VOLUME III—INPUT/OUTPUT

Chapter 24 Streams and Files

The Xerox Common Lisp file system supports multiple streams open simultaneously on the same file. This is an *incompatible change* to the semantics of Interlisp-D. You may have to modify old programs if they have not followed the guidelines listed in Sec 24.5 of the *Interlisp-D Reference Manual*. Some of the implications of this change for Interlisp programs are described below.

In prior releases of Interlisp-D, the system treated the *name* of an open file as a synonym for the *stream* open on the file. This meant that only one stream could be open at any time on a given file. In the Lyric release, a file name is no longer a unique name for an open stream. Thus, file names are no longer acceptable as the file/stream argument to any input/output or file system function that operates on an open stream (**READ, PRINT, CLOSEF**, **COPYBYTES**, etc). The only non-stream values acceptable as stream designators are the symbols **NIL** and **T**, designating the primary and terminal input/output streams. An attempt to use a litatom, even a "full file name," as a stream designator signals the error "LITATOM 'streams' no longer supported." Strings no longer designate an input stream whose source is the string itself—programs should call **OPENSTRINGSTREAM** instead, or use the comparable Common Lisp forms, such as **CL:WITH-INPUT-FROM-STRING**.

The functions **OPENFILE** and **OPENSTREAM** are now synonymous—both return a stream instead of a "full file name." The functions **INPUT** and **OUTPUT** also return streams. One exception to this is that **INPUT** and **OUTPUT** return **T** in the case where the primary input or output stream was previously directed to the terminal. However, this special behavior is for the Lyric release only; we recommend that you convert old code that depended on testing **(EQ (OUTPUT) T)**. Note that the values of the variables ***STANDARD-INPUT*** and ***STANDARD-OUTPUT*** are always streams, even if they are directed to the terminal.

The function **FULLNAME** can be used to obtain the name of a stream. For your convenience, the print syntax of streams now includes the name of the stream (if to a file) and its access (input, output, etc.). Functions, such as **UNPACKFILENAME**, that manipulate file names generally accept a stream as well, extracting the name of the file from the stream.

INFILEP still returns a full file name, as it is merely recognizing a file, not opening a stream to it. Programmers should be wary of code that subsequently tries to use the value of **INFILEP** as a stream argument. And, of course, the *FILENAME* argument to **OPENSTREAM** is still a name (a symbol or string), not a stream. **OPENSTREAM** also accepts a Common Lisp pathname as its *FILENAME* argument.

The function **CLOSEALL** is no longer implemented. The function **OPENP** returns **NIL** when passed a file name (or anything else but an open stream). However, for the Lyric release, **(OPENP NIL)** still returns a list of all streams open to files.

The functions **GETFILEINFO** and **SETFILEINFO** can still be given either an open stream or a file name. However, in the latter case, the request refers to the file, not to any stream open on the file. Thus, requesting the value of the attribute **LENGTH** for a file name may return a different value than requesting the value of the attribute **LENGTH** for a stream currently open on the file. **GETFILEINFO** returns **NIL** if given a file name and an attribute that only makes sense for streams (e.g., **ACCESS**, **ENDOFSTREAMOP**).

There is no difference between Common Lisp and Interlisp streams. A stream opened by an Interlisp function can be passed as argument to a Common Lisp input/output function, and vice versa.

Even though multiple streams per file are supported, the streams must still obey consistent access rules. That is, if a stream is open for output, no other streams on that file can be opened. It is not possible to **RENAMEFILE** or **DELFILE** a file that has *any* open stream on it.

The RS-232 or TTY ports are inherently single-user devices (rather than real files) thus, multiple streams cannot be open simultaneously on RS-232 or TTY.

Section 24.15 Deleting, Copying, and Renaming Files

(III:24.15)

The support of multiple streams per file now makes it possible to use **COPYFILE** without worrying about there being other readers of the file, in particular even when there is already a stream open on the file for sequential-only access (a case that failed in prior releases). Of course, **COPYFILE** cannot be used if the file already has an *output* stream open.

