CSL reference

A C Norman

March 7, 2010

1 Introduction

This is reference material for CSL. The Lisp identifiers mentioned here are the ones that are initially present in a raw CSL image. Some proportion of them are not really intended to be used by end-users but are merely the internal components of some feature.

2 Command-line options

The items shown here are the ones that are recognized on the CSL command line. In general an option that requires an argument can be written as either `-x yyy` or as `-xyyy`. Arguments should be case insensitive.

2.1 `-a`

`-a` is a curious option, not intended for general or casual use. If given it causes the `(batchp)` function to return the opposite result from normal! Without "attfamily -a" `(batchp)` returns `T` either if at least one file was specified on the command line, or if the standard input is "not a tty" (under some operating systems this makes sense – for instance the standard input might not be a "tty" if it is provided via file redirection). Otherwise (ie primary input is directly from a keyboard) `(batchp)` returns `nil`. Sometimes this judgement about how "batch" the current run is will be wrong or unhelpful, so `-a` allows the user to coax the system into better behaviour. I hope that this is never used!

2.2 `-b`

`-b` tells the system to avoid any attempt to recolour prompts and input text. It will mainly be needed on X terminals that have been set up so that they use colours that make the defaults here unhelpful. Specifically white-on-black and so on.

`-b` can be followed by colour specifications to make things yet more specific. It is supposed to be the idea that three colours can be specified after

it for output, input and prompts, with the letters KRGYbMCW standing for blacK, Red, Green, Yellow, blue, Magenta, Cyan and White. This may not fully work yet!

2.3 `-c`

Displays a notice retalting to the authorship of CSL.

2.4 `-d`

A command line entry `-Dname=value` or `-D name=value` sets the value of the named lisp variable to the value (as a string).

2.5 `-e`

A "spare" option used from tim eto time to activate experiments within CSL.

2.6 `-f`

At one stage CSL could run as a socket server, and `-f portnumber` activated that mode. `-f-` used a default port, 1206 (a number inspired by an account number on Titan that I used in the 1960s). The code that supports this may be a useful foundation to others who want to make a network service out of this code-base.

2.7 `-g`

In line with the implication of this option for C compilers, this enables a debugging mode. It sets a lisp variable `!*backtrace` and arranges that all backtraces are displayed notwithstanding use of `errorset`.

2.8 `-h`

This option is a left-over. When the X-windows version of the code first started to use Xft it viewed that as optional and could allow a build even when it was not available. And then even if Xft was detected and liable to be used by default it provided this option to disable its use. The remnants of the switch that disabled use of Xft (relating to fonts living on the Host or the Server) used this switch, but it now has no effect.

2.9 `-i`

CSL and Reduce use image files to keep both initial heap images and "fasl" loadable modules. By default if the executable launched has some name, say xxx, then an image file xxx.img is used. But to support greater generality

`-i` introduces a new image, `-i-` indicates the default one and a sequence of such directives list image files that are searched in the order given. These are read-only. The similar option `-o` equally introduces image files that are scanned for input, but that can also be used for output. Normally there would only be one `-o` directive.

2.10 `-j`

Follow this directive with a file-name, and a record of all the files read during the Lisp run will be dumped there with a view that it can be included in a Makefile to document dependencies.

2.11 `-k`

`-K nnn` sets the size of heap to be used. If it is given then that much memory will be allocated and the heap will never expand. Without this option a default amount is used, and (on many machines) it will grow if space seems tight.

The extended version of this option is `-K nnn/ss` and then ss is the number of "CSL pages" to be allocated to the Lisp stack. The default value (which is 1) should suffice for almost all users, and it should be noted that the C stack is separate from and independent of this one and it too could overflow.

A suffix K, M or G on the number indicates units of kilobytes, megabytes or gigabytes, with megabytes being the default. So `-K200M` might represent typical usage.

2.12 `-l`

This is to send a copy of the standard output to a named log file. It is very much as if the Lisp function `(spool ``logfile'')` had been invoked at the start of the run.

2.13 `-m`

Memory trace mode. An option that represents an experiment from the past, and no longer reliably in use.

2.14 `-n`

Normally when the system is started it will run a "restart function" as indicated in its heap image. There can be cases where a heap image has been created in a bad way such that the saved restart function always fails abruptly, and hence working out what was wrong becomes hard. In such cases it may be useful to give the `-n` option that forces CSL to ignore any

startup function and merely always begin in a minimal Lisp-style read-eval-print loop.

2.15 -o

See `-i`.

2.16 -p

If a suitable profile option gets implemented one day this will activate it, but for now it has no effect.

2.17 -q

This option sets `!*echo` to `nil` and switches off garbage collector messages to give a slightly quieter run.

2.18 -r

The random-number generator in CSL is normally initialised to a value based on the time of day and is hence not reproducible from run to run. In many cases that behavious is desirable, but for debugging it can be useful to force a seed. The directive `-r nnn,mmm` sets the seed to up to 64 bits taken from the values nnn and mmm. The second value if optional, and specifying `-r0` explicitly asks for the non-reproducible behaviour (I hope). Note that the main Reduce-level random number source is coded at a higher level and does not get reset this way – this is the lower level CSL generator.

2.19 -s

Sets the Lisp variable `!*plap` and hence the compiler generates an assembly listing.

2.20 -t

`-t name` reports the time-stamp on the named module, and then exits. This is for use in perl scripts and the like, and is needed because the stamps on modules within an image or library file are not otherwise instantly available.

Note that especially on windowed systems it may be necessary to use this with `-- filename` since the information generated here goes to the default output, which in some cases is just the screen.

2.21 -u

See -d, but this forcibly undefines a symbol. There are probably very very few cases where it is useful since I do not have a large number of system-specific predefined names.

2.22 -v

An option to make things mildly more verbose. It displays more of a banner at startup and switches garbage collection messages on.

2.23 -w

On a typical system if the system is launched it creates a new window and uses its own windowed intarface in that. If it is run such that at startup the standard input or output are associated with a file or pipe, or under X the variable DISPLAY is not set it will try to start up in console mode. The flag -w indicates that the system should run in console more regadless, while -w+ attempts a window even if that seems doomed to failure. When running the system to obey a script it will often make sense to use the -w option. Note that on Windows the system is provided as two separate (but almost identical) binaries. For example the file csl.exe is linked in windows mode. A result is that if launched from the command line it detaches from its console, and if launched by double-clicking it does not create a console. It is in fact very ugly when double clicking on an application causes an unwanted console window to appear. In contrast csl.com is a console mode version of just the same program, so when launched from a command line it can communicate with the console in the ordinary expected manner.

