# Using Version 8.10 of Icon Under VMS

Gregg M. Townsend and Sandra L. Miller

Department of Computer Science, The University of Arizona

## 1. Introduction

This report describes the use of Version 8.10 of Icon [1, 2] under the VAX/VMS operating system.

Several commands for running Icon programs are described here. These commands are not a standard part of VMS and so they must be defined before they can be used. Unless the system administrator incorporates definitions into the system login procedure, explicit setup action is needed. This is performed by the command

```
$ @[directory]DEFICON
```

which you may wish to put in your LOGIN.COM file for convenience. *directory* is the directory containing the Icon executables and is dependent on the particular installation.

## 2. Using the Interpreter

### Translation and Linking

Icon source programs must be transformed into an internal code before they can be executed. This is done through use of the ICONT program. Its command arguments consist of options (if any), then one or more files, then −x and any program arguments if execution is desired. It is important to note that the options must precede all file names. The command may be summarized as:

```
$ ICONT [options] files [−x arguments]
```

Used in its simplest form, ICONT 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 that of the first input file but with a .ICX file type. 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 of type .ICN are assumed to be Icon source programs; .ICN is assumed if no file type is specified. These programs are translated, and the intermediate code is left in two ucode files of the same name with .U1 and .U2 as file types. The ucode files normally are deleted when ICONT completes. Icon source programs may be read from standard input by giving "−" (including the quotation marks) as a file name. In this case, the ucode files are named STDIN.U1 and STDIN.U2 and the icode file is named STDIN.ICX.

Files of type of .U, .U1, or .U2 indicate inclusion of ucode from a previously translated source file. Only one of these types should be named; the corresponding .U1 and .U2 files are both read. These files are included in the linking phase after any .ICN files have been translated. Explicitly named ucode files are not deleted.

The following options are recognized by ICONT:

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

−e *efile*
Connect *efile* to &errout (similar to defining SYS$ERROR).

−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.

**Program Execution**

An icode file produced by ICONT is run by calling the interpreter, ICONX, with the linked program as its argument. Additional arguments may be given and are passed to the Icon program. For example, the command

```
$ ICONX MYPROG PARAM1 PARAM2 PARAM3
```

executes the file MYPROG.ICX with three parameters.

Program arguments of the form <*ifile* and >*ofile*, called file redirection arguments, cause *ifile* and *ofile* to be used for &input and &output respectively (SYS$INPUT and SYS$OUTPUT). An argument of the form "−e *efile*" immediately following the ICONX keyword connects *efile* to &errout (SYS$ERROR). File redirection arguments are not passed to the Icon program.

The logical name IPATH controls the location of files specified in link directives. The value of IPATH should be blank-separated sequences *p1 p2 … pn* where the *pi* name directories. Each directory is searched in turn to locate ucode files named in link directives. The current directory always is searched first, regardless of the value of IPATH.

When an Icon program is executed, several logical names are examined to determine certain execution parameters. These logical names should have numeric values. The variables that affect execution and the interpretations of their values are as follows:

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 NOERRBUF is defined, &errout is not buffered.

STRSIZE
The size of the initial string region, in bytes. The default value of STRSIZE is 65000.

BLKSIZE
The size of the initial allocated block region, in bytes. The default value of BLKSIZE is 65000. HEAPSIZE and BLOCKSIZE are synonyms for BLKSIZE.

COEXPSIZE
The size, in words, of each co-expression block. The default value of COEXPSIZE is 2000.

MSTKSIZE
The size, in words, of the main interpreter stack. The default value of MSTKSIZE is 10000.

QLSIZE
The size, in bytes, of the region used for pointers to strings during garbage collection. The default value of QLSIZE is 5000.

## 3. Using the Compiler

Version 8.10 of Icon also includes a compiler, ICONC [3]. This compiler generates C code and hence its use also requires a C compiler. The command arguments for the Icon compiler consist of options (if any), then one or more files, then −x and any program arguments if execution is desired. It is important to note that the options must precede all file names. The command may be summarized as:

```
$ ICONC [options] files [−x arguments]
```

Used in its simplest form, ICONC produces a single .EXE file. Unless the −o option is specified, the name of the resulting .EXE file is that of the first input file but with a .EXE file type. If the −x argument is used, the file is automatically executed and any arguments following the −x are passed as execution arguments to the Icon program itself.

Files of type .ICN are assumed to be Icon source programs; .ICN is assumed if no file type is specified. The compiler produces a C program source file and include file. These intermediate files are left in two files of the same name with .C and .H as file types. These files normally are deleted when ICONC completes. Icon source programs may be read from standard input by giving "−" (including the quotation marks) as a file name. In this case, the intermediate files are named STDIN.C and STDIN.H and the executable file is named STDIN.EXE.

The following options are recognized by ICONC:

−c Stop compilation after producing C code. The C program source and include files are not deleted.

−e *efile*
Connect *efile* to &errout (similar to defining SYS$ERROR).

−f *s* Enable features as indicated by the letters in *s*.

| | |
|---|---|
| a | all; equivalent to delns |
| d | enable debugging features, including 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 *s* Disable specific optimizations as indicated by the letters in *s*.

| | |
|---|---|
| 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 interface |

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

−p *arg*
Pass *arg* on to the C compiler used by ICONC.

−r *path*
Use the run-time system at *path*.

−s Suppress informative messages from the compiler. 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.

–v *i* Set verbosity level of informative messages to *i*.

″–C″ *prg*
Have ICONC use the C compiler given by *prg*.

The logical name LPATH controls the location of files specified in link directives. The value of LPATH should be blank-separated sequences *p1 p2 … pn* where the *pi* name directories. Each directory is searched in turn to locate source files named in link directives. The current directory always is searched first, regardless of the value of LPATH.

## 4. The Programming Environment

The IEXE command, which takes a file name as its single argument, defines a command that allows an Icon program to be executed by name. For example,

```
$ IEXE MYPROG
```

defines MYPROG as a command equivalent to RUN MYPROG if MYPROG.EXE is an executable built with the Icon compiler. If MYPROG.EXE doesn't exist, it defines MYPROG as a command equivalent to ICONX MYPROG if MYPROG.ICX is an icode file built with the Icon translator.

IEXE can handle disk/directory specifications associated with a file name. For example,

```
$IEXE DISK:[DIR]MYPROG.ICX
```

defines MYPROG as a command equivalent to

```
ICONX MYPROG
```

The main procedure is always called with a single argument that is a list of strings. This list contains any arguments passed to the program by the command that executed it. When there are no such arguments, the list is empty.

If the main procedure returns or fails, the DCL status is set to 1 indicating normal termination. If stop(s) is called, the value is hexadecimal 10000000, indicating an error but producing no additional messages. A call to exit(i) terminates the program with the status specified.

The call system(s) executes s as a command and produces its DCL status.

The call open(s,"rp") spawns a process to execute command s and produces a file that will read the standard output of that command. Similarly, open(s,"wp") spawns a command that will read its input from data written to the file produced by the call.

## 5. Warnings and Known Problems

Ucode and icode files produced under earlier versions of Icon are incompatible with this version. Such programs must be recompiled.

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

## References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, 1990.
2. R. E. Griswold, C. L. Jeffery and G. M. Townsend, *Version 8.10 of the Icon Programming Language*, The Univ. of Arizona Icon Project Document IPD212, 1993.
3. R. E. Griswold, *Using Version 8.10 of the Icon Compiler*, The Univ. of Arizona Icon Project Document IPD214, 1993.