Chapter 25 Input/Output Functions

Variables Affecting Input/Output

There are several implicit parameters that affect the behavior of the input/output functions: the numeric print base, the primary output file, etc. In Common Lisp, these parameters are controlled by binding special variables. In Interlisp they are controlled by a functional interface; e.g., an output expression is wrapped in `(RESETFORM (RADIX 8) --)` to cause numbers to be printed in octal.

Where the input/output parameters in Common Lisp and Interlisp have essentially the same semantics, they have been integrated in

Lisp. That is, binding the Common Lisp special variable and calling the Interlisp function are equivalent operations, and they affect both Interlisp and Common Lisp input/output. However, it is considerably more efficient to bind a special variable than to call a function in a **RESETFORM** expression. In addition, binding a variable has only a local effect, whereas calling a function to side-effect the input/output parameters can also affect other processes. Thus, you are encouraged to use special variable binding to change parameters formerly changed via functional interface.

All of these variables are accessible in both the Common Lisp and Interlisp packages, so no package qualifier is required when typing them.

These parameters are as follows:

PRINT-BASE* vs RADIX** Binding ***PRINT-BASE to an integer *n* from 2 to 36 tells the printing functions to print numbers in base n. This is equivalent to (**RADIX** *n*). Note: this variable should not be confused with ***PRINT-RADIX***, another Common Lisp variable that controls whether Common Lisp functions include radix specifiers when printing numbers.

STANDARD-INPUT* vs INPUT** Binding ***STANDARD-INPUT to an input stream specifies the stream from which to read when an input function's stream argument is **NIL** or omitted. Evaluating ***STANDARD-INPUT*** is the same as evaluating (**INPUT**), except that (**INPUT**) returns **T** if the primary input for the process is the same as the terminal input stream (this compatibility feature is for the Lyric release only).

STANDARD-OUTPUT* vs OUTPUT** Binding ***STANDARD-OUTPUT to an output stream specifies the stream to which to print when an output function's stream argument is NIL or omitted. Evaluating ***STANDARD-OUTPUT*** is the same as evaluating (**OUTPUT**) except that (**OUTPUT**) returns **T** if the primary output for the process is the same as the terminal output stream (this compatibility feature is for the Lyric release only).

PRINT-LEVEL* & *PRINT-LENGTH* vs PRINTLEVEL** Binding ***PRINT-LEVEL to a positive integer *a* and ***PRINT-LENGTH*** to a positive integer *d* is equivalent to calling (**PRINTLEVEL** *a d*). Binding either variable to **NIL** is equivalent to specifying a value of -1 for the corresponding argument to **PRINTLEVEL,** i.e., it specifies infinite depth or length. Note that in Interlisp, print level is "triangular"—the print length decreases as the depth increases. In Common Lisp, the two are independent. Thus, although print level for both Interlisp and Common Lisp is controlled by a common pair of variables, the Interlisp and Common Lisp print functions interpret them (specifically ***PRINT-LENGTH***) slightly differently. In addition, Interlisp observes print level only when printing to the terminal, whereas Common Lisp observes it on all output.

READTABLE* vs SETREADTABLE** Binding ***READTABLE to a readtable specifies the readtable to use in any input/output function with a readtable argument omitted or specified as **NIL**. Evaluating ***READTABLE*** is the same as evaluating (**GETREADTABLE**). There is no longer a "NIL" or "T" readtable in Interlisp. See the discussion of readtables for more details.

Although the binding style is to be preferred to the **RESETFORM** expression, one difference should be noted with respect to error checking. In a form such as

(**RESETFORM** (RADIX *n*)
some-printing-code)

the value of *n* is checked immediately for validity, and an error is signalled if *n* is not an integer between 2 and 36. However, in

(**LET** ((***PRINT-BASE*** *n*))
some-printing-code)

there is no error checking at the time of the binding; rather, an error will not be signalled until the code attempts to print a number.

