Icon Programs and Procedures
for
Macintosh Programmer's Workshop

Robert J. Alexander

IPD229

The distribution disk for MPW Icon Version 8.10 contains a mini-library of programs, procedures, and MPW scripts that serve a dual purpose: (1) they are examples of Icon code, and (2) they are useful MPW utilities. Building the tools and experimenting with them will help to provide insight into the capabilities of MPW Icon and a feel for using Icon in the MPW environment.

These files are provided on an as-is basis, with no formal support provided as such. They are, however, reasonably error-free — we would like to hear about any suggestions or bugs.

The files in the MPW Helpers folder are of the following types:

- MPW tools written in Icon
- MPW Shell scripts to construct custom menu items that increase the utility of the Icon tools
- Icon procedures you can use in your own tools

The files contain sufficient documentation to use them. This document provides an overview explaining how the parts fit together. Some of the tools described in this document are now part of the Icon Program Library, a sizeable collection of programs and procedures written in Icon . The Library is available from the Icon Project, University of Arizona.

The Tools

The tools provided are

colm.icn	a tool that formats lines of standard input into columns to be displayed in a window or printed.
FixIncludes.icn	a tool that changes the filename formats int #include directive from UNIX to Mac format.
FixMake.icn	a tool that assists in converting a UNIX Makefile to MPW Make format.
format.icn	a tool that “word wraps” input text to conform to given specifications. It

also formats program comments, preserving any required comment sequence a the start of each line.

icom.icn	a tool that “comments-out” or “uncomments-in” a section of Icon code — for use with the Encomment and Decomment menu items for operating on Icon programs in MPW windows.

localx.icn	a utility to report undeclared local identifiers in Icon programs (used in conjunctions with the locals script).

IconProg.icn	a filter that turns an Icon program fragment or whole program into input suitable for translation by icont (specifically for use in conjunction with one of the supplied script: DoIcon).

macsf.icn	a tool that creates an aesthetically pleasing list of filenmes separated by suffix.

SegAlloc.icn	a tool that generates linker commands to allocate program modules to segments so that no segment exceeds the maximum for a segment.

The tools are documented in their source files, and can be used as MPW commands. They are put to use in the scripts and menus that are described below. Note that these tools are not limited to the Macintosh environment — the author has found them useful in MS-DOS and UNIX environments (especially format in conjunction with vi or emacs for formatting program comments).

The MPW Shell Menus

The UserStartup scripts create custom menu items for your MPW environment. They are

UserStartup•IconMenu
UserStartup•Format

These files are intended to be incorporated into the startup scenario of MPW. With MPW 3.0, all that is necessary to put them into service is to copy the files into your MPW folder. They will automatically be executed when MPW starts up. With older versions of MPW, you will have to explicitly execute them by inserting explicit Execute commands into your UserStartup file.

Each of the script files allows you to specify which menu its items will appear under. They can be either under an existing menu, or a new menu name can be specified. The menu name is specified within the script text, and is changed by editing the script.

Icon Menu Items

UserStartup•IconMenu creates four menu items specifically for use with the Icon Programming Language. They are

Do Icon allows an Icon program or sequence of expressions to be executed directly from any MPW window, usually the Worksheet. It is useful for quickly executing small Icon segments, often just to experimentally determine what Icon will do in certain circumstances. Simply select the text of an Icon program that has been typed into a window and choose the Do Icon menu item or, more expediently, hit command-I. (Like the MPW Shell Enter key, if no text is selected the whole line will be evaluated). There is no way to pass command-line arguments to such a program.

Write Empty Procedure creates an Icon program template in the active MPW window, usually used quickly creating a program structure for the Do icon menu item:

```
procedure main()
  |
end
```

The | indicates that the insertion point is positioned for immediate text entry after choosing Write Empty Procedure.

List Undeclared creates a local declaration for each undeclared local identifier in an Icon procedure or program. To use it, select an entire Icon procedure in an MPW window and choose the List Undeclared menu item. The local line will appear below the selected code. Cut and paste the local line into the procedure body. The List Undeclared utility is unaware of globals you have declared, and might include them in the local declaration. This will probably cause your program to malfunction — you must manually remove global variable names that accidentally appear in the created local declaration.

Show ucode works similarly to Do Icon it that it operates on text in a MPW window. Instead of executing an Icon program, Show ucode displays its ucode file. An often overlooked fact is that the unlinked “object” files output by the Icon translator (icont) are human-readable text files that resemble assembly language source files. The target machine for ucode files is the Icon virtual machine, emulated by the Icon interpreter iconx. Often it is interesting to see the ucode that is generated by various Icon source code constructs. Show ucode provides a convenient mechanism to view generated ucode. Note that since ucode files are text files, the MPW Shell editor can also be used to view ucode files.