2.24 -x

-x is an option intended for use only by system support experts – it disables trapping if segment violations by errorset and so makes it easier to track down low level disasters – maybe! This can be valuable when running under a debugger since if the code traps signals in its usual way and tries to recover it can make it a lot harder to find out just what was going wrong.

2.25 -y

-y sets the variable !*hankaku, which causes the lisp reader convert a Zenkaku code to Hankaku one when read. I leave this option decoded on the command line even if the Kanji support code is not otherwise compiled into CSL just so I can reduce conditional compilation. This was part of the Internationalisation effort for CSL bu this is no longer supported.

2.26 -z

When bootstrapping it is necessary to start up the system for one initial time without the benefit of any image file at all. The option `-z` makes this happen, so when it is specified the system starts up with a minimal environment and only those capabilities that are present in the CSL kernel. It will normally make sense to start loading some basic Lisp definitions rather rapidly. The files `compat.lsp`, `extras.lsp` and `compiler.lsp` have Lisp source for the main things I use, and once they are loaded the Lisp compiler can be used to compile itself.

2.27 --help

It is probably obvious what this option does! But in particular it displays and explantion of the `--dump-source` option, and hence should count as a prominent and easy-to-find way of alerting people to their rights and obligations. Note that on Windows of the application was linked as a windows binary it carefully creates a console to display the help text in, and organizes a delay to give people a chance to read it.

2.28 --my-path

At some time I had felt the need for this option, but I now forget what I expected to use it for! It leads the executable to display the fully rooted name of the directory it was in and then terminate. It may be useful in some script?

2.29 --texmacs

If CSL/Reduce is launched from texmacs this command-line flag should be used to arrange that the `texmacs` flag is set in `lispsystem!*`, and the code may then do special things.

2.30 --

If the application is run in console mode then its standard output could be redirected to a file using shell facilities. But the `--` directive (followed by a file name) redirects output within the Lisp rather than outside it. If this is done a very limited capability for sending progress or status reports to stderr (or the title-bar when running in windowed mode) remains via the `report!-right` function.

The `-w` option may frequently make sense in such cases, but if that is not used and the system tries to run in a window it will create it starting off minimised.

3 Predefined variables

3.1 !!fleps1

There is a function safe!-fp!-plus that performs floating point arithmetic but guarantees never to raise an exception. This value was at one stage related to when small values created there got truncated to zero, but the current code does not use the Lisp variable at all and instead does things based on the bitwise representation of the numbers.

3.2 !$eof!$

The value of this variable is a special "character" used to denote an end-of-file condition.

3.3 !$eol!$

The value of this variable is an end-of-line character.

3.4 !*applyhook!*

If this is set it might be supposed to be the name of a function used by the interpreter as a callbackm but at presnet it does not actually do anything!

3.5 !*break!-loop!*

If the value of this is a symbol that is defined as a function of one argument then it is called during the processing on an error. This has not been used in anger and so its whole status may be dubious!

3.6 !*carcheckflag

In general CSL arranges that every `car` or `cdr` access is checked for validity. Once upon a time setting this variable to nil turned such checks off in the hope of gaining a little speed. But it no longer does that. It may have a minor effect on array access primitives.

3.7 !*comp

When set each function is compiled (into bytecodes) as it gets defined.

3.8 !*debug!-io!*

An I/O channel intended to be used for diagnostic interactions.

3.9 !*echo

When this is non-nil characters that are read from an input file are echoed to the standard output. This gives a more comlete transcript in a log file, but can sometimes amount to over-verbose output.

3.10 !*error!-messages!*

Has the value nil and does not do anything!

3.11 !*error!-output!*

An I/O channel intended for diagnostic output.

3.12 !*evalhook!*

See `!*applyhool!*`. This also does not do anything at present.

3.13 !*gc!-hook!*

If this is set to have as its value that is a function of one argument then that function is called with `nil` on every minor entry to the garbage collection, and with argument `t` at the end of a "genuine" full garbage collection.

3.14 !*hankaku

This was concerned with internationalisation to support a Japanese locale but has not been activated for some while.

3.15 !*loop!-print!*

Probably not used at present.

3.16 !*lower

Not yet written

3.17 !*macroexpand!-hook!*

Not yet written

3.18 !*math!-output!*

Not yet written

3.19 !*native_code

Not yet written

3.20 !*notailcall

Not yet written

3.21 !*package!*

Not yet written

3.22 !*pgwd

Not yet written

3.23 !*plap

Not yet written

3.24 !*pretty!-symmetric

Not yet written

3.25 !*prinl!-fn!*

Not yet written

3.26 !*prinl!-index!*

Not yet written

3.27 !*prinl!-visited!-nodes!*

Not yet written

3.28 !*print!-array!*

Not yet written

3.29 !*print!-length!*

Not yet written

3.30 !*print!-level!*

Not yet written

3.31 !*pwrds

Not yet written

3.32 !*query!-io!*

Not yet written

3.33 !*quotes

Not yet written

3.34 !*raise

Not yet written

3.35 !*redefmsg

Not yet written

3.36 !*resources!*

Not yet written

3.37 !*savedef

Not yet written

3.38 !*spool!-output!*

Not yet written

3.39 !*standard!-input!*

Not yet written

3.40 !*standard!-output!*

Not yet written

3.41 !*terminal!-io!*

Not yet written

3.42 !*trace!-output!*

Not yet written

3.43 !@cslbase

Not yet written

3.44 blank

Not yet written

3.45 bn

Not yet written

3.46 bufferi

Not yet written

3.47 buffero

Not yet written

3.48 common!-lisp!-mode

Not yet written

3.49 crbuf!*

Not yet written

3.50 emsg!*

Not yet written

3.51 eof!*

Not yet written

3.52 esc!*

Not yet written

3.53 indblanks

Not yet written

3.54 indentlevel

Not yet written

3.55 initialblanks

Not yet written

3.56 lispsystem!*

Not yet written

3.57 lmar

Not yet written

3.58 load!-source

Not yet written

3.59 nil

Not yet written

3.60 ofl!*

Not yet written

3.61 pendingrpars

Not yet written

3.62 program!*

Not yet written

3.63 rmar

Not yet written

3.64 rparcount

Not yet written

3.65 s!:gensym!-serial

Not yet written

3.66 stack

Not yet written

3.67 t

Not yet written

3.68 tab

Not yet written

3.69 thin!*

Not yet written

3.70 ttype!*

Not yet written

4 Items that can appear in lispsystem!*

There is a global variable called lispsystem!* whose value is reset in the process of CSL starting up. An effect of this is that if the user changes its value those changes do not survice a preserving and re-loading a heap image: this is deliberate since the heap image may be re-loaded on a different instance of CSL possibly on a quite different computer of with a different configuration. The value of lispsystem!* is a list of items, where each item is either an atomic tag of a pair whose first component is a key. In general it would be unwise to rely on exactly what information is present without review of the code that sets it up. The information may be of interest to anybody but some tags and keys are reflections of experiments rather than fullt stable facilities.