Similarly, the values of ***STANDARD-INPUT*** and ***STANDARD-OUTPUT*** must be actual streams, not the values that print functions coerce to streams, such as **NIL**, **T** or window objects.

Integration of Common Lisp and Interlisp Input/output Functions

Common Lisp and Interlisp have slightly different rules for reading and printing, regarding such things as escape characters, case sensitivity and number format. Each has two kinds of printing function, an escaped version (intended for reading back in) and an unescaped version. In order that Common Lisp and Interlisp programs can more freely intermix, Xerox Lisp isolates most of the reading/printing differences in the readtables used by both languages, rather than in the functions themselves. The exact rules have been chosen as a reasonable compromise between backward compatibility with Interlisp and integration with Common Lisp. This section outlines the details of this integration.

In what follows, the phrase "the readtable" generally refers to the readtable in force for the read or print operation being discussed. Specifically, this means an explicit readtable (other than NIL or T) passed as readtable argument to an Interlisp function, or else the current binding of ***READTABLE***. See the section on readtables for more details.

Section 25.2 Input Functions

The functions **IL:READ** and **CL:READ**, given the same readtable, interpret an input in exactly the same way. That is, the functions obey Common Lisp syntax rules when given a Common Lisp readtable, and Interlisp syntax when given an Interlisp readtable. Thus, the principal difference between the two is in the functionality provided by **CL:READ**'s extra arguments: end of file handling and the ability to specify that the read is recursive, which is mostly important when reading input containing circular structure references (the ## and #= macros). See *Common Lisp, the Language* for details of **CL:READ**'s optional arguments.

There is one further difference between **IL:READ** and **CL:READ**, in the handling of the terminating character. If the read terminates on a white space character, **CL:READ** consumes the character, while **IL:READ** leaves the character in the buffer, to be read by the next input operation. Thus, **IL:READ** is equivalent to **CL:READ-PRESERVING-WHITESPACE**. This difference is so that Interlisp

code that calls (**READC**) following a (**READ**) of a symbol will behave consistently between Koto and Lyric.

The Interlisp function **SKREAD** now defaults its readtable argument to the current readtable, viz., the value of ***READTABLE***, rather than **FILERDTBL**. This makes it consistent with all the other input functions, and is usually the correct thing, especially with the new reader environments used by the File Manager, but it is an incompatible change from Koto. **SKREAD** is also now implemented using Common Lisp's ***READ-SUPPRESS*** mechanism, which means that, unlike in Koto, it does something reasonable when it encounters read macros.

In the Medley release, reading in bitmaps from files is significantly faster.

Section 25.3 Output Functions

The discussion here is limited to the four basic printing functions: the unescaped and escaped Interlisp printing functions (**IL:PRIN1**, **IL:PRIN2**) and the corresponding Common Lisp functions (**CL:PRINC**, **CL:PRIN1**). All other print functions ultimately reduce to these. For example, **IL:PRINT** calls **IL:PRIN2**; **CL:FORMAT** with the ~S directive performs a **CL:PRIN1**.

IL:PRIN1 is essentially unchanged from previous releases. It uses no readtable at all, so is unaffected by the value of ***READTABLE***. It can be thought of as implicitly using the INTERLISP readtable.

Roughly speaking, **IL:PRIN2** and **CL:PRIN1** behave the same when given the same readtable. In particular, they both produce output acceptable to either **READ** function given the same readtable. Their minor differences are listed below.

CL:PRINC behaves like **CL:PRIN1**, except that it never prints escape characters or package prefixes. Thus, unlike **IL:PRIN1**, it *is* affected by the value of ***READTABLE***.

