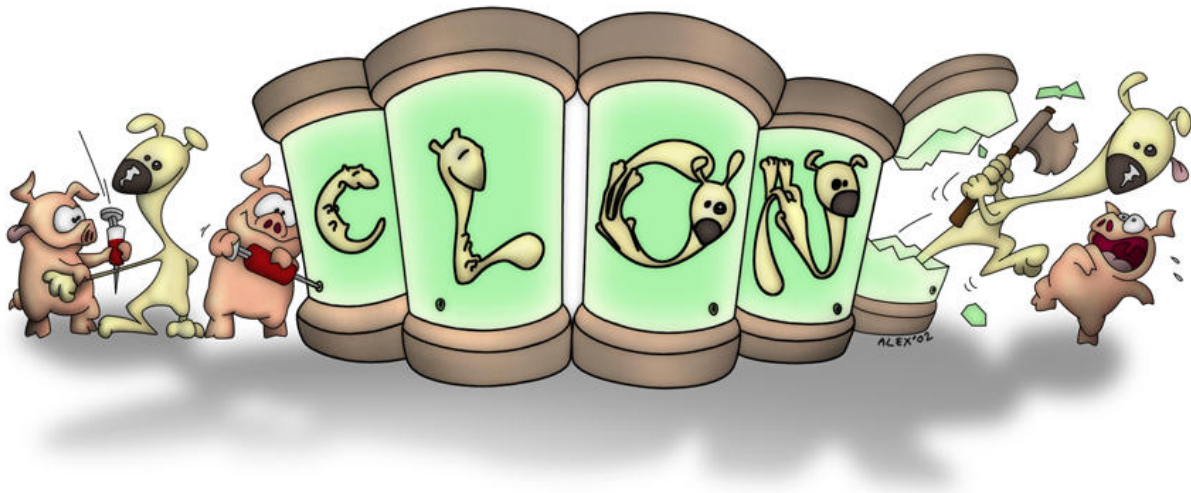


The Clon Reference Manual

The Command-Line Options Nuker, version 1.0 beta 25 "Michael Brecker"



Didier Verna <didier@didierverna.net>

This manual was generated automatically by Declt 3.0 "Montgomery Scott" on Thu Mar 25 20:06:20 2021 GMT+1.

Copyright © 2010-2012, 2015, 2017, 2020, 2021 Didier Verna

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the section entitled "Copying" is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be translated as well.

Table of Contents

Copying	1
1 Introduction	3
2 Systems	5
2.1 net.didierverna.clon.....	5
2.2 net.didierverna.clon.termio	5
2.3 net.didierverna.clon.setup/termio	6
2.4 net.didierverna.clon.core.....	7
2.5 net.didierverna.clon.setup.....	7
3 Modules	9
3.1 net.didierverna.clon.core/src	9
3.2 net.didierverna.clon.core/src/options.....	9
3.3 net.didierverna.clon.core/src/retrieval	10
3.4 net.didierverna.clon.core/src/output.....	10
3.5 net.didierverna.clon.setup/src.....	10
4 Files	11
4.1 Lisp.....	11
4.1.1 net.didierverna.clon.asd.....	11
4.1.2 net.didierverna.clon.termio.asd	11
4.1.3 net.didierverna.clon.core.asd	11
4.1.4 net.didierverna.clon.setup.asd.....	11
4.1.5 net.didierverna.clon.termio/sbcl/constants.lisp.....	11
4.1.6 net.didierverna.clon.termio/termio.lisp	11
4.1.7 net.didierverna.clon.core/package.lisp.....	12
4.1.8 net.didierverna.clon.core/src/util.lisp	12
4.1.9 net.didierverna.clon.core/src/item.lisp	13
4.1.10 net.didierverna.clon.core/src/text.lisp	13
4.1.11 net.didierverna.clon.core/src/options/option.lisp	13
4.1.12 net.didierverna.clon.core/src/options/flag.lisp.....	14
4.1.13 net.didierverna.clon.core/src/options/valued.lisp	14
4.1.14 net.didierverna.clon.core/src/options/negatable.lisp.....	15
4.1.15 net.didierverna.clon.core/src/options/switch-base.lisp.....	15
4.1.16 net.didierverna.clon.core/src/options/switch.lisp	15
4.1.17 net.didierverna.clon.core/src/options/stropt.lisp	16
4.1.18 net.didierverna.clon.core/src/options/lispobj.lisp.....	16
4.1.19 net.didierverna.clon.core/src/options/path.lisp.....	16
4.1.20 net.didierverna.clon.core/src/options/enum-base.lisp.....	17
4.1.21 net.didierverna.clon.core/src/options/enum.lisp.....	17
4.1.22 net.didierverna.clon.core/src/options/xswitch.lisp	17
4.1.23 net.didierverna.clon.core/src/container.lisp	18
4.1.24 net.didierverna.clon.core/src/group.lisp	18
4.1.25 net.didierverna.clon.core/src/retrieval/cmdline.lisp.....	18
4.1.26 net.didierverna.clon.core/src/retrieval/environ.lisp.....	19

4.1.27	net.didierverna.clon.core/src/synopsis.lisp	20
4.1.28	net.didierverna.clon.core/src/output/face.lisp	20
4.1.29	net.didierverna.clon.core/src/output/sheet.lisp	21
4.1.30	net.didierverna.clon.core/src/context.lisp	24
4.1.31	net.didierverna.clon.setup/package.lisp	25
4.1.32	net.didierverna.clon.setup/src/version.lisp	25
4.1.33	net.didierverna.clon.setup/src/configuration.lisp	26
4.1.34	net.didierverna.clon.setup/src/readtable.lisp	26
4.1.35	net.didierverna.clon.setup/src/termio.lisp	26
5	Packages	27
5.1	net.didierverna.clon	27
5.2	net.didierverna.clon.setup	39
6	Definitions	41
6.1	Exported definitions	41
6.1.1	Special variables	41
6.1.2	Macros	42
6.1.3	Functions	43
6.2	Internal definitions	49
6.2.1	Constants	49
6.2.2	Special variables	50
6.2.3	Macros	50
6.2.4	Functions	52
6.2.5	Generic functions	70
6.2.6	Conditions	91
6.2.7	Structures	98
6.2.8	Classes	100
Appendix A	Indexes	119
A.1	Concepts	119
A.2	Functions	121
A.3	Variables	128
A.4	Data types	130

Copying

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THIS SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

1 Introduction

`Clon` is a library for managing command-line options in standalone Common-Lisp applications. It provides a unified option syntax with both short and long names, automatic completion of partial names and automatic retrieval/conversion of option arguments from the command-line, associated environment variables, fallback or default values. `Clon` comes with a set of extensible option types (switches, paths, strings *etc.*). `Clon` also provides automatic generation and formatting of help strings, with support for highlighting on `tty`'s through ISO/IEC 6429 SGR. This formatting is customizable through *themes*.

Depending on the target audience, `Clon` stands for either “The Command-Line Options Nuker” or “The Common-Lisp Options Nuker”. `Clon` also has a recursive acronym: “`Clon` Likes Options Nuking”, and a reverse one: “Never Omit to Link with `Clon`”. Other possible expansions of the acronym are still being investigated.

This is the `Clon` reference manual, and as such, it is not meant to be read. It may help you find sleep in case of insomnia though. `Clon` comes with two human-readable manuals:

- the “end-user manual” (see *The Clon End-User Manual*) is for the `Clon` *end-user*, that is, the user of an application powered by `Clon`. It describes how to use the command-line of clonified¹ applications and how to customize `Clon`'s output. Everybody should read this manual first.
- the “user manual” (see *The Clon End-User Manual*) is for the `Clon` *user*, that is, the developer of a Common-Lisp application who wants to use `Clon` for command-line option management. It describes how to clonify your application and extend the library with your own option types.

¹ An application using `Clon` for its command-line option management is said to be *clonified*. It is also possible to say *clonfiscated*. However, we advise against using *clonistified*. The term *clonificated* is also considered bad style, and the use of *clonificationated* is strictly prohibited.

2 Systems

The main system appears first, followed by any subsystem dependency.

2.1 net.didierverna.clon

Long Name

The Command-Line Options Nuker

Author Didier Verna

Contact didier@didierverna.net

Home Page

<http://www.lrde.epita.fr/~didier/software/lisp/clon.php>

Source Control

<https://github.com/didierverna/clon>

License BSD

Description

Command-line options management for standalone Common Lisp applications

Long Description

Clon is a library for command-line options management. It is intended to ease the creation of standalone Common Lisp applications by providing a powerful and uniform command-line options interface. The most important features of Clon are the following.

- From the application programmer's point of view: centralized command-line options specification and management, including automatic generation of help strings, conversion from command-line / defaults / fallbacks / environment variables to application-level option values, global or on-demand option retrieval, and extensibility (the programmer can define his own option types).

- From the application user's point of view: uniform command-line option syntax across all Clon applications, customization of the help strings layout (with optional ISO6429 coloring on terminals that support it), automatic completion of abbreviated option names and short/long/pack syntax.

Defsystem Dependency

[net.didierverna.clon.setup/termio], page 6, (system)

Dependencies

- [net.didierverna.clon.core], page 7, (system)
- [net.didierverna.clon.termio], page 5, (system) (for feature net.didierverna.clon.termio)

Source [net.didierverna.clon.asd], page 11, (file)

2.2 net.didierverna.clon.termio

Long Name

The Command-Line Options Nuker, termio library

Author Didier Verna

Contact didier@didierverna.net

Home Page

<http://www.lrde.epita.fr/~didier/software/lisp/clon.php>

Source Control

<https://github.com/didierverna/clon>

License BSD

Description

Clon's support for termio (tty geometry and fontification)

Long Description

Clon's termio library provides automatic detection of tty geometry and ISO6429 coloring on terminals that support it. For a more complete description of Clon, see the net.didierverna.clon system.

If Feature net.didierverna.clon.termio

Defsystem Dependencies

- [net.didierverna.clon.setup/termio], page 6, (system)
- required module sb-grovel (for feature sbcl)
- cffi-grovel (for feature (or allegro clisp lispworks))

Dependencies

- sb-posix (for feature sbcl)
- cffi (for feature (and clisp net.didierverna.clon.termio))
- [net.didierverna.clon.setup], page 7, (system)
- [net.didierverna.clon.core], page 7, (system)

Source [net.didierverna.clon.termio.asd], page 11, (file)

Components

- [sbcl/constants.lisp], page 11, (file)
- [termio.lisp], page 11, (file)

2.3 net.didierverna.clon.setup/termio

Long Name

The Command-Line Options Nuker, termio setup

Author Didier Verna <didier@didierverna.net>

Contact didier@didierverna.net

Home Page

<http://www.lrde.epita.fr/~didier/software/lisp/clon.php>

Source Control

<https://github.com/didierverna/clon>

License BSD

Description

Clon's support for automatic configuration of termio support

Long Description

This is a virtual subsystem or Clon (no actual code). Its purpose is only to autodetect termio support and update Clon's preload configuration on load. For a more complete description of Clon, see the net.didierverna.clon system.

Dependency

[`net.didierverna.clon.setup`], page 7, (system)

Source [`net.didierverna.clon.setup.asd`], page 11, (file)

2.4 `net.didierverna.clon.core`

Long Name

The Command-Line Options Nuker, core library

Author Didier Verna

Contact `didier@didierverna.net`

Home Page

<http://www.lrde.epita.fr/~didier/software/lisp/clon.php>

Source Control

<https://github.com/didierverna/clon>

License BSD

Description

Clon's basic, platform-independent functionality

Long Description

Clon's core library provides the platform/feature independent part. For a more complete description of Clon, see the `net.didierverna.clon` system.

Dependencies

- required module `sb-posix` (for feature `sbcl`)
- [`net.didierverna.clon.setup`], page 7, (system)

Source [`net.didierverna.clon.core.asd`], page 11, (file)

Components

- [`package.lisp`], page 12, (file)
- [`src`], page 9, (module)

2.5 `net.didierverna.clon.setup`

Long Name

The Command-Line Options Nuker, setup library

Author Didier Verna

Contact `didier@didierverna.net`

Home Page

<http://www.lrde.epita.fr/~didier/software/lisp/clon.php>

Source Control

<https://github.com/didierverna/clon>

License BSD

Description

Clon's preload setup library

Long Description

The Clon setup library provides support for various preload configuration parameters and meta-utilities. For a more complete description of Clon, see the '`net.didierverna.clon`' system.

Dependency

`named-readtables`

Source `[net.didierverna.clon.setup.asd]`, page 11, (file)

Components

- `[package.lisp]`, page 25, (file)
- `[src]`, page 10, (module)

3 Modules

Modules are listed depth-first from the system components tree.

3.1 net.didierverna.clon.core/src

Dependency

[package.lisp], page 12, (file)

Parent

[net.didierverna.clon.core], page 7, (system)

Location

core/src/

Components

- [util.lisp], page 12, (file)
- [item.lisp], page 13, (file)
- [text.lisp], page 13, (file)
- [options], page 9, (module)
- [container.lisp], page 18, (file)
- [group.lisp], page 18, (file)
- [retrieval], page 10, (module)
- [synopsis.lisp], page 20, (file)
- [output], page 10, (module)
- [context.lisp], page 24, (file)

3.2 net.didierverna.clon.core/src/options

Dependency

[text.lisp], page 13, (file)

Parent

[src], page 9, (module)

Location

core/src/options/

Components

- [option.lisp], page 13, (file)
- [flag.lisp], page 14, (file)
- [valued.lisp], page 14, (file)
- [negatable.lisp], page 15, (file)
- [switch-base.lisp], page 15, (file)
- [switch.lisp], page 15, (file)
- [stropt.lisp], page 16, (file)
- [lispobj.lisp], page 16, (file)
- [path.lisp], page 16, (file)
- [enum-base.lisp], page 17, (file)
- [enum.lisp], page 17, (file)
- [xswitch.lisp], page 17, (file)

3.3 net.didierverna.clon.core/src/retrieval

Dependency

[options], page 9, (module)

Parent [src], page 9, (module)

Location core/src/retrieval/

Components

- [cmdline.lisp], page 18, (file)
- [environ.lisp], page 19, (file)

3.4 net.didierverna.clon.core/src/output

Dependencies

- [synopsis.lisp], page 20, (file)
- [retrieval], page 10, (module)

Parent [src], page 9, (module)

Location core/src/output/

Components

- [face.lisp], page 20, (file)
- [sheet.lisp], page 21, (file)

3.5 net.didierverna.clon.setup/src

Dependency

[package.lisp], page 25, (file)

Parent [net.didierverna.clon.setup], page 7, (system)

Location setup/src/

Components

- [version.lisp], page 25, (file)
- [configuration.lisp], page 26, (file)
- [readtable.lisp], page 26, (file)
- [termio.lisp], page 26, (file)

4 Files

Files are sorted by type and then listed depth-first from the systems components trees.

4.1 Lisp

4.1.1 net.didierverna.clon.asd

Location net.didierverna.clon.asd

Systems [net.didierverna.clon], page 5, (system)

4.1.2 net.didierverna.clon.termio.asd

Location termio/net.didierverna.clon.termio.asd

Systems [net.didierverna.clon.termio], page 5, (system)

4.1.3 net.didierverna.clon.core.asd

Location core/net.didierverna.clon.core.asd

Systems [net.didierverna.clon.core], page 7, (system)

4.1.4 net.didierverna.clon.setup.asd

Location setup/net.didierverna.clon.setup.asd

Systems

- [net.didierverna.clon.setup/termio], page 6, (system)
- [net.didierverna.clon.setup], page 7, (system)

4.1.5 net.didierverna.clon.termio/sbcl/constants.lisp

Parent [net.didierverna.clon.termio], page 5, (system)

Location termio/sbcl/constants.lisp

4.1.6 net.didierverna.clon.termio/termio.lisp

Dependency

[sbcl/constants.lisp], page 11, (file)

Parent [net.didierverna.clon.termio], page 5, (system)

Location termio/termio.lisp

Internal Definitions

- [stream-ioctl-output-handle], page 88, (generic function)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-line-width], page 69, (function)

4.1.7 net.didierverna.clon.core/package.lisp

Parent [net.didierverna.clon.core], page 7, (system)

Location core/package.lisp

Packages [net.didierverna.clon], page 27,

Exported Definitions

[nickname-package], page 48, (function)

4.1.8 net.didierverna.clon.core/src/util.lisp

Parent [src], page 9, (module)

Location core/src/util.lisp

Exported Definitions

- [*executablep*], page 41, (special variable)
- [cmdline], page 43, (function)
- [dump], page 42, (macro)
- [executablep], page 44, (function)
- [exit], page 44, (function)

Internal Definitions

- [abstract-class], page 100, (class)
- [accumulate], page 50, (macro)
- [beginning-of-string-p], page 53, (function)
- [closest-match], page 54, (function)
- [complete-string], page 54, (function)
- [copy-instance], page 75, (generic function)
- [copy-instance], page 75, (method)
- [declare-valid-superclass], page 50, (macro)
- [defabstract], page 50, (macro)
- [econd], page 51, (macro)
- [endpush], page 51, (macro)
- [error-string], page 76, (method)
- [(setf error-string)], page 76, (method)
- [getenv], page 57, (function)
- [home-directory], page 58, (function)
- [home-directory], page 93, (condition)
- [list-to-string], page 58, (function)
- [macosp], page 58, (function)
- [maybe-push], page 52, (macro)
- [putenv], page 64, (function)
- [remove-keys], page 65, (function)
- [replace-in-keys], page 52, (macro)
- [replace-key], page 65, (function)
- [replace-keys], page 66, (function)
- [select-keys], page 69, (function)

4.1.9 net.didierverna.clon.core/src/item.lisp

Dependency

[util.lisp], page 12, (file)

Parent [src], page 9, (module)

Location core/src/item.lisp

Internal Definitions

- [help-spec], page 77, (generic function)
- [help-spec], page 78, (method)
- [hiddenp], page 78, (method)
- [item], page 107, (class)
- [traversedp], page 90, (method)
- [(setf traversedp)], page 90, (method)
- [untraverse], page 90, (generic function)
- [untraverse], page 91, (method)

4.1.10 net.didierverna.clon.core/src/text.lisp

Dependency

[item.lisp], page 13, (file)

Parent [src], page 9, (module)

Location core/src/text.lisp

Exported Definitions

[make-text], page 48, (function)

Internal Definitions

- [contents], page 74, (method)
- [help-spec], page 78, (method)
- [make-internal-text], page 61, (function)
- [text], page 115, (class)
- [untraverse], page 91, (method)

4.1.11 net.didierverna.clon.core/src/options/option.lisp

Parent [options], page 9, (module)

Location core/src/options/option.lisp

Internal Definitions

- [check-name-clash], page 72, (generic function)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 73, (method)
- [description], page 75, (method)
- [env-var], page 76, (method)
- [help-spec], page 78, (method)
- [long-name], page 80, (method)
- [match-option], page 62, (function)

- [negated-pack-char], page 82, (generic function)
- [negated-pack-char], page 82, (method)
- [option], page 83, (method)
- [option], page 108, (class)
- [option-abbreviation-distance], page 62, (function)
- [option-error], page 96, (condition)
- [option-sticky-distance], page 83, (generic function)
- [option-sticky-distance], page 83, (method)
- [potential-pack-char], page 63, (function)
- [short-name], page 87, (method)
- [short-pack-char], page 87, (generic function)
- [short-pack-char], page 87, (method)
- [untraverse], page 91, (method)

4.1.12 net.didierverna.clon.core/src/options/flag.lisp

Dependency

[option.lisp], page 13, (file)

Parent [options], page 9, (module)

Location core/src/options/flag.lisp

Exported Definitions

[make-flag], page 45, (function)

Internal Definitions

- [flag], page 106, (class)
- [make-internal-flag], page 59, (function)

4.1.13 net.didierverna.clon.core/src/options/valued.lisp

Dependency

[option.lisp], page 13, (file)

Parent [options], page 9, (module)

Location core/src/options/valued.lisp

Internal Definitions

- [*item-names*], page 50, (special variable)
- [argument], page 70, (method)
- [argument-name], page 70, (method)
- [argument-required-p], page 70, (method)
- [check], page 71, (generic function)
- [comment], page 74, (method)
- [comment], page 74, (method)
- [convert], page 74, (generic function)
- [default-value], page 75, (method)
- [defoption], page 51, (macro)
- [fallback-value], page 76, (method)
- [help-spec], page 78, (method)

- [invalid-argument], page 94, (condition)
- [invalid-value], page 95, (condition)
- [option-sticky-distance], page 83, (method)
- [read-argument], page 64, (function)
- [read-value], page 65, (function)
- [restartable-check], page 66, (function)
- [restartable-convert], page 67, (function)
- [short-pack-char], page 87, (method)
- [short-syntax-help-spec-prefix], page 87, (generic function)
- [short-syntax-help-spec-prefix], page 88, (method)
- [stringify], page 88, (generic function)
- [value], page 91, (method)
- [valued-option], page 115, (class)

4.1.14 net.didierverna.clon.core/src/options/negatable.lisp

Dependency

[valued.lisp], page 14, (file)

Parent [options], page 9, (module)

Location core/src/options/negatable.lisp

Internal Definitions

- [negatable], page 108, (class)
- [negated-pack-char], page 82, (method)
- [short-syntax-help-spec-prefix], page 88, (method)

4.1.15 net.didierverna.clon.core/src/options/switch-base.lisp

Dependency

[negatable.lisp], page 15, (file)

Parent [options], page 9, (module)

