Building Version 8.8 of MPW Icon

Ralph E. Griswold
Department of Computer Science, The University of Arizona

Robert J. Alexander

1. Background

The implementation of Version 8.8 of the Icon programming language is written almost entirely in C, and it is designed to be portable to a wide range of computers and operating systems. This document concerns the compilation of Icon for Macintosh under MPW (MPW Icon).

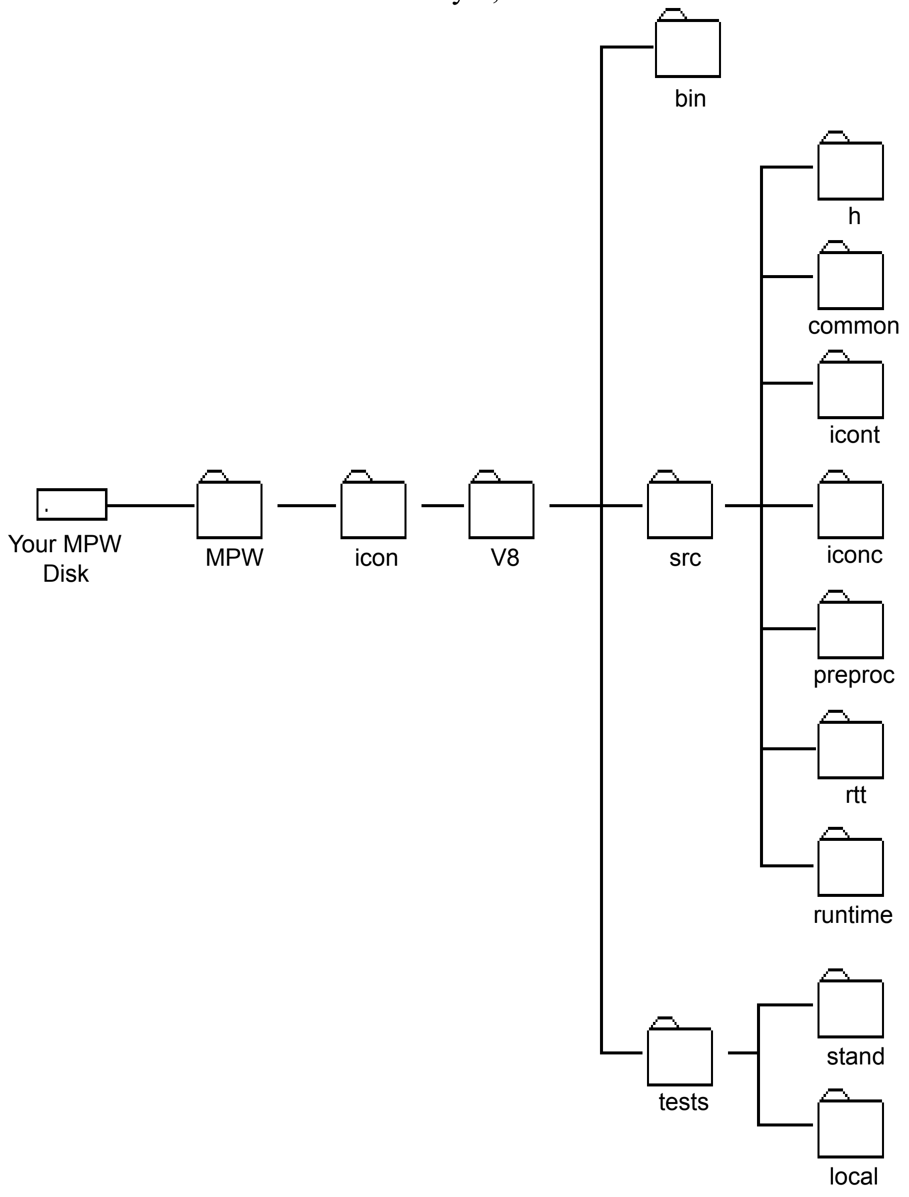
Version 8.8 of Icon requires at least a 1024K (1M) Macintosh to run. It runs all but very large programs well if the MPW Shell is given a MultiFinder partition size of 1024K (and can run small programs in as little as 600K). Of course, programs with very large code size or that accumulate large amounts of data will require that the partition size be increased. Compiling Version 8.8 of Icon requires MPW, MPW C (Version 3.0 or beyond), and the MPW assembler, as well as HFS. At least 2M of RAM is recommended for Version 3.x of MPW C — building MPW Icon Version 8.8 with less available memory has not been attempted as of this writing.

The Icon language book [1] and a technical report provided with this distribution [2] together comprise a complete description of Version 8.8 of the Icon programming language. See Section 4 for information about the implementation itself.

2. Organization of the Implementation

The source code for Icon is organized in a hierarchy. The distribution is on two 800K disks. The files are in a compressed archive format created by the StuffIt utility. The “freeware” utility, UnStuffIt, to unstuff the files is provided on the first distribution disk.

The illustration that follows shows the folder hierarchy normally used to work on MPW Icon.



If a different arrangement is used, it may be necessary to make changes to Makefiles and other supporting files.

To install the Icon source-related files:

1. Create the folders as shown above using either the Finder or MPW Shell.

2. Extract and decompress the files from their archive files (the .sit files) using UnStuffIt. Start UnStuffIt by double-clicking on its icon in the Finder's window for the distribution disk.

3. Choose UnStuffIt's Open Archive menu item.

4. Open common.sit on the source distribution disk.

5. Click the Extract button.

6. In the Extract dialog box, navigate to the destination folder for the common files, which is folder common in your Icon hierarchy. To do this, you will probably have to click the Drive button, then double click on all the folder names leading to the destination folder (in this case, MPW, icon, V8, src, and finally, common).

