Installation and Maintenance Guide for Version 5.9 of Icon*

Ralph E. Griswold

William H. Mitchell

TR 84-13a

August 24, 1984; Revised October 12, 1984

Department of Computer Science

The University of Arizona

Tucson, Arizona 85721

*This work was supported by the National Science Foundation under Grants MCS81-01916 and DCR 8401831.

Installation and Maintenance Guide
for Version 5.9 of Icon

This document describes how to install Version 5.9 of the Icon programming language [1, 2]. The distribution of Version 5.9 contains source code for Icon itself, documentation, sample programs and tests, the Icon program library [3], the procedures from the Icon book [1], test suites to aid in porting Icon [4, 5], and various support material.

The installation procedure for Icon is simple; it requires unloading the distribution tape onto the target machine, the setting of site-specific constants, and the compilation of the Icon system itself. Read Sections 1 and 2 of this document and the overview of Version 5.9 [2] before beginning the installation process.

If Version 5.9 is being installed at a site that is running Version 5.8, read Section 3.2 before beginning the installation process.

1. System Requirements

This distribution of Version 5 Icon is targeted for VAX-11s and PDP-11s (with separate instruction and data spaces) running the UNIX* operating system. This distribution package has been tested under V7 and 4.2bsd, and no problems should be encountered when installing Icon under one of these versions of UNIX.

When the system is unloaded, it requires about 2,700 kilobytes of disk space. During compilation, about 3,700 kilobytes are required. To install subsidiary components and test the entire system, about 4,800 kilobytes are required. After the removal of unnecessary files, about 3,800 kilobytes are required. These figures vary slightly depending upon the logical organization of a particular file system.

Additional space can be saved by omitting the Icon program library, test programs, and so on. See Sections 6.1, 6.2, and Appendix A for more information about the directories on the distribution tape. On systems with severely limited disk space, source code also can be deleted after Icon is installed.

2. Installation Procedure

2.1 Unloading the Distribution Tape

The system is distributed as a *tar* archive on magnetic tape. The *tar* hierarchy is rooted at the directory v5g. If you already have a v5g directory from an earlier version of Icon, you may wish to move this old directory to a new location, such as v5g.old.

Mount the distribution tape and do a *cd* to the directory that is to hold the system hierarchy.

The precise *tar* command to unload the distribution tape depends on the local environment. On a 4.*n*bsd VAX with a 1600 bpi distribution tape, the following command should extract the contents of the tape:

```
tar x
```

On a V7 PDP-11 with a 1600 bpi distribution tape, use:

```
tar xfb /dev/rmt0 20
```

For example, if the system is to reside at /usr/src/local/icon/v5g on a VAX, type:

*UNIX is a trademark of AT&T Bell Laboratories

```
cd /usr/src/local
mkdir icon
cd icon
tar x
```

Note: File names used in the following sections usually are relative to the root directory for the Icon hierarchy. For example, if the Icon system is unloaded as described above, the root directory is

```
/usr/src/local/icon/v5g
```

and the file name v5g/bin/Makefile refers to

```
/usr/src/local/icon/v5g/bin/Makefile
```

2.2 Configuring the Icon System

The installer must perform a site-specific configuration of the Icon system. This configuration is done by the shell script Icon-setup. Icon-setup accepts a number of parameters and modifies several source files to produce a ready-to-compile Icon system tailored as specified by the parameters.

Before Icon-setup modifies a file, it copies a generic version of the file into place and works on it. Thus, Icon-setup can be run a number of times and only the last run has any lasting effect. Icon-setup is like any other UNIX command and thus all of its arguments must be specified on one logical command line. Icon-setup has the following synopsis:

```
Icon-setup
   {-vax,-pdp11,-port}
   -host string
   [-sets]
   [-xpx]
   [-hz rate]
   [-nofp]
   [-interpex]
   [-vfork]
   [-usg]
   [-ibin library directory for Icon]
   [-iconx location of the Icon interpreter]
   [-debug]
```

The parameters have the following meanings:

-vax, -pdp11, or -port
: These are mutually exclusive options that control the selection of machine-dependent portions of code. Select one as appropriate. -port is used to set up Icon for transporting to a computer for which it previously has not been implemented.

-host *string*
: The Icon system has a keyword, &host, whose value should be the name of the host machine where the system is running. On some systems, notably 4.2bsd, System III, and System V, it is possible to determine the name of the system at run-time via a system call. On other systems, 4.1bsd for example, the file /usr/include/whoami.h contains the name of the host in a #define statement. On 4.2bsd, specify gethost for *string*. This causes the *gethostname(2)* function to be used. On System III, System V, or some other system that supports the *uname*(2) system call, specify uname for *string*. On a system with a /usr/include/whoami.h file that has a #define for sysname, then specify whoami for *string*. If none of these are available on your machine, or to give &host some value besides that of the machine name, specify an arbitrary string (quotes around it may be needed) for *string*, for example: -host UNIX.

-sets
: This parameter incorporates code to support the set data type, an extension to standard Version 5 that is described in [6]. This extension adds about 3 kilobytes to the size of the run-time system.

-xpx
This parameter incorporates a number of other extensions to standard Version 5 as described in [6]. These extensions add about 2 kilobytes to the size of the Icon run-time system.