Location core/src/options/switch-base.lisp

Internal Definitions

- [argument-style], page 70, (method)
- [argument-styles], page 71, (method)
- [(setf argument-styles)], page 71, (method)
- [no-values], page 82, (method)
- [(setf no-values)], page 82, (method)
- [switch-base], page 112, (class)
- [yes-values], page 91, (method)
- [(setf yes-values)], page 91, (method)

4.1.16 net.didierverna.clon.core/src/options/switch.lisp

Dependency

[switch-base.lisp], page 15, (file)

Parent [options], page 9, (module)

Location `core/src/options/switch.lisp`

Exported Definitions

[`make-switch`], page 47, (function)

Internal Definitions

- [`check`], page 72, (method)
- [`convert`], page 75, (method)
- [`make-internal-switch`], page 61, (function)
- [`stringify`], page 89, (method)
- [`switch`], page 112, (class)

4.1.17 `net.didierverna.clon.core/src/options/stropt.lisp`

Dependency

[`valued.lisp`], page 14, (file)

Parent [`options`], page 9, (module)

Location `core/src/options/stropt.lisp`

Exported Definitions

[`make-stropt`], page 47, (function)

Internal Definitions

- [`check`], page 72, (method)
- [`convert`], page 75, (method)
- [`make-internal-stropt`], page 60, (function)
- [`stringify`], page 89, (method)
- [`stropt`], page 112, (class)

4.1.18 `net.didierverna.clon.core/src/options/lispobj.lisp`

Dependency

[`valued.lisp`], page 14, (file)

Parent [`options`], page 9, (module)

Location `core/src/options/lispobj.lisp`

Exported Definitions

[`make-lispobj`], page 46, (function)

Internal Definitions

- [`check`], page 72, (method)
- [`convert`], page 74, (method)
- [`lispobj`], page 108, (class)
- [`make-internal-lispobj`], page 59, (function)
- [`stringify`], page 88, (method)
- [`typespec`], page 90, (method)

4.1.19 `net.didierverna.clon.core/src/options/path.lisp`

Dependency

[`valued.lisp`], page 14, (file)

Parent [`options`], page 9, (module)

Location `core/src/options/path.lisp`

Exported Definitions

[`make-path`], page 46, (function)

Internal Definitions

- [`check`], page 72, (method)
- [`convert`], page 74, (method)
- [`directory-pathname-p`], page 55, (function)
- [`make-internal-path`], page 60, (function)
- [`path`], page 110, (class)
- [`path-type`], page 84, (method)
- [`pathname-component-null-p`], page 63, (function)
- [`split-path`], page 69, (function)
- [`stringify`], page 88, (method)

4.1.20 `net.didierverna.clon.core/src/options/enum-base.lisp`

Parent [`options`], page 9, (module)

Location `core/src/options/enum-base.lisp`

Internal Definitions

- [`enum`], page 75, (method)
- [`enum-base`], page 103, (class)

4.1.21 `net.didierverna.clon.core/src/options/enum.lisp`

Dependencies

- [`valued.lisp`], page 14, (file)
- [`enum-base.lisp`], page 17, (file)

Parent [`options`], page 9, (module)

Location `core/src/options/enum.lisp`

Exported Definitions

[`make-enum`], page 45, (function)

Internal Definitions

- [`check`], page 72, (method)
- [`convert`], page 74, (method)
- [`enum`], page 102, (class)
- [`make-internal-enum`], page 59, (function)
- [`stringify`], page 88, (method)

4.1.22 `net.didierverna.clon.core/src/options/xswitch.lisp`

Dependencies

- [`valued.lisp`], page 14, (file)
- [`switch-base.lisp`], page 15, (file)
- [`enum-base.lisp`], page 17, (file)

Parent [`options`], page 9, (module)

Location `core/src/options/xswitch.lisp`

Exported Definitions

[make-xswitch], page 48, (function)

Internal Definitions

- [check], page 72, (method)
- [convert], page 74, (method)
- [make-internal-xswitch], page 61, (function)
- [stringify], page 88, (method)
- [xswitch], page 116, (class)

4.1.23 net.didierverna.clon.core/src/container.lisp**Dependency**

[options], page 9, (module)

Parent [src], page 9, (module)

Location core/src/container.lisp

Internal Definitions

- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [container], page 100, (class)
- [help-spec], page 78, (method)
- [items], page 80, (method)
- [untraverse], page 91, (method)

4.1.24 net.didierverna.clon.core/src/group.lisp**Dependency**

[container.lisp], page 18, (file)

Parent [src], page 9, (module)

Location core/src/group.lisp

Exported Definitions

- [defgroup], page 42, (macro)
- [make-group], page 46, (function)

Internal Definitions

- [%defgroup], page 50, (macro)
- [group], page 107, (class)
- [header], page 77, (method)
- [help-spec], page 78, (method)

4.1.25 net.didierverna.clon.core/src/retrieval/cmdline.lisp

Parent [retrieval], page 10, (module)

Location core/src/retrieval/cmdline.lisp

Internal Definitions

- [argument], page 70, (method)
- [argument-popable-p], page 53, (function)

- [cmdline-convert], page 54, (function)
- [cmdline-error], page 91, (condition)
- [cmdline-option-error], page 92, (condition)
- [invalid-cmdline-argument], page 94, (condition)
- [invalid-negated-syntax], page 95, (condition)
- [item], page 79, (method)
- [maybe-pop-argument], page 51, (macro)
- [missing-cmdline-argument], page 96, (condition)
- [name], page 81, (method)
- [option-call-p], page 62, (function)
- [restartable-cmdline-convert], page 66, (function)
- [restartable-invalid-negated-syntax-error], page 52, (macro)
- [restartable-spurious-cmdline-argument-error], page 52, (macro)
- [retrieve-from-long-call], page 85, (generic function)
- [retrieve-from-long-call], page 85, (method)
- [retrieve-from-long-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (generic function)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-short-call], page 86, (generic function)
- [retrieve-from-short-call], page 86, (method)
- [retrieve-from-short-call], page 86, (method)
- [spurious-cmdline-argument], page 96, (condition)

4.1.26 net.didierverna.clon.core/src/retrieval/environ.lisp

Parent [retrieval], page 10, (module)

Location core/src/retrieval/environ.lisp

Internal Definitions

- [env-val], page 76, (method)
- [env-var], page 76, (method)
- [environment-convert], page 55, (function)
- [environment-error], page 93, (condition)
- [environmental-option-error], page 93, (condition)
- [invalid-environment-value], page 94, (condition)
- [read-env-val], page 65, (function)
- [restartable-environment-convert], page 67, (function)
- [retrieve-from-environment], page 85, (generic function)
- [retrieve-from-environment], page 85, (method)
- [retrieve-from-environment], page 85, (method)
- [retrieve-from-environment], page 85, (method)

4.1.27 `net.didierverna.clon.core/src/synopsis.lisp`

Dependency

[`group.lisp`], page 18, (file)

Parent [`src`], page 9, (module)

Location `core/src/synopsis.lisp`

Exported Definitions

- [`*synopsis*`], page 42, (special variable)
- [`defsynopsis`], page 42, (macro)
- [`make-synopsis`], page 47, (function)

Internal Definitions

- [`clon-options-group`], page 73, (method)
- [`do-options`], page 51, (macro)
- [`help-spec`], page 77, (method)
- [`mapoptions`], page 81, (generic function)
- [`mapoptions`], page 81, (method)
- [`mapoptions`], page 81, (method)
- [`mapoptions`], page 81, (method)
- [`mapoptions`], page 81, (method)
- [`negated-pack`], page 82, (method)
- [`postfix`], page 84, (method)
- [`potential-pack`], page 84, (method)
- [`potential-pack-p`], page 84, (generic function)
- [`potential-pack-p`], page 84, (method)
- [`short-pack`], page 87, (method)
- [`synopsis`], page 114, (class)

4.1.28 `net.didierverna.clon.core/src/output/face.lisp`

Parent [`output`], page 10, (module)

Location `core/src/output/face.lisp`

Internal Definitions

- [`*highlight-properties*`], page 50, (special variable)
- [`add-subface`], page 52, (function)
- [`attach-face-tree`], page 53, (function)
- [`background`], page 71, (method)
- [`blink`], page 71, (method)
- [`bottom-padding`], page 71, (method)
- [`concealedp`], page 74, (method)
- [`crossed-out-p`], page 75, (method)
- [`face`], page 103, (class)
- [`face-highlight-property-set-p`], page 56, (function)
- [`face-highlight-property-value`], page 56, (function)
- [`foreground`], page 76, (method)

- [framedp], page 77, (method)
- [intensity], page 79, (method)
- [inversep], page 79, (method)
- [italicp], page 79, (method)
- [item-separator], page 79, (method)
- [left-padding], page 80, (method)
- [make-face-tree], page 80, (generic function)
- [make-face-tree], page 81, (method)
- [make-face-tree], page 81, (method)
- [make-raw-face-tree], page 61, (function)
- [name], page 81, (method)
- [parent], page 83, (method)
- [parent-generation], page 63, (function)
- [right-padding], page 86, (method)
- [search-branch], page 67, (function)
- [search-face], page 68, (function)
- [subface], page 89, (generic function)
- [subface], page 89, (method)
- [subface], page 89, (method)
- [subfaces], page 89, (method)
- [top-padding], page 90, (method)
- [underline], page 90, (method)
- [visiblep], page 91, (method)

4.1.29 net.didierverna.clon.core/src/output/sheet.lisp

Dependency

[face.lisp], page 20, (file)

Parent

[output], page 10, (module)

Location

core/src/output/sheet.lisp

Internal Definitions

- [available-right-margin], page 53, (function)
- [close-frame], page 73, (generic function)
- [close-frame], page 73, (method)
- [close-frame], page 73, (method)
- [close-line], page 53, (function)
- [close-sface], page 53, (function)
- [column], page 73, (method)
- [(setf column)], page 73, (method)
- [copy-frame], page 55, (function)
- [copy-highlight-frame], page 55, (function)
- [copy-highlight-property-instance], page 55, (function)
- [current-frame], page 55, (function)
- [current-left-margin], page 55, (function)

- [current-right-margin], page 55, (function)
- [current-sface], page 55, (function)
- [find-sface], page 56, (function)
- [flush-sheet], page 56, (function)
- [frame], page 98, (structure)
- [frame-left-margin], page 56, (function)
- [(setf frame-left-margin)], page 56, (function)
- [frame-p], page 56, (function)
- [frame-right-margin], page 56, (function)
- [(setf frame-right-margin)], page 56, (function)
- [frame-sface], page 56, (function)
- [(setf frame-sface)], page 56, (function)
- [frames], page 77, (method)
- [(setf frames)], page 77, (method)
- [get-bottom-padding], page 77, (generic function)
- [get-bottom-padding], page 77, (method)
- [get-bottom-padding], page 77, (method)
- [get-top-padding], page 57, (function)
- [help-spec-items-will-print], page 57, (function)
- [help-spec-will-print], page 78, (generic function)
- [help-spec-will-print], page 78, (method)
- [help-spec-will-print], page 78, (method)
- [help-spec-will-print], page 78, (method)
- [highlight-frame], page 99, (structure)
- [highlight-frame-highlight-property-instances], page 57, (function)
- [(setf highlight-frame-highlight-property-instances)], page 57, (function)
- [highlight-frame-left-margin], page 57, (function)
- [(setf highlight-frame-left-margin)], page 57, (function)
- [highlight-frame-p], page 57, (function)
- [highlight-frame-right-margin], page 57, (function)
- [(setf highlight-frame-right-margin)], page 57, (function)
- [highlight-frame-sface], page 57, (function)
- [(setf highlight-frame-sface)], page 57, (function)
- [highlight-property-ecase], page 51, (macro)
- [highlight-property-instance], page 99, (structure)
- [highlight-property-instance-escape-sequence], page 57, (function)
- [highlight-property-instance-name], page 57, (function)
- [(setf highlight-property-instance-name)], page 57, (function)
- [highlight-property-instance-p], page 58, (function)
- [highlight-property-instance-value], page 58, (function)
- [(setf highlight-property-instance-value)], page 58, (function)
- [highlightp], page 79, (method)

- [line-width], page 80, (method)
- [make-frame], page 58, (function)
- [make-highlight-frame], page 58, (function)
- [make-highlight-property-instance], page 59, (function)
- [make-raw-sface], page 62, (function)
- [make-sheet], page 62, (function)
- [map-frames], page 51, (macro)
- [open-frame], page 82, (generic function)
- [open-frame], page 83, (method)
- [open-frame], page 83, (method)
- [open-line], page 62, (function)
- [open-next-line], page 62, (function)
- [open-sface], page 62, (function)
- [output-stream], page 83, (method)
- [pop-frame], page 63, (function)
- [princ-char], page 63, (function)
- [princ-highlight-property-instances], page 63, (function)
- [princ-spaces], page 63, (function)
- [princ-string], page 63, (function)
- [print-faced-help-spec], page 64, (function)
- [print-help], page 64, (function)
- [print-help-spec], page 84, (generic function)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 85, (method)
- [print-help-spec], page 85, (method)
- [print-string], page 64, (function)
- [push-frame], page 64, (function)
- [reach-column], page 64, (function)
- [read-sface-tree], page 65, (function)
- [safe-left-margin], page 67, (function)
- [safe-right-margin], page 67, (function)
- [sface], page 110, (class)
- [sface-tree], page 86, (method)
- [sheet], page 111, (class)
- [sibling], page 88, (method)
- [top-padding], page 90, (method)
- [top-padding], page 90, (method)
- [try-read-sface-tree], page 69, (function)
- [try-read-theme], page 69, (function)

4.1.30 net.didierverna.clon.core/src/context.lisp

Dependency

[output], page 10, (module)

Parent

[src], page 9, (module)

Location

core/src/context.lisp

Exported Definitions

- [*context*], page 41, (special variable)
- [cmdline-options-p], page 43, (function)
- [cmdline-p], page 43, (function)
- [do-cmdline-options], page 42, (macro)
- [getopt], page 44, (function)
- [getopt-cmdline], page 44, (function)
- [help], page 44, (function)
- [make-context], page 45, (function)
- [multiple-value-getopt-cmdline], page 43, (macro)
- [progname], page 48, (function)
- [remainder], page 48, (function)
- [with-context], page 43, (macro)

Internal Definitions

- [argument], page 70, (method)
- [clon-options-group], page 73, (method)
- [cmdline-junk-error], page 92, (condition)
- [cmdline-option], page 98, (structure)
- [cmdline-option-name], page 54, (function)
- [(setf cmdline-option-name)], page 54, (function)
- [cmdline-option-option], page 54, (function)
- [(setf cmdline-option-option)], page 54, (function)
- [cmdline-option-p], page 54, (function)
- [cmdline-option-source], page 54, (function)
- [(setf cmdline-option-source)], page 54, (function)
- [cmdline-option-value], page 54, (function)
- [(setf cmdline-option-value)], page 54, (function)
- [cmdline-options], page 73, (method)
- [(setf cmdline-options)], page 73, (method)
- [context], page 101, (class)
- [copy-cmdline-option], page 55, (function)
- [error-handler], page 76, (method)
- [exit-abnormally], page 56, (function)
- [highlight], page 78, (method)
- [invalid-negated-equal-syntax], page 95, (condition)
- [invalid-short-equal-syntax], page 95, (condition)
- [junk], page 80, (method)

- [line-width], page 80, (method)
- [make-cmdline-option], page 58, (function)
- [mapoptions], page 81, (method)
- [name], page 81, (method)
- [negated-call], page 81, (method)
- [negated-pack], page 82, (method)
- [postfix], page 84, (method)
- [potential-pack-p], page 84, (method)
- [print-error], page 64, (function)
- [read-call], page 65, (function)
- [read-long-name], page 65, (function)
- [restart-on-error], page 66, (function)
- [restartable-cmdline-junk-error], page 66, (function)
- [search-option], page 68, (function)
- [search-option-by-abbreviation], page 68, (function)
- [search-option-by-name], page 68, (function)
- [search-path], page 86, (method)
- [search-sticky-option], page 68, (function)
- [short-call], page 87, (method)
- [short-pack], page 87, (method)
- [synopsis], page 89, (method)
- [theme], page 89, (method)
- [unknown-cmdline-option-error], page 97, (condition)
- [unrecognized-negated-call-error], page 97, (condition)
- [unrecognized-short-call-error], page 97, (condition)
- [untraverse], page 90, (method)
- [with-context-error-handler], page 52, (macro)

4.1.31 net.didierverna.clon.setup/package.lisp

Parent [net.didierverna.clon.setup], page 7, (system)

Location setup/package.lisp

Packages [net.didierverna.clon.setup], page 39,

4.1.32 net.didierverna.clon.setup/src/version.lisp

Parent [src], page 10, (module)

Location setup/src/version.lisp

Exported Definitions

- [*copyright-years*], page 41, (special variable)
- [*release-major-level*], page 41, (special variable)
- [*release-minor-level*], page 41, (special variable)
- [*release-name*], page 41, (special variable)
- [*release-status*], page 41, (special variable)
- [*release-status-level*], page 42, (special variable)

- [version], page 49, (function)

Internal Definitions

- [%version], page 52, (function)
- [release-status-number], page 65, (function)

4.1.33 net.didierverna.clon.setup/src/configuration.lisp

Parent [src], page 10, (module)

Location setup/src/configuration.lisp

Exported Definitions

- [configuration], page 43, (function)
- [configure], page 44, (function)

Internal Definitions

[*configuration*], page 50, (special variable)

4.1.34 net.didierverna.clon.setup/src/readtable.lisp

Dependency

[configuration.lisp], page 26, (file)

Parent [src], page 10, (module)

Location setup/src/readtable.lisp

Internal Definitions

- [clindent], page 53, (function)
- [defindent], page 51, (macro)
- [i-reader], page 58, (function)
- [~-reader], page 70, (function)

4.1.35 net.didierverna.clon.setup/src/termio.lisp

Dependency

[configuration.lisp], page 26, (file)

Parent [src], page 10, (module)

Location setup/src/termio.lisp

Exported Definitions

[setup-termio], page 48, (function)

Internal Definitions

[restrict-because], page 67, (function)

5 Packages

Packages are listed by definition order.

5.1 net.didierverna.clon

The Clon library's package.

Source [package.lisp], page 12, (file)

Nickname clon

Use List

- [net.didierverna.clon.setup], page 39,
- common-lisp

Exported Definitions

- [*context*], page 41, (special variable)
- [*executablep*], page 41, (special variable)
- [*synopsis*], page 42, (special variable)
- [cmdline], page 43, (function)
- [cmdline-options-p], page 43, (function)
- [cmdline-p], page 43, (function)
- [defgroup], page 42, (macro)
- [defsynopsis], page 42, (macro)
- [do-cmdline-options], page 42, (macro)
- [dump], page 42, (macro)
- [executablep], page 44, (function)
- [exit], page 44, (function)
- [getopt], page 44, (function)
- [getopt-cmdline], page 44, (function)
- [help], page 44, (function)
- [make-context], page 45, (function)
- [make-enum], page 45, (function)
- [make-flag], page 45, (function)
- [make-group], page 46, (function)
- [make-lispobj], page 46, (function)
- [make-path], page 46, (function)
- [make-stropt], page 47, (function)
- [make-switch], page 47, (function)
- [make-synopsis], page 47, (function)
- [make-text], page 48, (function)
- [make-xswitch], page 48, (function)
- [multiple-value-getopt-cmdline], page 43, (macro)
- [nickname-package], page 48, (function)
- [progname], page 48, (function)
- [remainder], page 48, (function)

- [with-context], page 43, (macro)

Internal Definitions

- [%defgroup], page 50, (macro)
- [*highlight-properties*], page 50, (special variable)
- [*item-names*], page 50, (special variable)
- [+tiocgwinsz+], page 49, (constant)
- [abstract-class], page 100, (class)
- [accumulate], page 50, (macro)
- [add-subface], page 52, (function)
- [allocate-winsize], page 53, (function)
- [argument], page 70, (generic function)
- [argument], page 70, (method)
- [argument], page 70, (method)
- [argument], page 70, (method)
- [argument-name], page 70, (generic function)
- [argument-name], page 70, (method)
- [argument-popable-p], page 53, (function)
- [argument-required-p], page 70, (generic function)
- [argument-required-p], page 70, (method)
- [argument-style], page 70, (generic function)
- [argument-style], page 70, (method)
- [argument-styles], page 71, (generic function)
- [argument-styles], page 71, (method)
- [(setf argument-styles)], page 71, (method)
- [(setf argument-styles)], page 71, (generic function)
- [attach-face-tree], page 53, (function)
- [available-right-margin], page 53, (function)
- [background], page 71, (generic function)
- [background], page 71, (method)
- [beginning-of-string-p], page 53, (function)
- [blink], page 71, (generic function)
- [blink], page 71, (method)
- [bottom-padding], page 71, (generic function)
- [bottom-padding], page 71, (method)
- [check], page 71, (generic function)
- [check], page 72, (method)
- [check], page 72, (method)
- [check], page 72, (method)
- [check], page 72, (method)
- [check], page 72, (method)
- [check], page 72, (method)
- [check], page 72, (method)
- [check-name-clash], page 72, (generic function)
- [check-name-clash], page 72, (method)

- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 73, (method)
- [clon-options-group], page 73, (generic function)
- [clon-options-group], page 73, (method)
- [clon-options-group], page 73, (method)
- [close-frame], page 73, (generic function)
- [close-frame], page 73, (method)
- [close-frame], page 73, (method)
- [close-line], page 53, (function)
- [close-sface], page 53, (function)
- [closest-match], page 54, (function)
- [cmdline-convert], page 54, (function)
- [cmdline-error], page 91, (condition)
- [cmdline-junk-error], page 92, (condition)
- [cmdline-option], page 98, (structure)
- [cmdline-option-error], page 92, (condition)
- [cmdline-option-name], page 54, (function)
- [(setf cmdline-option-name)], page 54, (function)
- [cmdline-option-option], page 54, (function)
- [(setf cmdline-option-option)], page 54, (function)
- [cmdline-option-p], page 54, (function)
- [cmdline-option-source], page 54, (function)
- [(setf cmdline-option-source)], page 54, (function)
- [cmdline-option-value], page 54, (function)
- [(setf cmdline-option-value)], page 54, (function)
- [cmdline-options], page 73, (generic function)
- [cmdline-options], page 73, (method)
- [(setf cmdline-options)], page 73, (method)
- [(setf cmdline-options)], page 73, (generic function)
- [column], page 73, (generic function)
- [column], page 73, (method)
- [(setf column)], page 73, (method)
- [(setf column)], page 73, (generic function)
- [comment], page 74, (generic function)
- [comment], page 74, (method)
- [comment], page 74, (method)
- [complete-string], page 54, (function)
- [concealedp], page 74, (generic function)
- [concealedp], page 74, (method)
- [container], page 100, (class)

- [contents], page 74, (generic function)
- [contents], page 74, (method)
- [context], page 101, (class)
- [convert], page 74, (generic function)
- [convert], page 74, (method)
- [convert], page 74, (method)
- [convert], page 74, (method)
- [convert], page 74, (method)
- [convert], page 75, (method)
- [convert], page 75, (method)
- [copy-cmdline-option], page 55, (function)
- [copy-frame], page 55, (function)
- [copy-highlight-frame], page 55, (function)
- [copy-highlight-property-instance], page 55, (function)
- [copy-instance], page 75, (generic function)
- [copy-instance], page 75, (method)
- [crossed-out-p], page 75, (generic function)
- [crossed-out-p], page 75, (method)
- [current-frame], page 55, (function)
- [current-left-margin], page 55, (function)
- [current-right-margin], page 55, (function)
- [current-sface], page 55, (function)
- [declare-valid-superclass], page 50, (macro)
- [defabstract], page 50, (macro)
- [default-value], page 75, (generic function)
- [default-value], page 75, (method)
- [defoption], page 51, (macro)
- [description], page 75, (generic function)
- [description], page 75, (method)
- [directory-pathname-p], page 55, (function)
- [do-options], page 51, (macro)
- [econd], page 51, (macro)
- [endpush], page 51, (macro)
- [enum], page 75, (generic function)
- [enum], page 75, (method)
- [enum], page 102, (class)
- [enum-base], page 103, (class)
- [env-val], page 76, (generic function)
- [env-val], page 76, (method)
- [env-var], page 76, (generic function)
- [env-var], page 76, (method)
- [env-var], page 76, (method)
- [environment-convert], page 55, (function)

- [environment-error], page 93, (condition)
- [environmental-option-error], page 93, (condition)
- [error-handler], page 76, (generic function)
- [error-handler], page 76, (method)
- [error-string], page 76, (generic function)
- [error-string], page 76, (method)
- [(setf error-string)], page 76, (method)
- [(setf error-string)], page 76, (generic function)
- [exit-abnormally], page 56, (function)
- [face], page 103, (class)
- [face-highlight-property-set-p], page 56, (function)
- [face-highlight-property-value], page 56, (function)
- [fallback-value], page 76, (generic function)
- [fallback-value], page 76, (method)
- [find-sface], page 56, (function)
- [flag], page 106, (class)
- [flush-sheet], page 56, (function)
- [foreground], page 76, (generic function)
- [foreground], page 76, (method)
- [frame], page 98, (structure)
- [frame-left-margin], page 56, (function)
- [(setf frame-left-margin)], page 56, (function)
- [frame-p], page 56, (function)
- [frame-right-margin], page 56, (function)
- [(setf frame-right-margin)], page 56, (function)
- [frame-sface], page 56, (function)
- [(setf frame-sface)], page 56, (function)
- [framedp], page 77, (generic function)
- [framedp], page 77, (method)
- [frames], page 77, (generic function)
- [frames], page 77, (method)
- [(setf frames)], page 77, (method)
- [(setf frames)], page 77, (generic function)
- [get-bottom-padding], page 77, (generic function)
- [get-bottom-padding], page 77, (method)
- [get-bottom-padding], page 77, (method)
- [get-top-padding], page 57, (function)
- [getenv], page 57, (function)
- [group], page 107, (class)
- [header], page 77, (generic function)
- [header], page 77, (method)
- [help-spec], page 77, (generic function)
- [help-spec], page 77, (method)

- [help-spec], page 78, (method)
- [help-spec], page 78, (method)
- [help-spec], page 78, (method)
- [help-spec], page 78, (method)
- [help-spec], page 78, (method)
- [help-spec], page 78, (method)
- [help-spec-items-will-print], page 57, (function)
- [help-spec-will-print], page 78, (generic function)
- [help-spec-will-print], page 78, (method)
- [help-spec-will-print], page 78, (method)
- [help-spec-will-print], page 78, (method)
- [hiddenp], page 78, (generic function)
- [hiddenp], page 78, (method)
- [highlight], page 78, (generic function)
- [highlight], page 78, (method)
- [highlight-frame], page 99, (structure)
- [highlight-frame-highlight-property-instances], page 57, (function)
- [(setf highlight-frame-highlight-property-instances)], page 57, (function)
- [highlight-frame-left-margin], page 57, (function)
- [(setf highlight-frame-left-margin)], page 57, (function)
- [highlight-frame-p], page 57, (function)
- [highlight-frame-right-margin], page 57, (function)
- [(setf highlight-frame-right-margin)], page 57, (function)
- [highlight-frame-sface], page 57, (function)
- [(setf highlight-frame-sface)], page 57, (function)
- [highlight-property-ecase], page 51, (macro)
- [highlight-property-instance], page 99, (structure)
- [highlight-property-instance-escape-sequence], page 57, (function)
- [highlight-property-instance-name], page 57, (function)
- [(setf highlight-property-instance-name)], page 57, (function)
- [highlight-property-instance-p], page 58, (function)
- [highlight-property-instance-value], page 58, (function)
- [(setf highlight-property-instance-value)], page 58, (function)
- [highlightp], page 79, (generic function)
- [highlightp], page 79, (method)
- [home-directory], page 58, (function)
- [home-directory], page 93, (condition)
- [intensity], page 79, (generic function)
- [intensity], page 79, (method)
- [invalid-argument], page 94, (condition)
- [invalid-cmdline-argument], page 94, (condition)
- [invalid-environment-value], page 94, (condition)

- [invalid-negated-equal-syntax], page 95, (condition)
- [invalid-negated-syntax], page 95, (condition)
- [invalid-short-equal-syntax], page 95, (condition)
- [invalid-value], page 95, (condition)
- [inversep], page 79, (generic function)
- [inversep], page 79, (method)
- [italicp], page 79, (generic function)
- [italicp], page 79, (method)
- [item], page 79, (generic function)
- [item], page 79, (method)
- [item], page 107, (class)
- [item-separator], page 79, (generic function)
- [item-separator], page 79, (method)
- [items], page 79, (generic function)
- [items], page 80, (method)
- [junk], page 80, (generic function)
- [junk], page 80, (method)
- [left-padding], page 80, (generic function)
- [left-padding], page 80, (method)
- [line-width], page 80, (generic function)
- [line-width], page 80, (method)
- [line-width], page 80, (method)
- [lispobj], page 108, (class)
- [list-to-string], page 58, (function)
- [long-name], page 80, (generic function)
- [long-name], page 80, (method)
- [macosp], page 58, (function)
- [make-cmdline-option], page 58, (function)
- [make-face-tree], page 80, (generic function)
- [make-face-tree], page 81, (method)
- [make-face-tree], page 81, (method)
- [make-frame], page 58, (function)
- [make-highlight-frame], page 58, (function)
- [make-highlight-property-instance], page 59, (function)
- [make-internal-enum], page 59, (function)
- [make-internal-flag], page 59, (function)
- [make-internal-lispobj], page 59, (function)
- [make-internal-path], page 60, (function)
- [make-internal-stropt], page 60, (function)
- [make-internal-switch], page 61, (function)
- [make-internal-text], page 61, (function)
- [make-internal-xswitch], page 61, (function)
- [make-raw-face-tree], page 61, (function)

- [make-raw-sface], page 62, (function)
- [make-sheet], page 62, (function)
- [map-frames], page 51, (macro)
- [mapoptions], page 81, (generic function)
- [mapoptions], page 81, (method)
- [mapoptions], page 81, (method)
- [mapoptions], page 81, (method)
- [mapoptions], page 81, (method)
- [mapoptions], page 81, (method)
- [match-option], page 62, (function)
- [maybe-pop-argument], page 51, (macro)
- [maybe-push], page 52, (macro)
- [missing-cmdline-argument], page 96, (condition)
- [name], page 81, (generic function)
- [name], page 81, (method)
- [name], page 81, (method)
- [name], page 81, (method)
- [negatable], page 108, (class)
- [negated-call], page 81, (generic function)
- [negated-call], page 81, (method)
- [negated-pack], page 82, (generic function)
- [negated-pack], page 82, (method)
- [negated-pack], page 82, (method)
- [negated-pack-char], page 82, (generic function)
- [negated-pack-char], page 82, (method)
- [negated-pack-char], page 82, (method)
- [no-values], page 82, (generic function)
- [no-values], page 82, (method)
- [(setf no-values)], page 82, (method)
- [(setf no-values)], page 82, (generic function)
- [offset-of-winsize-ws-col], page 49, (constant)
- [offset-of-winsize-ws-row], page 49, (constant)
- [offset-of-winsize-ws-xpixel], page 49, (constant)
- [offset-of-winsize-ws-ypixel], page 49, (constant)
- [open-frame], page 82, (generic function)
- [open-frame], page 83, (method)
- [open-frame], page 83, (method)
- [open-line], page 62, (function)
- [open-next-line], page 62, (function)
- [open-sface], page 62, (function)
- [option], page 83, (generic function)
- [option], page 83, (method)
- [option], page 108, (class)

- [option-abbreviation-distance], page 62, (function)
- [option-call-p], page 62, (function)
- [option-error], page 96, (condition)
- [option-sticky-distance], page 83, (generic function)
- [option-sticky-distance], page 83, (method)
- [option-sticky-distance], page 83, (method)
- [output-stream], page 83, (generic function)
- [output-stream], page 83, (method)
- [parent], page 83, (generic function)
- [parent], page 83, (method)
- [parent-generation], page 63, (function)
- [path], page 110, (class)
- [path-type], page 83, (generic function)
- [path-type], page 84, (method)
- [pathname-component-null-p], page 63, (function)
- [pop-frame], page 63, (function)
- [postfix], page 84, (generic function)
- [postfix], page 84, (method)
- [postfix], page 84, (method)
- [potential-pack], page 84, (generic function)
- [potential-pack], page 84, (method)
- [potential-pack-char], page 63, (function)
- [potential-pack-p], page 84, (generic function)
- [potential-pack-p], page 84, (method)
- [potential-pack-p], page 84, (method)
- [princ-char], page 63, (function)
- [princ-highlight-property-instances], page 63, (function)
- [princ-spaces], page 63, (function)
- [princ-string], page 63, (function)
- [print-error], page 64, (function)
- [print-faced-help-spec], page 64, (function)
- [print-help], page 64, (function)
- [print-help-spec], page 84, (generic function)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 84, (method)
- [print-help-spec], page 85, (method)
- [print-help-spec], page 85, (method)
- [print-string], page 64, (function)
- [push-frame], page 64, (function)
- [putenv], page 64, (function)
- [reach-column], page 64, (function)
- [read-argument], page 64, (function)

- [read-call], page 65, (function)
- [read-env-val], page 65, (function)
- [read-long-name], page 65, (function)
- [read-sface-tree], page 65, (function)
- [read-value], page 65, (function)
- [remove-keys], page 65, (function)
- [replace-in-keys], page 52, (macro)
- [replace-key], page 65, (function)
- [replace-keys], page 66, (function)
- [restart-on-error], page 66, (function)
- [restartable-check], page 66, (function)
- [restartable-cmdline-convert], page 66, (function)
- [restartable-cmdline-junk-error], page 66, (function)
- [restartable-convert], page 67, (function)
- [restartable-environment-convert], page 67, (function)
- [restartable-invalid-negated-syntax-error], page 52, (macro)
- [restartable-spurious-cmdline-argument-error], page 52, (macro)
- [retrieve-from-environment], page 85, (generic function)
- [retrieve-from-environment], page 85, (method)
- [retrieve-from-environment], page 85, (method)
- [retrieve-from-environment], page 85, (method)
- [retrieve-from-long-call], page 85, (generic function)
- [retrieve-from-long-call], page 85, (method)
- [retrieve-from-long-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (generic function)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-short-call], page 86, (generic function)
- [retrieve-from-short-call], page 86, (method)
- [retrieve-from-short-call], page 86, (method)
- [right-padding], page 86, (generic function)
- [right-padding], page 86, (method)
- [safe-left-margin], page 67, (function)
- [safe-right-margin], page 67, (function)
- [search-branch], page 67, (function)
- [search-face], page 68, (function)
- [search-option], page 68, (function)
- [search-option-by-abbreviation], page 68, (function)
- [search-option-by-name], page 68, (function)
- [search-path], page 86, (generic function)
- [search-path], page 86, (method)
- [search-sticky-option], page 68, (function)

- [select-keys], page 69, (function)
- [sface], page 110, (class)
- [sface-tree], page 86, (generic function)
- [sface-tree], page 86, (method)
- [sheet], page 111, (class)
- [short-call], page 87, (generic function)
- [short-call], page 87, (method)
- [short-name], page 87, (generic function)
- [short-name], page 87, (method)
- [short-pack], page 87, (generic function)
- [short-pack], page 87, (method)
- [short-pack], page 87, (method)
- [short-pack-char], page 87, (generic function)
- [short-pack-char], page 87, (method)
- [short-pack-char], page 87, (method)
- [short-syntax-help-spec-prefix], page 87, (generic function)
- [short-syntax-help-spec-prefix], page 88, (method)
- [short-syntax-help-spec-prefix], page 88, (method)
- [sibling], page 88, (generic function)
- [sibling], page 88, (method)
- [size-of-winsize], page 49, (constant)
- [split-path], page 69, (function)
- [spurious-cmdline-argument], page 96, (condition)
- [stream-ioctl-output-handle], page 88, (generic function)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-ioctl-output-handle], page 88, (method)
- [stream-line-width], page 69, (function)
- [stringify], page 88, (generic function)
- [stringify], page 88, (method)
- [stringify], page 88, (method)
- [stringify], page 88, (method)
- [stringify], page 88, (method)
- [stringify], page 89, (method)
- [stringify], page 89, (method)
- [stropt], page 112, (class)
- [subface], page 89, (generic function)
- [subface], page 89, (method)
- [subface], page 89, (method)
- [subfaces], page 89, (generic function)
- [subfaces], page 89, (method)
- [switch], page 112, (class)

- [switch-base], page 112, (class)
- [synopsis], page 89, (generic function)
- [synopsis], page 89, (method)
- [synopsis], page 114, (class)
- [text], page 115, (class)
- [theme], page 89, (generic function)
- [theme], page 89, (method)
- [top-padding], page 89, (generic function)
- [top-padding], page 90, (method)
- [top-padding], page 90, (method)
- [top-padding], page 90, (method)
- [traversedp], page 90, (generic function)
- [traversedp], page 90, (method)
- [(setf traversedp)], page 90, (method)
- [(setf traversedp)], page 90, (generic function)
- [try-read-sface-tree], page 69, (function)
- [try-read-theme], page 69, (function)
- [typespec], page 90, (generic function)
- [typespec], page 90, (method)
- [underline], page 90, (generic function)
- [underline], page 90, (method)
- [unknown-cmdline-option-error], page 97, (condition)
- [unrecognized-negated-call-error], page 97, (condition)
- [unrecognized-short-call-error], page 97, (condition)
- [untraverse], page 90, (generic function)
- [untraverse], page 90, (method)
- [untraverse], page 91, (method)
- [untraverse], page 91, (method)
- [untraverse], page 91, (method)
- [untraverse], page 91, (method)
- [value], page 91, (generic function)
- [value], page 91, (method)
- [valued-option], page 115, (class)
- [visiblep], page 91, (generic function)
- [visiblep], page 91, (method)
- [winsize-ws-col], page 69, (function)
- [(setf winsize-ws-col)], page 69, (function)
- [winsize-ws-row], page 69, (function)
- [(setf winsize-ws-row)], page 69, (function)
- [winsize-ws-xpixel], page 69, (function)
- [(setf winsize-ws-xpixel)], page 69, (function)
- [winsize-ws-ypixel], page 70, (function)
- [(setf winsize-ws-ypixel)], page 70, (function)

- `[with-context-error-handler]`, page 52, (macro)
- `[with-winsize]`, page 52, (macro)
- `[xswitch]`, page 116, (class)
- `[yes-values]`, page 91, (generic function)
- `[yes-values]`, page 91, (method)
- `[(setf yes-values)]`, page 91, (method)
- `[(setf yes-values)]`, page 91, (generic function)

5.2 `net.didierverna.clon.setup`

The Clon setup library's package.

Source `[package.lisp]`, page 25, (file)

Use List `common-lisp`

Used By List

`[net.didierverna.clon]`, page 27,

Exported Definitions

- `[*copyright-years*]`, page 41, (special variable)
- `[*release-major-level*]`, page 41, (special variable)
- `[*release-minor-level*]`, page 41, (special variable)
- `[*release-name*]`, page 41, (special variable)
- `[*release-status*]`, page 41, (special variable)
- `[*release-status-level*]`, page 42, (special variable)
- `[configuration]`, page 43, (function)
- `[configure]`, page 44, (function)
- `[setup-termio]`, page 48, (function)
- `[version]`, page 49, (function)

Internal Definitions

- `[%version]`, page 52, (function)
- `[*configuration*]`, page 50, (special variable)
- `[clindent]`, page 53, (function)
- `[defindent]`, page 51, (macro)
- `[i-reader]`, page 58, (function)
- `[release-status-number]`, page 65, (function)
- `[restrict-because]`, page 67, (function)
- `[~-reader]`, page 70, (function)

6 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

6.1 Exported definitions

6.1.1 Special variables

- *context*** [Special Variable]
 The current context.
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- *copyright-years*** [Special Variable]
 A string denoting the copyright years for the whole project.
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)
- *executablep*** [Special Variable]
 Whether the current Lisp image is a standalone executable.
 This information is needed in several implementations to distinguish user options from implementation-specific ones on the command-line.
 It is set automatically to T by the ‘dump’ function, which see.
 If the image is dumped by ASDF’s program-op, this variable is ignored. In any other case, that is, when dumping via an implementation-specific function, it must be set manually to T just before dumping.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- *release-major-level*** [Special Variable]
 The major level of this release.
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)
- *release-minor-level*** [Special Variable]
 The minor level of this release.
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)
- *release-name*** [Special Variable]
 The name of this release.
 The general naming theme for Clon is "Great Jazz musicians".
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)
- *release-status*** [Special Variable]
 The status of this release.
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)

release-status-level [Special Variable]
 The status level of this release.

Package [net.didierverna.clon.setup], page 39,

Source [version.lisp], page 25, (file)

synopsis [Special Variable]
 The current synopsis.

Package [net.didierverna.clon], page 27,

Source [synopsis.lisp], page 20, (file)

6.1.2 Macros

defgroup (&rest *KEYS* &key *HEADER HIDDEN*) &body *FORMS* [Macro]
 Define a new group.

KEYS are initalargs to MAKE-GROUP (currently, only :header).

Each form in *FORMS* will be treated as a new :item.

The CAR of each form is the name of the operation to perform: TEXT, GROUP, or an option class name. The rest are the arguments to the MAKE-<OP> function or the DEFGROUP macro.

Package [net.didierverna.clon], page 27,

Source [group.lisp], page 18, (file)

defsynopsis (&rest *KEYS* &key *POSTFIX MAKE-DEFAULT*) &body [Macro]
FORMS

Define a new synopsis.

Package [net.didierverna.clon], page 27,

Source [synopsis.lisp], page 20, (file)

do-commandline-options (*OPTION NAME VALUE SOURCE* &key [Macro]
CONTEXT) &body *BODY*

Evaluate *BODY* over all command-line options in *CONTEXT*.

OPTION, *NAME* and *VALUE* are bound to each option's object, name used on the command-line and retrieved value.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

dump *NAME FUNCTION* &rest *ARGS* [Macro]

Dump a standalone executable named *NAME* starting with *FUNCTION*.

ARGS may be any arguments understood by the underlying implementation's dumping facility. They will simply be passed along. Note that DUMP already passes some such arguments. Some of them are critical for the dumping facility (e.g. :executable) and cannot be overridden. Some others, however, will be if you provide them as well (e.g. :load-init-file).

Since executable dumping is not available in all supported implementations, this function behaves differently in some cases, as described below.

- ECL doesn't create executables by dumping a Lisp image, but relies on having toplevel code to execute instead, so this macro simply expands to a call to *FUNCTION*. This also means that *ARGS* is unused.