Mark Icon Procedures creates MPW Shell marks for every procedure and record declared in the Icon program file in the active MPW window. MPW marks allow you to move quickly to any procedure or record definition by choosing its name from the MPW Shell’s Mark menu. Additionally, marks are created for all global and link declaration lines, so the Mark menu will contain a complete index of your Icon program. If your program composed of multiple files, this facility in conjunction with the Mark Browser make it easy to navigate large programs.

Encomment and Decomment menu items comment out or decomment in the selected range of Icon program lines in an MPW editor window.

Output Option... presents a dialog box (a list box, really) to allow you to choose between normal window output and fast output. The default, normal output is usually adequate, but if large amounts of output are written to a window a significant speedup is obtained by selecting the faster option. Nothing is free, though — when the fast option is used output to different files (like &output and &errout) which are written to the same window will not be ordered strictly chronologically; i.e. they get a bit jumbled.

Trace Option... presents a list box to allow you to turn run-time procedure tracing on and off.

Format Menu Items

UserStartup•Format creates four menu items for performing word-wrap operations on text in MPW windows.

To use any of the Format menu items, first select the range of text you want wrapped, then select the menu item. The Format items operate on mono-spaced fonts only, such as Monaco and Courier (the MPW default font is Monaco).

Format Paragraph simply wraps the selected text according to the current word-wrap parameters (parameters are discussed below).

Format Comments will word-wrap a range of program comments, leaving the comment characters intact and in the proper place. Format comments will work with comment text that has its "comment indicator" string at the beginning of each comment line (such as Icon).

```
#  This is an
#  Icon program.
```

will work, as will

```
//  This is a
//  C++ program.
```

This will not work

```
{ This is a }
{ Pascal program }
```

but this will

```
/*
 *  This is a
 *  C program
 */
```

as long as only the second and third lines are selected.

The comment marker can be preceded by white space. When selecting comment text to wrap, make sure that the first line actually contains comment text, so that the format utility can determine the format of text to follow. For example, given

```
#
#  getopt() --  The getopt procedure…
```

the text selection should start at the second line, not the first.

Another feature of Format Comments is helpful when inserting a large amount of new text into a program comment: it is not necessary to include the “comment indicator” string in lines following the first (format establishing) line. If Format Comments is applied to the following text

```
#  This is a comment string
containing a whole lot of
new text and I was too
lazy to put a # at the
start of each line
```

it will come out something like this

```
#  This is a comment string containing a whole lot of new
#  text and I was too lazy to put a # at the start of each line
```

Justify Paragraph works similarly to Format Paragraph except that the text is expanded (by random insertion of space characters) so that it is flush to both the left and right margins. The last line of the justified output is not expanded.

Justify Last line Too works like Justify Paragraph except that all output is expanded including the last line.

Note that justified text can be un-justified or re-justified simply by applying any of the other word-wrap operations. Also, the MPW Undo operation will unwrap wrapped text if it is performed immediately after the wrap operation.

Word wrapping parameters (line width, tab settings, and other options) are controlled by MPW Shell environment variables. They are set to default values in the UserStartup•Format script. The defaults can be changed by editing the script, or they can be temporarily overridden by resetting the variable values. Refer to the script for details.

The Scripts

lz (el-zee) is a MPW Shell script to provide a command similar to the UNIX ls command (directed to a console). It outputs file names in columnar fashion, with file names arranged alphabetically vertically within columns. It uses the colm tool to arrange the columns, and does so in a quite sophisticated way. The width of each column is adjusted individually depending on its contents, and the total width is adjusted to the current size of the active window. The output format is superior to the MPW files -m n command. The disadvantage: speed.

The lz command is useful when a large number of files are to be listed, and especially if the list is to be printed. There are few other automated methods to fit as many file names, readably formatted, into a smaller space! lz also works well if the -r option is used (to list files in all included directories) — the directory name headings and the file names of each directory are kept separate and are listed in a pleasantly readable, compact format.

The sf command lists files, separated by filename suffix. Like lz, the listing format is compact.

Locals determines the undeclared local identifiers in an Icon procedure. It is used in conjunction with the List Undeclared menu item.

The provided script, MarkIcon, is used in conjunction with the Mark Icon Procedures menu item.

The Procedures

The procedures contain programmer's documentation within their text. They are

options.icn
shquote.icn
filename.icn
colmize.icn
hex.icn
isort.icn
fmacunix.icn

The options procedure provides a convenient mechanism for processing command-line options.

The mpwquote procedure (in shquote.icn) properly quotes strings (if necessary) that are to be recognized as single words within the MPW command language. It performs the same transformation as does the quote MPW tool. Since Icon can be put to good use in generating text to be executed by the MPW Shell, mpwquote often comes in handy.

suffix (in filename.icn) breaks a file name into its base part and suffix parts and returns the result as a two-member list. suffix("suffix.icn") returns ["suffix","icn"]. suffix("xxx") returns ["xxx",&null], but suffix("yyy.") returns ["yyy",""].

colmize does most of the work for the utility colm, described in this document. The colmize procedure is useful in a variety of programs — see the source file for detailed documentation.