LispDoc
Manual

4. April 1995

Heiko Kirschke
kirschke@informatik.uni-hamburg.de
Beim Alten Schützenhof 4
D 22083 Hamburg

A POSTSCRIPT version of this document is available by WWW from the page
http://lki-www.informatik.uni-hamburg.de/~kirschke/invlit.html

University of Hamburg • Computer Science Department • Vogt-Kölln-Straße 30 • D 22527 Hamburg

No Warranty

1. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program ‘as is’ without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

2. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

Contents

Chapter 1

Introduction

The package `lisp-doc` can be used for generating semi-automated documentation from LISP source code. Its way of processing could best be described by the term 'Inverse Literate Programming'. To begin with the non-inverse approach, Literate Programming means generating both program documentation and source code from one 'meta-source' file; a good example for literate programming is WEB. A WEB file is broken down by a 'compilation' process into a TEX file containing the program's documentation and a Pascal or C file containing the program's source code (among providing the poor Pascal programmers with the benefits of a textual preprocessor). Alas, `lisp-doc` works a little bit the other way round: It takes a LISP source code and generates a TEX documentation for it. I developed this strange looking way of processing 'the other way round' w.r.t. Literate Programming because my diploma thesis [Kirschke 94] [Kirschke 95] supervisor raised the request to make a good-looking manual for my diploma thesis at the very end of programming around 500 KBytes of LISP code; so I thought it would be a good idea to use the nice syntax, structuring and its regular treatment of documentating items inherent to the LISP language to generate a manual right from the source code itself, especially because LISP programmers are encouraged to document their 'products' directly in the source code. This means too that `lisp-doc` is 'only' a spin-off product of my diploma thesis and will only be supported by me if I have the time to do so. It was thought by me of being a use-once product, but meanwhile it is used by some people in the Computer Science Department at the University of Hamburg. To benefit from `lisp-doc`, follow these steps:

1. Write your LISP code almost as usual, but bear in mind that all documentation strings will be processed later by TEX.[1] This has the advantage that all your documentation can contain really any TEX commands and that the output will look very pretty, if you know how to use TEX. The last meant constraint is also the disadvantage for non-TEX-expirienced programmers. Hints on how the documentation should look like are given in section 1.3 (p. 3).

2. Compile & load `lisp-doc` as usual into your LISP system; let it scan your source code as explained in section 1.1 (p. 2).

3. Include the generated TEX file in a main TEX file as shown in the following example file.

[1] Only documentation strings are processed, comments are not.

4. Call LaTeX with the name of the main file.

5. Print and distribute the hopefully pretty looking manual.

This is the documentation about the `lisp-doc` module itself; the next chapter was generated from its own documentation strings. Use this LaTeX file as an example how to include the documented TeX source into a 'surrounding' manual.

A minimal necessary 'main' LaTeX file working on a file generated from `lisp-doc` looks like this:

```
\documentclass{report}
\usepackage[english]{babel}%    Otherwise lispdoc will fail ...
\usepackage{lispdoc,crossref}
%
\begin{document}
%
\tableofcontents%    The table of contents is useful as an index too
\clearpage
\sloppy%             \sloppy lets the output look a little bit better
\numberingoff%       Switch section numbering off
\include{...}%       \include here the \TeX\ file generated by lisp-doc
\numberingon%        Switch section numbering on again
\fussy
%
\end{document}
```

The class file `lispdoc.cls` was adpated to use NFSS, but care was taken that it should also be working without NFSS. You can use the original `cmr` or the POSTSCRIPT fonts; add the `times` package in front of the `lispdoc` packages to get POSTSCRIPT fonts.

1.1 Generating the TeX source from LISP documentation

For this, use function **scan-files** [*21*]; see its documentation. The text in the next chapter of this documentation was generated by calling the function **scan-myself** [*22*].

Please note that TeX commands embedded into the documentation strings mut be preceeded by *two* backslashes since the LISP reader handles the backslash character as a quoting character; the scanning process of `lisp-doc` 'collapses' each pair of two backslashes into one backslash. Look into file `lisp-doc` itself for examples.