4.1 (c!-code . count)

This will be present if code has been optimised into C through the source files u01.c to u12.c, and in that case the value tells you how many functions have been optimised in this manner.

4.2 common!-lisp

For a project some while ago a limited Common Lisp compatibility mode was being developed, and this tag indicated that it was active. In that case all entries are in upper case and the variable is called *FEATURES* rather than lispsystem!*. But note that this Lisp has never even aspired to be a full Common Lisp, since its author considers Common Lisp to have been a sad mistake that must bear significant responsibility for the fact that interest in Lisp has faded dramatically since its introduction.

4.3 (compiler!-command . command)

The value associated with this key is a string that was used to compile the files of C code making up CSL. It should contain directives to set up search

paths and predefined symbols. It is intended to be used in an experiment that generates C code synamically, uses a command based on this string to compile it and then dynamically links the resulting code in with the running system.

4.4 `csl`

A simple tag intended to indicate that this Lisp system is CSL and not any other. This can of course only work properly if all other Lisp systems agree not to set this tag! In the context of Reduce I note that the PSL Lisp system sets a tag `psl` on `lispsystem!*` and the realistic use of this is to discriminate between CSL and PSL hosted copies of Reduce.

4.5 `debug`

If CSL was compiled with debugging options this is present, and one can imagine various bits of code being more cautious or more verbose if it is detected.

4.6 `(executable . name)`

The value is the fully rooted name of the executable file that was launched.

4.7 `fox`

Used to be present if the FOX GUI toolkit was detected and incorporated as part of CSL, but now probably never used!

4.8 `(linker . type)`

Intended for use in association with `compiler!-command`, the value is `win32` on Windows, `x86_64` on 64-bit Linux and other things on other systems, as detected using the program `objtype.c`.

4.9 `(name . name)`

Some indication of the platform. For instance on one system I use it is `linux-gnu:x86_64` and on anther it is just `win32`.

4.10 `(native . tag)`

One of the many experiments within CSL that were active at one stage but are not current involved compilation directly into machine code. The strong desire to ensure that image files coudl be used on a cross-platform basis led to saved compiled code being tagged with a numeric "native code tag", and this key/value pair identified the value to be used on the current machine.

4.11 `(opsys . operating-system)`

Some crude indication of the host operating system.

4.12 `pipes`

In the earlier days of CSL there were computers where pipes were not supported, so this tag notes when they are present and hance the facility to create sub-tasks through them can be used.

4.13 `record_get`

An an extension to the CSL profiling scheme it it possible to compile a special version that tracks and counts each use of property-list access functions. This can be useful because there are ways to give special treatment to a small number of flags and a small number of properties. The special-case flage end up stored as a bitmap in the symbol-header so avoid need for property-list searching. But of course recording this extra information slows things down. This tag notes when the slow version is in use. It might be used to trigger a display of statistics at the end of a calculation.

4.14 `reduce`

This is intended to report if the initial heap image is for Reduce rather than merely for Lisp.

4.15 `(shortname . name)`

Gives the short name of the current executable, without its full path.

4.16 `showmath`

If the "showmath" capability has been compiled into CSL this will be present so that Lisp code can know it is reasonable to try to use it.

4.17 `sixty!-four`

Present if the Lisp was compiled for a 64-bit computer.

4.18 `termed`

Present if a cursor-addressable console was detected.

4.19 `texmacs`

Present if the system was launched with the `--texmacs` flag. The intent is that this should only be done when it has been launched with texmacs as a front-end.

4.20 `(version . ver)`

The CSL version number.

4.21 `win32`

Present on Windows platforms, both the 32 and 64-bit variants!

4.22 `windowed`

Present if CSL is running in its own window rather than in console mode.

5 Flags and Properties

The tags here are probably not much use to end-users, but I am noting them as a matter of completeness.

5.1 `s!:ppchar and s!:ppformat`

These are used in the prettyprint code found in `extras.red`. A name is given a property `s!:ppformat` if in prettyprinted display its first few arguments should appear on the same line as it if at all possible. The `s!:ppchar` property is used to make the display of bracket characters a little more tide in the source code.

5.2 `switch`

In the Reduce parser some names are "switches", and then directives such as `on xxx` and `off xx` have the effect of setting or clearing the value of a variable `!*xxx`. This is managed by setting the `switch` flag om `xxx`. CSL sets some things as switches ready for when they may be used by the Reduce parser.

5.3 `lose`

If a name is flagged as ttfamily lose then a subsequent attempt to define or redefine it will be ignored.

5.4 `!~magic!-internal!-symbol!~`

CSL does not have a clear representation for functions that is separated from the representation of an identifier, and so when you ask to get the value of a raw function you get an identifier (probably a gensym) and this tag is used to link such values with the symbols they were originally extracted from.

6 Functions and Special Forms

Each line here shows a name and then one of the words `expr`, `fexpr` or `macro`. In some cases there can also be special treatment of functions by the compiler so that they get compiled in-line.

