Using Version 8 of Icon under VMS

Gregg M. Townsend
Sandra L. Miller

Department of Computer Science
The University of Arizona

Introduction

This paper describes the use of Icon 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 binaries and is dependent on the particular installation.

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 call for 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*
Connects *efile* to &errout.

–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.

–L Enable linker debugging.

Icon has a number of 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*cn*", where the key character *c* specifies the table and *n* is the number of storage units to allocate for the table. This option must be quoted because it and its suboptions are case sensitive. Key characters, meanings, and 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 logical name IPATH controls the location of files specified in link directives. The value of IPATH should be a blank-separated 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 the current directory.

Program Execution

An Icon program 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* cause *ifile* and *ofile* to be used for &input and &output respectively. An argument of the form "–e *efile*" immediately following the ICONX keyword connects *efile* to &errout. File redirection arguments are not passed to the Icon program.

The IEXE command, which takes a file name as its single argument, defines a command allowing an icode file to be executed by name. For example,

```
$ IEXE MYPROG
```

defines MYPROG as a command equivalent to ICONX MYPROG.

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 initial size of the string space, in bytes. The string space grows if necessary, but it never shrinks. The default value of STRSIZE is 65000.

BLOCKSIZE
: The initial size of the allocated block region, in bytes. The block region grows if necessary, but it never shrinks. The default value of BLOCKSIZE is 65000. HEAPSIZE and BLKSIZE are synonyms for BLOCKSIZE.

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.

STATSIZE
: The size, in bytes, of the static region in which co-expression blocks are allocated. The default value of STATSIZE is 20480.

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.

MEMMON
: The name of an output file, if memory allocation information is to be recorded.

MAXMEM
: The maximum amount of memory, in bytes, that can be allocated in total by all of Icon's memory regions. The default value of MAXMEM is 1000000.

The Programming Environment

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.

Memory Monitoring

The Icon interpreter is instrumented to provide a history of memory allocations. This tracing is enabled by setting the MEMMON logical name. The MMPS program can be used to produce PostScript snapshots of the interpreter's memory layout. This facility is described in more detail in [3]

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, *Version 8 of Icon*, The Univ. of Arizona Tech. Rep. 90-1, 1990.
3. G. M. Townsend, *The Icon Memory Monitoring System*, The Univ. of Arizona Icon Project Document IPD113, 1990.