CSL reference

A C Norman

November 17, 2011

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.

`--` 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.

`--help` It is probably obvious what this option does! Note that on Windows the application was linked as a windows binary so it carefully creates a console to display the help text in, and organizes a delay to give people a chance to read it.

`--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?

`--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.

`-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!

`-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!

`-c` Displays a notice relating to the authorship of CSL. Note that this is an authorship statement not a Copyright notice, because if any (L)GPL code is involved that would place requirements on what was displayed in a Copyright Notice.

`-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). Note that the value set is a *string* so if you wish to retrieve it and use it as a symbold or number within your code you will have to perform some conversion.

`-e` A "spare" option used from time to time to activate experiments within CSL.

`-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, but is currently disabled.

`-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`.

`-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.

`-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.

`-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.

`-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 for common-sized computations. In general CSL will automatically expand its heap, and so it should normally never be necessary to use this option.

`-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.

`-m` Memory trace mode. An option that represents an experiment from the past, and no longer reliably in use. It make it possible to force an exception at stages whene reference to a specified part of memory was made and that could be useful for some low level debugging. It is not supported at present.

`-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. This is intended for experts to do disaster recovery and diagnosis of damaged image files.

`-o` See `-i`. This specifies an image file used for output via `faslout` and `reserve`.

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

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

`-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.

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

`-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.

`-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.

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

`-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.

`-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.

`-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.

`-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.

3 Predefined variables

`!!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.

`!$eof!$` The value of this variable is a pseudo-character returned from various read functions to signal end-of-file.

`!$eol!$` The value of this variable is an end-of-line character.

`!*plap` Not yet written

`!*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!

`!*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!

`!*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.

`!*comp` When set each function is compiled (into bytecodes) as it gets defined.

`!*debug!-io!*` An I/O channel intended to be used for diagnostic interactions.

`!*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.

`!*error!-messages!*` Has the value nil and does not do anything!

`!*error!-output!*` An I/O channel intended for diagnostic output.

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

`!*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.

`!*hankaku` This was concerned with internationalisation to support a Japanese locale but has not been activated for some while. In the fullness of time I hope to migrate CSL to use an UTF8 representation of Unicode characters internally, but that upgrade is at present an ideal and a project not a reality. Volunteers to help welcomed.

`!*loop!-print!*` Probably not used at present.

`!*lower` Not yet written

`!*macroexpand!-hook!*` Not yet written

`!*math!-output!*` Not yet written

`!*native_code` Not yet written

`!*notailcall` Not yet written

`!*package!*` Not yet written

`!*pgwd` Not yet written

`!*pretty!-symmetric` Not yet written

`!*prinl!-fn!*` Not yet written

`!*prinl!-index!*` Not yet written

`!*prinl!-visited!-nodes!*` Not yet written

`!*print!-array!*` Not yet written

`!*print!-length!*` Not yet written

`!*print!-level!*` Not yet written

`!*pwrds` Not yet written

`!*query!-io!*` Not yet written

`!*quotes` Not yet written

`!*raise` Not yet written

`!*redefmsg` Not yet written

`!*resources!*` Not yet written

`!*savedef` Not yet written

`!*spool!-output!*` Not yet written

`!*standard!-input!*` Not yet written

`!*standard!-output!*` Not yet written

`!*terminal!-io!*` Not yet written

`!*trace!-output!*` Not yet written

`!@cslbase` Not yet written
]pendingrpars]

`pendingrpars` Not yet written

`blank` The value of this variable is an space or blank character. This might otherwise be written as "`! `".

`bn` Not yet written

`bufferi` Not yet written

`bufferp` Not yet written

`common!-lisp!-mode` Not yet written

`crbuf!*` Not yet written

`emsg!*` Not yet written

`eof!*` Not yet written

`esc!*` The value of this variable is the character "escape". As a non-printing character use of this is to be viewed as delicate.

`indblanks` Not yet written

`indentlevel` Not yet written

`initialblanks` Not yet written

`lispsystem!*` Not yet written

`lmar` Not yet written

`load!-source` Not yet written

`nil` Not yet written

`ofl!*` Not yet written

`program!*` Not yet written

`rmar` Not yet written

`rparcount` Not yet written

`s!:gensym!-serial` Not yet written

`stack` Not yet written

`t` Not yet written

`tab` The value of this variable is a tab character.

