NAME

icon − compile and interpret Icon programs

SYNOPSIS

iconc [option ...] file ... [−x arg ...] iconc [option ...] file ... [−x arg ...]

DESCRIPTION

The program icont is a command processor for running Version 8.5 Icon programs. Used in its simplest form, it produces a file suitable for interpretation by the Icon interpreter. Processing consists of two phases: *translation* and *linking*. During translation, each Icon source file is translated into an intermediate language called *ucode*; during linking, the one or more ucode files are combined and a single *icode* file is produced. Unless the −o option is specified, the name of the resulting icode file is formed by deleting the suffix of the first input file named on the command line. If the −x argument is used, the file is automatically executed by the interpreter and any arguments following the −x are passed as execution arguments to the Icon program itself.

Files whose names end in .icn are assumed to be Icon source programs. The .icn suffix may be omitted; it is supplied automatically. These programs are translated, and the intermediate code is left in two ucode files of the same name with .u1 and .u2 substituted for .icn. The ucode files normally are deleted when icont completes. Files whose names end in .u1 are assumed to refer to ucode files from a previous translation; these files and the corresponding .u2 files are included in the linking phase after any .icn files have been translated. The suffix .u can be used in place of .u1; in this case the 1 is supplied automatically. A .u1 or .u2 file that is explicitly named is not deleted. 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 following options are recognized by icont:

−c Suppress the linking phase. The ucode code files are not deleted.

−m Preprocess each .icn source file with the *m4(1)* macro processor before translation.

−o *output*
Name the icode file *output*.

−s Suppress informative messages from the translator and linker. Normally, both informative messages and error messages are sent to standard error output.

−t Arrange for &trace to have an initial value of −1 when the program is executed. Normally, &trace has an initial value of 0.

−u Issue warning messages 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 their sizes can be changed, if necessary, by 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. The tables and their default sizes are:

c	constant table	100
f	field table	100
g	global symbol table	200
i	identifier table	500
l	local symbol table	100
n	line number space	1000
r	field table for records	100
s	string space	20000
t	tree space	15000
C	code buffer	15000
F	file names	10

L	labels	500

The units depend on the table involved, but the default values can be used as a general guide for appropriate settings of −S options without knowing the units.

The environment variable IPATH controls the location of files specified in link declarations. The value of IPATH should be a blank-separated string of the form *p1 p2 ... pn* where the *pi* name directories. Each directory is searched in turn to locate files named in link directives. The default value for IPATH is . , that is, the current directory. The current directory is always searched first, regardless of the value of IPATH.

The icode file produced by the Icon linker is executable. For example, the command

```
icont hello.icn
```

produces a file named hello that can be run by the command

```
hello
```

Arguments can be passed to the Icon program by following the program name with the arguments. Any such arguments are passed to the main procedure as a list of strings.

The location of iconx, the executor for icode files, is built into an icode file when it is produced. This location can be overridden by setting the environment variable ICONX as described below. If ICONX is not set and iconx is not found on the built-in path, PATH is searched for it.

When an Icon program is executed, several environment variables are examined to determine certain execution parameters. Except for ICONX, NOERRBUF, and ICONCORE, the values assigned to these variables should be numbers. The variables that affect execution and the interpretations of their values follow. Numbers in parentheses are the default values.

ICONX
: If this environment variable is set, it specifies the location of iconx to use to execute an icode file.

TRACE
: Initialize the value of &trace. If this variable has a value, it overrides the translation-time −t option.

NOERRBUF
: By default, &errout is buffered. If this variable is set, &errout is not buffered.

ICONCORE
: If set, a core dump is produced for error termination.

STRSIZE (65000)
: The initial size of the string space, in bytes. The string space grows if necessary, but it never shrinks.

HEAPSIZE (65000)
: The initial size of the allocated block region, in bytes. The block region grows if necessary, but it never shrinks.

COEXPSIZE (2000)
: The size, in words, of each co-expression block.

MSTKSIZE (10000)
: The size, in words, of the main interpreter stack.

STATSIZE (20480)
: The size, in bytes, of the static region in which co-expression blocks are allocated. If co-expressions are not implemented, the default size is 1024.

STATINCR
: The size of the increment used when the static region is expanded. The default increment is one-fourth of the initial size of the static region.

QLSIZE (5000)

The size, in bytes, of the region used for pointers to strings during garbage collection (fixed-regions implementations only).

MEMMON

The name of the output file for memory monitoring.

FILES

iconc	Icon translator
icont	Icon translator
iconx	Icon executor

SEE ALSO

The Icon Programming Language, Ralph E. Griswold and Madge T. Griswold, Prentice-Hall Inc., Englewood Cliffs, New Jersey, Second Edition, 1990.

Version 8.5 of Icon, Ralph E. Griswold, Clinton L. Jeffery, Gregg M. Townsend, and Kenneth Walker, IPD174, Department of Computer Science, The University of Arizona, 1992.

m4(1), icon_vt(1)

LIMITATIONS AND BUGS

Icode files do not stand alone; the Icon run-time system must be present.

Stack overflow is checked using a heuristic that is not always effective.

If the −m option is used, line numbers reported in error messages or tracing messages are from the file after, not before, preprocessing.