- ABCL can't dump executables at all because of the underlying Java implementation, so this macro expands to just (PROGN) but creates a Java class file with a main function that creates an interpreter, loads the file in which this macro call appears and calls FUNCTION. This also means that ARGS is unused.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

multiple-value-getopt-cmdline (*OPTION NAME VALUE SOURCE* [Macro]
&key *CONTEXT*) &body *BODY*

Get the next command-line option in *CONTEXT*. and evaluate *BODY*. *OPTION*, *NAME* and *VALUE* are bound to the values returned by GETOPT-CMDLINE. *BODY* is executed only if there is a next command-line option.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

with-context *CONTEXT* &body *BODY* [Macro]

Execute *BODY* with *context* bound to *CONTEXT*.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

6.1.3 Functions

cmdline () [Function]

Get the current application's command-line.

This command-line is not supposed to contain any Lisp implementation specific option; only user-level ones. When a standalone executable is dumped, this is always the case. When used interactively, this depends on the underlying Lisp implementation. See appendix A.5 of the user manual for more information.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

cmdline-options-p &key *CONTEXT* [Function]

Return T if *CONTEXT* has any unprocessed options left.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

cmdline-p &key *CONTEXT* [Function]

Return T if *CONTEXT* has anything on its command-line.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

configuration *KEY* [Function]

Return *KEY*'s value in the current Clon configuration.

Package [net.didierverna.clon.setup], page 39,

Source [configuration.lisp], page 26, (file)

- configure** *KEY VALUE* [Function]
 Set KEY to VALUE in the current Clon configuration.
- Package** [net.didierverna.clon.setup], page 39,
Source [configuration.lisp], page 26, (file)
- executablep** () [Function]
 Return T if the current Lisp image is a standalone executable.
 This function detects executables dumped by ASDF's program-op operation, those dumped by Clon's 'dump' function (which see), and those in which '*executablep*' (which see) has been set to T manually.
- Package** [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- exit** &optional *STATUS* [Function]
 Quit the current application with STATUS.
 This function is considered deprecated. Please use UIOP:QUIT instead.
- Package** [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- getopt** &rest *KEYS* &key *CONTEXT SHORT-NAME LONG-NAME* [Function]
OPTION
 Get an option's value in CONTEXT.
 The option can be specified either by SHORT-NAME, LONG-NAME, or directly via an OPTION object.
 Return two values:
 - the retrieved value,
 - the value's source.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- getopt-cmdline** &key *CONTEXT* [Function]
 Get the next command-line option in CONTEXT.
 When there is no next command-line option, return nil. Otherwise, return four values:
 - the option object,
 - the option's name used on the command-line,
 - the retrieved value,
 - the value source.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- help** &key *CONTEXT ITEM OUTPUT-STREAM SEARCH-PATH* [Function]
THEME LINE-WIDTH HIGHLIGHT
 Print CONTEXT's help.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)

make-context &rest *KEYS* &key *SYNOPSIS CMDLINE PROGNAME* [Function]
MAKE-CURRENT

Make a new context.

- *SYNOPSIS* is the program synopsis to use in that context.

It defaults to **SYNOPSIS**.

- *CMDLINE* is the argument list (strings) to process.

It defaults to a POSIX conformant argv.

- *PROGNAME* is an alternate value for argv[0].

It defaults to NIL, in which case the actual argv[0] is used. Otherwise, it can be a non-empty string, standing for itself,

or *:environment* meaning to retrieve the value of the *__CL_ARGV0* environment variable (ignored if it's empty).

value.

- If *MAKE-CURRENT*, make the new context current. This is the default.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

make-enum &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM
ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN

Make a new enum option.

- *SHORT-NAME* is the option's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the option's long name (without the double-dash).

It defaults to nil.

- *DESCRIPTION* is the option's description appearing in help strings.

It defaults to nil.

- *ARGUMENT-NAME* is the option's argument name appearing in help strings. - *ARGUMENT-TYPE* is one of *:required*, *:mandatory* or *:optional* (*:required* and *:mandatory* are synonyms).

It defaults to *:optional*.

- *ENUM* is the set of possible values.

- *ENV-VAR* is the option's associated environment variable.

It defaults to nil.

- *FALLBACK-VALUE* is the option's fallback value (for missing optional arguments), if any.

- *DEFAULT-VALUE* is the option's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [enum.lisp], page 17, (file)

make-flag &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ENV-VAR HIDDEN

Make a new flag.

- *SHORT-NAME* is the option's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the option's long name (without the double-dash). It defaults to nil.

- *DESCRIPTION* is the option's description appearing in help strings. It defaults to nil.

- *ENV-VAR* is the flag's associated environment variable.

It defaults to nil.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [flag.lisp], page 14, (file)

make-group &rest *KEYS* &key *HEADER ITEM HIDDEN* [Function]
 Make a new group.

Package [net.didierverna.clon], page 27,

Source [group.lisp], page 18, (file)

make-lispobj &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR
TYPESPEC FALLBACK-VALUE DEFAULT-VALUE HIDDEN

Make a new lispobj option.

- *SHORT-NAME* is the option's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the option's long name (without the double-dash).

It defaults to nil.

- *DESCRIPTION* is the option's description appearing in help strings.

It defaults to nil.

- *ARGUMENT-NAME* is the option's argument name appearing in help strings. -
ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory
 are synonyms).

It defaults to :optional.

- *ENV-VAR* is the option's associated environment variable.

It defaults to nil.

- *TYPESPEC* is a type specifier the option's value should satisfy.

- *FALLBACK-VALUE* is the option's fallback value (for missing optional arguments), if any.

- *DEFAULT-VALUE* is the option's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [lispobj.lisp], page 16, (file)

make-path &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR
FALLBACK-VALUE DEFAULT-VALUE TYPE HIDDEN

Make a new path option.

- *SHORT-NAME* is the option's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the option's long name (without the double-dash).

It defaults to nil.

- *DESCRIPTION* is the option's description appearing in help strings.

It defaults to nil.

- *ARGUMENT-NAME* is the option's argument name appearing in help strings. -
ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory
 are synonyms).

It defaults to :optional.

- *ENV-VAR* is the option's associated environment variable.

It defaults to nil.

- *FALLBACK-VALUE* is the option's fallback value (for missing optional arguments), if any.

- *DEFAULT-VALUE* is the option's default value, if any.

- *TYPE* is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil
 meaning that everything is allowed.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [path.lisp], page 16, (file)

make-stropt &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR
FALLBACK-VALUE DEFAULT-VALUE HIDDEN

Make a new string option.

- *SHORT-NAME* is the option's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the option's long name (without the double-dash).

It defaults to nil.

- *DESCRIPTION* is the option's description appearing in help strings.

It defaults to nil.

- *ARGUMENT-NAME* is the option's argument name appearing in help strings. -
ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory
are synonyms).

It defaults to :optional.

- *ENV-VAR* is the option's associated environment variable.

It defaults to nil.

- *FALLBACK-VALUE* is the option's fallback value (for missing optional arguments), if any.

- *DEFAULT-VALUE* is the option's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [stropt.lisp], page 16, (file)

make-switch &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR
DEFAULT-VALUE HIDDEN

Make a new switch.

- *SHORT-NAME* is the switch's short name (without the dash).

It defaults to nil.

- *LONG-NAME* is the switch's long name (without the double-dash).

It defaults to nil.

- *DESCRIPTION* is the switch's description appearing in help strings.

It defaults to nil.

- *ARGUMENT-STYLE* is the switch's argument display style. It can be one of :yes/no,
:on/off, :true/false, :yup/nope or :yeah/nah.

It defaults to :yes/no.

- *ARGUMENT-TYPE* is one of :required, :mandatory or :optional (:required and :mandatory
are synonyms).

It defaults to :optional.

- *ENV-VAR* is the switch's associated environment variable.

It defaults to nil.

- *DEFAULT-VALUE* is the switch's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [switch.lisp], page 15, (file)

make-synopsis &rest *KEYS* &key *POSTFIX ITEM MAKE-DEFAULT* [Function]
Make a new SYNOPSIS.

- *POSTFIX* is a string to append to the program synopsis, in case it accepts a remainder.

- If *MAKE-DEFAULT*, make the new synopsis the default one.

Package [net.didierverna.clon], page 27,

- Source** [synopsis.lisp], page 20, (file)
- make-text** &rest *KEYS* &key *CONTENTS HIDDEN* [Function]
 Make a new text.
 - CONTENTS is the actual text to display.
 - When HIDDEN, the text doesn't appear in help strings.
- Package** [net.didierverna.clon], page 27,
Source [text.lisp], page 13, (file)
- make-xswitch** &rest *KEYS* &key *SHORT-NAME LONG-NAME* [Function]
DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM
ENV-VAR DEFAULT-VALUE HIDDEN
 Make a new xswitch.
 - SHORT-NAME is the xswitch's short name (without the dash).
 It defaults to nil.
 - LONG-NAME is the xswitch's long name (without the double-dash).
 It defaults to nil.
 - DESCRIPTION is the xswitch's description appearing in help strings. It defaults to nil.
 - ARGUMENT-NAME is the option's argument name appearing in help strings. -
 ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory
 are synonyms).
 It defaults to :optional.
 - ENUM is the set of possible non-boolean values.
 - ENV-VAR is the xswitch's associated environment variable.
 It defaults to nil.
 - DEFAULT-VALUE is the xswitch's default value, if any.
 - When HIDDEN, the option doesn't appear in help strings.
- Package** [net.didierverna.clon], page 27,
Source [xswitch.lisp], page 17, (file)
- nickname-package** &optional *NICKNAME* [Function]
 Add NICKNAME (:CLON by default) to the :NET.DIDIERVERNA.CLON package.
- Package** [net.didierverna.clon], page 27,
Source [package.lisp], page 12, (file)
- programe** &key *CONTEXT* [Function]
 Return CONTEXT's program name.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- remainder** &key *CONTEXT* [Function]
 Return CONTEXT's remainder.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- setup-termio** () [Function]
 Autodetect termio support.
 Update Clon configuration and *FEATURES* accordingly.
- Package** [net.didierverna.clon.setup], page 39,
Source [termio.lisp], page 26, (file)

version &optional *TYPE* [Function]

Return the current version of Clon.

TYPE can be one of :number, :short or :long.

A version number is computed as $\text{major} * 10000 + \text{minor} * 100 + \text{patchlevel}$, leaving two digits for each level. Alpha, beta and rc status are ignored in version numbers.

A short version is something like 1.3{a,b,rc}4, or 1.3.4 for patchlevel. Alpha, beta or rc levels start at 1. Patchlevels start at 0 but are ignored in the output, so that 1.3.0 appears as just 1.3.

A long version is something like

1.3 {alpha,beta,release candidate,patchlevel} 4 "Michael Brecker". As for the short version, a patchlevel of 0 is ignored in the output.

Package [net.didierverna.clon.setup], page 39,

Source [version.lisp], page 25, (file)

6.2 Internal definitions

6.2.1 Constants

+tiocgwinsz+ [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

offset-of-winsize-ws-col [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

offset-of-winsize-ws-row [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

offset-of-winsize-ws-xpixel [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

offset-of-winsize-ws-ypixel [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

size-of-winsize [Constant]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

6.2.2 Special variables

configuration [Special Variable]

The Clon configuration settings.

This variable contains a property list of configuration options. Current options are:

- :swank-eval-in-emacs (Boolean)
- :restricted (Boolean)
- :dump (Boolean)

See section A.1 of the user manual for more information.

Package [net.didierverna.clon.setup], page 39,

Source [configuration.lisp], page 26, (file)

highlight-properties [Special Variable]

The highlight face properties.

Package [net.didierverna.clon], page 27,

Source [face.lisp], page 20, (file)

item-names [Special Variable]

The list of defined item names.

Package [net.didierverna.clon], page 27,

Source [valued.lisp], page 14, (file)

6.2.3 Macros

%defgroup *INTERNALP* (&rest *KEYS* &key *HEADER HIDDEN*) &body [Macro]
FORMS

Define a new group.

Package [net.didierverna.clon], page 27,

Source [group.lisp], page 18, (file)

accumulate (*INITIAL-VALUE*) &body *BODY* [Macro]

Accumulate *BODY* forms in a list beginning with *INITIAL-VALUE*. *INITIAL-VALUE* is not evaluated. *BODY* forms are accumulated only when their value is non-nil.

If nothing to accumulate, then return nil instead of the list of *INITIAL-VALUE*.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

declare-valid-superclass *CLASS SUPERCLASS* [Macro]

Validate *SUPERCLASS* classes for *CLASS* classes.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

defabstract *CLASS SUPER-CLASSES SLOTS* &rest *OPTIONS* [Macro]

Like *DEFCLASS*, but define an abstract class.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

- defindent** *SYMBOL INDENT* [Macro]
 Wrapper around ‘clindent’ to avoid quoting SYMBOL and INDENT.
- Package** [net.didierverna.clon.setup], page 39,
Source [readtable.lisp], page 26, (file)
- defoption** *CLASS SUPERCLASSES SLOTS &rest OPTIONS* [Macro]
 Create a new option CLASS and register it with Clon.
- Package** [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)
- do-options** (*OPT THERE*) &body *BODY* [Macro]
 Execute BODY with OPT bound to every option in THERE.
- Package** [net.didierverna.clon], page 27,
Source [synopsis.lisp], page 20, (file)
- econd** &body *CLAUSES* [Macro]
 Like COND, but signal an error if no clause evaluates to t.
- Package** [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- endpush** *OBJECT PLACE* [Macro]
 Like push, but at the end.
- Package** [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- highlight-property-ecase** *PROPERTY VALUE &body CLAUSES* [Macro]
 Create an ECASE form to extract PROPERTY’s VALUE escape sequence.
 Each clause looks like: (PROPERTY-NAME (VALUE-OR-VALUE-LIST ESCAPE-SEQUENCE)*). The value-matching part will itself be enclosed in an ECASE expression.
 In addition, the special clause syntax (BOOLEAN <PROPERTY-NAME> <YES> <NO>) is a shortcut for: (PROPERTY-NAME ((on t) YES) ((off nil) NO)).
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- map-frames** *FUNCTION (SHEET &key REVERSE)* [Macro]
 Map FUNCTION over SHEET’s frames. If REVERSE, map in reverse order.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- maybe-pop-argument** *CMDLINE OPTION CMDLINE-ARGUMENT* [Macro]
 Pop OPTION’s argument from CMDLINE if needed. If so, store it in CMDLINE-ARGUMENT.
- Package** [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)

maybe-push *OBJECT PLACE* [Macro]

Like push, but only if OBJECT is non-nil.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

replace-in-keys (*KEY VAL*) *KEYS THE-KEY FORM* [Macro]

Replace every occurrence of THE-KEY in KEYS with FORM.

At every KEYS round, KEY and VAL are bound to the current key-value pair. FORM is evaluated each time and should return a key-value list.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

restartable-invalid-negated-syntax-error (*OPTION*) &body *BODY* [Macro]

Restartably throw an invalid-negated-syntax error.

The error relates to the command-line use of OPTION.

BODY constitutes the body of the only restart available, use-short-call, and should act as if OPTION had been normally called by short name.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

restartable-spurious-cmdline-argument-error (*OPTION NAME ARGUMENT*) &body *BODY* [Macro]

Restartably throw a spurious-cmdline-argument error.

The error relates to the command-line use of OPTION called by NAME with ARGUMENT.

BODY constitutes the body of the only restart available, discard-argument, and should act as if ARGUMENT had not been provided.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

with-context-error-handler *CONTEXT* &body *BODY* [Macro]

Execute BODY with CONTEXT's error handler bound for CONDITION.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

with-winsize *VAR24* (&rest *FIELD-VALUES-25*) &body *BODY26* [Macro]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

6.2.4 Functions

%version *TYPE MAJOR MINOR STATUS LEVEL NAME* [Function]

Package [net.didierverna.clon.setup], page 39,

Source [version.lisp], page 25, (file)

add-subface *FACE SUBFACE* [Function]

Add SUBFACE to FACE's subfaces and return it.

Package [net.didierverna.clon], page 27,

Source [face.lisp], page 20, (file)

`allocate-winsize ()` [Function]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

`argument-popable-p CMDLINE` [Function]

Return true if the first *CMDLINE* item is an argument.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

`attach-face-tree FACE FACE-TREE &aux NEW-TREE` [Function]

Create a copy of *FACE-TREE*, attach it to *FACE* and return it.

Apart from the parenting information, the copied faces share slot values with the original ones.

Package [net.didierverna.clon], page 27,

Source [face.lisp], page 20, (file)

`available-right-margin SHEET` [Function]

Return *SHEET*'s available right margin.

This margin is the first non-self margin specified by a frame. All inner self frames can potentially write until the available right margin.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

`beginning-of-string-p BEGINNING STRING &optional
IGNORE-CASE &aux LENGTH` [Function]

Check that *STRING* starts with *BEGINNING*. If *IGNORE-CASE*, well, ignore case.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

`clindent SYMBOL INDENT` [Function]

Send *SYMBOL*'s *INDENT*ation information to Emacs.

Emacs will set the 'common-lisp-indent-function property.

If *INDENT* is a symbol, use its indentation definition. Otherwise, *INDENT* is considered as an indentation definition.

Package [net.didierverna.clon.setup], page 39,

Source [readtable.lisp], page 26, (file)

`close-line SHEET` [Function]

Close all frames on *SHEET*'s current line and go to next line.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

`close-sface SHEET` [Function]

Close *SHEET*'s current sface.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

- `closest-match` *MATCH LIST &key IGNORE-CASE KEY &aux* [Function]
MATCH-LENGTH SHORTEST-DISTANCE CLOSEST-MATCH
 Return the LIST element closest to MATCH, or nil.
 If IGNORE-CASE, well, ignore case.
 KEY should provide a way to get a string from each LIST element.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- `cmdline-convert` *VALUED-OPTION CMDLINE-NAME* [Function]
CMDLINE-ARGUMENT
 Convert CMDLINE-ARGUMENT to VALUED-OPTION's value.
 This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME, and intercepts invalid-argument errors to raise the higher level invalid-cmdline-argument error instead.
Package [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
- `cmdline-option-name` *INSTANCE* [Function]
 (setf cmdline-option-name) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `cmdline-option-option` *INSTANCE* [Function]
 (setf cmdline-option-option) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `cmdline-option-p` *OBJECT* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `cmdline-option-source` *INSTANCE* [Function]
 (setf cmdline-option-source) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `cmdline-option-value` *INSTANCE* [Function]
 (setf cmdline-option-value) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `complete-string` *BEGINNING COMPLETE* [Function]
 Complete BEGINNING with the rest of COMPLETE in parentheses. For instance, completing 'he' with 'help' will produce 'he(lp)'.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)

- `copy-cmdline-option` *INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- `copy-frame` *INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `copy-highlight-frame` *INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `copy-highlight-property-instance` *INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `current-frame` *SHEET* [Function]
Return SHEET's current frame.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `current-left-margin` *SHEET* [Function]
Return SHEET's current left margin.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `current-right-margin` *SHEET* [Function]
Return SHEET's current right margin.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `current-sface` *SHEET* [Function]
Return SHEET's current sface or nil.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `directory-pathname-p` *PATHNAME* [Function]
Return true if PATHNAME denotes a directory.
Package [net.didierverna.clon], page 27,
Source [path.lisp], page 16, (file)
- `environment-convert` *VALUED-OPTION ENV-VAL* [Function]
Convert ENV-VAL to VALUED-OPTION's value.
This function is used when the conversion comes from an environment variable associated with VALUED-OPTION, and intercepts invalid-argument errors to raise the higher level invalid-environment-value error instead.
Package [net.didierverna.clon], page 27,
Source [environ.lisp], page 19, (file)

- exit-abnormally** *ERROR* [Function]
 Print *ERROR* on **ERROR-OUTPUT** and exit with status code 1.
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- face-highlight-property-set-p** *FACE PROPERTY* [Function]
 Return t if *PROPERTY* is set explicitly in *FACE*.
Package [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
- face-highlight-property-value** *FACE PROPERTY* [Function]
 Return *PROPERTY*'s value in *FACE*.
 Since faces inherit highlight properties, the actual value might come from one of *FACE*'s ancestors.
 if *PROPERTY* is not et, return nil.
Package [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
- find-sface** *SFACE NAME* &aux *SIBLING SUB-SFACE* [Function]
 Find an sface starting at *SFACE* named *NAME*.
 If the sface can't be found in *SFACE*'s face tree, find one in *SFACE*'s sibling instead, and make a copy of it.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- flush-sheet** *SHEET* [Function]
 Flush *SHEET*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- frame-left-margin** *INSTANCE* [Function]
(setf frame-left-margin) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- frame-p** *OBJECT* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- frame-right-margin** *INSTANCE* [Function]
(setf frame-right-margin) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- frame-sface** *INSTANCE* [Function]
(setf frame-sface) *VALUE INSTANCE* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)

- `get-top-padding` *SFACE ITEMS* [Function]
 Return top padding of the next item in *ITEMS* that will print under *SFACE*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `getenv` *VARIABLE* [Function]
 Get environment *VARIABLE*'s value. *VARIABLE* may be null.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- `help-spec-items-will-print` *SFACE ITEMS* [Function]
 Return t if at least one of *ITEMS* will print under *SFACE*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-frame-highlight-property-instances` *INSTANCE* [Function]
 (setf highlight-frame-highlight-property-instances) *VALUE* [Function]
INSTANCE
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-frame-left-margin` *INSTANCE* [Function]
 (setf highlight-frame-left-margin) *VALUE* *INSTANCE* [Function]
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-frame-p` *OBJECT* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-frame-right-margin` *INSTANCE* [Function]
 (setf highlight-frame-right-margin) *VALUE* *INSTANCE* [Function]
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-frame-sface` *INSTANCE* [Function]
 (setf highlight-frame-sface) *VALUE* *INSTANCE* [Function]
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-property-instance-escape-sequence` *INSTANCE* [Function]
 Return highlight property *INSTANCE*'s escape sequence.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- `highlight-property-instance-name` *INSTANCE* [Function]
 (setf highlight-property-instance-name) *VALUE* *INSTANCE* [Function]
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)

- highlight-property-instance-p** *OBJECT* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- highlight-property-instance-value** *INSTANCE* [Function]
(setf highlight-property-instance-value) VALUE INSTANCE [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- home-directory** () [Function]
 Return user's home directory in canonical form.
 If the user's home directory cannot be computed, signal a warning and return NIL.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- i-reader** *STREAM SUBCHAR ARG* [Function]
 Construct a call to 'defindent' by reading an argument list from STREAM. This dispatch macro character function is installed on #i in the NET.DIDIERVERNA.CLON named readtable.
Package [net.didierverna.clon.setup], page 39,
Source [readtable.lisp], page 26, (file)
- list-to-string** *LIST &key KEY SEPARATOR* [Function]
 Return a SEPARATOR-separated string of all LIST elements.
 - KEY should provide a way to get a string from each LIST element. - SEPARATOR is the string to insert between elements.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- macosp** () [Function]
 Return t if running on Mac OS.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- make-cmdline-option** *&key (NAME NAME) (OPTION OPTION) (VALUE VALUE) (SOURCE SOURCE)* [Function]
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- make-frame** *&key (SPACE SPACE) (LEFT-MARGIN LEFT-MARGIN) (RIGHT-MARGIN RIGHT-MARGIN)* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- make-highlight-frame** *&key (SPACE SPACE) (LEFT-MARGIN LEFT-MARGIN) (RIGHT-MARGIN RIGHT-MARGIN) (HIGHLIGHT-PROPERTY-INSTANCES HIGHLIGHT-PROPERTY-INSTANCES)* [Function]
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)

`make-highlight-property-instance` *&key (NAME NAME) (VALUE VALUE)* [Function]

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

`make-internal-enum` *LONG-NAME DESCRIPTION &rest KEYS &key ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) enum option.

- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)

- DESCRIPTION is the options's description.

- ARGUMENT-NAME is the option's argument name appearing in help strings.

- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- ENUM is the set of possible values.

- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.

- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.

- DEFAULT-VALUE is the option's default value, if any.

- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [enum.lisp], page 17, (file)

`make-internal-flag` *LONG-NAME DESCRIPTION &rest KEYS &key ENV-VAR HIDDEN* [Function]

Make a new internal (Clon-specific) flag.

- LONG-NAME is the flag's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)

- DESCRIPTION is the flag's description.

- ENV-VAR is the flag's associated environment variable, sans the 'CLON_' prefix. It default to nil.

- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [flag.lisp], page 14, (file)

`make-internal-lispobj` *LONG-NAME DESCRIPTION &rest KEYS &key ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR TYPESPEC FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

Make a new internal (Clon-specific) string option.

- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)

- DESCRIPTION is the options's description.

- ARGUMENT-NAME is the option's argument name appearing in help strings.

- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.

- TYPESPEC is a type specifier the option's value should satisfy.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [lispobj.lisp], page 16, (file)

make-internal-path *LONG-NAME DESCRIPTION* &rest *KEYS* &key [Function]
ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE
DEFAULT-VALUE TYPE HIDDEN

Make a new internal (Clon-specific) path option.

- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- TYPE is the pathname type. It can be one of :file, :directory, :file-list, :directory-list or nil meaning that everything is allowed.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [path.lisp], page 16, (file)

make-internal-stropt *LONG-NAME DESCRIPTION* &rest *KEYS* [Function]
&key *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR*
FALLBACK-VALUE DEFAULT-VALUE HIDDEN

Make a new internal (Clon-specific) string option.

- LONG-NAME is the option's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)
- DESCRIPTION is the options's description.
- ARGUMENT-NAME is the option's argument name appearing in help strings.
- ARGUMENT-TYPE is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- ENV-VAR is the option's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.
- FALLBACK-VALUE is the option's fallback value (for missing optional arguments), if any.
- DEFAULT-VALUE is the option's default value, if any.
- When HIDDEN, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [stropt.lisp], page 16, (file)

make-internal-switch *LONG-NAME DESCRIPTION* &rest *KEYS* [Function]
 &key *ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR*
DEFAULT-VALUE HIDDEN

Make a new internal (Clon-specific) switch.

- *LONG-NAME* is the switch's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)

- *DESCRIPTION* is the switch's description.

- *ARGUMENT-STYLE* is the switch's argument display style. It can be one of :yes/no, :on/off, :true/false, :yup/nope or :yeah/nah.

It defaults to :yes/no.

- *ARGUMENT-TYPE* is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- *ENV-VAR* is the switch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.

- *DEFAULT-VALUE* is the switch's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [switch.lisp], page 15, (file)

make-internal-text &rest *KEYS* &key *CONTENTS HIDDEN* [Function]

Package [net.didierverna.clon], page 27,

Source [text.lisp], page 13, (file)

make-internal-xswitch *LONG-NAME DESCRIPTION* &rest *KEYS* [Function]
 &key *ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR*
DEFAULT-VALUE HIDDEN

Make a new internal (Clon-specific) xswitch.

- *LONG-NAME* is the xswitch's long-name, sans the 'clon-' prefix. (Internal options don't have short names.)

- *DESCRIPTION* is the xswitch's description.

- *ARGUMENT-NAME* is the option's argument name appearing in help strings.

- *ARGUMENT-TYPE* is one of :required, :mandatory or :optional (:required and :mandatory are synonyms).

It defaults to :optional.

- *ENUM* is the set of possible non-boolean values.

- *ENV-VAR* is the xswitch's associated environment variable, sans the 'CLON_' prefix. It defaults to nil.

- *DEFAULT-VALUE* is the xswitch's default value, if any.

- When *HIDDEN*, the option doesn't appear in help strings.

Package [net.didierverna.clon], page 27,

Source [xswitch.lisp], page 17, (file)

make-raw-face-tree &optional *FACE-CLASS* [Function]

Make a raw (boring yet functional) face tree.

Package [net.didierverna.clon], page 27,

Source [face.lisp], page 20, (file)

- make-raw-sface** *SIBLING* &aux *SFACE* [Function]
 Return a new SFace based on SIBLING.
 This function does not consider SIBLING as a face tree:
 only face properties are copied; the face parent and children are set to nil.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- make-sheet** &rest *KEYS* &key *OUTPUT-STREAM SEARCH-PATH* [Function]
THEME LINE-WIDTH HIGHLIGHT
 Make a new SHEET.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- match-option** *OPTION* &key *SHORT-NAME LONG-NAME* [Function]
 Try to match OPTION against SHORT-NAME, LONG-NAME. If OPTION matches, return
 the name that matched.
Package [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
- open-line** *SHEET* [Function]
 Open all frames on SHEET's current line.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- open-next-line** *SHEET* [Function]
 Close SHEET's current line and open the next one.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- open-sface** *SHEET SFACE* [Function]
 Create a frame for SFACE and open it.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- option-abbreviation-distance** *OPTION PARTIAL-NAME* [Function]
 Return the distance between OPTION's long name and PARTIAL-NAME. If PARTIAL-
 NAME does not abbreviate OPTION's long name, return MOST-POSITIVE-FIXNUM.
Package [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
- option-call-p** *STR* [Function]
 Return true if STR looks like an option call.
Package [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)

- parent-generation** *FACE PARENT-NAME* [Function]
Return *FACE*'s parent generation for *PARENT-NAME*.
That is, 1 if *PARENT-NAME* names *FACE*'s parent, 2 if it names its grand-parent etc. If *PARENT-NAME* does not name one of *FACE*'s ancestors, trigger an error.
- Package** [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
- pathname-component-null-p** *COMPONENT* [Function]
Return true if *COMPONENT* is either null or :unspecific.
- Package** [net.didierverna.clon], page 27,
Source [path.lisp], page 16, (file)
- pop-frame** *SHEET* [Function]
Pop *SHEET*'s current frame.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- potential-pack-char** *OPTION* &optional *AS-STRING* [Function]
Return *OPTION*'s potential pack character, if any. If *AS-STRING*, return a string of that character.
- Package** [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
- princ-char** *SHEET CHAR* [Function]
Princ *CHAR* on *SHEET*'s stream and increment the column position.
The effect of printing *CHAR* must be exactly to move right by one column, so control characters, as well as newlines and tabs are forbidden here.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- princ-highlight-property-instances** *SHEET INSTANCES* [Function]
Princ highlight proeprty *INSTANCES* on *SHEET*'s stream.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- princ-spaces** *SHEET NUMBER* [Function]
Princ *NUMBER* spaces to *SHEET*'s stream and update the column position.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- princ-string** *SHEET STRING* [Function]
Princ *STRING* on *SHEET*'s stream and update the column position.
The effect of printing *STRING* must be exactly to move right by the corresponding string length, so control characters, as well as newlines and tabs are forbidden here.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)

- print-error** *ERROR* &optional *INTERACTIVEP* &aux *STREAM* **PRINT-ESCAPE** [Function]
 Print *ERROR* on **ERROR-OUTPUT**.
 When *INTERACTIVEP*, print on **QUERY-IO** instead.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- print-faced-help-spec** *SHEET SFACE ITEMS* [Function]
 Print all help specification *ITEMS* on *SHEET* with *SFACE*.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- print-help** *SHEET HELP* [Function]
 Open the toplevel help face and print *HELP* on *SHEET* with it.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- print-string** *SHEET STRING* [Function]
 Output *STRING* to *SHEET*.
STRING is output within the current frame's bounds.
 Spacing characters are honored but newlines might replace spaces when the output reaches the rightmost bound.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- push-frame** *SHEET FRAME* [Function]
 Push a new frame to *SHEET*'s frames.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- putenv** *VARIABLE VALUE* [Function]
 Set environment *VARIABLE* to *VALUE*.
- Package** [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- reach-column** *SHEET COLUMN* [Function]
 Reach *COLUMN* on *SHEET* by princ'ing spaces.
- Package** [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- read-argument** () [Function]
 Read an option argument from standard input.
- Package** [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)

- read-call** *&optional NEGATED* [Function]
 Read an option's call or pack from standard input.
 If *NEGATED*, read a negated call or pack. Otherwise, read a short call or pack.
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- read-env-val** *ENV-VAR* [Function]
 Read *ENV-VAR*'s new value from standard input.
Package [net.didierverna.clon], page 27,
Source [environ.lisp], page 19, (file)
- read-long-name** () [Function]
 Read an option's long name from standard input.
Package [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- read-sface-tree** *PATHNAME* [Function]
 Read an sface tree from *PATHNAME*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
- read-value** () [Function]
 Read an option value from standard input.
Package [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)
- release-status-number** *RELEASE-STATUS* [Function]
Package [net.didierverna.clon.setup], page 39,
Source [version.lisp], page 25, (file)
- remove-keys** *KEYS &rest REMOVED* [Function]
 Return a new property list from *KEYS* without *REMOVED* ones.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
- replace-key** *REPLACEMENT KEYS* [Function]
 Return a new property list from *KEYS* with *REPLACEMENT*.
REPLACEMENT can take the following forms:
 - *:KEY*
 The effect is to remove *:KEY* from *KEYS*, as per *REMOVE-KEYS*.
 - (*:KEY* *:NEW-KEY*)
 The effect is to replace *:KEY* with *:NEW-KEY*, leaving the values unchanged. - (*:KEY* *:NEW-KEY* (*VAL-OR-VALS* *NEW-VAL*)*), with *VAL-OR-VALS* being either a value or a list of values. The effect is to replace *:KEY* with *:NEW-KEY* and a value matching one of the *VAL-OR-VALS* with the corresponding *NEW-VAL*. Values not matching any *VAL-OR-VALS* remain unchanged. - (*:KEY* (*VAL-OR-VALS* *:NEW-KEY* *NEW-VAL*...)*), with *VAL-OR-VALS* as above. The effect is the same as above, but *:NEW-KEY* additionally depends on the matched value. If multiple *:NEW-KEY* *NEW-VAL* couples

are provided, that many new keys are inserted along with their values. For values not matching any VAL-OR-VALS, :KEY and its value remain unchanged.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

replace-keys *KEYS* &rest *REPLACEMENTS* &aux *NEW-KEYS* [Function]

Return a new property list from KEYS with REPLACEMENTS.

See REPLACE-KEY for more information on the replacement syntax.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

restart-on-error *ERROR* [Function]

Print ERROR and offer available restarts on *QUERY-IO*.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

restartable-check *VALUED-OPTION* *VALUE* [Function]

Restartably check that VALUE is valid for VALUED-OPTION.

The only restart available, use-value, offers to try a different value from the one that was provided.

Package [net.didierverna.clon], page 27,

Source [valued.lisp], page 14, (file)

restartable-cmdline-convert *VALUED-OPTION* *CMDLINE-NAME* *CMDLINE-ARGUMENT* [Function]

Restartably convert CMDLINE-ARGUMENT to VALUED-OPTION's value.

This function is used when the conversion comes from a command-line usage of VALUED-OPTION, called by CMDLINE-NAME.

As well as conversion errors, this function might raise a missing-cmdline-argument error if CMDLINE-ARGUMENT is nil and an argument is required.

Available restarts are (depending on the context):

- use-fallback-value: return FALLBACK-VALUE,
- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument.

Return two values: VALUED-OPTION's value and the actual value source. The value source may be :cmdline, :fallback or :default.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

restartable-cmdline-junk-error *JUNK* [Function]

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

restartable-convert *VALUED-OPTION ARGUMENT* [Function]

Restartably convert ARGUMENT to VALUED-OPTION's value. Available restarts are:

- use-default-value: return OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument.

Package [net.didierverna.clon], page 27,

Source [valued.lisp], page 14, (file)

restartable-environment-convert *VALUED-OPTION ENV-VAL* [Function]

Restartably convert ENV-VAL to VALUED-OPTION's value.

This function is used when the conversion comes from an environment variable associated with VALUED-OPTION.

Available restarts are:

- use-default-value: return VALUED-OPTION's default value,
- use-value: return another (already converted) value,
- use-argument: return the conversion of another argument,
- modify-env: modify the environment variable's value.

Package [net.didierverna.clon], page 27,

Source [environ.lisp], page 19, (file)

restrict-because *REASON* [Function]

Put Clon in restricted mode because of REASON.

Package [net.didierverna.clon.setup], page 39,

Source [termio.lisp], page 26, (file)

safe-left-margin *SHEET MARGIN* [Function]

Return either MARGIN or a safe value instead.

To be safe, margin must be greater than the current left margin and smaller than the currently available margin.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

safe-right-margin *SHEET LEFT-MARGIN MARGIN* [Function]

Return either MARGIN or a safe value instead.

To be safe, margin must be greater than LEFT-MARGIN and smaller than the currently available right margin.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

search-branch *FACE NAMES* [Function]

Search for a branch of faces named NAMES starting at FACE.

The branch is searched for as a direct subbranch of FACE, or as a direct subbranch of FACE's ancestors.

If a branch is found, return its leaf face. Otherwise return nil.

Package [net.didierverna.clon], page 27,

Source [face.lisp], page 20, (file)

- search-face** *FACE NAME* &optional *ERROR-ME* [Function]
 Search for a face named *NAME* starting at *FACE*.
 The face is looked for as a direct subface of *FACE* (in which case it is simply returned), or up in the hierarchy and by successive upper branches (in which case it is copied and attached to *FACE*).
 If *ERROR-ME*, trigger an error if no face is found; otherwise, return nil.
- Package** [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
- search-option** *CONTEXT* &rest *KEYS* &key *SHORT-NAME* [Function]
LONG-NAME PARTIAL-NAME
 Search for an option in *CONTEXT*.
 The search is done with *SHORT-NAME*, *LONG-NAME*, or *PARTIAL-NAME*.
 In case of a *PARTIAL-NAME* search, look for an option the long name of which begins with it.
 In case of multiple matches by *PARTIAL-NAME*, the longest match is selected. When such an option exists, return two values:
 - the option itself,
 - the name used to find the option, possibly completed if partial.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- search-option-by-abbreviation** *CONTEXT PARTIAL-NAME* [Function]
 Search for option abbreviated with *PARTIAL-NAME* in *CONTEXT*. When such an option exists, return two values:
 - the option itself,
 - the completed name.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- search-option-by-name** *CONTEXT* &rest *KEYS* &key *SHORT-NAME* [Function]
LONG-NAME
 Search for option with either *SHORT-NAME* or *LONG-NAME* in *CONTEXT*. When such an option exists, return two values:
 - the option itself,
 - the name that matched.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- search-sticky-option** *CONTEXT NAMEARG* [Function]
 Search for a sticky option in *CONTEXT*, matching *NAMEARG*.
NAMEARG is the concatenation of the option's short name and its argument. In case of multiple matches, the option with the longest name is selected. When such an option exists, return two values:
 - the option itself,
 - the argument part of *NAMEARG*.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)

select-keys *KEYS* &rest *SELECTED* [Function]

Return a new property list from *KEYS* with only *SELECTED* ones.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

split-path *PATH* [Function]

Split *PATH* into a list of directories.

Package [net.didierverna.clon], page 27,

Source [path.lisp], page 16, (file)

stream-line-width *STREAM* &aux *HANDLE* [Function]

Get *STREAM*'s line width.

Return two values:

- the stream's line width, or nil if it can't be computed (typically when the stream does not denote a tty), - an error message if the operation failed.

Package [net.didierverna.clon], page 27,

Source [termio.lisp], page 11, (file)

try-read-sface-tree *PATHNAME* [Function]

Read an sface tree from *PATHNAME* if it exists or return nil.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

try-read-theme *PATHNAME* [Function]

Read a theme from *PATHNAME* or *PATHNAME*.cth if it exists or return nil.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

winsize-ws-col *STRUCT* [Function]

(setf winsize-ws-col) *VAR-21 STRUCT* [Function]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

winsize-ws-row *STRUCT* [Function]

(setf winsize-ws-row) *VAR-0 STRUCT* [Function]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

winsize-ws-xpixel *STRUCT* [Function]

(setf winsize-ws-xpixel) *VAR-22 STRUCT* [Function]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

`winsize-ws-yapixel` *STRUCT* [Function]
 (`setf winsize-ws-yapixel`) *VAR-23 STRUCT* [Function]

Package [net.didierverna.clon], page 27,

Source /Users/didier/.cache/common-lisp/sbcl-2.1.2.72-f990ff7ad-macosx-x64/Users/didier/Lisp/clon/termio/sbcl/constants.lisp-temp

`~-reader` *STREAM CHAR* [Function]
 Read a series of ~"string" to be concatenated together.

Package [net.didierverna.clon.setup], page 39,

Source [readtable.lisp], page 26, (file)

6.2.5 Generic functions

`argument` *CONDITION* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

`argument` (*CONDITION* *unknown-cmdline-option-error*) [Method]
Source [context.lisp], page 24, (file)

`argument` (*CONDITION* *spurious-cmdline-argument*) [Method]
Source [cmdline.lisp], page 18, (file)

`argument` (*CONDITION* *invalid-argument*) [Method]
Source [valued.lisp], page 14, (file)

`argument-name` *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

`argument-name` (*VALUED-OPTION* *valued-option*) [Method]
 The option's argument display name.
Source [valued.lisp], page 14, (file)

`argument-required-p` *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

`argument-required-p` (*VALUED-OPTION* *valued-option*) [Method]
 Whether the option's argument is required.
Source [valued.lisp], page 14, (file)

`argument-style` *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

`argument-style` (*SWITCH-BASE* *switch-base*) [Method]
 The selected argument style.
Source [switch-base.lisp], page 15, (file)

- argument-styles** *OBJECT* [Generic Function]
 (setf argument-styles) *NEW-VALUE OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- argument-styles** (*SWITCH-BASE* switch-base) [Method]
 (setf argument-styles) *NEW-VALUE* (*SWITCH-BASE* switch-base) [Method]
 The possible argument styles.
 The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.
Source [switch-base.lisp], page 15, (file)
- background** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- background** (*FACE* face) [Method]
 The face background.
Source [face.lisp], page 20, (file)
- blink** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- blink** (*FACE* face) [Method]
 The face's blink speed.
Source [face.lisp], page 20, (file)
- bottom-padding** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- bottom-padding** (*FACE* face) [Method]
 The face bottom padding.
 This property can take the following forms:
 - nil: the next output can start right at the end of this face's, - 0: the next output should start on the next line,
 - N>0: there should be N empty lines before the next output.
Source [face.lisp], page 20, (file)
- check** *VALUED-OPTION VALUE* [Generic Function]
 Check that *VALUE* is valid for *VALUED-OPTION*.
 If *VALUE* is valid, return it. Otherwise, raise an invalid-value error.
Package [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)
Methods

- check** (*XSWITCH* *xswitch*) *VALUE* [Method]
 Check that *VALUE* is valid for *XSWITCH*.
Source [xswitch.lisp], page 17, (file)
- check** (*ENUM* *enum*) *VALUE* [Method]
 Check that *VALUE* is valid for *ENUM*.
Source [enum.lisp], page 17, (file)
- check** (*PATH* *path*) *VALUE* [Method]
 Check that *VALUE* is valid for *PATH*.
Source [path.lisp], page 16, (file)
- check** (*LISPOBJ* *lispobj*) *VALUE* [Method]
 Check that *VALUE* is valid for *LISPOBJ*.
Source [lispobj.lisp], page 16, (file)
- check** (*STROPT* *stropt*) *VALUE* [Method]
 Check that *VALUE* is valid for *STROPT*.
Source [stropt.lisp], page 16, (file)
- check** (*SWITCH* *switch*) *VALUE* [Method]
 Check that *VALUE* is valid for *SWITCH*.
Source [switch.lisp], page 15, (file)
- check-name-clash** *ITEM1* *ITEM2* [Generic Function]
 Check for name clash between *ITEM1*'s options and *ITEM2*'s options.
Package [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
Methods
- check-name-clash** (*CONTAINER1* *container*) [Method]
 (*CONTAINER2* *container*)
 Check for name clash between *CONTAINER1*'s options and *CONTAINER2*'s ones.
Source [container.lisp], page 18, (file)
- check-name-clash** *ITEM1* (*CONTAINER* *container*) [Method]
 Check for name clash between *ITEM1*'s options and *CONTAINER*'s ones.
Source [container.lisp], page 18, (file)
- check-name-clash** (*CONTAINER* *container*) *ITEM2* [Method]
 Check for name clash between *CONTAINER*'s options and *ITEM2*'s ones.
Source [container.lisp], page 18, (file)
- check-name-clash** *ITEM1* (*TEXT* *text*) [Method]
 Do nothing (no name clash with a text object).
- check-name-clash** (*TEXT* *text*) *ITEM2* [Method]
 Do nothing (no name clash with a text object).