For the benefit of user-defined print functions, **IL:PRIN2** and **CL:PRIN1** bind ***PRINT-ESCAPE*** to **T,** while **IL:PRIN1** and **CL:PRINC** bind it to **NIL**. Thus, the print function can always examine ***PRINT-ESCAPE*** to decide whether it needs to print objects in a way that will read back correctly (Common Lisp user print functions may want to use **CL:WRITE** to pass ***PRINT-ESCAPE*** through transparently; Interlisp functions should choose **IL:PRIN2** or **IL:PRIN1** appropriately).

Printing Differences Between IL:PRIN2 and CL:PRIN1

There are two respects in which the Interlisp print functions (both **IL:PRIN1** and **IL:PRIN2**) differ from the Common Lisp ones, independent of readtable:

Line Length. The Interlisp functions respect the output stream's line length, while the Common Lisp functions all ignore it (they never insert newline characters when output approaches the right margin).

Print Level. The Interlisp functions respect the print level variables only when printing to the terminal (unless **PLVLFILEFLG** is true, see the *Interlisp-D Reference Manual*) or when printing with a Common Lisp readtable, whereas the Common Lisp functions respect them on *all* output.

Internal Printing Functions

Interlisp has several functions (e.g., **NCHARS**, **STRINGWIDTH**, **CHCON**, **MKSTRING**) that operate on the "prin1 pname" of an object, or optionally on its "prin2 pname" when given an extra flag and readtable as arguments. These functions are essentially unchanged in Lyric.

In terms of the discussion above, the "prin1 pname" of an object continues to be the characters that would be produced by a call to **IL:PRIN1** at infinite print level and line length, and with ***PRINT-BASE*** bound to 10 (unless **PRXFLG** is true, see *Interlisp-d Reference Manual*). The "prin2 pname" of an object is the list of characters that would be produced by a call to **IL:PRIN2** (or **CL:PRIN1**) using the specified readtable (or ***READTABLE*** if none is given), again at infinite print level and line length.

Exception: the function **STRINGWIDTH** computes the width of the expression as if it were printed at the current ***PRINT-LEVEL*** and ***PRINT-LENGTH***.

Printing Differences between Koto and Lyric

The Common Lisp and Interlisp printing functions use the same strategy for escaping characters in symbol names. Because of this, symbols may print differently in Lyric than they did in Koto, for two reasons: the use of the Common Lisp multiple escape character, and the escaping of numeric print names. Although the appearance is different, the functionality is the same—symbols are still printed in a way that allows them to be correctlyread.

Roughly speaking, the multiple escape character is used to escape symbol names that would require two or more single escape characters. Thus, for example, a symbol that printed as `%(OH% NO%)` in Koto will print in Lyric as `|(OH NO)|`. However, in the old readtables that lack a multiple escape character (e.g., OLD-INTERLISP-T), the single escapes are still used. Multiple escapes are also used to print a symbol containing lower-case letters when the readtable is case-insensitive, e.g., `|Small|` in a Common Lisp readtable. Keep in mind also that some additional characters are now "special", e.g., colon in all new readtables, semi-colon in Common Lisp. Thus, the typical NS FIle "full name" will be printed with the multiple escape character.

Since it is now possible to create symbols that have "numeric" print names, such symbols must be printed with suitable escape characters, so that on input they are not misinterpreted as numbers. For example, the symbol whose print name is "1.2E3" is printed as `|1.2E3|`. In read tables lacking a multiple escape character, the single escape character is used instead, e.g.,

`%1.2E3` in the old Interlisp T readtable. A print name is considered numeric according to the definition of "potential number" in Common Lisp (p. 341). Even if such a symbol is not readable in the current system as a number, it still needs to be escaped in case it is read into another system that treats it as numeric (either another Common Lisp implementation, or a future implementation of Xerox Lisp). Thus, some old Interlisp symbols now print escaped where they didn't in Koto; e.g., the **PRINTOUT** directive `|.P2|` is a potential number.