6.1 `abs expr`

Not yet written

6.2 `acons expr`

Not yet written

6.3 `acos expr`

Not yet written

6.4 `acosd expr`

Not yet written

6.5 `acosh expr`

Not yet written

6.6 `acot expr`

Not yet written

6.7 `acotd expr`

Not yet written

6.8 `acoth expr`

Not yet written

6.9 `acsc expr`

Not yet written

6.10 `acscd expr`

Not yet written

6.11 `acsch expr`

Not yet written

6.12 `add1 expr`

Not yet written

6.13 `and fexpr`

Not yet written

6.14 `append expr`

Not yet written

6.15 `apply expr`

Not yet written

6.16 `apply0 expr`

Not yet written

6.17 `apply1 expr`

Not yet written

6.18 `apply2 expr`

Not yet written

6.19 `apply3 expr`

Not yet written

6.20 `asec expr`

Not yet written

6.21 `asecd expr`

Not yet written

6.22 `asech expr`

Not yet written

6.23 `ash expr`

Not yet written

6.24 `ash1 expr`

Not yet written

6.25 `asin expr`

Not yet written

6.26 `asind expr`

Not yet written

6.27 `asinh expr`

Not yet written

6.28 `assoc expr`

Not yet written

6.29 `assoc!*!* expr`

Not yet written

6.30 `atan expr`

Not yet written

6.31 `atan2 expr`

Not yet written

6.32 `atan2d expr`

Not yet written

6.33 `atand expr`

Not yet written

6.34 `atanh expr`

Not yet written

6.35 `atom expr`

Not yet written

6.36 `atsoc expr`

Not yet written

6.37 `batchp expr`

Not yet written

6.38 `binary_close_input expr`

Not yet written

6.39 `binary_close_output expr`

Not yet written

6.40 `binary_open_input expr`

Not yet written

6.41 `binary_open_output expr`

Not yet written

6.42 `binary_prin1 expr`

Not yet written

6.43 `binary_prin2 expr`

Not yet written

6.44 `binary_prin3 expr`

Not yet written

6.45 `binary_prinbyte expr`

Not yet written

6.46 `binary_princ expr`

Not yet written

6.47 `binary_prinfloat expr`

Not yet written

6.48 `binary_read2 expr`

Not yet written

6.49 `binary_read3 expr`

Not yet written

6.50 `binary_read4 expr`

Not yet written

6.51 `binary_readbyte expr`

Not yet written

6.52 `binary_readfloat expr`

Not yet written

6.53 `binary_select_input expr`

Not yet written

6.54 `binary_terpri expr`

Not yet written

6.55 `binopen expr`

Not yet written

6.56 `boundp expr`

Not yet written

6.57 `bps!-getv expr`

Not yet written

6.58 `bps!-putv expr`

Not yet written

6.59 `bps!-upbv expr`

Not yet written

6.60 `bpsp expr`

Not yet written

6.61 `break!-loop expr`

Not yet written

6.62 `byte!-getv expr`

Not yet written

6.63 `bytecounts expr`

Not yet written

6.64 `c_out expr`

Not yet written

6.65 `caaaar expr`

Not yet written

6.66 `caaadr expr`

Not yet written

6.67 `caaar expr`

Not yet written

6.68 `caadar expr`

Not yet written

6.69 `caaddr expr`

Not yet written

6.70 `caadr expr`

Not yet written

6.71 `caar expr`

Not yet written

6.72 `cadaar expr`

Not yet written

6.73 `cadadr expr`

Not yet written

6.74 `cadar expr`

Not yet written

6.75 `caddar expr`

Not yet written

6.76 `cadddr expr`

Not yet written

6.77 `caddr expr`

Not yet written

6.78 `cadr expr`

Not yet written

6.79 `car expr`

Not yet written

6.80 `car!* expr`

Not yet written

6.81 `carcheck expr`

Not yet written

6.82 `catch fexpr`

Not yet written

6.83 `cbrt expr`

Not yet written

6.84 `cdaaar expr`

Not yet written

6.85 `cdaadr expr`

Not yet written

6.86 `cdaar expr`

Not yet written

6.87 `cdadar expr`

Not yet written

6.88 `cdaddr expr`

Not yet written

6.89 `cdadr expr`

Not yet written

6.90 `cdar expr`

Not yet written

6.91 `cddaar expr`

Not yet written

6.92 `cddadr expr`

Not yet written

6.93 `cddar expr`

Not yet written

6.94 `cdddar expr`

Not yet written

6.95 `cddddr expr`

Not yet written

6.96 `cdddr expr`

Not yet written

6.97 `cddr expr`

Not yet written

6.98 `cdr expr`

Not yet written

6.99 `ceiling expr`

Not yet written

6.100 `char!-code expr`

Not yet written

6.101 `char!-downcase expr`

Not yet written

6.102 `char!-upcase expr`

Not yet written

6.103 `chdir expr`

Not yet written

6.104 `check!-c!-code expr`

Not yet written

6.105 `checkpoint expr`

Not yet written

6.106 `cl!-equal expr`

Not yet written

6.107 `close expr`

Not yet written

6.108 `close!-library expr`

Not yet written

6.109 `clrhash expr`

Not yet written

6.110 `code!-char expr`

Not yet written

6.111 `codep expr`

Not yet written

6.112 `compile expr`

Not yet written

6.113 `compile!-all expr`

Not yet written

6.114 `compress expr`

Not yet written

6.115 `cond fexpr`

Not yet written

6.116 `cons expr`

Not yet written

6.117 `consp expr`

Not yet written

6.118 `constantp expr`

Not yet written

6.119 `contained expr`

Not yet written

6.120 `convert!-to!-evector expr`

Not yet written

6.121 `copy expr`

Not yet written

6.122 `copy!-module expr`

Not yet written

6.123 `copy!-native expr`

Not yet written

6.124 `cos expr`

Not yet written

6.125 `cosd expr`

Not yet written

6.126 `cosh expr`

Not yet written

6.127 `cot expr`

Not yet written

6.128 `cotd expr`

Not yet written

6.129 `coth expr`

Not yet written

6.130 `create!-directory expr`

Not yet written

6.131 `csc expr`

Not yet written

6.132 `cscd expr`

Not yet written

6.133 `csch expr`

Not yet written

6.134 `date expr`

Not yet written

6.135 `dated!-name expr`

Not yet written

6.136 `datelessp expr`

Not yet written

6.137 `datestamp expr`

Not yet written

6.138 `de fexpr`

Not yet written

6.139 `define!-in!-module expr`

Not yet written

6.140 `deflist expr`

Not yet written

6.141 `deleq expr`

Not yet written

6.142 `delete expr`

Not yet written

6.143 `delete!-file expr`

Not yet written

6.144 `delete!-module expr`

Not yet written

6.145 `difference expr`

Not yet written

6.146 `digit expr`

Not yet written

6.147 `directoryp expr`

Not yet written

6.148 `divide expr`

Not yet written

6.149 `dm fexpr`

Not yet written

6.150 `do macro`

Not yet written

6.151 `do!* macro`

Not yet written

6.152 `do!*_z2tw2evoft83 expr`

Not yet written

6.153 `do_tys294e5sboe expr`

Not yet written

6.154 `dolist macro`

Not yet written