`thin!*` Not yet written

`ttype!*` Not yet written

/*!! flags [04] Flags and Properties

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

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 fully stable facilities.

`(c!-code . count)` This will be present if code has been optimised into C through the source files u01.c to u60.c, and in that case the value tells you how many functions have been optimised in this manner.

`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.

`(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.

`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.

`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.

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

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

`(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`.

`(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`.

`(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.

`(opsys . operating-system)` Some crude indication of the host operating system.

operating system identity The name of the current operating system is put on the list. Exactly what form is not explicitly defined!

`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.

`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.

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

`(shortname . name)` Gives the short name of the current executable, without its full path.

`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.

`sixty!-four` Present if the Lisp was compiled for a 64-bit computer.

`termed` Present if a cursor-addressable console was detected.

`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.

`(version . ver)` The CSL version number.

`win32`, `win64` Any windows system puts `win32` in `lispsystem!*`. If 64-bit windows is is use then `win64` is also included

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

4 Flags and Properties

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

`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.

`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.

`!~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.

5 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.

`abs` ***expr*** Not yet written

acons expr Not yet written

acos expr Not yet written

acosd expr Not yet written

acosh expr Not yet written

acot expr Not yet written

acotd expr Not yet written

acoth expr Not yet written

acsc expr Not yet written

acscd expr Not yet written

acsch expr Not yet written

add1 expr Not yet written

and fexpr Not yet written

append expr Not yet written

apply expr Not yet written

apply0 expr Not yet written

apply1 expr Not yet written

apply2 expr Not yet written

apply3 expr Not yet written

asec expr Not yet written

asecd expr Not yet written

asech expr Not yet written

ash expr Not yet written

ash1 expr Not yet written

asin expr Not yet written

asind expr Not yet written

asinh expr Not yet written

assoc expr Not yet written

assoc!*!* expr Not yet written

atan expr Not yet written

atan2 expr Not yet written

atan2d expr Not yet written

atand expr Not yet written

atanh expr Not yet written

atom expr Not yet written

atsoc expr Not yet written

batchp expr Not yet written

binary_close_input expr Not yet written

binary_close_output expr Not yet written

binary_open_input expr Not yet written

binary_open_output expr Not yet written

binary_prin1 expr Not yet written

binary_prin2 expr Not yet written

binary_prin3 expr Not yet written

binary_prinbyte expr Not yet written

binary_princ expr Not yet written

binary_prinfloat expr Not yet written

binary_read2 expr Not yet written

binary_read3 expr Not yet written

binary_read4 expr Not yet written

binary_readbyte expr Not yet written

binary_readfloat expr Not yet written

binary_select_input expr Not yet written

binary_terpri expr Not yet written

binopen expr Not yet written

boundp expr Not yet written

bps!-getv expr Not yet written

bps!-putv expr Not yet written

bps!-upbv expr Not yet written

bpsp expr Not yet written

break!-loop expr Not yet written

byte!-getv expr Not yet written

bytecounts expr Not yet written

c_out expr Not yet written

carcheck expr Not yet written

catch fexpr Not yet written

cbrt expr Not yet written

ceiling expr Not yet written

char!-code expr Not yet written

char!-downcase expr Not yet written