- `check-name-clash` (*OPTION1* option) (*OPTION2* option) [Method]
 Ensure that there is no name clash between *OPTION1* and *OPTION2*.
- `clon-options-group` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `clon-options-group` (*CONTEXT* context) [Method]
 Return the Clon options group of *CONTEXT*'s synopsis.
Source [context.lisp], page 24, (file)
- `clon-options-group` (*SYNOPSIS* synopsis) [Method]
 The Clon options group.
Source [synopsis.lisp], page 20, (file)
- `close-frame` *SHEET FRAME* [Generic Function]
 Close *FRAME* on *SHEET*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
Method Combination
 progn (short method combination)
Options: :most-specific-last
- Methods**
- `close-frame` *SHEET* (*FRAME* frame) &aux *RIGHT-MARGIN* progn [Method]
 Reach *FRAME*'s right margin if it has one.
- `close-frame` *SHEET* (*FRAME* highlight-frame) progn [Method]
 Restore the upper frame's highlight properties.
- `cmdline-options` *OBJECT* [Generic Function]
 (setf `cmdline-options`) *NEW-VALUE OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `cmdline-options` (*CONTEXT* context) [Method]
 (setf `cmdline-options`) *NEW-VALUE* (*CONTEXT* context) [Method]
 The options from the command-line.
Source [context.lisp], page 24, (file)
- `column` *OBJECT* [Generic Function]
 (setf `column`) *NEW-VALUE OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `column` (*SHEET* sheet) [Method]
 (setf `column`) *NEW-VALUE* (*SHEET* sheet) [Method]
 The sheet's current column.
Source [sheet.lisp], page 21, (file)

- `comment` *CONDITION* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- `comment` (*CONDITION* invalid-argument) [Method]

Source [valued.lisp], page 14, (file)
 - `comment` (*CONDITION* invalid-value) [Method]

Source [valued.lisp], page 14, (file)
- `concealedp` *OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- `concealedp` (*FACE* face) [Method]

The face's concealed status.

Source [face.lisp], page 20, (file)
- `contents` *OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- `contents` (*TEXT* text) [Method]

The actual text string.

Source [text.lisp], page 13, (file)
- `convert` *VALUED-OPTION ARGUMENT* [Generic Function]
- Convert ARGUMENT to VALUED-OPTION's value.
If ARGUMENT is invalid, raise an invalid-argument error.
- Package** [net.didierverna.clon], page 27,
- Source** [valued.lisp], page 14, (file)
- Methods**
- `convert` (*XSWITCH* xswitch) *ARGUMENT* [Method]

Convert ARGUMENT to an XSWITCH value.

Source [xswitch.lisp], page 17, (file)
 - `convert` (*ENUM* enum) *ARGUMENT* [Method]

Convert ARGUMENT to an ENUM value.

Source [enum.lisp], page 17, (file)
 - `convert` (*PATH* path) *ARGUMENT* [Method]

Convert ARGUMENT to a PATH value.

Source [path.lisp], page 16, (file)
 - `convert` (*LISPOBJ* lispobj) *ARGUMENT* [Method]

Convert ARGUMENT to a LISPOBJ value.

Source [lispobj.lisp], page 16, (file)

- convert** (*STROPT* stropt) *ARGUMENT* [Method]
 Convert *ARGUMENT* to an *STROPT* value.
Source [stropt.lisp], page 16, (file)
- convert** (*SWITCH* switch) *ARGUMENT* [Method]
 Convert *ARGUMENT* to a *SWITCH* value.
Source [switch.lisp], page 15, (file)
- copy-instance** *INSTANCE* &optional *SUBCLASS* [Generic Function]
 Return a copy of *INSTANCE*.
 Copy is either an object of *INSTANCE*'s class, or *INSTANCE*'s *SUBCLASS* if given.
Package [net.didierverna.clon], page 27,
Source [util.lisp], page 12, (file)
Methods
- copy-instance** *INSTANCE* &optional *SUBCLASS* [Method]
 Return a copy of *INSTANCE*.
 Both instances share the same slot values.
- crossed-out-p** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- crossed-out-p** (*FACE* face) [Method]
 The face's crossed out status.
Source [face.lisp], page 20, (file)
- default-value** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- default-value** (*VALUED-OPTION* valued-option) [Method]
 The option's default value.
Source [valued.lisp], page 14, (file)
- description** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- description** (*OPTION* option) [Method]
 The option's description.
Source [option.lisp], page 13, (file)
- enum** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- enum** (*ENUM-BASE* enum-base) [Method]
 The set of possible values.
Source [enum-base.lisp], page 17, (file)

- `env-val` *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `env-val` (*CONDITION* *invalid-environment-value*) [Method]
 Source [environ.lisp], page 19, (file)
- `env-var` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `env-var` (*CONDITION* *environment-error*) [Method]
 Source [environ.lisp], page 19, (file)
 `env-var` (*OPTION* *option*) [Method]
 The option's associated environment variable.
 Source [option.lisp], page 13, (file)
- `error-handler` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `error-handler` (*CONTEXT* *context*) [Method]
 The behavior to adopt on option retrieval errors.
 Source [context.lisp], page 24, (file)
- `error-string` *CONDITION* [Generic Function]
`(setf error-string) NEW-VALUE CONDITION` [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `error-string` (*CONDITION* *home-directory*) [Method]
 `(setf error-string) NEW-VALUE (CONDITION
 home-directory) [Method]
 Source [util.lisp], page 12, (file)`
- `fallback-value` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `fallback-value` (*VALUED-OPTION* *valued-option*) [Method]
 The option's fallback value.
 Source [valued.lisp], page 14, (file)
- `foreground` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
 `foreground` (*FACE* *face*) [Method]
 The face foreground.
 Source [face.lisp], page 20, (file)

- framedp** *OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- framedp** (*FACE* face) [Method]
The face's framed status.
- Source** [face.lisp], page 20, (file)
- frames** *OBJECT* [Generic Function]
(setf frames) *NEW-VALUE OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- frames** (*SHEET* sheet) [Method]
(setf frames) *NEW-VALUE (SHEET sheet)* [Method]
The stack of currently open frames.
- Source** [sheet.lisp], page 21, (file)
- get-bottom-padding** *SFACE HELP-SPEC* [Generic Function]
Get HELP-SPEC's bottom-padding under SFACE.
- Package** [net.didierverna.clon], page 27,
- Source** [sheet.lisp], page 21, (file)
- Methods**
- get-bottom-padding** *SFACE HELP-SPEC* [Method]
Basic help specifications (chars, strings etc) don't provide a bottom padding.
- get-bottom-padding** *SFACE (HELP-SPEC list)* [Method]
Return the bottom padding of HELP-SPEC's face.
- header** *OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- header** (*GROUP* group) [Method]
The group's header.
- Source** [group.lisp], page 18, (file)
- help-spec** *ITEM &key PROGRAM UNHIDE &allow-other-keys* [Generic Function]
Return ITEM's help specification.
- Package** [net.didierverna.clon], page 27,
- Source** [item.lisp], page 13, (file)
- Methods**
- help-spec** (*SYNOPSIS synopsis*) &key *PROGRAM* [Method]
Return SYNOPSIS's help specification.
- Source** [synopsis.lisp], page 20, (file)

- help-spec** (*GROUP* group) &key [Method]
Return GROUP's help specification.
Source [group.lisp], page 18, (file)
- help-spec** (*CONTAINER* container) &key [Method]
Return CONTAINER's help specification.
Source [container.lisp], page 18, (file)
- help-spec** (*OPTION* valued-option) &key [Method]
Return OPTION's help specification.
Source [valued.lisp], page 14, (file)
- help-spec** (*OPTION* option) &key [Method]
Return OPTION's help specification.
Source [option.lisp], page 13, (file)
- help-spec** (*TEXT* text) &key [Method]
Return TEXT's help specification.
Source [text.lisp], page 13, (file)
- help-spec** (*ITEM* item) &key *UNHIDE* around [Method]
Call the actual method only when ITEM is not hidden or UNHIDE.
- help-spec-will-print** *SFACE HELP-SPEC* [Generic Function]
Return t if HELP-SPEC will print under FACE.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
Methods
- help-spec-will-print** *SFACE HELP-SPEC* before [Method]
- help-spec-will-print** *SFACE HELP-SPEC* [Method]
Basic help specifications (chars, strings etc) do print.
- help-spec-will-print** *SFACE (HELP-SPEC list)* [Method]
Return t if HELP-SPEC's items will print under HELP-SPEC's face.
- hiddenp** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- hiddenp** (*ITEM* item) [Method]
Whether the item is hidden in help strings.
Source [item.lisp], page 13, (file)
- highlight** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- highlight** (*CONTEXT* context) [Method]
Clon's output highlight mode.
Source [context.lisp], page 24, (file)

- highlightp** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- highlightp** (*SHEET* sheet) [Method]
Whether to highlight SHEET's output.
Source [sheet.lisp], page 21, (file)
- intensity** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- intensity** (*FACE* face) [Method]
The face intensity.
Source [face.lisp], page 20, (file)
- inversep** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- inversep** (*FACE* face) [Method]
The face's inverse video status.
Source [face.lisp], page 20, (file)
- italicp** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- italicp** (*FACE* face) [Method]
The face's italic status.
Source [face.lisp], page 20, (file)
- item** *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- item** (*CONDITION* cmdline-error) [Method]
Source [cmdline.lisp], page 18, (file)
- item-separator** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- item-separator** (*FACE* face) [Method]
The face item separator.
Source [face.lisp], page 20, (file)
- items** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods

- items** (*CONTAINER* container) [Method]
 The items in the container.
Source [container.lisp], page 18, (file)
- junk** *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- junk** (*CONDITION* cmdline-junk-error) [Method]
Source [context.lisp], page 24, (file)
- left-padding** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- left-padding** (*FACE* face) [Method]
 The face left padding.
 This property can take the following forms:
 - <NUMBER>: the padding is relative to the enclosing face,
 - SELF: the padding is set to wherever the face happens to be opened, -
 (<NUMBER> ABSOLUTE): the padding is set in absolute value,
 - (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.
Source [face.lisp], page 20, (file)
- line-width** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- line-width** (*CONTEXT* context) [Method]
 The line width for help display.
Source [context.lisp], page 24, (file)
- line-width** (*SHEET* sheet) [Method]
 The sheet's line width.
Source [sheet.lisp], page 21, (file)
- long-name** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- long-name** (*OPTION* option) [Method]
 The option's long name.
Source [option.lisp], page 13, (file)
- make-face-tree** *DEFINITION* &optional *FACE-CLASS* [Generic Function]
 Make a FACE-CLASS face tree from DEFINITION.
Package [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
Methods

- `make-face-tree` (*DEFINITION* list) &optional *FACE-CLASS* [Method]
 Make a *FACE-CLASS* face tree from a list of face name and initargs.
- `make-face-tree` (*NAME* symbol) &optional *FACE-CLASS* [Method]
 Create a face named *NAME*.
- `mapoptions` *FUNC THERE* [Generic Function]
 Map *FUNC* over all options in *THERE*.
- Package** [net.didierverna.clon], page 27,
Source [synopsis.lisp], page 20, (file)
Methods
- `mapoptions` *FUNC (CONTEXT context)* [Method]
 Map *FUNC* over all options in *CONTEXT* synopsis.
Source [context.lisp], page 24, (file)
- `mapoptions` *FUNC ELSEWHERE* [Method]
 Do nothing by default.
- `mapoptions` *FUNC (ITEM item) after* [Method]
 Mark *TRAVERSABLE* as traversed.
- `mapoptions` *FUNC (CONTAINER container)* [Method]
 Map *FUNC* over all containers or options in *CONTAINER*.
- `mapoptions` *FUNC (OPTION option)* [Method]
 Call *FUNC* on *OPTION*.
- `name` *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `name` (*CONDITION* unknown-cmdline-option-error) [Method]
Source [context.lisp], page 24, (file)
- `name` (*FACE* face) [Method]
 The face name.
Source [face.lisp], page 20, (file)
- `name` (*CONDITION* cmdline-option-error) [Method]
Source [cmdline.lisp], page 18, (file)
- `negated-call` *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `negated-call` (*CONDITION* unrecognized-negated-call-error) [Method]
Source [context.lisp], page 24, (file)

- negated-pack** *OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- negated-pack** (*CONTEXT* context) [Method]
Return the negated pack of *CONTEXT*'s synopsis.
- Source** [context.lisp], page 24, (file)
- negated-pack** (*SYNOPSIS* synopsis) [Method]
The negated pack string.
- Source** [synopsis.lisp], page 20, (file)
- negated-pack-char** *OPTION* &optional *AS-STRING* [Generic Function]
Return *OPTION*'s negated pack character, if any. If *AS-STRING*, return a string of that character.
- Package** [net.didierverna.clon], page 27,
- Source** [option.lisp], page 13, (file)
- Methods**
- negated-pack-char** (*NEGATABLE* negatable) &optional *AS-STRING* [Method]
Return *NEGATABLE*'s negated pack character, if any.
- Source** [negatable.lisp], page 15, (file)
- negated-pack-char** (*OPTION* option) &optional *AS-STRING* [Method]
Return nil (only the switch hierarchy is negated-pack'able).
- no-values** *OBJECT* [Generic Function]
- (setf no-values) *NEW-VALUE OBJECT* [Generic Function]
- Package** [net.didierverna.clon], page 27,
- Methods**
- no-values** (*SWITCH-BASE* switch-base) [Method]
- (setf no-values) *NEW-VALUE* (*SWITCH-BASE* switch-base) [Method]
The possible 'no' values.
- Source** [switch-base.lisp], page 15, (file)
- open-frame** *SHEET FRAME* [Generic Function]
Open *FRAME* on *SHEET*.
- Package** [net.didierverna.clon], page 27,
- Source** [sheet.lisp], page 21, (file)
- Method Combination**
progn (short method combination)
- Options:** :most-specific-last
- Methods**

- `open-frame` *SHEET* (*FRAME* frame) progn [Method]
Reach the frame's left margin.
- `open-frame` *SHEET* (*FRAME* highlight-frame) progn [Method]
Reach the frame's left margin and output its highlight properties.
- `option` *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `option` (*CONDITION* option-error) [Method]
Source [option.lisp], page 13, (file)
- `option-sticky-distance` *OPTION* *NAMEARG* [Generic Function]
Try to match *OPTION*'s short name with a sticky argument against *NAMEARG*. If *OPTION* matches, return the length of *OPTION*'s short name; otherwise 0.
Package [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
Methods
- `option-sticky-distance` (*OPTION* valued-option) [Method]
NAMEARG
Try to match *OPTION*'s short name with a sticky argument against *NAMEARG*. If *OPTION* matches, return its short name's length; otherwise 0.
Source [valued.lisp], page 14, (file)
- `option-sticky-distance` (*OPTION* option) *NAMEARG* [Method]
Return 0 (non-valued options don't take any argument, sticky or not).
- `output-stream` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `output-stream` (*SHEET* sheet) [Method]
The sheet's output stream.
Source [sheet.lisp], page 21, (file)
- `parent` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `parent` (*FACE* face) [Method]
The face parent.
Source [face.lisp], page 20, (file)
- `path-type` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods

- `path-type` (*PATH* path) [Method]
 The path type.
Source [path.lisp], page 16, (file)
- `postfix` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `postfix` (*CONTEXT* context) [Method]
 Return the postfix of *CONTEXT*'s synopsis.
Source [context.lisp], page 24, (file)
- `postfix` (*SYNOPSIS* synopsis) [Method]
 A postfix to the program synopsis.
Source [synopsis.lisp], page 20, (file)
- `potential-pack` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `potential-pack` (*SYNOPSIS* synopsis) [Method]
 The potential pack string.
Source [synopsis.lisp], page 20, (file)
- `potential-pack-p` *PACK THERE* [Generic Function]
 Return t if *PACK* is a potential pack in *THERE*.
Package [net.didierverna.clon], page 27,
Source [synopsis.lisp], page 20, (file)
Methods
- `potential-pack-p` *PACK* (*CONTEXT* context) [Method]
 Return t if *PACK* (a string) is a potential pack in *CONTEXT*.
Source [context.lisp], page 24, (file)
- `potential-pack-p` *PACK* (*SYNOPSIS* synopsis) [Method]
 Return t if *PACK* is a potential pack for *SYNOPSIS*.
- `print-help-spec` *SHEET HELP-SPEC* [Generic Function]
 Print *HELP-SPEC* on *SHEET*.
Package [net.didierverna.clon], page 27,
Source [sheet.lisp], page 21, (file)
Methods
- `print-help-spec` *SHEET HELP-SPEC* before [Method]
- `print-help-spec` *SHEET* (*CHAR* character) [Method]
 Print *CHAR* on *SHEET* with the current face.
- `print-help-spec` *SHEET* (*CHAR-VECTOR* simple-vector) [Method]
 Print *CHAR-VECTOR* on *SHEET* with the current face.

- `print-help-spec SHEET (STRING string)` [Method]
Print STRING on SHEET with the current face.
- `print-help-spec SHEET (HELP-SPEC list)` [Method]
Open HELP-SPEC's face and print all of its items with it.
- `retrieve-from-environment OPTION ENV-VAL` [Generic Function]
Retrieve OPTION's value from the environment.
ENV-VAL is the value stored in the associated environment variable.
- Package** [net.didierverna.clon], page 27,
Source [environ.lisp], page 19, (file)
Methods
- `retrieve-from-environment OPTION ENV-VAL before` [Method]
Assert that ENV-VAL is not null.
- `retrieve-from-environment (FLAG flag) ENV-VAL` [Method]
- `retrieve-from-environment (OPTION valued-option) ENV-VAL` [Method]
- `retrieve-from-long-call OPTION CMDLINE-NAME` [Generic Function]
&optional *CMDLINE-ARGUMENT* *CMDLINE*
Retrieve OPTION's value from a long call.
CMDLINE-NAME is the name used on the command-line.
CMDLINE-ARGUMENT is a potentially already parsed cmdline argument. Otherwise, CMDLINE is where to find an argument.
This function returns three values:
- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).
- Package** [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
Methods
- `retrieve-from-long-call (OPTION option) CMDLINE-NAME &optional CMDLINE-ARGUMENT CMDLINE` [Method]
- `retrieve-from-long-call (OPTION valued-option) CMDLINE-NAME &optional CMDLINE-ARGUMENT CMDLINE` [Method]
- `retrieve-from-negated-call OPTION` [Generic Function]
Retrieve OPTION's value from a negated call.
- Package** [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
Methods
- `retrieve-from-negated-call (OPTION option)` [Method]
`retrieve-from-negated-call (OPTION valued-option)` [Method]
`retrieve-from-negated-call (NEGATABLE negatable)` [Method]

retrieve-from-short-call *OPTION* &optional [Generic Function]
CMDLINE-ARGUMENT CMDLINE

Retrieve *OPTION*'s value from a short call.

CMDLINE-ARGUMENT is a potentially already parsed cmdline argument. Otherwise, *CMDLINE* is where to find an argument.

This function returns three values:

- the retrieved value,
- the value source,
- the new command-line (possibly with the first item popped if the option requires an argument).

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

Methods

retrieve-from-short-call (*OPTION* option) &optional [Method]
CMDLINE-ARGUMENT CMDLINE

retrieve-from-short-call (*OPTION* valued-option) [Method]
 &optional *CMDLINE-ARGUMENT CMDLINE*

right-padding *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

right-padding (*FACE* face) [Method]

The face right padding.

This property can take the following forms:

- <NUMBER>: the padding is relative to the enclosing face,
- SELF: the padding is set to wherever the face happens to be closed, - (<NUMBER> ABSOLUTE): the padding is set in absolute value,
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set relatively to a parent face named FACE-NAME.

Source [face.lisp], page 20, (file)

search-path *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

search-path (*CONTEXT* context) [Method]

The search path for Clon files.

Source [context.lisp], page 24, (file)

sface-tree *OBJECT* [Generic Function]

Package [net.didierverna.clon], page 27,

Methods

sface-tree (*SHEET* sheet) [Method]

The sheet's sface tree.