6.155 `dolist_2oc4v2mwnrv2 expr`

Not yet written

6.156 `dotimes macro`

Not yet written

6.157 `dotimes_cm3wu6zfgv79 expr`

Not yet written

6.158 `double!-execute expr`

Not yet written

6.159 `egetv expr`

Not yet written

6.160 `eject expr`

Not yet written

6.161 `enable!-backtrace expr`

Not yet written

6.162 `encapsulatedp expr`

Not yet written

6.163 `endp expr`

Not yet written

6.164 `eputv expr`

Not yet written

6.165 `eq expr`

Not yet written

6.166 `eq!-safe expr`

Not yet written

6.167 `eqcar expr`

Not yet written

6.168 `eql expr`

Not yet written

6.169 `eqlhash expr`

Not yet written

6.170 `eqn expr`

Not yet written

6.171 `equal expr`

Not yet written

6.172 `equalcar expr`

Not yet written

6.173 `equalp expr`

Not yet written

6.174 `error expr`

Not yet written

6.175 `error1 expr`

Not yet written

6.176 `errorset expr`

Not yet written

6.177 `eupbv expr`

Not yet written

6.178 `eval expr`

Not yet written

6.179 `eval!-when fexpr`

Not yet written

6.180 `evectorp expr`

Not yet written

6.181 `evenp expr`

Not yet written

6.182 `evlis expr`

Not yet written

6.183 `exp expr`

Not yet written

6.184 `expand expr`

Not yet written

6.185 `explode expr`

Not yet written

6.186 `explode2 expr`

Not yet written

6.187 `explode2lc expr`

Not yet written

6.188 `explode2lcn expr`

Not yet written

6.189 `explode2n expr`

Not yet written

6.190 `explode2uc expr`

Not yet written

6.191 `explode2ucn expr`

Not yet written

6.192 `explodebinary expr`

Not yet written

6.193 `explodec expr`

Not yet written

6.194 `explodecn expr`

Not yet written

6.195 `explodehex expr`

Not yet written

6.196 `exploden expr`

Not yet written

6.197 `explodeoctal expr`

Not yet written

6.198 `expt expr`

Not yet written

6.199 `faslout expr`

Not yet written

6.200 `fetch!-url expr`

Not yet written

6.201 `fgetv32 expr`

Not yet written

6.202 `fgetv64 expr`

Not yet written

6.203 `file!-length expr`

Not yet written

6.204 `file!-readablep expr`

Not yet written

6.205 `file!-writeablep expr`

Not yet written

6.206 `filedate expr`

Not yet written

6.207 `filep expr`

Not yet written

6.208 `fix expr`

Not yet written

6.209 `fixp expr`

Not yet written

6.210 `flag expr`

Not yet written

6.211 `flagp expr`

Not yet written

6.212 `flagp!*!* expr`

Not yet written

6.213 `flagpcar expr`

Not yet written

6.214 `float expr`

Not yet written

6.215 `floatp expr`

Not yet written

6.216 `floor expr`

Not yet written

6.217 `fluid expr`

Not yet written

6.218 `fluidp expr`

Not yet written

6.219 `flush expr`

Not yet written

6.220 `format macro`

Not yet written

6.221 `format_vqx39lgqssd1 expr`

Not yet written

6.222 `fp!-evaluate expr`

Not yet written

6.223 `fputv32 expr`

Not yet written

6.224 `fputv64 expr`

Not yet written

6.225 `frexp expr`

Not yet written

6.226 `funcall expr`

Not yet written

6.227 `funcall!* expr`

Not yet written

6.228 `function fexpr`

Not yet written

6.229 `gcdn expr`

Not yet written

6.230 `gctime expr`

Not yet written

6.231 `gensym expr`

Not yet written

6.232 `gensym1 expr`

Not yet written

6.233 `gensym2 expr`

Not yet written

6.234 `gensymp expr`

Not yet written

6.235 `geq expr`

Not yet written

6.236 `get expr`

Not yet written

6.237 get!* expr

Not yet written

6.238 get!-current!-directory expr

Not yet written

6.239 get!-lisp!-directory expr

Not yet written

6.240 getd expr

Not yet written

6.241 getenv expr

Not yet written

6.242 gethash expr

Not yet written

6.243 getv expr

Not yet written

6.244 getv16 expr

Not yet written

6.245 getv32 expr

Not yet written

6.246 getv8 expr

Not yet written

6.247 global expr

Not yet written

6.248 globalp expr

Not yet written

6.249 `go fexpr`

Not yet written

6.250 `greaterp expr`

Not yet written

6.251 `hash!-table!-p expr`

Not yet written

6.252 `hashcontents expr`

Not yet written

6.253 `hashtagged!-name expr`

Not yet written

6.254 `hypot expr`

Not yet written

6.255 `iadd1 expr`

Not yet written

6.256 `idapply expr`

Not yet written

6.257 `idifference expr`

Not yet written

6.258 `idp expr`

Not yet written

6.259 `iequal expr`

Not yet written

6.260 `if fexpr`

Not yet written

6.261 igeq expr

Not yet written

6.262 igreaterp expr

Not yet written

6.263 ileq expr

Not yet written

6.264 ilessp expr

Not yet written

6.265 ilogand expr

Not yet written

6.266 ilogor expr

Not yet written

6.267 ilogxor expr

Not yet written

6.268 imax expr

Not yet written

6.269 imin expr

Not yet written

6.270 iminus expr

Not yet written

6.271 iminusp expr

Not yet written

6.272 indirect expr

Not yet written

6.273 inorm expr

Not yet written

6.274 input!-libraries fexpr

Not yet written

6.275 instate!-c!-code expr

Not yet written

6.276 integerp expr

Not yet written

6.277 intern expr

Not yet written

6.278 internal!-open expr

Not yet written

6.279 intersection expr

Not yet written

6.280 ionep expr

Not yet written

6.281 iplus expr

Not yet written

6.282 iplus2 expr

Not yet written

6.283 iquotient expr

Not yet written

6.284 iremainder expr

Not yet written

6.285 `irightshift expr`

Not yet written

6.286 `is!-console expr`

Not yet written

6.287 `isub1 expr`

Not yet written

6.288 `itimes expr`

Not yet written

6.289 `itimes2 expr`

Not yet written

6.290 `izerop expr`

Not yet written

6.291 `last expr`

Not yet written

6.292 `lastcar expr`

Not yet written

6.293 `lastpair expr`

Not yet written

6.294 `lcmn expr`

Not yet written

6.295 `length expr`

Not yet written

6.296 `lengthc expr`

Not yet written

6.297 `leq expr`

Not yet written

6.298 `lessp expr`

Not yet written

6.299 `let!* fexpr`

Not yet written

6.300 `library!-members expr`

Returns a list of all the modules that could potentially be loaded using `load!-module`. See `list!-modules` to get a human readable display that looks more like the result of listing a directory, or `modulep` for checking the state of a particular named module.