char!-upcase expr Not yet written

chdir expr Not yet written

check!-c!-code expr Not yet written

checkpoint expr Not yet written

cl!-equal expr Not yet written

close expr Not yet written

close!-library expr Not yet written

clrhash expr Not yet written

code!-char expr Not yet written

codep expr Not yet written

compile expr Not yet written

compile!-all expr Not yet written

compress expr Not yet written

cond fexpr Not yet written

cons expr Not yet written

consp expr Not yet written

constantp expr Not yet written

contained expr Not yet written

convert!-to!-evector expr Not yet written

copy expr Not yet written

copy!-module expr Not yet written

copy!-native expr Not yet written

cos expr Not yet written

cosd expr Not yet written

cosh expr Not yet written

cot expr Not yet written

cotd expr Not yet written

coth expr Not yet written

create!-directory expr Not yet written

csc expr Not yet written

cscd expr Not yet written

csch expr Not yet written

date expr Not yet written

dated!-name expr Not yet written

datelessp expr Not yet written

datestamp expr Not yet written

de fexpr Not yet written

define!-in!-module expr Not yet written

deflist expr Not yet written

deleq expr Not yet written

delete expr Not yet written

delete!-file expr Not yet written

delete!-module expr Not yet written

difference expr Not yet written

digit expr Not yet written

directoryp expr Not yet written

divide expr Not yet written

dm fexpr Not yet written

do macro Not yet written

do!* macro Not yet written

dolist macro Not yet written

dotimes macro Not yet written

double!-execute expr Not yet written

egetv expr Not yet written

eject expr Not yet written

enable!-backtrace expr Not yet written

enable!-errorset expr Not yet written

encapsulatedp expr Not yet written

endp expr Not yet written

eputv expr Not yet written

eq expr Not yet written

eq!-safe expr Not yet written

eqcar expr Not yet written

eql expr Not yet written

eqlhash expr Not yet written

eqn expr Not yet written

equal expr Not yet written

equalcar expr Not yet written

equalp expr Not yet written

error expr Not yet written

error1 expr Not yet written

errorset expr Not yet written

eupbv expr Not yet written

eval expr Not yet written

eval!-when fexpr Not yet written

evectorp expr Not yet written

evenp expr Not yet written

evlis expr Not yet written

exp expr Not yet written

expand expr Not yet written

explode expr Not yet written

explode2 expr Not yet written

explode2lc expr Not yet written

explode2lcn expr Not yet written

explode2n expr Not yet written

explode2uc expr Not yet written

explode2ucn expr Not yet written

explodebinary expr Not yet written

explodec expr Not yet written

explodecn expr Not yet written

explodehex expr Not yet written

exploden expr Not yet written

explodeoctal expr Not yet written

expt expr Not yet written

faslout expr Not yet written

fetch!-url expr Not yet written

fgetv32 expr Not yet written

fgetv64 expr Not yet written

file!-length expr Not yet written

file!-readablep expr Not yet written

file!-writeablep expr Not yet written

filedate expr Not yet written

filep expr Not yet written

fix expr Not yet written

fixp expr Not yet written

flag expr Not yet written

flagp expr Not yet written

flagp!*!* expr Not yet written

flagpcar expr Not yet written

float expr Not yet written

floatp expr Not yet written

floor expr Not yet written

fluid expr Not yet written

fluidp expr Not yet written

flush expr Not yet written

format macro Not yet written

fp!-evaluate expr Not yet written

fputv32 expr Not yet written

fputv64 expr Not yet written

frexp expr Not yet written

funcall expr Not yet written

funcall!* expr Not yet written

function fexpr Not yet written

gcdn expr Not yet written

gctime expr Not yet written

gensym expr Not yet written

gensym1 expr Not yet written

gensym2 expr Not yet written

gensymp expr Not yet written

geq expr Not yet written

get expr Not yet written

get!* expr Not yet written

get!-current!-directory expr Not yet written

get!-lisp!-directory expr Not yet written

getd expr Not yet written

getenv expr Not yet written

gethash expr Not yet written

getv expr Not yet written

getv16 expr Not yet written

getv32 expr Not yet written

getv8 expr Not yet written

global expr Not yet written

globalp expr Not yet written

