Installing Version 9.0 of Icon under VMS

Gregg M. Townsend and Sandra L. Miller

Department of Computer Science, The University of Arizona

1. Introduction

Version 9.0 [1] is the current version of Icon, superseding Version 8.10. Version 9.0 contains new features, enhanced graphics faciliies [2], and significant changes to the implementation. These changes are described in more detail in an associated document [3].

This report provides the information necessary to install Version 9.0 of Icon under VMS. The installation process for Version 9.0 is similar to that for Version 8.10.

In addition to Icon itself, there is an optional program library [4]. You may want to review the technical reports describing these optional components before beginning the installation. In any event, the installation of optional components can be done separately after Icon itself is installed.

The instructions that follow are for the installation of both the interpreter and compiler. If you don't have a C compiler, if your resources are limited, or if you are new to Icon, you may wish to install only the interpreter initially. See Section 4 for information on installation of the interpreter or compiler separately.

2. General Notes

VMS Icon has been built and tested using a recent release of VMS as indicated in the status file.

The full Icon tape requires about 35,300 blocks when copied to disk. The .EXE, .COM and other files required for running Icon take about 6,600 blocks, and the optional Icon program library occupies around 26,100 blocks.

.EXE files are included on the distribution tape, so recompilation is usually not required. *Note:* The executable files as distributed require the DECWindows shared library. If this library is not installed locally, it can be obtained from DEC or Icon can be rebuilt without X-Icon capabilities. See Step 3 in Section 3.

Complete source code and build procedures are included for optional use on systems having a C compiler. A total of approximately 80,000 disk blocks are needed for a full rebuild.

Icon requires no special privileges to build. We recommend that Icon be built under whatever account owns the files; problems have been observed in trying to build Icon from other accounts, even SYSTEM, that depend on file access privileges.

The Icon Source Code

The source code for the Icon system is in the public domain. You may use and copy this material freely. This extends to modifications, although any modified version of this system given to a third party should clearly identify your modifications as well as the original source.

The responsibility for the use of this material resides entirely with you. We make no warranty of any kind concerning this material, nor do we make any claim as to the suitability of Icon for any application.

3. The Installation Process

Step 1: Unloading the Distribution Tape

These instructions assume that your tape drive is named MT:. The drive on your system may have a different name, and if so you should adjust the instructions accordingly.

The Icon distribution tape is a BACKUP tape containing four levels of directories. To unload the tape, first set your default directory to a parent directory that can contain the Icon hierarchy. Copy Icon to disk using these commands:

```
$ MOUNT/FOREIGN MT:
$ BACKUP MT:ICON9 [...]
$ SET DEFAULT [.V9]
```

These commands create a subdirectory V9 containing the Icon system and make it the current directory. For convenience, subsequent sections refer to this directory as [V9], though its actual location is system dependent and it is not necessarily a top-level directory.

Check the current status of the VMS distribution, and the VMS version under which it was prepared, by entering

```
$ TYPE STATUS.
```

This command and all subsequent ones assume that V9 is the default directory. Note that if a command script fails, it may leave the default directory set incorrectly.

Step 2: Configuring the Host Name

The value of &host seen by an Icon program is the logical symbol ICON_HOST if it exists; otherwise SYS$NODE if it exists; otherwise the string "VAX/VMS". Check to see if SYS$NODE is defined on your system by typing

```
$ SHOW LOGICAL SYS$NODE
```

If SYS$NODE is not defined (and you can’t persuade your system administrator to define it), edit [V9.BIN]DEFICON.COM to supply a value for ICON_HOST. Insert

$ DEFINE/NOLOG ICON_HOST *hostname::*

immediately before the EXIT line.

Step 3: Rebuilding Icon

The VMS distribution of Icon includes prebuilt binaries of the Icon system in [V9.BIN]. Rebuilding may be needed if your VMS system is an older version than the one indicated in the status file or if your system does not have the shareable DECwindows library installed. Otherwise, this section can be skipped.

A complete rebuild is possible if your system has a C compiler. This takes about two hours on a moderately loaded pair of VAX 4000/300’s. The following commands recompile and relink the Icon system:

```
$ SET VERIFY
$ @MAKE
```

If you do not have X-Windows capability on your system, you may need to rebuild Icon so that it will not try to link with the shareable DECwindows library at run time. To do this, edit the file [V9.SRC.H]DEFINE.H and delete the line

```
#define Graphics 1
```

Then recompile and relink the Icon system as explained above.

Step 4: Performing Some Simple Tests

A few simple tests usually are sufficient to confirm that Icon is running properly. A suite of nine tests is provided for testing the Icon system. It is run by these commands:

```
$ SET NOVERIFY
$ SET DEFAULT [.TESTS]
$ @SAMPLES
$ SET DEFAULT [–]
```

Each program's name is printed as it is run. Unless problems are found, there is no other output.