Bitmap Syntax

Bitmaps are printed in a new syntax in Lyric. When ***PRINT-ARRAY*** is **NIL** (the default at top level), a bitmap prints in roughly the same compact form as previously:

`#<BITMAP @` *nn,nnnnnn*`>`

If ***PRINT-ARRAY*** is **T**, a bitmap prints in a manner that allows it to be read back:

`#*(`*Width Height* [*BitsPerPixel*]`)` *XXXXXXXXX*...

Width and *Height* are measured in pixels; *BitsPerPixel* is supplied for bitmaps of other than the default of 1 bit per pixel. Each *X* represents four bits of a row of the bitmap; the characters `@` and `A` through `O` are used in this encoding. Thus, there are 4*⌈*Width***BitsPerPixel*/16⌉ *X*'s for each row.

MAKEFILE binds ***PRINT-ARRAY*** to **T** so that bitmaps print readably in files. E.g., if the value of FOO is a bitmap, the command (VARS FOO) dumps something like

```
(RPAQQ FOO #*(10 10)ADSDKJFDKJH...)
```

Note that with this new format, bitmaps are readable even inside a complex list structure. This means you need no longer use the **UGLYVARS** command when dumping a list containing bitmaps if the bitmaps were previously the only "unprintable" part of the list.

Section 25.8 Readtables

(III:25.34)

The input/output syntaxes of Common Lisp and Interlisp differ in a few significant ways. For example, Common Lisp uses "\" as the escape character, whereas Interlisp uses "%". Common Lisp input is case-insensitive (lower-case letters are read as upper-case), whereas Interlisp is case-sensitive. In Xerox Lisp, these differences are accommodated by having different readtables for the two dialects. Which syntax is used for input or output depends on which readtable is being used (either as an explicit argument to the read/print function or by being the "current" readtable).

Interlisp readtables have been extended to include features of Common Lisp syntax. There is a registry of named readtables to make it easier to choose a readtable. The default Interlisp readtable has been modified to make it look a little closer to Common Lisp.

Also, Lisp has a new mechanism for maintaining read/print consistency. This means that even though Koto files may contain characters that are now "special", such as colon, you need make no changes to them—the File Manager knows how to load them correctly. See *IRM*, Chapter 17, Reader Environments and File Manager for details of this mechanism.

Differences Between Interlisp and Common Lisp Read Tables

When reading or printing, the readtable dictates the syntax rules being followed. As in past releases, the readtable indicates which characters must be escaped when printing a symbol (and ***PRINT-ESCAPE*** is true). In addition, in Lyric the readtable specifies such things as which escape character to use (\ or %) and the package delimiter to print on package-qualified symbols. The less obvious rules are detailed below.

Printing numbers. Numbers are always printed in the current print base (the value of the variable ***PRINT-BASE***, or equivalently the value of (**RADIX**). Whether to print a radix specifier is determined by the readtable. In Common Lisp, a radix specifier is printed exactly when the value of ***PRINT-RADIX*** is true. The radix specifier is a suffix decimal point in base 10, or a prefix using # for other bases. In Interlisp, a radix specifier is printed if the base is not 10, ***PRINT-ESCAPE*** is true, and the number is not less than the print base. The radix specifier is a suffix Q for octal, or a prefix using # (or | in old Interlisp readtables) for other bases. There is no decimal radix specifier.

Reading numbers. In Common Lisp, numbers are read in the current value of ***READ-BASE***, and a trailing decimal point is interpreted as a decimal radix specifier. In Interlisp, numbers are always read in base 10, and trailing decimal point denotes a floating-point number.

Case conversion. In a case-insensitive readtable (as Common Lisp is), the value of ***PRINT-CASE*** controls how upper-case symbols are printed, and lower-case letters in symbols are escaped. In a case-sensitive readtable (as Interlisp is), ***PRINT-CASE*** is ignored, so all letters in symbols are printed verbatim. ***PRINT-CASE*** is also ignored by **PRIN1**, which implicitly uses an Interlisp readtable.