-hz *rate*
This parameter specifies the cycle rate of the electrical environment. The rate defaults to 60 Hz. -hz does not need to be specified in a 60-Hz electrical environment. If the rate is incorrect, the value of &time will be wrong.

-nofp
Specify this on a PDP-11 that does not have floating-point hardware.

-interpex
Specify this option on a 4.*n*bsd system. This option causes the use of a feature of the *exec*(2) system call to make interpretable files directly executable. Do *not* specify this option on other systems.

-vfork
Specify this option if the operating system supports the *vfork(2)* system call; this should be specified for 4.*n*bsd systems.

-usg
Specify this option to adapt for minor incompatibilities between V7 UNIX systems and UNIX Software Group UNIX systems. In particular, specify this for System III or System V UNIX.

-ibin *directory*
The directory bin contains executable versions of the Icon translator (itran), the Icon linker (ilink), and the run-time system (iconx). The path name of this directory is built into icont, the program that controls the translation of Icon programs. By default, the fully qualified name of bin is used for -ibin. Specifying -ibin causes icont to use the specified directory as its library directory. If an alternate directory is specified, after the Icon system is built, copy v5g/bin/itran, v5g/bin/ilink, v5g/bin/iconx, and v5g/bin/iconx.hdr into the specified directory.

-iconx *interpreter-location*
By default, the *interpreter-location* is the fully qualified name of v5g/bin/iconx. If -ibin is specified, then *interpreter-location* defaults to *ibin-directory*/iconx. See Section 3.2 for considerations in chosing the location of the interpreter. If the fully qualified name of iconx is longer than 29 characters and -interpex is specified, Icon-setup considers the name to be erroneous. If this happens, consult Section 3.1. If an alternate location is specified for iconx, after the Icon system is built, copy v5g/bin/iconx to the specified location.

-debug
This option enables some debugging code; it normally is not used.

The only required options are -host and one of -vax, -pdp11, or -port. For example, on a VAX running 4.2bsd, use

```
Icon-setup -vax -interpex -host gethost -vfork
```

On a PDP-11 without floating-point hardware, running Version 7 UNIX, and on which &host is to be UNIX Version 7, use

```
Icon-setup -pdp11 -nofp -host "Unix Version 7"
```

On a VAX running System V in a 50-Hz electrical environment, on which the library directory for Icon is to be /usr/lib/icon, the Icon interpreter is to reside at /usr/bin/iconx, and all language extensions are to be included, use

```
Icon-setup -vax -hz 50 -sets -xpx -host uname -usg -ibin /usr/lib/icon -iconx /usr/bin/iconx
```

2.3 Compiling Icon

The Icon distribution tape contains no executable binary files and no object files, so the system must be completely recompiled from the source. After running Icon-setup, compile the system by doing a *make*. The complete system construction process is:

```
cd v5g
Icon-setup appropriate arguments
make Icon
```

2.4 Installing Icon for Public Use

A copy of v5g/bin/icont may be placed in a public directory such as /usr/bin or /usr/local to provide general user access to Icon. If -ibin and/or -iconx are specified in Icon-setup, copy the appropriate files to the specified location. For example, if the last Icon-setup specification in Section 2.2 were used, the following would be done:

```
cd v5g/bin
cp itran ilink iconx iconx.hdr /usr/lib/icon
cp iconx /usr/bin/icon
```

The manual pages icont.1 and icon-pi.1 in v5g/docs may be copied into /usr/man/man1.

2.5 Testing Icon

At this point, Icon system should be working. To test the system,

```
cd v5g
make Stdtest
```

which runs a number of standard test programs in the directory v5g/samples and compares them to results from a VAX-11. While these tests are not exhaustive, any major installation problems should show up when they are run. The output of the sample program hello.icn is certain to be different, since it contains time and site-specific code.

The sets and experimental extensions can be tested by

```
cd v5g
make Settest
```

and

```
cd v5g
make Xpxtest
```

respectively. Alternatively, all the tests may be run by

```
cd v5g
make Alltest
```

There is also a battery of tests in the directory v5g/test. These can be run, if desired, by

```
cd v5g/test
make Std
```

There are also entries in the Makefile in the test directory for Set, Xpx, and All. See [5] for information on interpreting the results of these tests.

There are other tests in v5g/port. See [5] for details.

3. Special Installation Considerations

3.1 Direct Execution of Interpretable Files

When an Icon program is processed by the translator and linker using the *icont* command, the result is a file containing opcodes and data in a format that the Icon interpreter understands. Rather than having the user "execute" this interpretable file by running the Icon interpreter with the file as an argument, the Icon system uses one of two methods to make the interpretable files appear to be directly executable.

In 4.1bsd and 4.2bsd systems, a feature of *exec*(2) system call can be used to enable the interpretable file produced by the linker to appear to be directly executable. When *exec* is called with a file to execute, it examines the first two characters of the file. If the first two characters are #!, *exec* assumes that the next argument on the line is the name of a program for which the file is to serve as input. The program then is executed with the named file (the file that is being "executed") as its argument. Thus, instead of having to run the Icon interpreter with the interpretable file as input, the interpretable file can be executed directly.

An alternative method is used on systems whose *exec*(2) system call doesn't have this feature. An executable file is prepended to the data used by the interpreter. The executable portion of the file merely runs the Icon interpreter with the file itself and any supplied arguments as the arguments for the interpreter.

If -interpex is specified for Icon-setup, the former method is used, otherwise, the latter method is used. The first method is preferable in that the interpretable files are smaller and they start executing more quickly.

