire
stars	Display field of stars
strpsgml	Strip SGML tags from file
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 (see also queens)
xtable	Print character translation tables
yescr	Add carriage returns
zipsort	Sort labels by ZIP code

6.2 Procedures

adjuncts	Utilities for gettext.icn and idxtext.icn
adlutils	Utilities for processing address lists
allof	Perform iterative conjunction
ansi	Control ANSI terminal
asciinam	Produce ASCII of unprintable character
bincvt	Convert binary data
bold	Enbolden and underscore text
buffer	Buffered I/O
codeobj	Encode and decode Icon values as strings
collate	Collate and decollate strings
colmize	Arrange data in columns
commaize	Insert commas in numbers
complete	Complete partial string

complex	Perform complex arithmetic
currency	Format in American currency
dif	Generate differences
dosfiles	Get MS-DOS file names
ebcdic	Translate between EBCDIC and ASCII character sets
escape	Interpret Icon literal escapes
fcopy	Copy file
feval	Evaluation string for function call
filename	Parse file name
findre	Find regular expression
fullimag	Produce full image of Icon value (see also image and ximage)
gcd	Compute greatest common divisor
gdl	Get directory list
gener	Generate various strings
getchlib	Provide keyboard support for UNIX
getkeys	Get keys for gettext.icn file
getpaths	Generate paths
gettext	Utilities for text-base files
gmean	Compute geometric mean
hexcvt	Convert hexadecimal numbers
hostname	Get hostname
ibench	Utilities for benchmarking Icon programs
identity	Produce identities for Icon types
ifncs	Procedure wrappers for function tracing
iftrace	Function tracing
image	Produce image of Icon value
inbits	Read variable-length characters
inserts	Build tables with duplicate keys
instring	Create string from raw bits of an integer
iolib	Provide generalized screen support for MS-DOS and UNIX
ipause	Pause
irandom	Set random-number generator seed
iscreen	Provide screen support for UNIX
isort	Sort with customization
ispf	Communicate between Icon and ISPF
itlib	Provide Icon-based term-lib screen output for UNIX
itlibdos	Termlib utilities
ivalue	Convert string to Icon value
largint	Perform arbitrary-precision integer arithmetic
lastname	Get last name

lmap	Map list elements
longstr	Match longest string
lscan	Perform list scanning
mapbit	Map string into its bit representation
matchlib	Matching procedures
math	Perform mathematical computations
morse	Convert string to Morse code
namepfx	Get name prefix
ngrams	Tabulate n-grams in a text file
numbers	Format and convert numbers
object	Encode and decode Icon values as strings
options	Process command-line options
outbits	Output variable-length characters
packunpk	Pack and unpack packed-decimal strings
parscond	Condense parse tree
patch	Provide UNIX-like patch
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
plural	Produce plural form of singular noun
printcol	Print columnar data
printf	Format in C printf style
radcon	Convert radix
rational	Perform rational arithmetic
readtbl	Read stripsgml table
rec2tab	Convert record fields to tab-separated string
recog	Main procedure for recognizers
rewrap	Wrap lines
segment	Segment string
sentence	Find sentences in file
seqimage	Produce string image of Icon result sequence
shquote	Quote words for shells
shuffle	Shuffle string or list
signed	Put bits of characters into a signed integer
slashbal	Match balanced string with escapes
snapshot	Show state of Icon string scanning
statemap	Produce two-way table of states and their abbreviations

strings	Perform operations on strings
strip	Strip characters from a string
stripcom	Strip comments from a line of Icon code
stripunb	Strip unbalanced material
structs	Perform operations on structures
tab2rec	Convert tab-separated string to record
tblset	Perform set-theoretic table manipulations
tclass	Classify Icon values
tempname	Produce a temporary file name
title	Get title from name
titleset	Produce a set of possible titles
tuple	Simulate n-tuples
typecode	Produce type code for Icon value
unsigned	Put bits of characters into an unsigned integer
usage	Service utilities
version	Produce Icon version number
wildcard	Match UNIX wild-card patterns
wrap	Wrap text lines
ximage	Produce image of Icon value

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
address.doc	Documentation for address lists
dylan.txt	Text file
farber.sen	Farberisms
hebcalen.dat	Data for Jewish/civil calendar programs
hebcalen.hlp	Help for Jewish/civil calendar programs
hebcalpi.hlp	ProIcon help for Jewish/civil calendar programs
icon.wrd	Words containing the substring ‘‘icon’’
ihelp.dat	Data file used by ihelp
ipp.doc	Documentation for ipp.icn
joyce1.txt	Text file
joyce2.txt	Text file

joyce3.txt	Text file
palin.sen	Palindromic sentences
termcap.dos	Termcap data for MS-DOS
termcap2.dos	Alternative termcap data for MS-DOS

7. Contributions to the Icon Program Library

New 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 and may not carry copyright notices, even if accompanied by permission for unlimited copying.

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 will be consulted in the case of substantial changes.

8. 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) 621-8448 (voice)
(602) 621-4246 (fax)

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

Acknowledgements

Dozens of persons have contributed material to this release of the Icon program library. See the program material itself for authorship information.

References

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