If your system does not have a C compiler, or if you have not installed the Icon compiler, you can test the interpreter only by using

```
$ @SAMPLES–ICONT
```

instead of

```
$ @SAMPLES
```

Step 5: Extensive Testing

If you want to run more extensive tests, do:

```
$ SET NOVERIFY
$ SET DEFAULT [.TESTS]
$ @GENERAL–ICONT
$ @GENERAL–ICONC (if you have the Icon compiler)
$ SET DEFAULT [–]
```

Each program's name is printed as it is run. Some differences are to be expected, since tests include date, time, and local host information. There also may be insignificant differences in the format of floating-point numbers and the order of random numbers. Testing icont takes about twenty minutes; testing iconc takes about three hours.

Step 6: Installing Icon

The files needed to run Icon are placed in [V9.BIN] as the result of building Icon:

ICONC.EXE	Icon compiler
ICONT.EXE	Icon translator for interpreter
ICONX.EXE	Icon executor for interpreter

Files needed by iconc also are placed in [V9.BIN]:

RT.OLB	compiler library
RT.DB	compiler database
RT.H	include file
X11.OPT	options file
DLRGINT.OBJ	object file for large integers

Some other files related to installing Icon and the optional components mentioned earlier also are placed in [V9.BIN].

DEFICON.COM	used to define commands and logical names
DEF_ICON_BIN.COM	used during build
GRTTIN.H	include file for rtt
ICON.HLB	Icon help file
IEXE.COM	used to establish an Icon executable as a command
RTT.EXE	used to build iconx and iconc

The directory [V9.BIN] holds all files required for compilation and execution of Icon programs. This directory can be left in place or moved as a unit to a system area. DEFICON.COM assumes that all needed files reside in the same directory.

Separate documentation explains the use of Icon, but it is worth noting here that any use of Icon requires the execution of [V9.BIN]DEFICON.COM to define commands and logical names. A system administrator can incorporate DEFICON.COM into the system login procedure, making Icon available to all users automatically and relieving them of the need to run the definition script.

4. Installing the Interpreter and Compiler Separately

As mentioned earlier, the interpreter and compiler can be built separately. If one is installed first, the other can be added without rebuilding the former.

Building the interpreter or compiler separately is very similar to building both at the same time. Steps 1 and 2 in Section 3 apply to both the interpreter and compiler and need be done only once.

For subsequent steps, there are .COM files that are the same as for the combined build, but with the suffixes –ICONT and –ICONC to differentiate the interpreter and compiler.

For example, to build only the interpreter, the command is

```
@ MAKE–ICONT
```

5. Icon Program Library

The Icon program library contains a variety of programs and procedures. This library not only is useful in its own right, but it provides numerous examples of programming techniques which may be helpful to novice Icon programmers. This optional component is in [V9.IPL]. It can be built by

```
$ SET NOVERIFY
$ @IPLMAKE
```

and tested by

```
$ SET NOVERIFY
$ @IPLTEST
```

The test procedure generates four DIFF outputs which should report no differences.

The Icon program library can be used with both the interpreter and the compiler. However, its use under the compiler requires command-line options in some programs to enable features that are not enabled by default when using the compiler. Because of this problem, the installation of the the Icon program library presently is supported only for the interpreter.

6. Cleaning Up

After Icon has been built and installed, type

```
$ @CLEAN
```

to remove duplicate versions, intermediate files, test results, etc.

Only [V9.BIN] is needed for running Icon programs. The library directory [V9.IPL] may be retained if desired. All other subdirectories of [V9] can be deleted unless you wish to keep the source code online.

7. Communicating with the Icon Project

Please let us know of any problems you encounter (and your solutions) so that we may make things easier for others. Our mailing address is

Icon Project
Department of Computer Science
Gould–Simpson Building
The University of Arizona
Tucson, Arizona 85721
U.S.A.

(602) 621–8448 (voice)
(602) 621–4246 (fax)

icon–project@cs.arizona.edu (Internet)

8. Acknowledgements

Chris Janton and Jan Trumbo provided assistance in the implementation of Icon for VMS.

References

1. R. E. Griswold, C. L. Jeffery and G. M. Townsend, *Version 9.0 of the Icon Programming Language*, The Univ. of Arizona Icon Project Document IPD236, 1994.
2. C. L. Jeffery, G. M. Townsend and R. E. Griswold, *Graphics Facilities for the Icon Programming Language; Version 9.0*, The Univ. of Arizona Icon Project Document IPD255, 1994.
3. R. E. Griswold, *Supplementary Information for the Implementation of Version 9.0 of Icon*, The Univ. of Arizona Icon Project Document IPD239, 1994.
4. R. E. Griswold, *The Icon Program Library; Version 9.0*, The Univ. of Arizona Icon Project Document IPD242, 1994.