Supplementary Information for the Implementation of Version 9 of Icon

Ralph E. Griswold

Department of Computer Science, The University of Arizona

1. Introduction

The Icon programming language [1] is fairly stable now, although the implementation of Version 9 of Icon contains major changes from preceding versions. The most notable change is the use of RTL [2] for writing the run-time system. The run-time system now also supports both the compiler and interpreter, with conditional code as appropriate.

RTL is a superset of C and contains constructions for describing the abstract semantics of Icon that are needed by the Icon compiler. RTL also has constructions for expressing type checking, conversion, and selection in ways that are close to the semantics of Icon.

Functions and operations written in RTL look substantially different from their former C versions, especially in their headers. However, most of the actual code is the same or similar to that for earlier versions.

Some aspects of data representation and the way parts of the run-time system work are new with Version 9. These changes are summarized in the next section.

Several documents are be useful in understanding the implementation of Icon. The implementation of Version 6 is described in considerable detail in a book [3]. Although much has changed since Version 6, the material in this book still is sound and is the best source of information about the basic aspects of the implementation. A supplementary report [4] describes changes made between Version 6 and 8 and also lists corrections to [3]. The Icon compiler is described in [5] and RTL is described in [2].

2. Significant Changes in Version 9

The information that follows assumes a basic knowledge of the implementation of Icon and the terminology used in it.

File Hierarchy

The organization of the files for the implementation of Icon and its related components has changed in Version 9. See the appendix at the the end of this report

Type Checking, Conversion, and Selection

As mentioned above, RTL contains constructions for dealing with Icon type checking, conversion, and selection. The code for these operations is cast differently than before. See [2].

Descriptor Layout

Some descriptors in Version 9 are different from earlier implementations. There is now a flag in the d-word of descriptors that contain type codes (as opposed to string qualifiers and descriptors for plain variables).

The trapped-variable flag is no longer used to distinguish plain variables from special ones, such as for keywords. Instead, the latter are represented by type codes.

Keywords

Icon keywords are now implemented by C functions. There are no longer trapped-variable blocks for keywords to which assignment can be made.

Tended Pointers and Storage Allocation

RTL provides a tended declaration for values that may contain pointers to allocated storage that must be processed during garbage collection. Predictive need is no longer used to assure that adequate free space exists before an allocation request. Instead, an allocation request causes a garbage collection if there is not adequate free space.

Multiple Storage Regions

Storage regions do not expand. Additional storage regions are allocated if garbage collection fails to provide enough free space to satisfy an allocation request.

Translator Table Expansion

The tables used by the Icon translator and linker (icont) now are expanded automatically as necessary. The former –S option is no longer used. As a side effect of this change, the sizes of procedures are no longer given as part of the advisory output of icont.

Removal of Unreferenced Globals

The linker now removes unreferenced globals (including procedures and record constructors).

3. Summary of Changes to the Implementation Book

The list that follows indicates recent changes to the implementation of Icon that affect the material in [3]. *Note:* Almost all the code segments shown in the book have been changed in some way since Version 6.

Chapter 1:	Still valid.
Chapter 2:	See [1] and [6] for recent changes to the Icon language itself
Chapter 3:	Still valid.
Chapter 4:	See [4] for changes to the handling of integers. See Section 2 above concerning the representation of descriptors and trapped variables.
Chapter 5:	Mostly still valid. See [4] for changes related to csets.
Chapter 6:	See [4] for changes to the method for adding elements to lists and the use of pointers in place of descriptors to link blocks.
Chapter 7:	See [4] for a description of dynamic hashing that now is used for sets and tables, as well as for the use of pointers to link blocks.
Chapter 8:	Mostly still valid.
Chapter 9:	See [4] for improvements to the handling of scanning environments.
Chapter 10:	See [4] for improved handling of co-expression activation.
Chapter 11:	Mostly still valid. See [4] for fixed-region implementations and Section 2 above for multiple fixed regions. See Section 2 above concerning tended pointers and the removal of predictive need.
Chapter 12:	See [2] for the handling of type conversion and errors.
Appendix A:	In addition to the changes in Chapters 4, 5, and 6 mentioned above, there are numerous minor changes in block layouts.
Appendix B:	Keywords are now represented by names instead of numbers.
Appendix C:	Still valid.
Appendix D:	See Section 2 above regarding the file hierarchy. See [2] for information on adding functions, operations, and new data types to Version 9. See the source code itself for current macro definitions.