Ratios. The character slash (/) is interpreted as the ratio marker in all readtables except old Interlisp readtables (specifically, those whose **COMMONNUMSYNTAX** property is **NIL**). This is so that old files containing symbols with slashes are not misinterpreted as ratios. Thus, the characters "1/2" are read in new readtables as the ratio 1/2, but in old Interlisp readtables as the 3-character symbol |1/2| (| is the multiple-escape character, see below). Ratios are printed in old Interlisp readtables in the form |.(/ numerator denominator).

Packages. Symbols are interned with respect to the current package (the binding of ***PACKAGE***) except in old Interlisp readtables (specifically, those whose **USESILPACKAGE** property is **T**), where symbols are read with respect to the INTERLISP package, independent of the binding of ***PACKAGE***. Again, this is a backward-compatibility feature: Interlisp had no package system,

so programmers were not confronted with the need to read and print in a consistent package environment.

Print Level elision. When ***PRINT-LEVEL*** or ***PRINT-LENGTH*** is exceeded, the printing functions denote elided elements and elided tails by printing "&" and "--" with an Interlisp readtable, or "#" and "..." with a Common Lisp readtable.

Section 25.8.2 New Readtable Syntax Classes

The following new syntax classes are recognized by **GETSYNTAX** and **SETSYNTAX**:

MULTIPLE-ESCAPE This character inhibits any special interpretation of all characters (except the single escape character) up until the next occurrence of the multiple escape character. In Common Lisp and in the new Interlisp readtables this character is the vertical bar ("|"). For example, |(a)| is read as the 3-character symbol "(a)"; |x\|y\\z| is read as the 5 character symbol "x|y\z".

There is no multiple escape character in the old Interlisp readtables.

PACKAGEDELIM This character separates a package name from the symbol name in a package-qualified symbol. In Common Lisp and in the new Interlisp readtables this character is colon (":"). In the old Interlisp readtables the package delimiter is control-↑ ("↑↑"); it is not intended to be easily typed, but exists only to have a compatible way to print package-qualified symbols in obsolete readtables. See *Common Lisp, the Language* for details of package specification.

Additional Readtable Properties

Read tables have several additional properties in Xerox Lisp. These are accessible via the function **READTABLEPROP**:

(READTABLEPROP *RDTBL PROP NEWVALUE*) [Function]

Returns the current value of the property *PROP* of the readtable *RDTBL*. In addition, if *NEWVALUE* is specified, the property's value is set to *NEWVALUE*. The following properties are recognized:

NAME The name of the readtable (a string, case is ignored). The name is used for identification when printing the readtable object itself, and can be given to the function **FIND-READTABLE** to retrieve a particular readtable.

CASEINSENSITIVE If true, then unescaped lower-case letters in symbols are read as upper-case when this readtable is in effect. This property is true by default in Common Lisp readtables and false in Interlisp readtables.

COMMONLISP If true, then input/output obeys certain Common Lisp rules; otherwise it obeys Interlisp rules. This is described in more detail in the section on reading and printing. Setting this property to true also sets **COMMONNUMSYNTAX** true and **USESILPACKAGE** false.

COMMONNUMSYNTAX If true, then the Common Lisp rules for number parsing are followed; otherwise the old Interlisp rules are used. This affects the interpretation of "/" and the floating-point exponent specifiers "d",

"f", "l" and "s". It does not affect the interpretation of decimal point and ***READ-BASE***, which are controlled by the **COMMONLISP** property. **COMMONNUMSYNTAX** is true for Common Lisp readtables and the new Interlisp readtables; it is false for old Interlisp readtables.

USESILPACKAGE This is a backward compatibility feature. If **USESILPACKAGE** is true, then the Interlisp input/output functions read and print symbols with respect to the Interlisp package, independent of the current value of ***PACKAGE***. This property is true by default for old Interlisp readtables and false for others.