6.301 `library!-name expr`

Not yet written

6.302 `linelength expr`

Not yet written

6.303 `list fexpr`

Not yet written

6.304 `list!* fexpr`

Not yet written

6.305 `list!-directory expr`

Not yet written

6.306 `list!-modules expr`

This prints a human-readable display of the modules present in the current image files. This will include "InitialImage" which is the heap-image loaded at system startup. For example

```
> (list!-modules)

File d:\csl\csl.img (dirsize 8  length 155016, Writable):
  compat       Sat Jul 26 10:20:08 2008  position 556   size: 9320
  compiler     Sat Jul 26 10:20:08 2008  position 9880  size: 81088
  InitialImage Sat Jul 26 10:20:09 2008  position 90972 size: 64040

nil
```

See `library!-members` and `modulep` for functions that make it possible for Lisp code to discover about the loadable modules that are available.

6.307 `list!-to!-string expr`

Not yet written

6.308 `list!-to!-symbol expr`

Not yet written

6.309 `list!-to!-vector expr`

Not yet written

6.310 `list2 expr`

Not yet written

6.311 `list2!* expr`

Not yet written

6.312 `list3 expr`

Not yet written

6.313 `list3!* expr`

Not yet written

6.314 `list4 expr`

Not yet written

6.315 `liter expr`

Not yet written

6.316 `ln expr`

Not yet written

6.317 `load!-module expr`

Not yet written

6.318 `load!-source expr`

Not yet written

6.319 `log expr`

Not yet written

6.320 `log10 expr`

Not yet written

6.321 `logand expr`

Not yet written

6.322 `logb expr`

Not yet written

6.323 `logeqv expr`

Not yet written

6.324 `lognot expr`

Not yet written

6.325 `logor expr`

Not yet written

6.326 `logxor expr`

Not yet written

6.327 `lose!-precision expr`

Not yet written

6.328 `lposn expr`

Not yet written

6.329 `lsd expr`

Not yet written

6.330 `macro!-function expr`

Not yet written

6.331 `macroexpand expr`

Not yet written

6.332 `macroexpand!-1 expr`

Not yet written

6.333 `make!-bps expr`

Not yet written

6.334 `make!-function!-stream expr`

Not yet written

6.335 `make!-global expr`

Not yet written

6.336 `make!-native expr`

Not yet written

6.337 `make!-random!-state expr`

Not yet written

6.338 `make!-simple!-string expr`

Not yet written

6.339 `make!-special expr`

Not yet written

6.340 map expr

Not yet written

6.341 mapc expr

Not yet written

6.342 mapcan expr

Not yet written

6.343 mapcar expr

Not yet written

6.344 mapcon expr

Not yet written

6.345 maphash expr

Not yet written

6.346 maple_atomic_value expr

Not yet written

6.347 maple_component expr

Not yet written

6.348 maple_integer expr

Not yet written

6.349 maple_length expr

Not yet written

6.350 maple_string_data expr

Not yet written

6.351 maple_tag expr

Not yet written

6.352 `maplist expr`

Not yet written

6.353 `mapstore expr`

Not yet written

6.354 `math!-display expr`

Not yet written

6.355 `max expr`

Not yet written

6.356 `max2 expr`

Not yet written

6.357 `md5 expr`

Not yet written

6.358 `md60 expr`

Not yet written

6.359 `member expr`

Not yet written

6.360 `member!*!* expr`

Not yet written

6.361 `memq expr`

Not yet written

6.362 `min expr`

Not yet written

6.363 `min2 expr`

Not yet written

6.364 `minus expr`

Not yet written

6.365 `minusp expr`

Not yet written

6.366 `mkevect expr`

Not yet written

6.367 `mkfvect32 expr`

Not yet written

6.368 `mkfvect64 expr`

Not yet written

6.369 `mkhash expr`

Not yet written

6.370 `mkquote expr`

Not yet written

6.371 `mkvect expr`

Not yet written

6.372 `mkvect16 expr`

Not yet written

6.373 `mkvect32 expr`

Not yet written

6.374 `mkvect8 expr`

Not yet written

6.375 `mkxvect expr`

Not yet written

6.376 mod expr

Not yet written

6.377 modular!-difference expr

Not yet written

6.378 modular!-expt expr

Not yet written

6.379 modular!-minus expr

Not yet written

6.380 modular!-number expr

Not yet written

6.381 modular!-plus expr

Not yet written

6.382 modular!-quotient expr

Not yet written

6.383 modular!-reciprocal expr

Not yet written

6.384 modular!-times expr

Not yet written

6.385 modulep expr

This takes a single argument and checks whether there is a loadable module of that name. If there is not then `nil` is returned, otherwise a string that indicates the date-stamp on the module is given. See `datelessp` for working with such dates, and `library!-members` for finding a list of all modules that are available.

