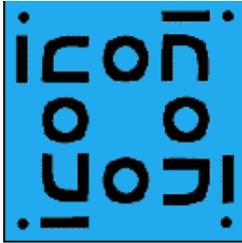


Icon 9 UNIX Manual Page

Ralph E. Griswold



Department of Computer Science
The University of Arizona
Tucson, Arizona

IPD244d

November 29, 1996

<http://www.cs.arizona.edu/icon/docs/ipd244.htm>

NAME

`icon` -- interpret or compile Icon programs

SYNOPSIS

`icont` [option ...] file ... [-x arg ...]

`iconc` [option ...] file ... [-x arg ...]

DESCRIPTION

`icont` and `iconc` each convert an Icon source program into executable form. `icont` translates quickly and provides interpretive execution. `iconc` takes longer to compile but produces programs that execute faster. `icont` and `iconc` for the most part can be used interchangeably.

This manual page describes both `icont` and `iconc`. Where there are differences in usage between `icont` and `iconc`, these are noted.

File Names: Files whose names end in `.icn` are assumed to be Icon source files. The `.icn` suffix may be omitted; if it is not present, it is supplied. The character `-` can be used to indicate an Icon source file given in standard input. Several source files can be given on the same command line; if so, they are combined to produce a single program.

The name of the executable file is the base name of the first input file, formed by deleting the suffix, if present. `stdin` is used for source programs given in standard input.

Processing: As noted in the synopsis above, `icont` and `iconc` accept options followed by file names, optionally followed by `-x` and arguments. If `-x` is given, the program is executed automatically and any following arguments are passed to it.

`icont`: The processing performed by `icont` consists of two phases: translation and linking. During translation, each Icon source file is translated into an intermediate language called ucode. Two ucode files are produced for each source file, with base names from the source file and suffixes `.u1` and `.u2`. During linking, the one or more pairs of ucode files are combined to produce a single icode file. The ucode files are deleted after the icode file is created.

Processing by `icont` can be terminated after translation by the `-c` option. In this case, the ucode files are not deleted. The names of `.u1` files from previous translations can be given on the `icont` command line. These files and the corresponding `.u2` files are included in the linking phase after the translation of any source files. The suffix `.u` can be used in place of `.u1`; in this case the `1` is supplied automatically. Ucode files that are explicitly named are not deleted.

`iconc`: The processing performed by `iconc` consists of two phases: code generation and compilation and linking. The code generation phase produces C code, consisting of a `.c` and a `.h` file, with the base name of the first source file. These files are then compiled and linked to produce an executable binary file. The C files normally are deleted after compilation and linking.

Processing by `iconc` can be terminated after code generation by the `-c` option. In this case, the C files are not deleted.

OPTIONS

The following options are recognized by `icont` and `iconc`:

`-c` Stop after producing intermediate files and do not delete them.

`-e file` Redirect standard error output to file.

`-f s` Enable full string invocation.

`-o name` Name the output file name.

`-s` Suppress informative messages. 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 and for `iconc` enable debugging features.

`-u` Issue warning messages for undeclared identifiers in the program.

`-v i` Set verbosity level of informative messages to `i`.

`-E` Direct the results of preprocessing to standard output and inhibit further processing.

The following additional options are recognized by `iconc`:

`-f string` Enable features as indicated by the letters in `string`:

`a` all, equivalent to `delns`

`d` enable debugging features: `display()`, `name()`, `variable()`, error trace back, and the effect of `-f n` (see below)

`e` enable error conversion

- l enable large-integer arithmetic
- n produce code that keeps track of line numbers and file names in the source code
- s enable full string invocation
- n *string* Disable specific optimizations. These are indicated by the letters in *string*:
 - a all, equivalent to cest
 - c control flow optimizations other than switch statement optimizations
 - e expand operations in-line when reasonable (keywords are always put in-line)
 - s optimize switch statements associated with operation invocations
 - t type inference
- p *arg* Pass *arg* on to the C compiler used by *iconc*.
- r *path* Use the run-time system at *path*, which must end with a slash.

ENVIRONMENT VARIABLES

When an Icon program is executed, several environment variables are examined to determine certain execution parameters. Values in parentheses are the default values.

BLKSIZE (500000)

The initial size of the allocated block region, in bytes.

COEXPSIZE (2000)

The size, in words, of each co-expression block.

DBLIST

The location of data bases for *iconc* to search before the standard one. The value of *DBLIST* should be a blank-separated string of the form *p1 p2 . . . pn* where the *pi* name directories.

ICONCORE

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

ICONX

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*. If *ICONX* is set, its value is used in place of the location built into the icode file.

IPATH

The location of ucode files specified in link declarations for *icont*. *IPATH* is a blank-separated list of directories. The current *dirVersion 9.3* of the Icon Programming Language is always searched first, regardless of the value of *IPATH*.

LPATH

The location of source files specified in preprocessor `$include` directives and in link declarations for `iconc`. `LPATH` is otherwise similar to `IPATH`.

`MSTKSIZE(10000)`

The size, in words, of the main interpreter stack for `icont`.

`NOERRBUF`

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

`QLSIZE(5000)`

The size, in bytes, of the region used for pointers to strings during garbage collection.

`STRSIZE(500000)`

The initial size of the string space, in bytes.

`TRACE`

The initial value of `&trace`. If this variable has a value, it overrides the translation-time `-t` option.

FILES

<code>icont</code>	Icon translator
<code>iconc</code>	Icon compiler
<code>iconx</code>	Icon executor

SEE ALSO

The Icon Programming Language, Ralph E. Griswold and Madge T. Griswold, Peer-to-Peer Communications, Inc., Third Edition, 1996.

Version 9.3 of Icon, Ralph E. Griswold, Clinton L. Jeffery, and Gregg M. Townsend, IPD278, Department of Computer Science, The University of Arizona, 1996.

Version 9 of the Icon Compiler, Ralph E. Griswold, IPD237, Department of Computer Science, The University of Arizona, 1995.

`icon_vt(1)`

LIMITATIONS AND BUGS

The `icode` files for the interpreter do not stand alone; the Icon run-time system (`iconx`) must be present.

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