Appendix E: Still valid except for extensions that require the addition of new data types. Some of the projects listed have been done since Version 6.

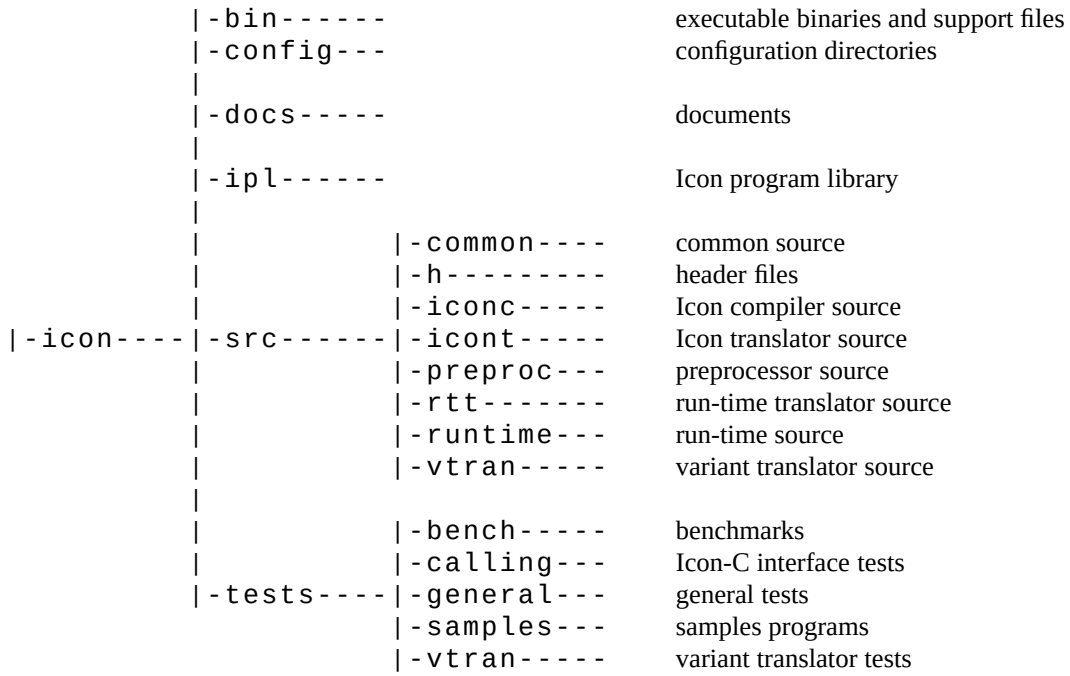
Acknowledgements

Many persons have contributed to the implementation of Icon. Clint Jeffery and Gregg Townsend collaborated with the author on the implementation of Version 9.

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, 1990.
2. K. Walker, *The Run-Time Implementation Language for Icon*, The Univ. of Arizona Icon Project Document IPD261, 1994.
3. R. E. Griswold and M. T. Griswold, *The Implementation of the Icon Programming Language*, Princeton University Press, 1986.
4. R. E. Griswold, *Supplementary Information for the Implementation of Version 8 of Icon*, The Univ. of Arizona Icon Project Document IPD112, 1995.
5. K. Walker, *The Implementation of an Optimizing Compiler for Icon*, The Univ. of Arizona Tech. Rep. 91-16 . Aug. 1991.
6. R. E. Griswold, C. L. Jeffery and G. M. Townsend, *Version 9.1 of the Icon Programming Language*, The Univ. of Arizona Icon Project Document IPD267, 1995.

Appendix — Icon Hierarchy



In some cases there is additional sub-structure not shown here.