go fexpr Not yet written

greaterp expr Not yet written

hash!-table!-p expr Not yet written

hashcontents expr Not yet written

hashtagged!-name expr Not yet written

hypot expr Not yet written

iadd1 expr Not yet written

idapply expr Not yet written

idifference expr Not yet written

idp expr Not yet written

iequal expr Not yet written

if fexpr Not yet written

igeq expr Not yet written

igreaterp expr Not yet written

ileq expr Not yet written

ilessp expr Not yet written

ilogand expr Not yet written

ilogor expr Not yet written

ilogxor expr Not yet written

imax expr Not yet written

imin expr Not yet written

iminus expr Not yet written

iminusp expr Not yet written

indirect expr Not yet written

inorm expr Not yet written

input!-libraries fexpr Not yet written

instate!-c!-code expr Not yet written

integerp expr Not yet written

internal!-open expr Not yet written

intern expr Not yet written

intersection expr Not yet written

ionep expr Not yet written

iplus expr Not yet written

iplus2 expr Not yet written

iquotient expr Not yet written

iremainder expr Not yet written

irightshift expr Not yet written

is!-console expr Not yet written

isub1 expr Not yet written

itimes expr Not yet written

itimes2 expr Not yet written

izerop expr Not yet written

last expr Not yet written

lastcar expr Not yet written

lastpair expr Not yet written

lcmn expr Not yet written

length expr Not yet written

lengthc expr Not yet written

leq expr Not yet written

lessp expr Not yet written

let!* fexpr Not yet written

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.

library!-name expr Not yet written

linelength expr Not yet written

list fexpr Not yet written

list!* fexpr Not yet written

list!-directory expr Not yet written

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.

list!-to!-string expr Not yet written

list!-to!-symbol expr Not yet written

list!-to!-vector expr Not yet written

list2 expr Not yet written

list2!* expr Not yet written

list3 expr Not yet written

list3!* expr Not yet written

list4 expr Not yet written

liter expr Not yet written

ln expr Not yet written

load!-module expr Not yet written

load!-source expr Not yet written

log expr Not yet written

log10 expr Not yet written

logand expr Not yet written

logb expr Not yet written

logeqv expr Not yet written

lognot expr Not yet written

logor expr Not yet written

logxor expr Not yet written

lose!-precision expr Not yet written

lposn expr Not yet written

lsd expr Not yet written

macro!-function expr Not yet written

macroexpand expr Not yet written

macroexpand!-1 expr Not yet written

make!-bps expr Not yet written

make!-function!-stream expr Not yet written

make!-global expr Not yet written

make!-native expr Not yet written

make!-random!-state expr Not yet written

make!-simple!-string expr Not yet written

make!-special expr Not yet written

map expr Not yet written

mapc expr Not yet written

mapcan expr Not yet written

mapcar expr Not yet written

mapcon expr Not yet written

maphash expr Not yet written

maple_atomic_value expr Not yet written

maple_component expr Not yet written

maple_integer expr Not yet written

maple_length expr Not yet written

maple_string_data expr Not yet written

maple_tag expr Not yet written

maplist expr Not yet written

mapstore expr Not yet written

math!-display expr Not yet written

max expr Not yet written

max2 expr Not yet written

md5 expr Not yet written

md60 expr Not yet written

member expr Not yet written

member!*!* expr Not yet written

memq expr Not yet written

min expr Not yet written

min2 expr Not yet written

minus expr Not yet written

minusp expr Not yet written

mkevect expr Not yet written

mkfvect32 expr Not yet written

mkfvect64 expr Not yet written

mkhash expr Not yet written

mkquote expr Not yet written

mkvect expr Not yet written

mkvect16 expr Not yet written

mkvect32 expr Not yet written

mkvect8 expr Not yet written

mkxvect expr Not yet written

mod expr Not yet written

modular!-difference expr Not yet written

modular!-expt expr Not yet written

modular!-minus expr Not yet written

modular!-number expr Not yet written

modular!-plus expr Not yet written

modular!-quotient expr Not yet written

modular!-reciprocal expr Not yet written

modular!-times expr Not yet written

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.