1.2 Generated TeX source

The style of presentation and the notation are 'borrowed' and extended from chapter 6 of [AMOP]; see [AMOP, p. 163] for details concerning the notation used for [generic] functions and methods. Here are the extensions and changes introducted to the style of chapter 6 of [AMOP]:

- Constants, variables, macros and classes are documented too.

- Since not only the external interface but also all internal entities are described, there is an addition to each section header if the entity is *external* or *internal*. The internal entities are described to document the system itself.

- Added to each section should be a subsection named 'SEE ALSO' with references to other entities; the references should be ordered according to their importance.
- References are done by using the `\fcite{}` macro; they generate an identifier naming the kind of entity, the symbol naming the entity and a page resp. bibliographic reference in square brackets, e.g.: `\Fcite{scan-files}` gives 'Function **scan-files** [*21*]'.
- If a form has a **(setf)** equivalent, the **(setf)** form is placed right behind the non-**(setf)** form.

1.3 Documenting source code

The TEX code is read directly from the documentation strings which may be given to many LISP top level expressions; it should be built up as follows:

```
(defun sample-function ( <argument 1> ... <argument n> )
  "
\\Argumentslabel
 \\isa{\\funarg{ <argument 1> }} for keyword arguments, use \\keyarg instead of \\funarg
       { Text explaining <argument 1>, e.g. a string}
 ...
 \\isa{\\funarg{ <argument n> }}
       { Text explaining <argument n>, e.g. a symbol}

\\Valueslabel
 Explain here the values returned by the function, e.g. Returns always \\lispt.
 Omit the \\Valueslabel section if the function returns no values at all; if it
 returns the value of one of its arguments, use \\retarg{\\funarg{ <returned
 argument> }}.

\\Purposelabel
 Explain here what the documented item does, e.g. This is a sample
 function showing how to document functions. References to function
 arguments should be enclosed in \\funarg{ <argument> } resp. \\keyarg{
 <argument> }; this will emphasize them.

\\Seealsolabel
 Give here references ordered by their importance to other documented entities. The
 \\Fcite macro capitilizes the first letter:
 \\Fcite{...}; \\fcite{...}."
  ...
  t)
```

This is an example of how to document a function. Actually, the documentation of almost all top-level expressions of the form `(def...)` will be put into the reference manual. To suppress a top-level expression from being put into the reference manual, put a `#-:Lisp-Doc` expression in front of it; for top-level expressions which are only for being put into the reference manual but are not intended to be compiled, use a `#+:Lisp-Doc` expression. The above text is shown as it would be found in a LISP documentation string, e.g. all backslashes are 'doubled'. Variables, constants, methods and slots should be documented only by a short text without any of the labels specified.

A table of contents is very useful and should be generated in the main LATEX file; since each documented item is handled as a LATEX `section`, the table of contents serves as a quite useful index too.

1.4 Cross-referencing

Cross-referencing is done with the `\flabel{}{}{}{}` and `\fcite{}` resp. `\Fcite{}` macros. The name 'fcite' means **f**unction **cite**, i.e. it was created by me to reference function names easily.

The `\flabel{}{}{}{}` macro defines an item which should be referenced in the text by the `\fcite{}` macro. You can think of using the `\flabel{}{}{}{}` macro is very similar to using the standard LaTeX `\label{}` macro. The difference is that references set with a `\label{}` macro expand at their usage with `\ref{}` to the number defined by the enclosing environment of the `\label{}` macro, i.e. a `\label{}` macro placed right behind a `\section{}` macro expands at its usage with `\ref{}` to the section number of the section in which the `\label{}` macro was placed. With `\flabel{}{}{}{}` you are free to define another text which should appear at referencing instead of the number defined by the enclosing environment. The `\flabel{}{}{}{}` macro takes 4 arguments:

1. A text describing the 2nd argument. Predefined are some shorthands which must be used to make the `\fcite{}` and `\Fcite{}` macros explained below work as expected:

Shorthand	**expands to**	**Shorthand**	**expands to**
`\crfchapter`	chapter	`\crfsection`	section
`\cls`	class	`\clsmc`	class metaobject class
`\clsmo`	class metaobject	`\const`	constant
`\fn`	function	`\gfn`	generic function
`\mac`	macro	`\mc`	metaobject class
`\mo`	metaobject	`\mtd`	method
`\mtdmc`	method metaobject class	`\mtdmo`	method metaobject
`\obj`	object	`\slt`	slot
`\sltmc`	slot metaobject class	`\sltmo`	slot metaobject
`\spfrm`	special form	`\var`	variable

   Please note that you must place a `\protect` macro right ahead the shorthands described above.

2. The label name. Normally this is the name of a function, macro or something similar.

3. The text which should appear when the label is referenced. Normally this is a `\cite{}` reference.

4. The text the label refers to.

All calls to `\flabel{}{}{}` are placed normally into the document's preamble and define references to external items, i.e. the standard LISP items described in [CLtLII].

Example: Assuming that the label of the bibliographic reference to [CLtLII] is `bib:st90`, the call

```
\flabel{\protect\var}{*print-case*}{\protect\cite[p. 560]{bib:st90}}
```

is referenced in the text by[2]

```
\fcite{*print-case*}
```

and will expand to the text 'variable ***print-case*** [CLtLII, p. 560]'. The `\Fcite{}` macro capitalizes the first letter of the referenced text, i.e.

```
\Fcite{*print-case*}
```

[2] Actually, `\fcite` should better be named `\fref`.

expands to ‘Variable ***print-case*** [CLtLII, p. 560]’. The `\Fcite{}` macro works only as expcted if the above shorthands are used as the first argument to `\flabel{}{}{}`.

For more examples see file `plob/tex/inputs/dipldefs.sty` on using `\flabel{}{}{}` and file `plob/tex/manual/plobrefg.tex` for examples on using `\fcite{}`.

1.5 Further ideas

Perhaps it would be a good idea to select a similar approach for generating HTML code from LISP source code, so that the friends of WWW can make an online-available program documentation.

1.6 Known bugs

This package works for now only with LISPWORKS Common LISP. Set the global variable ***print-case*** [CLtLII, p. 560] to `:downcase` to print all identifier names in lower case; otherwise they will appear in upper case.

When using the `[twoside]` class option, the page headings on the left pages are not as expected: They contain the name of the last documented item on the left page and not the first one. This is not a `lisp-doc` bug, but a good known LaTeX ‘feature’ which occurs when section names instead of chapter names are used in page headers of left pages (each documented item forms a LaTeX `\section`). To my opinion, using a chapter name in the page header makes it redundant, because the headers of all left pages of the reference manual would contain something like ‘Reference Guide’, which is bad for locating a documented item in the reference manual.

Chapter 2

LispDoc Reference Guide

CLOS Class Hierarchy *Class Hierarchy*

all-clos-classes *Internal Variable*

INITIAL VALUE
`nil`

PURPOSE
A list containing all scanned CLOS classes.

all-structure-classes *Internal Variable*

INITIAL VALUE
`nil`

PURPOSE
A list containing all scanned structure classes.

comment-class *Internal Class*

PURPOSE
A class for documenting classes.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

(`lisp-comment` [*17*])

DIRECT SLOTS

com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-class-midfix+
:accessor comment-midfix
```

superclasses-label *Direct Slot*

```
:initarg :superclasses-label
:initform +default-superclasses-label+
:accessor comment-superclasses-label
```

Contains the TEX label to use for slot **superclasses**.

superclasses *Direct Slot*

```
:initarg :superclasses
:initform nil
:accessor comment-superclasses
```

Contains a list of the direct superclasses of the documented class.

metaclass-label *Direct Slot*