Source [sheet.lisp], page 21, (file)

- short-call** *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- short-call** (*CONDITION* unrecognized-short-call-error) [Method]
Source [context.lisp], page 24, (file)
- short-name** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- short-name** (*OPTION* option) [Method]
The option's short name.
Source [option.lisp], page 13, (file)
- short-pack** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- short-pack** (*CONTEXT* context) [Method]
Return the short pack of *CONTEXT*'s synopsis.
Source [context.lisp], page 24, (file)
- short-pack** (*SYNOPSIS* synopsis) [Method]
The short pack string.
Source [synopsis.lisp], page 20, (file)
- short-pack-char** *OPTION* &optional *AS-STRING* [Generic Function]
Return *OPTION*'s short pack character, if any. If *AS-STRING*, return a string of that character.
Package [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
Methods
- short-pack-char** (*OPTION* valued-option) &optional *AS-STRING* [Method]
Return *OPTION*'s short pack character if *OPTION*'s argument is optional.
Source [valued.lisp], page 14, (file)
- short-pack-char** (*OPTION* option) &optional *AS-STRING* [Method]
Return *OPTION*'s potential pack character.
- short-syntax-help-spec-prefix** *OPTION* [Generic Function]
Return the help specification prefix for *OPTION*'s short call.
Package [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)
Methods

- `short-syntax-help-spec-prefix` (*OPTION* negatable) [Method]
Source [negatable.lisp], page 15, (file)
- `short-syntax-help-spec-prefix` (*OPTION* valued-option) [Method]
- `sibling` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `sibling` (*SFACE* sface) [Method]
 The SFace's raw sibling.
Source [sheet.lisp], page 21, (file)
- `stream-ioctl-output-handle` *STREAM* [Generic Function]
 Return STREAM's ioctl output handle or NIL.
Package [net.didierverna.clon], page 27,
Source [termio.lisp], page 11, (file)
Methods
- `stream-ioctl-output-handle` (*STREAM* synonym-stream) [Method]
`stream-ioctl-output-handle` (*STREAM* two-way-stream) [Method]
`stream-ioctl-output-handle` (*STREAM* fd-stream) [Method]
`stream-ioctl-output-handle` *STREAM* [Method]
- `stringify` *VALUED-OPTION* *VALUE* [Generic Function]
 Transform VALUED-OPTION's VALUE into an argument. This is the opposite of argument conversion.
Package [net.didierverna.clon], page 27,
Source [valued.lisp], page 14, (file)
Methods
- `stringify` (*XSWITCH* xswitch) *VALUE* [Method]
 Transform XSWITCH's VALUE into an argument.
Source [xswitch.lisp], page 17, (file)
- `stringify` (*ENUM* enum) *VALUE* [Method]
 Transform ENUM's VALUE into an argument.
Source [enum.lisp], page 17, (file)
- `stringify` (*PATH* path) *VALUE* [Method]
 Transform PATH's VALUE into an argument.
Source [path.lisp], page 16, (file)
- `stringify` (*LISPOBJ* lispobj) *VALUE* [Method]
 Transform LISPOBJ's VALUE into an argument.
Source [lispobj.lisp], page 16, (file)

- stringify** (*STROPT* stropt) *VALUE* [Method]
 Transform STROPT's VALUE into an argument.
Source [stropt.lisp], page 16, (file)
- stringify** (*SWITCH* switch) *VALUE* [Method]
 Transform SWITCH's VALUE into an argument.
Source [switch.lisp], page 15, (file)
- subface** *FACE* *name(s)* [Generic Function]
 Return subface of FACE named NAME(S) or nil.
 If a list of names is provided instead of a single one, follow a subface branch matching those names to find the leaf face.
Package [net.didierverna.clon], page 27,
Source [face.lisp], page 20, (file)
Methods
- subface** *FACE* (*NAME* symbol) [Method]
 Return FACE's subface named NAME, or nil.
- subface** *FACE* (*NAMES* list) &aux *BRANCH* [Method]
 Return the leaf face from FACE's subbranch matching NAMES, or nil.
- subfaces** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- subfaces** (*FACE* face) [Method]
 The face children.
Source [face.lisp], page 20, (file)
- synopsis** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- synopsis** (*CONTEXT* context) [Method]
 The program synopsis.
Source [context.lisp], page 24, (file)
- theme** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- theme** (*CONTEXT* context) [Method]
 The theme filename.
Source [context.lisp], page 24, (file)
- top-padding** *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods

- `top-padding` (*HELP-SPEC* list) [Method]
Source [sheet.lisp], page 21, (file)
- `top-padding` *OTHER* [Method]
Source [sheet.lisp], page 21, (file)
- `top-padding` (*FACE* face) [Method]
 The face top padding.
 This property can take the following forms:
 - nil: the output can start right away,
 - 0: the output should start on the next line,
 - N>0: there should be N empty lines before the output.
Source [face.lisp], page 20, (file)
- `traversedp` *OBJECT* [Generic Function]
 (setf `traversedp`) *NEW-VALUE OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `traversedp` (*ITEM* item) [Method]
 (setf `traversedp`) *NEW-VALUE (ITEM* item) [Method]
 The item's traversal state.
Source [item.lisp], page 13, (file)
- `typespec` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `typespec` (*LISPOBJ* lispobj) [Method]
 A type specifier the option's value should satisfy.
Source [lispobj.lisp], page 16, (file)
- `underline` *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- `underline` (*FACE* face) [Method]
 The face's underline level.
Source [face.lisp], page 20, (file)
- `untraverse` *ITEM* [Generic Function]
 Reset *ITEM*'s traversal state, and return *ITEM*.
Package [net.didierverna.clon], page 27,
Source [item.lisp], page 13, (file)
Methods
- `untraverse` (*CONTEXT* context) [Method]
 Untraverse *CONTEXT* synopsis.
Source [context.lisp], page 24, (file)

- `untraverse` (*CONTAINER* container) [Method]
 Untraverse all *CONTAINER* items.
Source [container.lisp], page 18, (file)
- `untraverse` (*OPTION* option) [Method]
OPTION is a terminal object: just return it.
Source [option.lisp], page 13, (file)
- `untraverse` (*TEXT* text) [Method]
TEXT is a terminal object: just return it.
Source [text.lisp], page 13, (file)
- `untraverse` (*ITEM* item) after [Method]
 Mark *ITEM* as untraversed.
- value *CONDITION* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- value (*CONDITION* invalid-value) [Method]
Source [valued.lisp], page 14, (file)
- visiblep *OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- visiblep (*FACE* face) [Method]
 Whether the face is visible.
Source [face.lisp], page 20, (file)
- yes-values *OBJECT* [Generic Function]
 (setf yes-values) *NEW-VALUE OBJECT* [Generic Function]
Package [net.didierverna.clon], page 27,
Methods
- yes-values (*SWITCH-BASE* switch-base) [Method]
 (setf yes-values) *NEW-VALUE (SWITCH-BASE*
switch-base) [Method]
 The possible 'yes' values.
Source [switch-base.lisp], page 15, (file)

6.2.6 Conditions

- cmdline-error () [Condition]
 An error related to a command-line item.
Package [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
Direct superclasses
 error (condition)

Direct subclasses

- [cmdline-option-error], page 92, (condition)
- [invalid-short-equal-syntax], page 95, (condition)
- [invalid-negated-equal-syntax], page 95, (condition)
- [cmdline-junk-error], page 92, (condition)
- [unrecognized-short-call-error], page 97, (condition)
- [unrecognized-negated-call-error], page 97, (condition)
- [unknown-cmdline-option-error], page 97, (condition)

Direct methods

[item], page 79, (method)

Direct slots

item [Slot]
 The concerned command-line item.

Initargs :item

Readers [item], page 79, (generic function)

cmdline-junk-error () [Condition]

An error related to a command-line piece of junk.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

Direct superclasses

[cmdline-error], page 91, (condition)

Direct methods

[junk], page 80, (method)

Direct slots

item [Slot]
 The piece of junk appearing on the command-line.

Initargs :junk, :item

Readers [junk], page 80, (generic function)

cmdline-option-error () [Condition]

An error related to a command-line (known) option.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

Direct superclasses

- [option-error], page 96, (condition)
- [cmdline-error], page 91, (condition)

Direct subclasses

- [spurious-cmdline-argument], page 96, (condition)
- [invalid-negated-syntax], page 95, (condition)
- [invalid-cmdline-argument], page 94, (condition)
- [missing-cmdline-argument], page 96, (condition)

- Direct methods**
 [name], page 81, (method)
- Direct slots**
 item [Slot]
 The option's name as it appears on the command-line.
 Initargs :name, :item
 Readers [name], page 81, (generic function)
- environment-error () [Condition]
 An error related to an environment variable.
 Package [net.didierverna.clon], page 27,
 Source [environ.lisp], page 19, (file)
 Direct superclasses
 error (condition)
 Direct subclasses
 [environmental-option-error], page 93, (condition)
 Direct methods
 [env-var], page 76, (method)
 Direct slots
 env-var [Slot]
 The concerned environment variable.
 Initargs :env-var
 Readers [env-var], page 76, (generic function)
- environmental-option-error () [Condition]
 An error related to an option's environment variable.
 Package [net.didierverna.clon], page 27,
 Source [environ.lisp], page 19, (file)
 Direct superclasses
 • [option-error], page 96, (condition)
 • [environment-error], page 93, (condition)
 Direct subclasses
 [invalid-environment-value], page 94, (condition)
- home-directory () [Condition]
 Package [net.didierverna.clon], page 27,
 Source [util.lisp], page 12, (file)
 Direct superclasses
 warning (condition)
 Direct methods
 • error-string (method)
 • [error-string], page 76, (method)

Direct slots

error-string [Slot]

Initargs :error-string

Readers [error-string], page 76, (generic function)

Writers [(setf error-string)], page 76, (generic function)

invalid-argument () [Condition]

An invalid argument error.

Package [net.didierverna.clon], page 27,

Source [valued.lisp], page 14, (file)

Direct superclasses

[option-error], page 96, (condition)

Direct subclasses

- [invalid-cmdline-argument], page 94, (condition)
- [invalid-environment-value], page 94, (condition)

Direct methods

- [comment], page 74, (method)
- [argument], page 70, (method)

Direct slots

argument [Slot]

The invalid argument.

Initargs :argument

Readers [argument], page 70, (generic function)

comment [Slot]

An additional comment about the error.

Initargs :comment

Readers [comment], page 74, (generic function)

invalid-cmdline-argument () [Condition]

An invalid command-line argument error.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

Direct superclasses

- [invalid-argument], page 94, (condition)
- [cmdline-option-error], page 92, (condition)

invalid-environment-value () [Condition]

An invalid environment variable's value error.

Package [net.didierverna.clon], page 27,

Source [environ.lisp], page 19, (file)

Direct superclasses

- [invalid-argument], page 94, (condition)

- [environmental-option-error], page 93, (condition)

Direct methods

[env-val], page 76, (method)

Direct slots

argument [Slot]

The invalid environment variable value.

Initargs :env-val, :argument

Readers [env-val], page 76, (generic function)

invalid-negated-equal-syntax () [Condition]

An error related to a negated-equal syntax.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

Direct superclasses

[cmdline-error], page 91, (condition)

invalid-negated-syntax () [Condition]

An invalid negated syntax error.

Package [net.didierverna.clon], page 27,

Source [cmdline.lisp], page 18, (file)

Direct superclasses

[cmdline-option-error], page 92, (condition)

invalid-short-equal-syntax () [Condition]

An error related to a short-equal syntax.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

Direct superclasses

[cmdline-error], page 91, (condition)

invalid-value () [Condition]

An invalid value error.

Package [net.didierverna.clon], page 27,

Source [valued.lisp], page 14, (file)

Direct superclasses

[option-error], page 96, (condition)

Direct methods

- [comment], page 74, (method)

- [value], page 91, (method)

Direct slots

value [Slot]

The invalid value.

Initargs :value

Readers [value], page 91, (generic function)

- comment** [Slot]
 An additional comment about the error.
- Initargs** :comment
Readers [comment], page 74, (generic function)
- missing-cmdline-argument** () [Condition]
 A missing command-line argument error.
- Package** [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
- Direct superclasses**
 [cmdline-option-error], page 92, (condition)
- option-error** () [Condition]
 An error related to an option.
- Package** [net.didierverna.clon], page 27,
Source [option.lisp], page 13, (file)
- Direct superclasses**
 error (condition)
- Direct subclasses**
- [invalid-value], page 95, (condition)
 - [invalid-argument], page 94, (condition)
 - [cmdline-option-error], page 92, (condition)
 - [environmental-option-error], page 93, (condition)
- Direct methods**
 [option], page 83, (method)
- Direct slots**
- option** [Slot]
 The concerned option.
- Initargs** :option
Readers [option], page 83, (generic function)
- spurious-cmdline-argument** () [Condition]
 A spurious command-line argument error.
- Package** [net.didierverna.clon], page 27,
Source [cmdline.lisp], page 18, (file)
- Direct superclasses**
 [cmdline-option-error], page 92, (condition)
- Direct methods**
 [argument], page 70, (method)
- Direct slots**
- argument** [Slot]
 The spurious argument.
- Initargs** :argument
Readers [argument], page 70, (generic function)

- unknown-cmdline-option-error ()** [Condition]
 An error related to an unknown command-line option.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- Direct superclasses**
 [cmdline-error], page 91, (condition)
- Direct methods**
- [argument], page 70, (method)
 - [name], page 81, (method)
- Direct slots**
- item** [Slot]
 The option's name as it appears on the command-line.
- Initargs** :name, :item
Readers [name], page 81, (generic function)
- argument** [Slot]
 The option's command-line argument.
- Initargs** :argument
Readers [argument], page 70, (generic function)
- unrecognized-negated-call-error ()** [Condition]
 An error related to an unrecognized negated call.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- Direct superclasses**
 [cmdline-error], page 91, (condition)
- Direct methods**
 [negated-call], page 81, (method)
- Direct slots**
- item** [Slot]
 The unrecognized negated call on the command-line.
- Initargs** :negated-call, :item
Readers [negated-call], page 81, (generic function)
- unrecognized-short-call-error ()** [Condition]
 An error related to an unrecognized short call.
- Package** [net.didierverna.clon], page 27,
Source [context.lisp], page 24, (file)
- Direct superclasses**
 [cmdline-error], page 91, (condition)
- Direct methods**
 [short-call], page 87, (method)

Direct slots

item		[Slot]
	The unrecognized short call on the command-line.	
Initargs	:short-call, :item	
Readers	[short-call], page 87, (generic function)	

6.2.7 Structures

cmdline-option () [Structure]

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

Direct superclasses

structure-object (structure)

Direct slots

name		[Slot]
Readers	[cmdline-option-name], page 54, (function)	
Writers	[(setf cmdline-option-name)], page 54, (function)	
option		[Slot]
Readers	[cmdline-option-option], page 54, (function)	
Writers	[(setf cmdline-option-option)], page 54, (function)	
value		[Slot]
Readers	[cmdline-option-value], page 54, (function)	
Writers	[(setf cmdline-option-value)], page 54, (function)	
source		[Slot]
Readers	[cmdline-option-source], page 54, (function)	
Writers	[(setf cmdline-option-source)], page 54, (function)	

frame () [Structure]

The FRAME structure.

This structure hold layout properties used for printing.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

Direct superclasses

structure-object (structure)

Direct subclasses

[highlight-frame], page 99, (structure)

Direct methods

- [close-frame], page 73, (method)
- [open-frame], page 83, (method)

Direct slots

sface		[Slot]
Readers	[frame-sface], page 56, (function)	
Writers	[(setf frame-sface)], page 56, (function)	

<code>left-margin</code>	[Slot]
Readers	[<code>frame-left-margin</code>], page 56, (function)
Writers	[(<code>setf frame-left-margin</code>)], page 56, (function)
<code>right-margin</code>	[Slot]
Readers	[<code>frame-right-margin</code>], page 56, (function)
Writers	[(<code>setf frame-right-margin</code>)], page 56, (function)
<code>highlight-frame ()</code>	[Structure]
The HIGHLIGHT-FRAME structure.	
This structure holds both layout and highlight properties used for printing.	
Package	[<code>net.didierverna.clon</code>], page 27,
Source	[<code>sheet.lisp</code>], page 21, (file)
Direct superclasses	[<code>frame</code>], page 98, (structure)
Direct methods	<ul style="list-style-type: none"> • [<code>close-frame</code>], page 73, (method) • [<code>open-frame</code>], page 83, (method)
Direct slots	
<code>highlight-property-instances</code>	[Slot]
Readers	[<code>highlight-frame-highlight-property-instances</code>], page 57, (function)
Writers	[(<code>setf highlight-frame-highlight-property-instances</code>)], page 57, (function)
<code>highlight-property-instance ()</code>	[Structure]
The HIGHLIGHT-PROPERTY-INSTANCE structure.	
Package	[<code>net.didierverna.clon</code>], page 27,
Source	[<code>sheet.lisp</code>], page 21, (file)
Direct superclasses	<code>structure-object</code> (structure)
Direct slots	
<code>name</code>	[Slot]
Readers	[<code>highlight-property-instance-name</code>], page 57, (function)
Writers	[(<code>setf highlight-property-instance-name</code>)], page 57, (function)
<code>value</code>	[Slot]
Readers	[<code>highlight-property-instance-value</code>], page 58, (function)
Writers	[(<code>setf highlight-property-instance-value</code>)], page 58, (function)

6.2.8 Classes

abstract-class () [Class]

The ABSTRACT-CLASS class.
This is the meta-class for abstract classes.

Package [net.didierverna.clon], page 27,

Source [util.lisp], page 12, (file)

Direct superclasses
standard-class (class)

Direct methods

- validate-superclass (method)
- validate-superclass (method)
- make-instance (method)

container () [Class]

The CONTAINER class.
This class is a mixin used in synopsis and groups to represent the program's command-line hierarchy.

Package [net.didierverna.clon], page 27,

Source [container.lisp], page 18, (file)

Direct superclasses
[item], page 107, (class)

Direct subclasses

- [group], page 107, (class)
- [synopsis], page 114, (class)

Direct methods

- [mapoptions], page 81, (method)
- initialize-instance (method)
- initialize-instance (method)
- [help-spec], page 78, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [check-name-clash], page 72, (method)
- [untraverse], page 91, (method)
- [items], page 80, (method)

Direct slots

items [Slot]

The items in the container.

Type list

Initargs :items

Readers [items], page 79, (generic function)

`context ()` [Class]

The CONTEXT class.

This class represents the association of a synopsis and a set of command-line options based on it.

Package [net.didierverna.clon], page 27,

Source [context.lisp], page 24, (file)

Direct superclasses

standard-object (class)

Direct methods

- initialize-instance (method)
- [untraverse], page 90, (method)
- [mapoptions], page 81, (method)
- [potential-pack-p], page 84, (method)
- [clon-options-group], page 73, (method)
- [negated-pack], page 82, (method)
- [short-pack], page 87, (method)
- [postfix], page 84, (method)
- [error-handler], page 76, (method)
- [highlight], page 78, (method)
- [line-width], page 80, (method)
- [theme], page 89, (method)
- [search-path], page 86, (method)
- cmdline-options (method)
- [cmdline-options], page 73, (method)
- [synopsis], page 89, (method)

Direct slots

`synopsis` [Slot]

The program synopsis.

Type net.didierverna.clon::synopsis

Initargs :synopsis

Initform net.didierverna.clon:*synopsis*

Readers [synopsis], page 89, (generic function)

`progrname` [Slot]

The program name as it appears on the command-line.

Type string

`cmdline-options` [Slot]

The options from the command-line.

Type list

Readers [cmdline-options], page 73, (generic function)

Writers [(setf cmdline-options)], page 73, (generic function)

remainder		[Slot]
	The non-Clon part of the command-line.	
Type	<code>list</code>	
search-path		[Slot]
	The search path for Clon files.	
Readers	[<code>search-path</code>], page 86, (generic function)	
theme		[Slot]
	The theme filename.	
Readers	[<code>theme</code>], page 89, (generic function)	
line-width		[Slot]
	The line width for help display.	
Readers	[<code>line-width</code>], page 80, (generic function)	
highlight		[Slot]
	Clon's output highlight mode.	
Readers	[<code>highlight</code>], page 78, (generic function)	
error-handler		[Slot]
	The behavior to adopt on option retrieval errors.	
Type	<code>symbol</code>	
Initform	<code>:quit</code>	
Readers	[<code>error-handler</code>], page 76, (generic function)	

Direct Default Initargs

Initarg	Value
<code>:cmdline</code>	<code>(net.didierverna.clon:cmdline)</code>

enum () [Class]

The ENUM class.

This class implements options whose values belong to a set of keywords.