mpi_allgather expr Not yet written

mpi_alltoall expr Not yet written

mpi_barrier expr Not yet written

mpi_bcast expr Not yet written

mpi_comm_rank expr Not yet written

mpi_comm_size expr Not yet written

mpi_gather expr Not yet written

mpi_iprobe expr Not yet written

mpi_irecv expr Not yet written

mpi_isend expr Not yet written

mpi_probe expr Not yet written

mpi_recv expr Not yet written

mpi_scatter expr Not yet written

mpi_send expr Not yet written

mpi_sendrecv expr Not yet written

mpi_test expr Not yet written

mpi_wait expr Not yet written

msd expr Not yet written

native!-address expr Not yet written

native!-getv expr Not yet written

native!-putv expr Not yet written

native!-type expr Not yet written

nconc expr Not yet written

ncons expr Not yet written

neq expr Not yet written

noisy!-setq fexpr Not yet written

not expr Not yet written

nreverse expr Not yet written

null expr Not yet written

numberp expr Not yet written

oblist expr Not yet written

oddp expr Not yet written

oem!-supervisor expr Not yet written

onep expr Not yet written

open expr Not yet written

open!-library expr Not yet written

open!-url expr Not yet written

or fexpr Not yet written

orderp expr Not yet written

ordp expr Not yet written

output!-library fexpr Not yet written

pagelength expr Not yet written

pair expr Not yet written

pairp expr Not yet written

parallel expr Not yet written

peekch expr Not yet written

pipe!-open expr Not yet written

plist expr Not yet written

plus fexpr Not yet written

plus2 expr Not yet written

plusp expr Not yet written

posn expr Not yet written

preserve expr Not yet written

prettyprint expr Not yet written

prin expr Not yet written

prin1 expr Not yet written

prin2 expr Not yet written

prin2a expr Not yet written

prinbinary expr Not yet written

princ expr Not yet written

princ!-downcase expr Not yet written

princ!-upcase expr Not yet written

princl expr Not yet written

prinhex expr Not yet written

prinl expr Not yet written

prinoctal expr Not yet written

prinraw expr Not yet written

print expr Not yet written

print!-config!-header expr Not yet written

print!-csl!-headers expr Not yet written

print!-imports expr Not yet written

printc expr Not yet written

printcl expr Not yet written

printl expr Not yet written

printprompt expr Not yet written

prog fexpr Not yet written

prog1 fexpr Not yet written

prog2 fexpr Not yet written

progn fexpr Not yet written

protect!-symbols expr Not yet written

protected!-symbol!-warn expr Not yet written

psetq macro Not yet written

put expr Not yet written

putc expr Not yet written

putd expr Not yet written

puthash expr Not yet written

putv expr Not yet written

putv!-char expr Not yet written

putv16 expr Not yet written

putv32 expr Not yet written

putv8 expr Not yet written

qcaar expr Not yet written

qcadr expr Not yet written

qcar expr Not yet written

qcdar expr Not yet written

qcddr expr Not yet written

qcdr expr Not yet written

qgetv expr Not yet written

qputv expr Not yet written

quote fexpr Not yet written

quotient expr Not yet written

random!-fixnum expr Not yet written

random!-number expr Not yet written

rassoc expr Not yet written

rational expr Not yet written

rdf expr Not yet written

rds expr Not yet written

read expr Not yet written

readb expr Not yet written

readch expr Not yet written

readline expr Not yet written

reclaim expr Not yet written

remainder expr Not yet written

remd expr Not yet written

remflag expr Not yet written

remhash expr Not yet written

remob expr Not yet written

remprop expr Not yet written

rename!-file expr Not yet written

representation expr Not yet written

resource!-exceeded expr Not yet written

resource!-limit expr Not yet written

restart!-csl expr Not yet written

restore!-c!-code expr Not yet written

return fexpr Not yet written

reverse expr Not yet written

reversip expr Not yet written

round expr Not yet written

rplacw expr Not yet written

rseek expr Not yet written

rtell expr Not yet written

s!:blankcount macro Not yet written

s!:blanklist macro Not yet written

s!:blankp macro Not yet written

s!:depth macro Not yet written

s!:do!-bindings expr Not yet written

s!:do!-endtest expr Not yet written

s!:do!-result expr Not yet written

s!:do!-updates expr Not yet written

s!:endlist expr Not yet written

s!:expand!-do expr Not yet written

s!:expand!-dolist expr Not yet written

s!:expand!-dotimes expr Not yet written