```
:initarg :metaclass-label
:initform +default-metaclass-label+
:accessor comment-metaclass-label
```

Contains the TEX label to use for slot **metaclass**.

metaclass *Direct Slot*

```
:initarg :metaclass
:initform nil
:accessor comment-metaclass
```

Contains the metaclass of the documented class.

slots-label *Direct Slot*

```
:initarg :slots-label
:initform +default-slots-label+
:accessor comment-slots-label
```

Contains the TEX label to use for slot **slots**.

slots *Direct Slot*

```
:initarg :slots
:initform nil
:accessor comment-slots
```

Contains a list of the direct slots of the documented class.

class-options-label *Direct Slot*

```
:initarg :class-options-label
:initform +default-class-options-label+
:accessor comment-class-options-label
```

Contains the TEX label to use for slot **class-options**.

class-options *Direct Slot*

```
:initarg :class-options
:initform nil
:accessor comment-class-options
```

Contains the class options of the documented class.

comment-constant *Internal Class*

PURPOSE
A class for documenting constants.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS
com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-constant-midfix+
:accessor comment-midfix
```

comment-documentation *Internal Class*

PURPOSE
A class for documenting-only items.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS

com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-documentation-midfix+
:accessor comment-midfix
```

comment-function *Internal Class*

PURPOSE
A class for documenting functions.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

(`lisp-comment` [*17*])

DIRECT SLOTS
com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-function-midfix+
:accessor comment-midfix
```

comment-generic-function *Internal Class*

PURPOSE
A class for documenting generic functions.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

(`lisp-comment` [*17*])

DIRECT SLOTS
com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-generic-function-midfix+
:accessor comment-midfix
```

methods-label *Direct Slot*

```
:initarg :methods-label
:initform +default-methods-label+
:accessor comment-methods-label
```

Contains the TEX label to use for slot **methods**.

methods *Direct Slot*

```
:initarg :methods
:initform nil
:accessor comment-methods
```

Contains a list of method documentations belonging to the generic function.

comment-macro *Internal Class*

PURPOSE
A class for documenting macros.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS

com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-macro-midfix+
:accessor comment-midfix
```

comment-method *Internal Class*

PURPOSE
A class for documenting methods.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS

metaobject *Direct Slot*

```
:initarg :metaobject
:initform nil
:accessor method-metaobject
```

The method metaobject of the documented method.

qualifiers *Direct Slot*

```
:initarg :qualifiers
:initform nil
:accessor method-qualifiers
```

specializers *Direct Slot*

```
:initarg :specializers
:initform nil
:accessor method-specializers
```

syntax-label *Direct Slot*

```
:initarg :syntax-label
:initform nil
:accessor comment-syntax-label
```

Since methods don't have a SYNTAX label, this slot is always initialized to `nil`.

comment-parameter *Internal Class*

PURPOSE
A class for documenting parameters.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS
com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-parameter-midfix+
:accessor comment-midfix
```

comment-slot *Internal Class*

PURPOSE
A class for documenting slots of classes.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS
metaobject *Direct Slot*

```
:initarg :metaobject
:initform nil
:accessor method-metaobject
```

The method metaobject of the documented slot.

syntax-label *Direct Slot*

```
:initarg :syntax-label
:initform nil
:accessor comment-syntax-label
```

comment-variable *Internal Class*

PURPOSE
A class for documenting variables.

SEE ALSO
Class **lisp-comment** [*17*].

DIRECT SUPERCLASSES