There is a potential complication in using the first method. The 4.1bsd and 4.2bsd *exec*(2) system calls impose a length limitation of 29 characters on the name of the program to be run. If the name exceeds 29 characters, execution of the interpretable file fails. For example, suppose the Icon interpreter (iconx) on a system is located at /usr/csc/local/icon/v5g/bin/iconx. This path name is longer than 29 characters, and is thus unsuitable for inclusion in interpretable files.

The length of the path to iconx is checked by Icon-setup and the path above would be rejected.

One way to solve the problem is to link /usr/csc/local/icon/v5g/bin/iconx to /usr/local/iconx, and have interpretable files reference /usr/local/iconx. Two things need to be done to accomplish this. First, find a location where a copy of bin/iconx can be referenced with a fully qualified path name that is no more than 29 characters long. Second, when configuring the system using Icon-setup, specify the new location of iconx using the -iconx option. For example:

```
Icon-setup other arguments -iconx /usr/local/iconx
```

Be sure to place a copy of iconx in the specified location after the system is completely built. It is also possible to get around this problem by not specifying -interpex and having Icon prepend the executable header on interpretable files.

3.2 The Effect of Version Changes on Interpretable Files

An interpretable file produced by icont contains a path to the interpreter, iconx. Thus iconx cannot be moved without invalidating existing interpretable files. Furthermore, the interpreter for Version 5.9 is incompatible with those for previous versions. If, for example, Version 5.8 has been in use at a site and Version 5.9 is installed with the new iconx at the same location as that for Version 5.8, all previous interpretable files will be invalidated.

To avoid this, it may be desirable to retain iconx for Version 5.8 and put iconx for Version 5.9 at a new location, such as /usr/lib/icon/5.9/iconx. See the -iconx option in Section 2.2.

4. Installing the Icon Program Library

The Icon program library is rooted in v5g/src. After Icon has been installed, the library can be installed by

```
cd v5g
make Library
```

See [1] for a description of program library hierarchy.

There are test programs for the library, which can be run by

```
cd v5g
make Libtest
```

These tests should be unnecessary if previous testing has been successful, but they are included in case unexpected problems arise. They also may be useful in resolving questions about library programs. Some tests of the library require the optional extensions that are available in Version 5.9 and will malfunction if these extensions are not installed. The test of *farb(6)* probably will produce output that is different from the "standard" output, since it is dependent on the time of day.

5. Personalized Interpreters

Version 5.9 contains a facility for building personalized interpreters for Icon that allow individuals to augment or modify the Icon run-time system easily and quickly. See [7] for details. To make this facility available for general user access, a copy of v5g/icon-pi may be placed in a public area.

A sample personalized interpreter may be built by

```
cd v5g
make Pidemo
```

The result is a personalized interpreter in v5g/pidemo. This personalized interpreter contains the C functions from the Icon program library. It may be tested by

```
cd v5g
make Pitest
```

This test compares the results produced locally by this personalized interpreter to standard results. Differences should be expected. In addition, some of the functions only are supported on VAXs. Tests of these will terminate prematurely on a PDP-11.

6. Maintenance Information

The following sections contain information that is not needed to install Icon, but which may be helpful in understanding the organization of the system and in maintaining it.

6.1 System Layout

The Icon system has several top-level directories branching off from v5g. Only the directories marked with asterisks are needed to build Icon:

bin*	Executable versions of various system components. Source code for programs that control Icon translation also resides here.
h*	Header files that are included in other source files.
tran*	Source code for the Icon translator.
link*	Source code for the Icon linker.
iconx*	Source code for the Icon interpreter.
functions*	Source code for the Icon built-in functions.
operators*	Source code for the Icon operators.
lib*	Source code for run-time support routines that are directly callable from an Icon program.
rt*	Source code for the Icon run-time system.
pifuncs*	Source code for the C functions in the Icon program library.
pilib*	Files used to build personalized interpreters.

samples	Sample programs and associated data.
pidemo	Sample personalized interpreter.
docs	Documentation and manual pages.
src	Source code for programs and procedures in the Icon program library [1].
ibin	Application programs and games from the Icon program library.
ilib	Linkable code for procedures in the Icon program library.
man	Manual pages for the Icon program library.
libtest	Tests for the Icon program library.
book	Source code for procedures appearing in the Icon book [2].
test	Tests programs that are more comprehensive than those in samples.
port	Support material for transporting Version 5.9 to other computers [3, 4].

6.2 Disk Utilization

As mentioned earlier, not all of the directories on the distribution tape are needed in order to install Icon. Once Icon is working satisfactorily,

```
cd v5g
make Clean
```

can be used to remove non-source files and test results. Additional disk space can be saved by deleting source code after the Icon system is built.