The following properties let the print functions know what characters are being used for certain variable syntax classes so that they can print objects in a way that will read back correctly. Note that it is possible for several characters to have the same syntax on input, but only one of the characters is used for output. Also note that only the three syntax classes **ESCAPE**, **MULTIPLE-ESCAPE** and **PACKAGEDELIM** are parameterized for output; the others (such as **LEFTPAREN** and **STRINGDELIM**) are hardwired —the same character is always used.

ESCAPECHAR This is the character code for the character to use for single escape. Setting this property also gives the designated character the syntax **ESCAPE** in the readtable.

MULTIPLE-ESCAPECHAR This is the character code for the character to use for multiple escape. Setting this property also gives the designated character the syntax **MULTIPLE-ESCAPE** in the readtable.

PACKAGECHAR This is the character code for the package delimiter. Setting this property also gives the designated character the syntax **PACKAGEDELIM** in the readtable.

(**FIND-READTABLE** *NAME*) [Function]

Returns the readtable whose name is *NAME*, which should be a symbol or string (case is ignored); returns **NIL** if no such readtable is registered. Readtables are registered by calling (**READTABLEPROP** *rdtbl* 'NAME *name*).

(**COPYREADTABLE** *RDTBL*) [Function]

COPYREADTABLE has been extended to accept a readtable name as its *RDTBL* argument (the old value ORIG could be considered a special case of this). For example, (**COPYREADTABLE** "INTERLISP") returns a copy of the INTERLISP readtable. **COPYREADTABLE** preserves all syntax settings and properties except NAME.

Section 25.8 Predefined Readtables

The following readtables are registered in the Lyric release of Lisp:

INTERLISP This is the "new" Interlisp readtable. It is used by default in the Interlisp Exec and by the File Manager to write new versions of pre-existing source files. It thus replaces the old T readtable, FILERDTBL, CODERDTBL and DEDITRDTBL. It differs from them in the following ways:

| (vertical bar) has syntax **MULTIPLE-ESCAPE** rather than being used as a variant of the Common Lisp dispatching # macro character.

is used as the Common Lisp dispatching # macro character. For example, to type a number in hexadecimal, the syntax is #xnnn rather than |xnnn.

: (colon) has syntax **PACKAGEDELIM**.

’ (quote) reads the next expression as (QUOTE expression).

‘ (backquote)
, (comma) are used to read backquoted expressions

In addition, the Common Lisp syntax for numbers is supported (the readtable has property **COMMUNNUMSYNTAX**). For example, the characters "1/2" denote a ratio, not a symbol. Note, however, that trailing decimal point still means floating point, rather than forcing a decimal read base for an integer.

The syntax for quote, backquote, and comma is the same as in OLD-INTERLISP-T, so you will not see any difference when typing to an Interlisp Exec. However, the fact that files are also written in the new INTERLISP readtable means that the prettyprinter is now able to print quoted and backquoted expressions much more attractively on files (and to the display as well).

LISP This readtable implements Common Lisp read syntax, exactly as described in *Common Lisp, the Language*.

XCL This readtable is the same as LISP, except that the characters with ASCII codes 1 thru 26 have white-space (**SEPRCHAR**) syntax. This readtable is intended for use in File Manager files, so that font information can be encoded on the file.

The following readtables are provided for backward compatibility. They are the same as the corresponding readtables in the Koto release, with the addition of the **USESILPACKAGE** property.

ORIG This is the same as the ORIG readtable described in the *Interlisp-D Reference Manual*. When using a readtable produced by (**COPYREADTABLE** ’ORIG), expressions will read and print exactly the same in Koto and Lyric.

OLD-INTERLISP-FILE This is the same as the FILERDTBL described in the *Interlisp-D Reference Manual*. This readtable is used to read source files produced in the Koto release. Note that in Lyric, FILERDTBL is no longer used when reading or writing new files; see the section on reader environments.