```
(lisp-comment [17])
```

DIRECT SLOTS

com-midfix *Direct Slot*

```
:initarg :midfix
:initform +default-variable-midfix+
:accessor comment-midfix
```

+default-class-midfix+ *Internal Constant*

VALUE
`"cl"`

PURPOSE
The text to put into the `\begin..com` command for classes.

+default-class-options-label+ *Internal Constant*

VALUE
`"\\Classoptionslabel"`

PURPOSE
The default TEX command string for generating the CLASS OPTIONS label.

+default-constant-midfix+ *Internal Constant*

VALUE
`"cn"`

PURPOSE
The text to put into the `\begin..com` command for constants.

+default-documentation-midfix+ *Internal Constant*

VALUE
`"dc"`

PURPOSE
The text to put into the `\begin..com` command for documented items.

+default-function-midfix+ *Internal Constant*

VALUE
`"fn"`

PURPOSE
The text to put into the `\begin..com` command for functions.

+default-generic-function-midfix+ *Internal Constant*

VALUE
`"gf"`

PURPOSE
The text to put into the `\begin..com` command for generic functions.

+default-macro-midfix+ *Internal Constant*

VALUE
`"mc"`

PURPOSE
The text to put into the `\begin..com` command for macros.

+default-metaclass-label+ *Internal Constant*

VALUE
`"\\Metaclasslabel"`

PURPOSE
The default TEX command string for generating the METAOBJECT CLASS label.

+default-method-text+ *Internal Constant*

VALUE
```
"%
% No behavior is specified for this method beyond that which is specified
% for the generic function."
```

PURPOSE
The default text for methods without any documentation.

+default-methods-label+ *Internal Constant*

VALUE
`"\\Methodslabel"`

PURPOSE
The default TEX command string for generating the *Methods* label.

+default-parameter-midfix+ *Internal Constant*

VALUE
`"pm"`

PURPOSE
The text to put into the `\begin..com` command for parameters.

+default-slots-label+ *Internal Constant*

VALUE
`"\\Directslotslabel"`

PURPOSE
The default TEX command string for generating the DIRECT SLOTS label.

+default-superclasses-label+ *Internal Constant*

VALUE
`"\\Directsuperclasseslabel"`

PURPOSE
The default TEX command string for generating the DIRECT SUPERCLASSES label.

+default-syntax-label+ *Internal Constant*

VALUE
`"\\Syntaxlabel"`

PURPOSE
The default TEX command string for generating the SYNTAX label.

+default-syntax-postfix+ *Internal Constant*

VALUE
`""`

PURPOSE
The default postfix to append after a syntax text.

+default-variable-midfix+ *Internal Constant*

VALUE
`"vr"`

PURPOSE
The text to put into the `\begin..com` command for variables.

+exported-label+ *Internal Constant*

VALUE
`"External"`

PURPOSE
The string used as *External* label in a document item header.

free item ... *Document Items with 'free' Text*

This is a document item not bound to a programming construct:

```
#+lisp-doc
(:defdoc
 "free item ..."      ; This could be also a symbol
 "Document Items with 'free' Text"
 "This is a document item not bound to a programming construct:
  ...")
```

It is only 'seen' by the LISP reader when `:lisp-doc` is on the `*features*` list; this is accomplished by `lisp-doc` during scanning a LISP file.

This item is sorted accroding to its first sort-key 'parameter' behind the `:defdoc` statement.

You can also use `\fcite{}` on these items by using the sort key as argument, e.g. `\fcite{free item ...}` gives ' **free item ...** [*16*]'.

generic-function-name->comment *Internal Variable*

INITIAL VALUE
`(make-hash-table :test 'equal)`

PURPOSE
A hash table mapping a generic function name to its documenting object.

insert-class *Internal Function*

SYNTAX
insert-class

the-class class-list

Insert *the-class* into *class-list*.

+internal-label+ *Internal Constant*

VALUE
`"Internal"`

PURPOSE
The string used as *Internal* label in a document item header.

lisp-comment *Internal Class*

PURPOSE
A class for representing comments. For each documented item, an instance of **lisp-comment** is created.

DIRECT SLOTS

com-midfix *Direct Slot*

```
:initarg :midfix
:initform nil
:accessor comment-midfix
```

The text to put into the `\begin..com` command.

key *Direct Slot*

```
:initarg :key
:initform nil
:accessor comment-key
```

The Keyword of the comment. The documentation is sorted according to this keyword.

key-attribute *Direct Slot*