The following table shows the approximate amount of disk space needed for the Icon system with the sets and experimental extensions included. Phase 1 refers to building Icon proper. Phase 2 refers to the installation of subsidiary components and testing in all directories. This data was obtained on a VAX running 4.2bsd. All figures are in kilobytes.

	as distributed	after phase 1	after phase 2	after clean up
bin*	9	161	161	154
h*	40	40	40	40
tran*	192	295	295	192
link*	83	147	147	83
iconx*	54	172	172	54
functions*	112	205	205	112
operators*	88	167	167	88
lib*	96	131	131	96
rt*	157	267	267	157
pifuncs*	14	14	26	14
pilib*	6	261	261	261
samples	48	49	65	49
pidemo	1	1	446	446
docs	479	479	479	479
src	125	125	125	125
ibin	1	1	85	85
ilib	1	1	143	143
man	319	319	319	319
libtest	163	163	232	171
book	192	192	192	192
test	246	247	348	247
port	257	260	454	260
total (including root)	2696	3714	4777	3784

6.3 Machine-Dependent Code

Since the same source is used to produce versions of Icon on both VAXs and PDP-11s, as well as for porting to other computers, the portions of the code that are machine dependent are bracketed with preprocessor control statements to select sections appropriate for each machine. The setup options -vax, -pdp11, and -port define the preprocessor symbols VAX, PDP11, and PORT, respectively.

Preprocessor control statements are used primarily to select appropriate assembly source code, but they also are used in a few cases for C source code. In some cases where differences are extensive (as for assembler files), the section of code for the VAX appears in its entirety and is followed by the code for the PDP-11 in its entirety. Preprocessor control statements for PORT bracket dummy code sections that are filled in as part of the porting process.

6.4 Recompilation of System Components

There is a Makefile in v5g for carrying out various tasks, as described in preceding sections. This Makefile typically performs corresponding *makes* in subdirectories.

The subdirectories v5g/iconx, v5g/tran, and v5g/link each contain code for a single component of Icon. Doing a *make* in any of these directories causes the particular component to be remade. The resulting component may then be copied into the v5g/bin directory.

The subdirectories v5g/functions, v5g/lib, v5g/operators, and v5g/rt each contain source code for a part of the Icon run-time system. The Icon interpreter, iconx, is formed by linking all the run-time subroutines together with the routines in v5g/iconx. When changes are made to the run-time system, all affected libraries must be rebuilt and then iconx must be rebuilt. For example, if the files v5g/operators/bang.c and v5g/functions/read.c have been modified, the following sequence of commands rebuilds the system.

```
cd v5g/functions
make
cd ../operators
make
cd ../iconx
make
cp iconx ../bin          # copy new version of interpreter to bin
```

Alternatively,

```
cd v5g
make Icon
```

has the same affect.

Note that if an alternate directory has been specified in Icon-setup via the -ibin option, the files in bin must then be copied to the alternate directory after the *make*.

6.5 PDP-11 Yacc Modifications for the Icon Translator

This section is relevant only if modifications are to be made to the Icon grammar, which is contained in the file tran/icon.g. The version of Yacc distributed with VAX systems is large enough to build the Icon parser, but it may be necessary to build a version of Yacc with larger parameters on a PDP-11. The following defined constants in the file dextern (in the yacc source directory) should be given the values listed below. Larger values are acceptable for all these constants, but are not necessary.

```
# ifdef HUGE
# define ACTSIZE 3000
# define MEMSIZE 6000
# define NSTATES 300
# define NTERMS 127
# define NPROD 200
# define NNONTERM 100
# define TEMPSIZE 1200
# define CNAMSZ 4100
# define LSETSIZE 200
# define WSETSIZE 200
# endif
```

The constant HUGE should be defined instead of MEDIUM at the end of the file yacc/files. Then Yacc should be rebuilt.

6.6 Obtaining Source Code Listings

Execution of the commands

```
cd v5g
make Listall
```

produces listings of all source files for the Icon system proper on standard output. Use the command

```
cd v5g
make List
```

to obtain listings of all such files that have been altered since the last make List or make Listall.

7. Electronic Mail and Problem Reporting

A mailbox has been established to facilitate communication with the Icon Project. Use the following addresses for electronic mail:

icon-project.arizona@csnet-relay	(CSNET and ARPANET)
arizona!icon-project	(Usenet and uucpnet)

The Icon Project currently has uucp connections established through noao, mcnc, ihnp4, and utah-cs.

If any problems are encountered with the Icon system, send electronic mail or telephone the Icon Project at 602-621-6613. If these forms of communication are not convenient, use the Trouble Report Forms supplied with the distribution package.

Acknowledgements

Rob McConeghy and Steve Wampler made a number of contributions to the procedures for installing and maintaining Icon.

References

1. Griswold, Ralph E. and Madge T. Griswold. *The Icon Programming Language.* Prentice-Hall Inc., Englewood Cliffs, New Jersey. 1983.

2. Griswold, Ralph E., Robert K. McConeghy, and William H. Mitchell. *Version 5.9 of Icon.* Technical report, Department of Computer Science, The University of Arizona. August 1984.

3. Griswold, Ralph E. *The Icon Program Library.* Technical Report TR 84-12, Department of Computer Science, The University of Arizona. August 1984.

4. Mitchell, William H. *Porting the UNIX Implementation of Icon.* Technical Report TR 83-10d, Department of Computer Science, The University of Arizona. August 1984.

5. Griswold, Ralph E. *An Overview of the Porting Process for Version 5.9 of Icon.* Department of Computer Science, The University of Arizona. October 1984.

6. Griswold, Ralph E., Robert K. McConeghy, and William H. Mitchell. *Extensions to Version 5 of the Icon Programming Language.* Technical Report TR 84-10a, Department of Computer Science, The University of Arizona. August 1984.