OLD-INTERLISP-T This is the same as the T readtable described in the *Interlisp-D Reference Manual*.

If you wish to change the syntax used by a standard readtable, it is recommended instead that you copy the readtable, give it a distinguished name, and make the change in the new readtable. This will reduce the likelihood that you will try to read another user’s files in an incompatible readtable, or that another user will fail reading yours. See chpater 17, Reader Environments and the File Manager, for more details.

Koto Compatibility Considerations

In order to consistently read a data structure that you have previously printed, it is important that **READ** and **PRINT** both use the same readtable and package. Code that calls **READ** or **PRINT** without explicitly specifying a readtable (via the *RDTBL* argument or by doing a **SETREADTABLE**) is thus in some danger of reading and printing inconsistently.

Specifying Readtables and Packages

In Koto, the "primary" (NIL) readtable was not significantly different from the other Interlisp readtables, and users tended not to make significant modifications to the primary readtable anyway. As a result, it was easy to write code that was not careful about readtables and get away with it. In Lyric, however, there are significant differences among commonly used readtables. Thus, if code using the default readtable called **PRINT** under, say, the Common Lisp Executive and tried to **READ** the expression back while running under an Interlisp Executive, it might very well get inconsistent results.

Lyric also introduces the extra complication of the default package, which is the other important parameter affecting the behavior of **READ** and **PRINT**.

Programmers are thus advised to fix any code that uses **READ** and **PRINT** as a way of storing and retrieving Lisp expressions to be sure to specify a readtable and package environment. For new code in Lyric, this can be done by binding the special variables ***READTABLE*** and ***PACKAGE***. If it is necessary to write code that works in both Koto and Lyric, the programmer should pass an explicit readtable to all **READ** and **PRINT** functions, or set the primary readtable using (**RESETFORM** (**SETREADTABLE** *rdtbl*) --). If the readtable chosen is either FILERDTBL or one derived as a copy of ORIG, then **READ** and **PRINT** will automatically use the INTERLISP package in Lyric, thereby avoiding any need to specify a binding for ***PACKAGE***.

The T Readtable

An additional possible incompatibility exists with regard to the Koto T readtable: The T readtable was "the readtable used when reading from the terminal". In Lyric, the T readtable is synonymous with NIL, and all Executives bind ***READTABLE*** to the appropriate value for the Exec. This is unlikely to be a major source of incompatibility, as few programs depend on printing something in the T readtable in a way that needs to read back consistently.

PQUOTE Printed Files

In Lyric, the prettyprinter automatically prints quoted and backquoted expressions attractively. Hence, the PQUOTE Lispusers module is now obsolete. However, if you have written files in the past with the PQUOTE module loaded into your environment, you need to do the following in Lyric in order to load those files:

(SETSYNTAX (CHARCODE "'") '(MACRO FIRST READQUOTE) FILERDTBL)

You can then load the old files. New files produced in Lyric by **MAKEFILE** will automatically be loadable, so you need only perform the **SETSYNTAX** change as long as you still have old files written with PQUOTE. Remember, of course, that as long as the **SETSYNTAX** is in effect (as with the old PQUOTE module), if you read old files that were written without PQUOTE you may read them incorrectly.

Back-Quote Facility

The back-quote facility now fully conforms with *Common Lisp the Language*. This means some cases of nested back-quote now work correctly. Back-quote forms are also more attractively displayed by the prettyprinter. Users should beware, however, that the back-quote facility does not attempt to create fresh list structures unless it is necessary to do so. Thus for example,

```
'(1 2 3)
```

is equivalent to

```
'(1 2 3)
```

not

```
(LIST 1 2 3)
```

If you need to avoid sharing structure you should explicitly use **LIST**, or **COPY** the output of the back-quote form.

Chapter 28 Windows and Menus

Section 28.5.1 Menu Fields

(III:28.38)

With the Medley release, multi-column menus can have rollout submenus.

[This page intentionally left blank]