```
:initarg :key-attribute
:initform nil
:accessor comment-key-attribute
```

The text attribute for key; normally, this contains either the *Internal* or *Exported* label.

syntax-label *Direct Slot*

```
:initarg :syntax-label
:initform +default-syntax-label+
:accessor comment-syntax-label
```

The TEX command string to generate the SYNTAX label for the document item.

syntax *Direct Slot*

```
:initarg :syntax
:initform nil
:accessor comment-syntax
```

The syntax description for the documented item.

text *Direct Slot*

```
:initarg :text
:initform nil
:accessor comment-text
```

Contains a textual description of key. This is normally the literal read documentation string.

make-class-options-comment *Internal Function*

SYNTAX
make-class-options-comment

class-options

ARGUMENTS
The *class-options* argument is a list with class-option value pairs.

PURPOSE
Returns *class-options* transformed into a TEX command sequence.

make-function-comment *Internal Function*

SYNTAX
make-function-comment

function-name lambda-list

ARGUMENTS
The *function-name* argument is a symbol.
The *lambda-list* argument is a λ-list.

PURPOSE
Transform the function named by *function-name* with λ-list *lambda-list* into a TEX command sequence.

SEE ALSO
Function **make-lambda-list-comment** [*19*].

make-key-attribute *Internal Function*

SYNTAX
make-key-attribute

name

PURPOSE
Check if *name* is an external name resp. to `*package*`.

SEE ALSO
Constant **+internal-label+** [*17*]; constant **+exported-label+** [*16*].

make-lambda-list-comment *Internal Function*

SYNTAX
make-lambda-list-comment

lambda-list

ARGUMENTS
The *lambda-list* argument is a λ-list.

PURPOSE
Returns *lambda-list* transformed into a TEX command sequence.

make-lisp-comment *Internal Generic Function*

SYNTAX
make-lisp-comment

keyword expression

ARGUMENTS
The *keyword* argument is a symbol.
The *expression* argument is a list.

VALUES
Returns either `nil` or an instance of [a subclass of] class **lisp-comment** [*17*] which contain comment items for *keyword*.

PURPOSE
Make a comment from *keyword*; *expression* contains the expression read from the source file.

METHODS
make-lisp-comment *Primary Method*

(*keyword* `(eql 'defclass))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defconstant))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defgeneric))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defmacro))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defmethod))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defparameter))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defstruct))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defun))` *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql 'defvar)`) *expression*

make-lisp-comment *Primary Method*

(*keyword* `(eql :defdoc)`) *expression*

make-lisp-comment *Primary Method*

keyword *expression*

Return always `nil`; this will generate no document text at all.

make-slot-comment

Internal Function

SYNTAX

make-slot-comment

class-name *expression* `&optional` *first-option*

ARGUMENTS

The *expression* argument is a slot-defining expression.

PURPOSE

Returns *expression* transformed into an instance of class **comment-slot** [*12*].

make-sort-key

Internal Function

SYNTAX

make-sort-key

raw-key

ARGUMENTS

The *raw-key* argument is either a symbol or a `setf`-expression.

PURPOSE

Transform *raw-key* into a string which is used for sorting the document items.

SEE ALSO

Function **make-lambda-list-comment** [*19*].

print-object

External Generic Function

SYNTAX

print-object

instance *stream*

See generic function **print-object** [CLtLII, p. 850].

METHODS

print-object *Primary Method*

(*object* `lisp-comment` [*17*]) *stream*

+qualifier-ordering+ *Internal Constant*

VALUE
`'(:around :before nil :after)`

PURPOSE