7. Griswold, Ralph E., Robert K. McConeghy, and William H. Mitchell. *Personalized Interpreters for Icon.* Technical Report TR 84-14, Department of Computer Science, The University of Arizona. August 1984.

Appendix A — Icon Hierarchy

v5g	root directory of the distribution
/bin	binaries and command processor
/h	header files
/tran	translator
/link	linker
/iconx	interpreter
/functions	built-in functions
/operators	built-in operators
/rt	run-time support
/lib	execution library
/pifuncs	C functions for personalized interpreters
/pilib	library for personalized interpreters
/samples	sample Icon programs
/distr	standard test results
/local	local test results
/pidemo	sample personalized interpreter
/docs	text for documents
/src	source code for the Icon program library
/cmd	Icon programs
/lib	Icon procedures
/libtest	ttest programs for the Icon program library
/distr	standard test results
/local	local test results
/ibin	executable code for Icon library programs
/ilib	linkable code for Icon program library procedures
/man	manual pages
/man0	text for front matter
/man1	text for Section 1
/man2	text for Section 2
/man3	text for Section 3
/man6	text for Section 6
/man7	text for Section 7
/man8	text for Section 8
/cat0	formatted pages for front matter
/cat1	formatted pages for Section 1
/cat2	formatted pages for Section 2
/cat3	formatted pages for Section 3
/cat6	formatted pages for Section 6
/cat7	formatted pages for Section 7
/cat8	formatted pages for Section 8
/test	test battery
/distr	standard test results
/local	local test results
/book	procedures from the Icon book
/01	procedures from Chapter 1
/02	procedures from Chapter 2
/03	procedures from Chapter 3
/04	procedures from Chapter 4
/05	procedures from Chapter 5

	/06	procedures from Chapter 6
	/07	procedures from Chapter 7
	/08	procedures from Chapter 8
	/09	procedures from Chapter 9
	/10	procedures from Chapter 10
	/11	procedures from Chapter 11
	/12	procedures from Chapter 12
	/13	procedures from Chapter 13
	/14	procedures from Chapter 14
	/15	procedures from Chapter 15
	/16	procedures from Chapter 16
	/17	procedures from Chapter 17
	/18	procedures from Chapter 18
	/19	procedures from Chapter 19
	/20	procedures from Chapter 20
	/f	procedures from Appendix F
/port		porting tests
	/distr	standard test results
	/local	local test results

Appendix B — Listing of Distributed Icon Files

Files names followed by a slash are directories. Asterisks identify executable files (shell scripts).

Icon-setup*	functions/	lib/	pifuncs/	test/
Makefile	h/	libtest/	pilib/	tran/
Pimakefile.gen	ibin/	link/	port/	
bin/	icon-pi.gen*	man/	rt/	
book/	iconx/	operators/	samples/	
docs/	ilib/	pidemo/	src/	
bin:				
Makefile	Makefile.gen	icont.c	ixhdr.c	
book:				
01/	08/	13/	18/	alpha.ls
02/	09/	14/	19/	f/
04/	10/	15/	20/	page.ls
05/	11/	16/	Makefile	
07/	12/	17/	READ.ME	
book/01:				
countm1.icn	hello1.icn	hello3.icn	hello5.icn	locate2.icn
countm2.icn	hello2.icn	hello4.icn	locate1.icn	
book/02:				
break1.icn	break2.icn	break3.icn	next.icn	
book/04:				
balop.icn	inset2.icn	minmax2.icn	vbars.icn	wordlist1.icn
icwrite.icn	lmark.icn	powers.icn	word1.icn	words1.icn
inset1.icn	minmax1.icn	section.icn	word2.icn	
book/05:				
array.icn	get.icn	tabwords1.icn	wordlen.icn	wordlist2.icn
book/07:				
exor1.icn	expr2.icn	fib2.icn	fword2.icn	
expr1.icn	fib1.icn	fword1.icn	nword.icn	
book/08:				
maxel.icn	words2.icn			
book/09:				
copy1.icn	copy2.icn	copy3.icn		
book/10:				
display.icn	shuffle1.icn			
book/11:				
fibseq1.icn	mark.icn	rtl.icn		
genword.icn	marker.icn	to.icn		
book/12:				
tabwords2.icn	words3.icn			
book/13:				
alt.icn	equalseq.icn	every.icn	filter1.icn	inter.icn
book/14:				
8q.icn	cross1.icn	limit1.icn	limit3.icn	stars.icn
break4.icn	cross2.icn	limit2.icn	posint.icn	
book/15:				
abc.icn	arbno.icn	parsexp.icn	recexp.icn	tab.icn
arb.icn	lmatch.icn	rarb.icn	shades.icn	

book/16:

eq.icn	lgraph.icn	stree.icn	visit.icn	
ldag.icn	ltree.icn	teq.icn		

book/17:

close.icn	disp.icn	reverse.icn	shuffle2.icn	swap.icn

book/18:

add1.icn	add2.icn	add3.icn	mpy.icn

book/19:

drv.icn	form1.icn	symadd.icn	
fix.icn	form2.icn	symop.icn	

book/20:

rsg1.icn

book/f:

8qp.icn	both.icn	fact.icn	infix.icn	rotate.icn
abcd.icn	btree.icn	filerev.icn	large.icn	rsg2.icn
acker1.icn	cdigit.icn	filter2.icn	lastline.icn	rsg3.icn
acker2.icn	charimage.icn	filter3.icn	limit4.icn	seqimage.icn
ackertr.icn	complex.icn	first.icn	locate3.icn	space.icn
allbal1.icn	dashes.icn	fixfunc.icn	nchars.icn	symmpy.icn
allbal2.icn	delete1.icn	form3.icn	oddline.icn	symsub.icn
allbal3.icn	delete2.icn	gener.icn	palseq.icn	tabwords3.icn
aver.icn	depth.icn	genpos.icn	pause.icn	tcopy.icn
bincop.icn	enrepl.icn	gensubstr.icn	qseq.icn	uscore.icn
boldface.icn	exor2.icn	hexcvt.icn	repalt.icn	vcount.icn

docs:

Makefile	icont.1	tr83-10a.roff	tr84-11	tr84-14
READ.ME	porting	tr83-10b.roff	tr84-11a.roff	version5.9
cover	reportform	tr83-10c.roff	tr84-11b.roff	
distpack	tmac.tr	tr83-3	tr84-12	
icon-pi.1	tr83-10	tr84-10	tr84-13	

functions:

Makefile	exit.c	member.c	reads.c	tab.c
abs.c	find.c	move.c	real.c	table.c
any.c	get.c	numeric.c	repl.c	trim.c
bal.c	image.c	open.c	reverse.c	type.c
center.c	insert.c	pop.c	right.c	upto.c
close.c	integer.c	pos.c	seq.c	write.c
collect.c	left.c	proc.c	set.c	writes.c
copy.c	list.c	pull.c	sort.c	
cset.c	many.c	push.c	stop.c	
delete.c	map.c	put.c	string.c	
display.c	match.c	read.c	system.c	

h:

Makefile	ctype.h	gc.h	pnames.h
config.h	defs.s	keyword.h	record.h
config.h.gen	err.h	pdef.h	rt.h

ibin:

.placeholder

iconx:

Makefile	init.c	main.c	pstop.s
Makefile.gen	interp.s	pstart.s	start.s

ilib:

.placeholder

lib:

Makefile	coret.s	esusp.s	limit.c	pfail.s
bscan.c	create.c	field.c	llist.c	pret.s
coact.s	efail.s	invoke.s	lsusp.s	psusp.s
cofail.s	escan.c	keywd.c	mkrec.c	

libtest:

Functest*	t-collate.dat	t-gener.dat	t-loadmap	t-shuffile.dat
Makefile	t-collate.icn	t-gener.icn	t-math.icn	t-shuffle.dat
Proctest*	t-complex.dat	t-getenv.icn	t-parens	t-shuffle.icn
Proctest.gen*	t-complex.icn	t-gpack.dat	t-patterns.dat	t-size.dat
Prog1test*	t-cppp	t-gpack.icn	t-patterns.icn	t-size.icn
Prog2test*	t-cppp.dat	t-gset	t-pdae.dat	t-snapshot.dat
READ.ME	t-cross.dat	t-i-psort.dat	t-pdae.icn	t-snapshot.icn
distr/	t-csgen.dat	t-i-split	t-pdco.dat	t-structs.dat
func.tlist	t-deal.dat	t-i-split.dat	t-pdco.icn	t-structs.icn
local/	t-delam	t-i-trfil	t-queens.dat	t-strutil.dat
pdef.h	t-delam.dat	t-i-trfil.dat	t-radcon.dat	t-strutil.icn
proc.tlist	t-delamc	t-i-xref.dat	t-radcon.icn	t-tablc.dat
prog1.tlist	t-delamc.dat	t-image.dat	t-roffcmds.dat	t-tablw.dat
prog2.tlist	t-edscript.dat	t-image.icn	t-rsg.dat	t-trig.icn
sizes.c	t-escape.dat	t-iscope.icn	t-seek.icn	t-trim.dat
t-bitops.dat	t-escape.icn	t-labels.dat	t-segment.dat	t-ttyctl.icn
t-bitops.icn	t-farb.dat	t-lam	t-segment.icn	t-ttyinit.dat
t-bold.dat	t-fset	t-lam.dat	t-seqimage.dat	t-ttyinit.icn
t-bold.icn	t-gcomp	t-ll.dat	t-seqimage.icn	t-worm

libtest/distr:

bitops.out	escape.out	i-xref.out	pdco.out	snapshot.out
bold.out	farb.out	image.out	queens.out	structs.out
collate.out	fset.out	iscope.out	radcon.out	strutil.out
complex.out	gcomp.out	labels.out	roffcmds.out	tablc.out
cppp.out	gener.out	lam.out	rsg.out	tablw.out
cross.out	getenv.out	ll.out	seek.out	trig.out
csgen.out	gpack.out	loadmap.out	segment.out	trim.out
deal.out	gset.out	math.out	seqimage.out	ttyctl.out
delam.out	i-psort.out	parens.out	shuffile.out	ttyinit.out
delamc.out	i-split.out	patterns.out	shuffle.out	worm.out
edscript.out	i-trfil.out	pdae.out	size.out	

libtest/local:
.placeholder

link:

Makefile	glob.c	lcode.c	lsym.c	
builtin.c	ilink.c	llex.c	opcode.c	
datatype.h	ilink.h	lmem.c	opcode.h	

man:

Makefile	cat3/	man0/	man6/	tmac.an.new
cat0/	cat6/	man1/	man7/	tmac.an6n
cat1/	cat7/	man2/	man8/	tmac.an6t
cat2/	cat8/	man3/	tmac.an	tmac.ilib

man/cat0:

ptx	toc

man/cat1:

cppp.1	fset.1	i-trfil.1	loadmap.1	tablc.1
csgen.1	gcomp.1	i-xref.1	parens.1	tablw.1
delam.1	gset.1	labels.1	roffcmds.1	trim.1
delamc.1	i-psort.1	lam.1	rsg.1	
edscript.1	i-split.1	ll.1	shuffile.1	

man/cat2:

bitops.2	escape.2	patterns.2	segment.2	snapshot.2
bold.2	gener.2	pdae.2	seqimage.2	structs.2
collate.2	gpack.2	pdco.2	shuffle.2	strutil.2
complex.2	image.2	radcon.2	size.2	ttyinit.2

man/cat3:

getenv.3	math.3	trig.3		
iscope.3	seek.3	ttyctl.3		

man/cat6:

cross.6	deal.6	farb.6	queens.6	worm.6

man/cat7:

i-hier.7				

man/cat8:

lman.8	uman.8			

man/man0:

ptx.in	toc.in	toc2	toc4	toc6	toc8
ptxx	toc1	toc3	toc5	toc7	

man/man1:

cppp.1	fset.1	i-trfil.1	loadmap.1	tablc.1
csgen.1	gcomp.1	i-xref.1	parens.1	tablw.1
delam.1	gset.1	labels.1	roffcmds.1	trim.1
delamc.1	i-psort.1	lam.1	rsg.1	
edscript.1	i-split.1	ll.1	shuffile.1	

man/man2:

bitops.2	escape.2	patterns.2	segment.2	snapshot.2
bold.2	gener.2	pdae.2	seqimage.2	structs.2
collate.2	gpack.2	pdco.2	shuffle.2	strutil.2
complex.2	image.2	radcon.2	size.2	ttyinit.2

man/man3:

getenv.3	math.3	trig.3		
iscope.3	seek.3	ttyctl.3		

man/man6:

cross.6	deal.6	farb.6	queens.6	worm.6

man/man7:

i-hier.7				

man/man8:

lman.8	uman.8			

operators:

Makefile	lconcat.c	mult.c	numle.c	sect.c
asgn.c	lexeq.c	neg.c	numlt.c	size.c
bang.c	lexge.c	neqv.c	numne.c	subsc.c
cat.c	lexgt.c	nonnull.c	plus.c	swap.c
compl.c	lexle.c	null.c	power.c	tabmat.c
diff.c	lexlt.c	number.c	random.c	toby.c
div.c	lexne.c	numeq.c	rasgn.c	unioncs.c
eqv.c	minus.c	numge.c	refresh.c	value.c
inter.c	mod.c	numgt.c	rswap.c	

pidemo:

.placeholder				

pifuncs:

Makefile	iscope.c	seek.c	ttyctl.c	
getenv.c	math.c	trig.c		

pilib:

Makefile	Pilib*			

port:

Linkchecker*	basis4.icn	esusp1.icn	lsusp1.icn	roman.icn
Linktest.gen*	basis5.icn	esusp2.icn	meander.icn	rsg.icn
Makefile	basis6.icn	fail.tlist	prefix.icn	seqimage.icn
Runtest.gen*	basis7.icn	fail1.icn	pret.tlist	set1.tlist
Runtestall*	basis8.icn	fail2.icn	pret1.icn	suspend.tlist
Trantest.gen*	btrees.icn	gc.tlist	pret2.icn	suspend1.icn
arith.tlist	cross.icn	gc1.icn	pret3.icn	suspend2.icn
arith1.icn	display.tlist	gc2.icn	proto.icn	wordcount.icn
basis.tlist	display1.icn	hello.icn	psusp.tlist	
basis1.icn	display2.icn	lit.icn	psusp1.icn	
basis2.icn	distr/	local/	psusp2.icn	
basis3.icn	esusp.tlist	lsusp.tlist	recogn.icn	

port/distr:

arith1.out	cross.u2	lit.u1	proto.u1	rsg.ux
basis1.out	cross.ux	lit.u2	proto.u2	seqimage.u1
basis2.out	display1.out	lit.ux	proto.ux	seqimage.u2
basis3.out	display2.out	lsusp1.out	psusp1.out	seqimage.ux
basis4.out	esusp1.out	meander.u1	psusp2.out	suspend1.out
basis5.out	esusp2.out	meander.u2	recogn.u1	suspend2.out
basis6.out	fail1.out	meander.ux	recogn.u2	wordcount.u1
basis7.out	fail2.out	prefix.u1	recogn.ux	wordcount.u2
basis8.out	gc1.out	prefix.u2	roman.u1	wordcount.ux
btrees.u1	gc2.out	prefix.ux	roman.u2	
btrees.u2	hello.u1	pret1.out	roman.ux	
btrees.ux	hello.u2	pret2.out	rsg.u1	
cross.u1	hello.ux	pret3.out	rsg.u2	

port/local:
.placeholder

rt:

Makefile	cvpos.c	doasgn.c	host.c	pow.c
addmem.c	cvreal.c	dump.c	ldfps.s	putstr.c
alc.c	cvstr.c	equiv.c	lexcmp.c	qtos.c
anycmp.c	dblocks.c	exp.c	locate.c	setbound.s
arith.s	defany.c	fail.s	log.c	strprc.c
cplist.c	defcset.c	floor.c	memb.c	suspend.s
csv.s	deffile.c	gc.c	mkint.c	sweep.c
ctype.c	defint.c	gcollect.s	mkreal.c	trace.c
cvcset.c	defshort.c	gcvt.c	mksubs.c	
cvint.c	defstr.c	getstr.c	numcmp.c	
cvnum.c	deref.c	hash.c	outimage.c	

samples:

Makefile	diffwords.icn	parallel.dat	recogn.icn	sieve.icn
Test.gen*	distr/	parallel.icn	roman.dat	std.tlist
btrees.dat	hello.dat	pdco.dat	roman.icn	wordcount.dat
btrees.icn	hello.icn	pdco.icn	seqimage.dat	wordcount.icn
cross.dat	local/	prefix.dat	seqimage.icn	xpx.tlist
cross.icn	meander.dat	prefix.icn	set.tlist	
diffwords.dat	meander.icn	recogn.dat	sieve.dat	

samples/distr:

btrees.out	hello.out	pdco.out	roman.out	wordcount.out
cross.out	meander.out	prefix.out	seqimage.out	
diffwords.out	parallel.out	recogn.out	sieve.out	

samples/local:
.placeholder

src:

Makefile	cmd/	lib/

src/cmd:

Makefile	delam.icn	gset.icn	lam.icn	rsg.icn
Makefile.gen	delamc.icn	i-psort.icn	ll.icn	shuffile.icn
cppp.icn	edscript.icn	i-split.icn	loadmap.icn	tablc.icn
cross.icn	farb.icn	i-trfil.icn	parens.icn	tablw.icn
csgen.icn	fset.icn	i-xref.icn	queens.icn	trim.icn
deal.icn	gcomp.icn	labels.icn	roffcmds.icn	worm.icn

src/lib:

Makefile	complex.icn	patterns.icn	seqimage.icn	strutil.icn
Makefile.gen	escape.icn	pdae.icn	shuffle.icn	ttyinit.icn
bitops.icn	gener.icn	pdco.icn	size.icn	
bold.icn	gpack.icn	radcon.icn	snapshot.icn	
collate.icn	image.icn	segment.icn	structs.icn	

test:

Makefile	std10.icn	std30.icn	std50.icn	std70.icn
READ.ME	std11.icn	std31.icn	std51.icn	std71.icn
Test.gen*	std12.icn	std32.icn	std52.icn	std72.icn
buildt.icn	std13.icn	std33.icn	std53.icn	std73.icn
dismem.icn	std14.icn	std34.icn	std54.icn	std74.icn
distr/	std15.icn	std35.icn	std55.icn	std75.icn
local/	std16.icn	std36.icn	std56.icn	std76.icn
set.tlist	std17.icn	std37.icn	std57.icn	std77.icn
set01.icn	std18.icn	std38.icn	std58.icn	std78.icn
set02.icn	std19.icn	std39.icn	std59.icn	std79.icn
std.tlist	std20.icn	std40.icn	std60.icn	std80.icn
std01.icn	std21.icn	std41.icn	std61.icn	std81.icn
std02.icn	std22.icn	std42.icn	std62.icn	std82.icn
std03.icn	std23.icn	std43.icn	std63.icn	std83.icn
std04.icn	std24.icn	std44.icn	std64.icn	std84.icn
std05.icn	std25.icn	std45.icn	std65.icn	std85.icn
std06.icn	std26.icn	std46.icn	std66.icn	xpx.tlist
std07.icn	std27.icn	std47.icn	std67.icn	xpx01.icn
std08.icn	std28.icn	std48.icn	std68.icn	xpx02.icn
std09.icn	std29.icn	std49.icn	std69.icn	xpx03.icn

test/distr:

set01.out	std17.out	std35.out	std53.out	std71.out
set02.out	std18.out	std36.out	std54.out	std72.out
std01.out	std19.out	std37.out	std55.out	std73.out
std02.out	std20.out	std38.out	std56.out	std74.out
std03.out	std21.out	std39.out	std57.out	std75.out
std04.out	std22.out	std40.out	std58.out	std76.out
std05.out	std23.out	std41.out	std59.out	std77.out
std06.out	std24.out	std42.out	std60.out	std78.out
std07.out	std25.out	std43.out	std61.out	std79.out
std08.out	std26.out	std44.out	std62.out	std80.out
std09.out	std27.out	std45.out	std63.out	std81.out
std10.out	std28.out	std46.out	std64.out	std82.out
std11.out	std29.out	std47.out	std65.out	std83.out
std12.out	std30.out	std48.out	std66.out	std84.out
std13.out	std31.out	std49.out	std67.out	std85.out
std14.out	std32.out	std50.out	std68.out	xpx01.out
std15.out	std33.out	std51.out	std69.out	xpx02.out
std16.out	std34.out	std52.out	std70.out	xpx03.out

test/local:
.placeholder

```
tran:
Makefile        err.c           lex.c           mktoktab.icn    sym.h
Makefile.gen    icon.g          lex.h           optab           synerr.h
char.c          itran.c         lfile.h         optab.c         token.h
char.h          itran.h         lnklist.c       parse.c         tokens
code.c          keyword.c       mem.c           pscript         toktab.c
code.h          keywords        mkkeytab.icn    sym.c           tree.h
```