6.386 mpi_allgather expr

Not yet written

6.387 mpi_alltoall expr

Not yet written

6.388 mpi_barrier expr

Not yet written

6.389 mpi_bcast expr

Not yet written

6.390 mpi_comm_rank expr

Not yet written

6.391 mpi_comm_size expr

Not yet written

6.392 mpi_gather expr

Not yet written

6.393 mpi_iprobe expr

Not yet written

6.394 mpi_irecv expr

Not yet written

6.395 mpi_isend expr

Not yet written

6.396 mpi_probe expr

Not yet written

6.397 mpi_recv expr

Not yet written

6.398 mpi_scatter expr

Not yet written

6.399 mpi_send expr

Not yet written

6.400 mpi_sendrecv expr

Not yet written

6.401 mpi_test expr

Not yet written

6.402 mpi_wait expr

Not yet written

6.403 msd expr

Not yet written

6.404 native!-address expr

Not yet written

6.405 native!-getv expr

Not yet written

6.406 native!-putv expr

Not yet written

6.407 native!-type expr

Not yet written

6.408 nconc expr

Not yet written

6.409 ncons expr

Not yet written

6.410 neq expr

Not yet written

6.411 noisy!-setq fexpr

Not yet written

6.412 not expr

Not yet written

6.413 nreverse expr

Not yet written

6.414 null expr

Not yet written

6.415 numberp expr

Not yet written

6.416 oblist expr

Not yet written

6.417 oddp expr

Not yet written

6.418 oem!-supervisor expr

Not yet written

6.419 onep expr

Not yet written

6.420 open expr

Not yet written

6.421 open!-library expr

Not yet written

6.422 open!-url expr

Not yet written

6.423 or fexpr

Not yet written

6.424 orderp expr

Not yet written

6.425 ordp expr

Not yet written

6.426 output!-library fexpr

Not yet written

6.427 pagelength expr

Not yet written

6.428 pair expr

Not yet written

6.429 pairp expr

Not yet written

6.430 parallel expr

Not yet written

6.431 peekch expr

Not yet written

6.432 pipe!-open expr

Not yet written

6.433 plist expr

Not yet written

6.434 plus fexpr

Not yet written

6.435 plus2 expr

Not yet written

6.436 plus_4lcok6r6bp3g expr

Not yet written

6.437 plusp expr

Not yet written

6.438 posn expr

Not yet written

6.439 preserve expr

Not yet written

6.440 prettyprint expr

Not yet written

6.441 prin expr

Not yet written

6.442 prin1 expr

Not yet written

6.443 prin2 expr

Not yet written

6.444 prin2a expr

Not yet written

6.445 prinbinary expr

Not yet written

6.446 princ expr

Not yet written

6.447 `princ!-downcase expr`

Not yet written

6.448 `princ!-upcase expr`

Not yet written

6.449 `princl expr`

Not yet written

6.450 `prinhex expr`

Not yet written

6.451 `prinl expr`

Not yet written

6.452 `prinoctal expr`

Not yet written

6.453 `prinraw expr`

Not yet written

6.454 `print expr`

Not yet written

6.455 `print!-config!-header expr`

Not yet written

6.456 `print!-csl!-headers expr`

Not yet written

6.457 `print!-imports expr`

Not yet written

6.458 `printc expr`

Not yet written

6.459 printcl expr

Not yet written

6.460 printl expr

Not yet written

6.461 printprompt expr

Not yet written

6.462 prog fexpr

Not yet written

6.463 prog1 fexpr

Not yet written

6.464 prog2 fexpr

Not yet written

6.465 progn fexpr

Not yet written

6.466 protect!-symbols expr

Not yet written

6.467 protected!-symbol!-warn expr

Not yet written

6.468 psetq macro

Not yet written

6.469 psetq_vg20v16gc5na expr

Not yet written

6.470 put expr

Not yet written

6.471 `putc expr`

Not yet written

6.472 `putd expr`

Not yet written

6.473 `puthash expr`

Not yet written

6.474 `putv expr`

Not yet written

6.475 `putv!-char expr`

Not yet written

6.476 `putv16 expr`

Not yet written

6.477 `putv32 expr`

Not yet written

6.478 `putv8 expr`

Not yet written

6.479 `qcaar expr`

Not yet written

6.480 `qcadr expr`

Not yet written

6.481 `qcar expr`

Not yet written

6.482 `qcdar expr`

Not yet written

6.483 qcddr expr

Not yet written

6.484 qcdr expr

Not yet written

6.485 qgetv expr

Not yet written

6.486 qputv expr

Not yet written

6.487 quote fexpr

Not yet written

6.488 quotient expr

Not yet written

6.489 random!-fixnum expr

Not yet written

6.490 random!-number expr

Not yet written

6.491 rassoc expr

Not yet written

6.492 rational expr

Not yet written

6.493 rdf expr

Not yet written

6.494 rds expr

Not yet written

6.495 `read expr`

Not yet written

6.496 `readb expr`

Not yet written

6.497 `readch expr`

Not yet written

6.498 `readline expr`

Not yet written

6.499 `reclaim expr`

Not yet written

6.500 `remainder expr`

Not yet written

6.501 `remd expr`

Not yet written

6.502 `remflag expr`

Not yet written

6.503 `remhash expr`

Not yet written

6.504 `remob expr`

Not yet written

6.505 `remprop expr`

Not yet written

6.506 `rename!-file expr`

Not yet written

6.507 representation expr

Not yet written

6.508 resource!-exceeded expr

Not yet written

6.509 resource!-limit expr

Not yet written

6.510 restart!-csl expr

Not yet written

6.511 restore!-c!-code expr

Not yet written

6.512 return fexpr

Not yet written

6.513 reverse expr

Not yet written

6.514 reversip expr

Not yet written

6.515 round expr

Not yet written

6.516 rplaca expr

Not yet written

6.517 rplacd expr

Not yet written

6.518 rplacw expr

Not yet written

6.519 `rseek expr`

Not yet written

6.520 `rtell expr`

Not yet written

6.521 `s!:blankcount macro`

Not yet written

6.522 `s!:blankcount_di4u8tiv3pra expr`

Not yet written

6.523 `s!:blanklist macro`

Not yet written

6.524 `s!:blanklist_3grr8hhc8kse expr`

Not yet written

6.525 `s!:blankp macro`

Not yet written

6.526 `s!:blankp_q4md8q4t32hd expr`

Not yet written

6.527 `s!:depth macro`

Not yet written

6.528 `s!:depth_nywe93u7asd2 expr`

Not yet written

6.529 `s!:do!-bindings expr`

Not yet written

6.530 `s!:do!-endtest expr`

Not yet written

6.531 s!:do!-result expr

Not yet written

6.532 s!:do!-updates expr

Not yet written

6.533 s!:endlist expr

Not yet written

6.534 s!:expand!-do expr

Not yet written

6.535 s!:expand!-dolist expr

Not yet written

6.536 s!:expand!-dotimes expr

Not yet written

6.537 s!:explodes expr

Not yet written