7. Click the Save All button, then watch and be entertained as UnStuffIt extracts the files.

8. Repeat the above steps 4 – 7 until all of the archive files have been "unstuffed" into their respective destination folders:

Archive ---->	**Folder**
common.sit	common
h.sit	h
iconc.sit	iconc
icont.sit	icont
preproc.sit	preproc
rtt.sit	rtt
runtime.sit	runtime
tests.sit	V8

2.1 Source Files

The seven source-code folders under src contain files related to the various components of Icon as follows:

common	source code for modules common to several components of Icon.
h	header files used by files in the other folders.
iconc	source code for the optimizing compiler for Icon (this program has not yet been configured for MPW and is not considered here).
icont	source code for the Icon translator/linker for the interpreter. The translator converts source-language programs to *ucode*, an assembly language for an abstract "Icon machine". The linker combines one or more ucode files into a single binary *icode* file in executable format for the Icon machine.
preproc	source code for the run-time system translator.
rtt	source code for a translator that builds files for the run-time system.
runtime	source code for the Icon run-time system, including the interpreter.

The seven source-code folders under src contain files for related to the various components of Icon as follows:

There are three executable components related to building and running Icon:

The Translator and Linker

The translator and linker, icont, performs both source code translation and linking functions. The translator is relatively straightforward. It contains a lexical analyzer, a parser, a code generator, and support routines. The translator produces printable *ucode* files. The linker is somewhat more complex than the translator. It reads ucode files and outputs binary code and data structures that are needed during execution.

The Run-Time Translator

The run-time translator, rtt, translates files for Icon's run-time system, which are written in a superset of C, to standard C. It is used only to build the interpreter.

The Interpreter

The interpreter, iconx, is large and complex. It includes code for all the operations in the Icon language. In addition, it manages storage dynamically.

The Compiler

As indicated above, there is an optimizing compiler for Icon. This compiler has not yet been configured for Icon and is not treated here. If you are interested in trying to build the compiler, contact the Icon Project as listed in Section 6.

2.2 Binary Files

The bin subdirectory in V8 is where the executable files for Icon will reside after compilation and linking.

3. Building the Icon Interpreter

Building the Icon interpreter is straightforward. Go to these subdirectories in src in the following order:

```
common
icont
rtt
runtime
```

In each of these subdirectories, do the following:

1. Enter a make command.

2. Look for resulting `make` commands in the output. If there are any, recursively select and execute these first.

3. Finally, select the resulting compilation, linking, and other commands and execute them.

As a result, the executable for the Icon interpreter will be placed in the bin subdirectory of V8.

4. Testing Icon

The folder tests contains a large battery of Icon programs and a folder stand that contains the "standard" output of running these programs. Files whose names end in .icn are the test program source files, those ending in .dat are files containing test data, and .lst files are lists of test programs to be run as a group. The lists are:

intrcoex.lst	co-expressions
intrlarg.lst	large-integer arithemtic
intrmain.lst	main features

(There are corresponding lists for the Icon compiler.)

Normally, the tests are run by using the script Test-icont, which is in the tests folder.For example,

```
Test-icont intrmain
```

tests the main features of the Icon interpreter.

Taken together, these tests run for quite a while. You may wish to redirect the output of the scripts to a file or files for easier examination.

Since the standard test results were obtained from a UNIX implementation, several differences will exist between the test output produced by MPW Icon and the standard test files (if they don't, you've done something wrong!). The differences are due to

- Differences in the time of day, date, name of the host machine, and other minor implementation differences.

- Differences in internal processing capacity and external formatting of floating point numeric output (real numbers).

All of the above differences will be reported as discrepancies when the test scripts are run. The reported differences must be scanned to determine whether they are due to the above causes or are real errors.

5. The Implementation Book

If you are interested in the larger view of the implementation of Icon, or if you are interested in modifying or extending Icon, you may want to acquire the book *The Implementation of the Icon Programming Language*. This book concentrates on the run-time system and covers data structures, the virtual machine, the interpreter, the implementation of generators, and storage management. It also contains material specifically related to making modifications to the source code.

The publication information is: *The Implementation of the Icon Programming Language*, by Griswold and Griswold, Princeton University Press, ISBN 0-691-08431-9, hardbound, 336 pages, $53.00. The book may be ordered from the Icon Project, a local bookstore, or directly from the publisher:

Princeton University Press
3175 Princeton Pike
Lawrenceville, NJ 08648

(609) 896-1344

The implementation book corresponds to Version 6.2 of the Icon source code. There have been several changes in the source code between Version 6.2 and the present version. Reports describing these changes are available free of charge from the Icon Project. Ask for IPD112 and IPD190.

6. Trouble Reports and Feedback

If you run into problems, contact the Icon Project:

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

(602) 621-2018 (voice)
(602) 621-4246 (fax)

icon-project@arizona.edu (Internet)
... uunet!arizona!icon-project (uucp)

We cannot promise to solve your problems, but we will try. We also may be able to place you in contact with other persons who are compiling Icon and who may have similar problems.

Please also let us know of any suggestions for improvements to the compilation process or its documentation.

Acknowledgments

Many persons have been involved in the implementation of Icon. Bob Alexander did most of the work to adapt Icon for use on the Macintosh under MPW.

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, second edition, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

2. R. E. Griswold, Clinton L. Jeffery, Gregg M. Townsend, and Kenneth Walker, *Version 8.8 of the Icon Programming Language*, The Univeristy of Arizona, technical report IPD210, 1992.