s!:explodes expr Not yet written

s!:finishpending expr Not yet written

s!:format expr Not yet written

s!:indenting macro Not yet written

s!:make!-psetq!-assignments expr Not yet written

s!:make!-psetq!-bindings expr Not yet written

s!:make!-psetq!-vars expr Not yet written

s!:newframe macro Not yet written

s!:oblist expr Not yet written

s!:oblist1 expr Not yet written

s!:overflow expr Not yet written

s!:prindent expr Not yet written

s!:prinl0 expr Not yet written

s!:prinl1 expr Not yet written

s!:prinl2 expr Not yet written

s!:prvector expr Not yet written

s!:putblank expr Not yet written

s!:putch expr Not yet written

s!:quotep expr Not yet written

s!:setblankcount macro Not yet written

s!:setblanklist macro Not yet written

s!:setindenting macro Not yet written

s!:stamp expr Not yet written

s!:top macro Not yet written

safe!-fp!-pl expr Not yet written

safe!-fp!-pl0 expr Not yet written

safe!-fp!-plus expr Not yet written

safe!-fp!-quot expr Not yet written

safe!-fp!-times expr Not yet written

sample expr Not yet written

sassoc expr Not yet written

schar expr Not yet written

scharn expr Not yet written

sec expr Not yet written

secd expr Not yet written

sech expr Not yet written

seprp expr Not yet written

set expr Not yet written

set!-autoload expr Not yet written

set!-help!-file expr Not yet written

set!-print!-precision expr Not yet written

set!-small!-modulus expr Not yet written setpchar expr Not yet written

setq fexpr Not yet written

silent!-system expr Not yet written

simple!-string!-p expr Not yet written

simple!-vector!-p expr Not yet written

sin expr Not yet written

sind expr Not yet written

sinh expr Not yet written

smemq expr Not yet written

sort expr Not yet written

sortip expr Not yet written

spaces expr Not yet written

special!-char expr Not yet written

special!-form!-p expr Not yet written

spool expr Not yet written

sqrt expr Not yet written

stable!-sort expr Not yet written

stable!-sortip expr Not yet written

start!-module expr Not yet written

startup!-banner expr Not yet written

stop expr Not yet written

streamp expr Not yet written

stringp expr Not yet written

sub1 expr Not yet written

subla expr Not yet written

sublis expr Not yet written

subst expr Not yet written

superprinm expr Not yet written

superprintm expr Not yet written

sxhash expr Not yet written

symbol!-argcode expr Not yet written

symbol!-argcount expr Not yet written

symbol!-env expr Not yet written

symbol!-fastgets expr Not yet written

symbol!-fn!-cell expr Not yet written

symbol!-function expr Not yet written

symbol!-make!-fastget expr Not yet written

symbol!-name expr Not yet written

symbol!-protect expr Not yet written

symbol!-restore!-fns expr Not yet written

symbol!-set!-definition expr Not yet written

symbol!-set!-env expr Not yet written

symbol!-set!-native expr Not yet written

symbol!-value expr Not yet written

symbolp expr Not yet written

system expr Not yet written

tagbody fexpr Not yet written

tan expr Not yet written

tand expr Not yet written

tanh expr Not yet written

terpri expr Not yet written

threevectorp expr Not yet written

throw fexpr Not yet written

time expr Not yet written

times fexpr Not yet written

times2 expr Not yet written

tmpnam expr Not yet written

trace expr Not yet written

trace!-all expr Not yet written

traceset expr Not yet written

traceset1 expr Not yet written

truename expr Not yet written

truncate expr Not yet written

ttab expr Not yet written

tyo expr Not yet written

undouble!-execute expr Not yet written

unfluid expr Not yet written

unglobal expr Not yet written

union expr Not yet written

unless fexpr Not yet written

unmake!-global expr Not yet written

unmake!-special expr Not yet written

unreadch expr Not yet written

untrace expr Not yet written

untraceset expr Not yet written

untraceset1 expr Not yet written

unwind!-protect fexpr Not yet written

upbv expr Not yet written

user!-homedir!-pathname expr Not yet written

vectorp expr Not yet written

verbos expr Not yet written

when fexpr Not yet written

where!-was!-that expr Not yet written

window!-heading expr Not yet written

writable!-libraryp expr Not yet written

write!-module expr Not yet written

wrs expr Not yet written

xassoc expr Not yet written

xcons expr Not yet written

xdifference expr Not yet written

xtab expr Not yet written

zerop expr Not yet written

!∼block fexpr Not yet written

!∼let fexpr Not yet written

!∼tyi expr Not yet written