6.538 s!:finishpending expr

Not yet written

6.539 s!:format expr

Not yet written

6.540 s!:indenting macro

Not yet written

6.541 s!:indenting_uugpn161oe9g expr

Not yet written

6.542 s!:make!-psetq!-assignments expr

Not yet written

6.543 s!:make!-psetq!-bindings expr

Not yet written

6.544 s!:make!-psetq!-vars expr

Not yet written

6.545 s!:newframe macro

Not yet written

6.546 s!:newframe_jj3e2mkec583 expr

Not yet written

6.547 s!:oblist expr

Not yet written

6.548 s!:oblist1 expr

Not yet written

6.549 s!:overflow expr

Not yet written

6.550 s!:prindent expr

Not yet written

6.551 s!:prinl0 expr

Not yet written

6.552 s!:prinl1 expr

Not yet written

6.553 s!:prinl2 expr

Not yet written

6.554 s!:prvector expr

Not yet written

6.555 s!:putblank expr

Not yet written

6.556 s!:putch expr

Not yet written

6.557 s!:quotep expr

Not yet written

6.558 s!:setblankcount macro

Not yet written

6.559 s!:setblankcount_wqtabtq2ayhf expr

Not yet written

6.560 s!:setblanklist macro

Not yet written

6.561 s!:setblanklist_yx45qh074fy7 expr

Not yet written

6.562 s!:setindenting macro

Not yet written

6.563 s!:setindenting_wlwn13x1f3y expr

Not yet written

6.564 s!:stamp expr

Not yet written

6.565 s!:top macro

Not yet written

6.566 s!:top_su2dv6yphmp9 expr

Not yet written

6.567 `safe!-fp!-pl expr`

Not yet written

6.568 `safe!-fp!-pl0 expr`

Not yet written

6.569 `safe!-fp!-plus expr`

Not yet written

6.570 `safe!-fp!-quot expr`

Not yet written

6.571 `safe!-fp!-times expr`

Not yet written

6.572 `sample expr`

Not yet written

6.573 `sassoc expr`

Not yet written

6.574 `schar expr`

Not yet written

6.575 `scharn expr`

Not yet written

6.576 `sec expr`

Not yet written

6.577 `secd expr`

Not yet written

6.578 `sech expr`

Not yet written

6.579 `seprp expr`

Not yet written

6.580 `set expr`

Not yet written

6.581 `set!-autoload expr`

Not yet written

6.582 `set!-help!-file expr`

Not yet written

6.583 `set!-print!-precision expr`

Not yet written

6.584 `set!-small!-modulus expr`

Not yet written

6.585 `setpchar expr`

Not yet written

6.586 `setq fexpr`

Not yet written

6.587 `silent!-system expr`

Not yet written

6.588 `simple!-string!-p expr`

Not yet written

6.589 `simple!-vector!-p expr`

Not yet written

6.590 `sin expr`

Not yet written

6.591 `sind expr`

Not yet written

6.592 `sinh expr`

Not yet written

6.593 `smemq expr`

Not yet written

6.594 `sort expr`

Not yet written

6.595 `sortip expr`

Not yet written

6.596 `spaces expr`

Not yet written

6.597 `special!-char expr`

Not yet written

6.598 `special!-form!-p expr`

Not yet written

6.599 `spool expr`

Not yet written

6.600 `sqrt expr`

Not yet written

6.601 `stable!-sort expr`

Not yet written

6.602 `stable!-sortip expr`

Not yet written

6.603 start!-module expr

Not yet written

6.604 startup!-banner expr

Not yet written

6.605 stop expr

Not yet written

6.606 streamp expr

Not yet written

6.607 stringp expr

Not yet written

6.608 sub1 expr

Not yet written

6.609 subla expr

Not yet written

6.610 sublis expr

Not yet written

6.611 subst expr

Not yet written

6.612 superprinm expr

Not yet written

6.613 superprintm expr

Not yet written

6.614 sxhash expr

Not yet written

6.615 `symbol!-argcode expr`

Not yet written

6.616 `symbol!-argcount expr`

Not yet written

6.617 `symbol!-env expr`

Not yet written

6.618 `symbol!-fastgets expr`

Not yet written

6.619 `symbol!-fn!-cell expr`

Not yet written

6.620 `symbol!-function expr`

Not yet written

6.621 `symbol!-make!-fastget expr`

Not yet written

6.622 `symbol!-name expr`

Not yet written

6.623 `symbol!-protect expr`

Not yet written

6.624 `symbol!-restore!-fns expr`

Not yet written

6.625 `symbol!-set!-definition expr`

Not yet written

6.626 `symbol!-set!-env expr`

Not yet written

6.627 `symbol!-set!-native expr`

Not yet written

6.628 `symbol!-value expr`

Not yet written

6.629 `symbolp expr`

Not yet written

6.630 `symerr expr`

Not yet written

6.631 `system expr`

Not yet written

6.632 `tagbody fexpr`

Not yet written

6.633 `tan expr`

Not yet written

6.634 `tand expr`

Not yet written

6.635 `tanh expr`

Not yet written

6.636 `terpri expr`

Not yet written

6.637 `threevectorp expr`

Not yet written

6.638 `throw fexpr`

Not yet written

6.639 `time expr`

Not yet written

6.640 `times fexpr`

Not yet written

6.641 `times2 expr`

Not yet written

6.642 `times_z6u5f3t8dwo4 expr`

Not yet written

6.643 `tmpnam expr`

Not yet written

6.644 `trace expr`

Not yet written

6.645 `trace!-all expr`

Not yet written

6.646 `traceset expr`

Not yet written

6.647 `traceset1 expr`

Not yet written

6.648 `truename expr`

Not yet written

6.649 `truncate expr`

Not yet written

6.650 `ttab expr`

Not yet written

6.651 `tyo expr`

Not yet written

6.652 `undouble!-execute expr`

Not yet written

6.653 `unfluid expr`

Not yet written

6.654 `unglobal expr`

Not yet written

6.655 `union expr`

Not yet written

6.656 `unless fexpr`

Not yet written

6.657 `unmake!-global expr`

Not yet written

6.658 `unmake!-special expr`

Not yet written

6.659 `unreadch expr`

Not yet written

6.660 `untrace expr`

Not yet written

6.661 `untraceset expr`

Not yet written

6.662 `untraceset1 expr`

Not yet written

6.663 unwind!-protect fexpr

Not yet written

6.664 upbv expr

Not yet written

6.665 user!-homedir!-pathname expr

Not yet written

6.666 vectorp expr

Not yet written

6.667 verbos expr

Not yet written

6.668 when fexpr

Not yet written

6.669 where!-was!-that expr

Not yet written

6.670 window!-heading expr

Not yet written

6.671 writable!-libraryp expr

Not yet written

6.672 write!-module expr

Not yet written

6.673 wrs expr

Not yet written

6.674 xassoc expr

Not yet written

6.675 `xcons expr`

Not yet written

6.676 `xdifference expr`

Not yet written

6.677 `xtab expr`

Not yet written

6.678 `zerop expr`

Not yet written

6.679 `!~block fexpr`

Not yet written

6.680 `!~let fexpr`

Not yet written

6.681 `!~tyi expr`

Not yet written