A list imposing an ordering on method qualifiers for computing the order of methods to be printed.

quote-tex-characters *Internal Function*

SYNTAX
quote-tex-characters

raw-string

ARGUMENTS
The *raw-string* argument is a string with TEX commands.

PURPOSE
Put a backslash before all TEX special characters.

+reader-lambda-list+ *Internal Constant*

VALUE
`'(instance)`

PURPOSE
The lambda list of a reader generic function.

scan-file *External Function*

SYNTAX
scan-file

`&rest` *all-args*

See function **scan-files** [*21*].

scan-files *External Function*

SYNTAX
scan-files

file-list `&optional` *to-stream*

ARGUMENTS
The *file-list* argument is a list with LISP source file names without `.lisp` extension.
The *to-stream* argument is either `nil` or a string or a Common LISP output stream.

VALUES
The output stream which was used for creating the TEX file is returned.

PURPOSE
Scan the LISP source files in *file-list* and write their documentation to *to-stream*.

scan-myself *External Function*

SYNTAX
scan-myself

`&optional` *my-path*

ARGUMENTS
The *my-path* argument is a path expression.

PURPOSE
Extract the documentation strings from module `lisp-doc`. The *my-path* argument contains the path information where the `lisp-doc` file is located.

SEE ALSO
Function **scan-files** [*21*].

scan-one-file *Internal Function*

SYNTAX
scan-one-file

filename

ARGUMENTS
The *filename* argument is a pathname.

PURPOSE
Scan file named by *filename* and extract its documentation.

SEE ALSO
Function **scan-files** [*21*].

ship-out *Internal Generic Function*

SYNTAX
ship-out

comment to-stream

ARGUMENTS
The *comment* argument is an instance of [a subclass of] class **lisp-comment** [*17*].
The *to-stream* argument is a Common LISP output stream.

PURPOSE
Write *comment* as TEX command sequence to *to-stream*.

METHODS

ship-out *Around-Method*

(*c* `comment-generic-function`[*10*]) *to-stream*

ship-out *Before-Method*

comment *to-stream*

Write out the `\begin..com` command.

ship-out *Primary Method*

(*c* `comment-class`[*8*]) *to-stream*

ship-out *Primary Method*

(*c* `comment-generic-function`[*10*]) *to-stream*

ship-out *Primary Method*

(*c* `lisp-comment`[*17*]) *to-stream*

ship-out *After-Method*

comment *to-stream*

Write out the `\endcom` command.

ship-out-class-tree *Internal Function*

SYNTAX

ship-out-class-tree

class-list *name* *to-stream*

Write *class-list* as a class tree to *to-stream*.

ship-out-label-and-argument *Internal Function*

SYNTAX

ship-out-label-and-argument

label *argument* *to-stream*

PURPOSE

Write *argument* as TEX command sequence following the *label* to *to-stream*.

SEE ALSO

Generic function **ship-out** [*22*].

+undocumented-item-prompt+ *Internal Constant*

VALUE

`";; *** Warning: Undocumented ~A ~S~%"`

PURPOSE

Prompt for `undocumented ...` warnings.

+unknown-qualifier-continue-prompt+ *Internal Constant*

VALUE
```
"Dont't order the methods."
```

PURPOSE
The continue prompt for unknown method qualifiers.

+unknown-qualifier-error-prompt+ *Internal Constant*

VALUE
```
"Found unknown qualifier ~A in method ~A."
```

PURPOSE
The error prompt for unknown method qualifiers.

+writer-lambda-list+ *Internal Constant*

VALUE
```
'(new-value instance)
```

PURPOSE
The lambda list of a writer generic function.

Bibliography

[AMOP] Gregor Kiczales, Jim des Rivières, Daniel G. Bobrow: The Art of the Metaobject Protocol, MIT Press, 1991

[CLtLII] Guy L. Steele Jr.: Common LISP the Language, Second Edition, Digital Press, 1990

[Kirschke 94] Heiko Kirschke: Persistenz in objekt-orientierten Programmiersprachen am Beispiel von CLOS. Diploma Thesis, Dept. of Computer Science, University of Hamburg, 1994

[Kirschke 95] Heiko Kirschke: Persistenz in der objekt-orientierten Programmiersprache CLOS am Beispiel des PLOB! Systems. Report FBI-HH-B-179/95, Dept. of Computer Science, University of Hamburg, 1995