Package [`net.didierverna.clon`], page 27,

Source [`enum.lisp`], page 17, (file)

Direct superclasses

- [`valued-option`], page 115, (class)
- [`enum-base`], page 103, (class)

Direct methods

- [`convert`], page 74, (method)
- [`check`], page 72, (method)
- [`stringify`], page 88, (method)

Direct slots

argument-name		[Slot]
Initform	<code>"type"</code>	

- enum-base ()** [Class]
- The ENUM-BASE abstract class.
This class provides support for options including enumerated values.
- Package** [net.didierverna.clon], page 27,
- Source** [enum-base.lisp], page 17, (file)
- Direct superclasses**
standard-object (class)
- Direct subclasses**
- [enum], page 102, (class)
 - [xswitch], page 116, (class)
- Direct methods**
- initialize-instance (method)
 - [enum], page 75, (method)
- Direct slots**
- enum [Slot]
The set of possible values.
- Initargs** :enum
- Readers** [enum], page 75, (generic function)
- face ()** [Class]
- The FACE class.
- Package** [net.didierverna.clon], page 27,
- Source** [face.lisp], page 20, (file)
- Direct superclasses**
standard-object (class)
- Direct subclasses**
[sface], page 110, (class)
- Direct methods**
- initialize-instance (method)
 - initialize-instance (method)
 - initialize-instance (method)
 - slot-unbound (method)
 - [parent], page 83, (method)
 - [subfaces], page 89, (method)
 - [background], page 71, (method)
 - [foreground], page 76, (method)
 - [framedp], page 77, (method)
 - [crossed-out-p], page 75, (method)
 - [concealedp], page 74, (method)
 - [inversep], page 79, (method)
 - [blink], page 71, (method)
 - [underline], page 90, (method)

- [italicp], page 79, (method)
- [intensity], page 79, (method)
- [item-separator], page 79, (method)
- [bottom-padding], page 71, (method)
- [top-padding], page 90, (method)
- [right-padding], page 86, (method)
- [left-padding], page 80, (method)
- [visiblep], page 91, (method)
- [name], page 81, (method)

Direct slots

name	[Slot]
The face name.	
Initargs	:name
Readers	[name], page 81, (generic function)
visiblep	[Slot]
Whether the face is visible.	
Initargs	:visible
Initform	t
Readers	[visiblep], page 91, (generic function)
left-padding	[Slot]
The face left padding.	
This property can take the following forms:	
- <NUMBER>: the padding is relative to the enclosing face,	
- SELF: the padding is set to wherever the face happens to be opened, -	
(<NUMBER> ABSOLUTE): the padding is set in absolute value,	
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set rela-	
tively to a parent face named FACE-NAME.	
Initargs	:padding-left
Initform	0
Readers	[left-padding], page 80, (generic function)
right-padding	[Slot]
The face right padding.	
This property can take the following forms:	
- <NUMBER>: the padding is relative to the enclosing face,	
- SELF: the padding is set to wherever the face happens to be closed, - (<NUM-	
BER> ABSOLUTE): the padding is set in absolute value,	
- (<NUMBER> :RELATIVE-TO <FACE-NAME>): the padding is set rela-	
tively to a parent face named FACE-NAME.	
Initargs	:padding-right
Initform	(quote net.didierverna.clon::self)
Readers	[right-padding], page 86, (generic function)

- top-padding** [Slot]
The face top padding.
This property can take the following forms:
- nil: the output can start right away,
- 0: the output should start on the next line,
- N>0: there should be N empty lines before the output.
Initargs :padding-top
Readers [top-padding], page 89, (generic function)
- bottom-padding** [Slot]
The face bottom padding.
This property can take the following forms:
- nil: the next output can start right at the end of this face's, - 0: the next output should start on the next line,
- N>0: there should be N empty lines before the next output.
Initargs :padding-bottom
Readers [bottom-padding], page 71, (generic function)
- item-separator** [Slot]
The face item separator.
Initargs :item-separator
Initform #\
Readers [item-separator], page 79, (generic function)
- intensity** [Slot]
The face intensity.
Initargs :intensity
Readers [intensity], page 79, (generic function)
- italicp** [Slot]
The face's italic status.
Initargs :italic
Readers [italicp], page 79, (generic function)
- underline** [Slot]
The face's underline level.
Initargs :underline
Readers [underline], page 90, (generic function)
- blink** [Slot]
The face's blink speed.
Initargs :blink
Readers [blink], page 71, (generic function)
- inversep** [Slot]
The face's inverse video status.
Initargs :inverse
Readers [inversep], page 79, (generic function)

<code>concealedp</code>	[Slot]
The face's concealed status.	
Initargs	<code>:concealed</code>
Readers	<code>[concealedp]</code> , page 74, (generic function)
<code>crossed-out-p</code>	[Slot]
The face's crossed out status.	
Initargs	<code>:crossed-out</code>
Readers	<code>[crossed-out-p]</code> , page 75, (generic function)
<code>framedp</code>	[Slot]
The face's framed status.	
Initargs	<code>:framed</code>
Readers	<code>[framedp]</code> , page 77, (generic function)
<code>foreground</code>	[Slot]
The face foreground.	
Initargs	<code>:foreground</code>
Readers	<code>[foreground]</code> , page 76, (generic function)
<code>background</code>	[Slot]
The face background.	
Initargs	<code>:background</code>
Readers	<code>[background]</code> , page 71, (generic function)
<code>subfaces</code>	[Slot]
The face children.	
Initargs	<code>:subfaces</code>
Readers	<code>[subfaces]</code> , page 89, (generic function)
<code>parent</code>	[Slot]
The face parent.	
Readers	<code>[parent]</code> , page 83, (generic function)
<code>flag ()</code>	[Class]
The FLAG class.	
This class implements options that don't take any argument.	
Package	<code>[net.didierverna.clon]</code> , page 27,
Source	<code>[flag.lisp]</code> , page 14, (file)
Direct superclasses	<code>[option]</code> , page 108, (class)
Direct methods	<code>[retrieve-from-environment]</code> , page 85, (method)

group () [Class]

The GROUP class.

This class groups other groups, options or strings together, effectively implementing hierarchical program command-line.

Package [net.didierverna.clon], page 27,

Source [group.lisp], page 18, (file)

Direct superclasses

[container], page 100, (class)

Direct methods

- [help-spec], page 78, (method)
- [header], page 77, (method)

Direct slots

header [Slot]

The group's header.

Initargs :header

Readers [header], page 77, (generic function)

item () [Class]

The ITEM class.

This class is the base class for all synopsis items.

Package [net.didierverna.clon], page 27,

Source [item.lisp], page 13, (file)

Direct superclasses

standard-object (class)

Direct subclasses

- [text], page 115, (class)
- [option], page 108, (class)
- [container], page 100, (class)

Direct methods

- [mapoptions], page 81, (method)
- [help-spec], page 78, (method)
- [untraverse], page 91, (method)
- [hiddenp], page 78, (method)
- traversedp (method)
- [traversedp], page 90, (method)

Direct slots

traversedp [Slot]

The item's traversal state.

Readers [traversedp], page 90, (generic function)

Writers [(setf traversedp)], page 90, (generic function)

- hiddenp** [Slot]
Whether the item is hidden in help strings.
- Initargs** :hidden
- Readers** [hiddenp], page 78, (generic function)
- lispobj ()** [Class]
The LISPOBJ class.
This class implements read-from-string options.
- Package** [net.didierverna.clon], page 27,
- Source** [lispobj.lisp], page 16, (file)
- Direct superclasses**
[valued-option], page 115, (class)
- Direct methods**
- [convert], page 74, (method)
 - [check], page 72, (method)
 - [stringify], page 88, (method)
 - [typespec], page 90, (method)
- Direct slots**
- argument-name** [Slot]
Initform "obj"
- typespec** [Slot]
A type specifier the option's value should satisfy.
- Initargs** :typespec
- Initform** t
- Readers** [typespec], page 90, (generic function)
- negatable ()** [Class]
The NEGATABLE Class.
This class implements the negated syntax for the switch-based hierarchy.
- Package** [net.didierverna.clon], page 27,
- Source** [negatable.lisp], page 15, (file)
- Direct superclasses**
standard-object (class)
- Direct subclasses**
[switch-base], page 112, (class)
- Direct methods**
- [retrieve-from-negated-call], page 85, (method)
 - [negated-pack-char], page 82, (method)
 - [short-syntax-help-spec-prefix], page 88, (method)
- option ()** [Class]
The OPTION class.
This is the base class for all options.
- Package** [net.didierverna.clon], page 27,

Source [option.lisp], page 13, (file)

Direct superclasses

[item], page 107, (class)

Direct subclasses

- [flag], page 106, (class)
- [valued-option], page 115, (class)

Direct methods

- [mapoptions], page 81, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-short-call], page 86, (method)
- [retrieve-from-long-call], page 85, (method)
- initialize-instance (method)
- initialize-instance (method)
- [negated-pack-char], page 82, (method)
- [short-pack-char], page 87, (method)
- [option-sticky-distance], page 83, (method)
- [check-name-clash], page 73, (method)
- [help-spec], page 78, (method)
- [untraverse], page 91, (method)
- [env-var], page 76, (method)
- [description], page 75, (method)
- [long-name], page 80, (method)
- [short-name], page 87, (method)

Direct slots

short-name [Slot]
The option's short name.

Type (or null string)

Initargs :short-name

Readers [short-name], page 87, (generic function)

long-name [Slot]
The option's long name.

Type (or null string)

Initargs :long-name

Readers [long-name], page 80, (generic function)

description [Slot]
The option's description.

Type (or null string)

Initargs :description

Readers [description], page 75, (generic function)

env-var [Slot]
 The option's associated environment variable.

Type (or null string)

Initargs :env-var

Readers [env-var], page 76, (generic function)

Direct Default Initargs

Initarg	Value
:internal	nil

path () [Class]

The PATH class.

This class implements options whose values are (colon-separated lists of) pathnames.

Package [net.didierverna.clon], page 27,

Source [path.lisp], page 16, (file)

Direct superclasses

[valued-option], page 115, (class)

Direct methods

- [convert], page 74, (method)
- [check], page 72, (method)
- [stringify], page 88, (method)
- [path-type], page 84, (method)

Direct slots

argument-name [Slot]
Initform "path"

path-type [Slot]
 The path type.

Initargs :type
Readers [path-type], page 83, (generic function)

sface () [Class]

The SFACE class.

An SFace is the association of a face and its raw sibling. The sibling is used to create subfaces which would be missing from the original, user defined one.

Package [net.didierverna.clon], page 27,

Source [sheet.lisp], page 21, (file)

Direct superclasses

[face], page 103, (class)

Direct methods

[sibling], page 88, (method)

Direct slots

sibling [Slot]
 The SFace's raw sibling.

Readers [sibling], page 88, (generic function)

sheet ()	[Class]
The SHEET class.	
This class implements the notion of sheet for printing Clon help.	
Package	[net.didierverna.clon], page 27,
Source	[sheet.lisp], page 21, (file)
Direct superclasses	
	standard-object (class)
Direct methods	
	<ul style="list-style-type: none"> • initialize-instance (method) • initialize-instance (method) • frames (method) • [frames], page 77, (method) • column (method) • [column], page 73, (method) • [sface-tree], page 86, (method) • [highlightp], page 79, (method) • [line-width], page 80, (method) • [output-stream], page 83, (method)
Direct slots	
output-stream	[Slot]
The sheet's output stream.	
Type	stream
Initargs	:output-stream
Readers	[output-stream], page 83, (generic function)
line-width	[Slot]
The sheet's line width.	
Type	(integer 1)
Initargs	:line-width
Readers	[line-width], page 80, (generic function)
highlightp	[Slot]
Whether to highlight SHEET's output.	
Initargs	:highlightp
Readers	[highlightp], page 79, (generic function)
sface-tree	[Slot]
The sheet's sface tree.	
Readers	[sface-tree], page 86, (generic function)
column	[Slot]
The sheet's current column.	
Type	(integer 0)
Initform	0

- Readers** [column], page 73, (generic function)
- Writers** [(setf column)], page 73, (generic function)
- frames** [Slot]
The stack of currently open frames.
- Type** list
- Readers** [frames], page 77, (generic function)
- Writers** [(setf frames)], page 77, (generic function)
- stropt ()** [Class]
The STROPT class.
This class implements options the values of which are strings.
- Package** [net.didierverna.clon], page 27,
- Source** [stropt.lisp], page 16, (file)
- Direct superclasses**
[valued-option], page 115, (class)
- Direct methods**
- [convert], page 75, (method)
 - [check], page 72, (method)
 - [stringify], page 89, (method)
- Direct slots**
- argument-name** [Slot]
Initform "str"
- switch ()** [Class]
The SWITCH class.
This class implements boolean options.
- Package** [net.didierverna.clon], page 27,
- Source** [switch.lisp], page 15, (file)
- Direct superclasses**
- [valued-option], page 115, (class)
 - [switch-base], page 112, (class)
- Direct methods**
- initialize-instance (method)
 - [convert], page 75, (method)
 - [check], page 72, (method)
 - [stringify], page 89, (method)
- switch-base ()** [Class]
The SWITCH-BASE abstract class.
This class provides support for options including boolean values.
- Package** [net.didierverna.clon], page 27,
- Source** [switch-base.lisp], page 15, (file)
- Direct superclasses**
[negatable], page 108, (class)

Direct subclasses

- [switch], page 112, (class)
- [xswitch], page 116, (class)

Direct methods

- initialize-instance (method)
- initialize-instance (method)
- [argument-style], page 70, (method)
- no-values (method)
- [no-values], page 82, (method)
- yes-values (method)
- [yes-values], page 91, (method)
- argument-styles (method)
- [argument-styles], page 71, (method)

Direct slots

argument-styles [Slot]

The possible argument styles.

The position of every argument style in the list must correspond to the position of the associated strings in the yes-values and no-values slots.

Type list

Initargs :argument-styles

Readers [argument-styles], page 71, (generic function)

Writers [(setf argument-styles)], page 71, (generic function)

yes-values [Slot]

The possible 'yes' values.

Type list

Initargs :yes-values

Readers [yes-values], page 91, (generic function)

Writers [(setf yes-values)], page 91, (generic function)

no-values [Slot]

The possible 'no' values.

Type list

Initargs :no-values

Readers [no-values], page 82, (generic function)

Writers [(setf no-values)], page 82, (generic function)

argument-style [Slot]

The selected argument style.

Type keyword

Initargs :argument-style

Initform :yes/no

Readers [argument-style], page 70, (generic function)

Direct Default Initargs

Initarg	Value
:argument-type	:optional
:argument-styles	(quote (:yes/no :on/off :true/false :yup/nope :yeah/nah))
:yes-values	(quote ("yes" "on" "true" "yup" "yeah"))
:no-values	(quote ("no" "off" "false" "nope" "nah"))

synopsis () [Class]

The SYNOPSIS class.

This class handles the description of the program's command-line options.

Package [net.didierverna.clon], page 27,

Source [synopsis.lisp], page 20, (file)

Direct superclasses

[container], page 100, (class)

Direct methods

- initialize-instance (method)
- initialize-instance (method)
- [potential-pack-p], page 84, (method)
- [help-spec], page 77, (method)
- [clon-options-group], page 73, (method)
- [potential-pack], page 84, (method)
- [negated-pack], page 82, (method)
- [short-pack], page 87, (method)
- [postfix], page 84, (method)

Direct slots

postfix [Slot]

A postfix to the program synopsis.

Type (or null string)

Initargs :postfix

Readers [postfix], page 84, (generic function)

short-pack [Slot]

The short pack string.

Type (or null string)

Readers [short-pack], page 87, (generic function)

negated-pack [Slot]

The negated pack string.

Type (or null string)

Readers [negated-pack], page 82, (generic function)

<code>potential-pack</code>	[Slot]
The potential pack string.	
Type	(or null string)
Readers	[potential-pack], page 84, (generic function)
<code>clon-options-group</code>	[Slot]
The Clon options group.	
Type	net.didierverna.clon:group
Initargs	:clon-options-group
Readers	[clon-options-group], page 73, (generic function)
<code>text ()</code>	[Class]
The TEXT class.	
This class implements plain text objects appearing in a synopsis.	
Package	[net.didierverna.clon], page 27,
Source	[text.lisp], page 13, (file)
Direct superclasses	
	[item], page 107, (class)
Direct methods	
	• [check-name-clash], page 72, (method)
	• [check-name-clash], page 72, (method)
	• [help-spec], page 78, (method)
	• [untraverse], page 91, (method)
	• [contents], page 74, (method)
Direct slots	
<code>contents</code>	[Slot]
The actual text string.	
Type	string
Initargs	:contents
Readers	[contents], page 74, (generic function)
<code>valued-option ()</code>	[Class]
The VALUED-OPTION class.	
This is the base class for options accepting arguments.	
Package	[net.didierverna.clon], page 27,
Source	[valued.lisp], page 14, (file)
Direct superclasses	
	[option], page 108, (class)
Direct subclasses	
	• [switch], page 112, (class)
	• [stropt], page 112, (class)
	• [lispobj], page 108, (class)
	• [path], page 110, (class)

- [enum], page 102, (class)
- [xswitch], page 116, (class)

Direct methods

- [retrieve-from-environment], page 85, (method)
- [retrieve-from-negated-call], page 85, (method)
- [retrieve-from-short-call], page 86, (method)
- [retrieve-from-long-call], page 85, (method)
- initialize-instance (method)
- initialize-instance (method)
- [help-spec], page 78, (method)
- [short-syntax-help-spec-prefix], page 88, (method)
- [short-pack-char], page 87, (method)
- [option-sticky-distance], page 83, (method)
- [default-value], page 75, (method)
- [fallback-value], page 76, (method)
- [argument-required-p], page 70, (method)
- [argument-name], page 70, (method)

Direct slots

<code>argument-name</code>	[Slot]
The option's argument display name.	
Initargs	:argument-name
Initform	"arg"
Readers	[argument-name], page 70, (generic function)
<code>argument-required-p</code>	[Slot]
Whether the option's argument is required.	
Readers	[argument-required-p], page 70, (generic function)
<code>fallback-value</code>	[Slot]
The option's fallback value.	
Initargs	:fallback-value
Readers	[fallback-value], page 76, (generic function)
<code>default-value</code>	[Slot]
The option's default value.	
Initargs	:default-value
Readers	[default-value], page 75, (generic function)

Direct Default Initargs

Initarg	Value
:argument-type	:required

`xswitch ()` [Class]
 The XSWITCH class.
 This class merges the functionalities of switches and enumerations. As such, the negated syntax is available for extended xswitches.

Package [net.didierverna.clon], page 27,

Source [xswitch.lisp], page 17, (file)

Direct superclasses

- [valued-option], page 115, (class)
- [enum-base], page 103, (class)
- [switch-base], page 112, (class)

Direct methods

- [convert], page 74, (method)
- [check], page 72, (method)
- [stringify], page 88, (method)

Direct slots

enum

The set of possible non-boolean values.

[Slot]

Appendix A Indexes

A.1 Concepts

F

- File, Lisp, net.didierverna.clon.asd..... 11
- File, Lisp, net.didierverna.clon.core.asd..... 11
- File, Lisp,
 - net.didierverna.clon.core/package.lisp.... 12
- File, Lisp,
 - net.didierverna.clon.core/src/container.lisp..18
- File, Lisp,
 - net.didierverna.clon.core/src/context.lisp..24
- File, Lisp,
 - net.didierverna.clon.core/src/group.lisp.. 18
- File, Lisp,
 - net.didierverna.clon.core/src/item.lisp... 13
- File, Lisp,
 - net.didierverna.clon.core/src/options/enum-base.lisp..17
- File, Lisp,
 - net.didierverna.clon.core/src/options/enum.lisp..17
- File, Lisp,
 - net.didierverna.clon.core/src/options/flag.lisp..14
- File, Lisp,
 - net.didierverna.clon.core/src/options/lispobj.lisp..16
- File, Lisp,
 - net.didierverna.clon.core/src/options/negatable.lisp..15
- File, Lisp,
 - net.didierverna.clon.core/src/options/option.lisp..13
- File, Lisp,
 - net.didierverna.clon.core/src/options/path.lisp..16
- File, Lisp,
 - net.didierverna.clon.core/src/options/stropt.lisp..16
- File, Lisp,
 - net.didierverna.clon.core/src/options/switch-base.lisp..15
- File, Lisp,
 - net.didierverna.clon.core/src/options/switch.lisp..15
- File, Lisp,
 - net.didierverna.clon.core/src/options/valued.lisp..14
- File, Lisp,
 - net.didierverna.clon.core/src/options/xswitch.lisp..17
- File, Lisp,
 - net.didierverna.clon.core/src/output/face.lisp..20
- File, Lisp,
 - net.didierverna.clon.core/src/output/sheet.lisp..21
- File, Lisp,
 - net.didierverna.clon.core/src/retrieval/cmdline.lisp..18
- File, Lisp,
 - net.didierverna.clon.core/src/retrieval/enviro... 19
- File, Lisp,
 - net.didierverna.clon.core/src/synopsis.lisp..20
- File, Lisp,
 - net.didierverna.clon.core/src/text.lisp... 13
- File, Lisp,
 - net.didierverna.clon.core/src/util.lisp... 12
- File, Lisp, net.didierverna.clon.setup.asd..... 11
- File, Lisp,
 - net.didierverna.clon.setup/package.lisp... 25
- File, Lisp,
 - net.didierverna.clon.setup/src/configuration.lisp..26
- File, Lisp,
 - net.didierverna.clon.setup/src/readtable.lisp..26
- File, Lisp,
 - net.didierverna.clon.setup/src/termio.lisp..26
- File, Lisp,
 - net.didierverna.clon.setup/src/version.lisp..25
- File, Lisp, net.didierverna.clon.termio.asd... 11
- File, Lisp,
 - net.didierverna.clon.termio/sbcl/constants.lisp..11
- File, Lisp,
 - net.didierverna.clon.termio/termio.lisp... 11

L

- Lisp File, net.didierverna.clon.asd..... 11
- Lisp File, net.didierverna.clon.core.asd..... 11
- Lisp File,
 - net.didierverna.clon.core/package.lisp.... 12
- Lisp File,
 - net.didierverna.clon.core/src/container.lisp..18
- Lisp File,
 - net.didierverna.clon.core/src/context.lisp..24
- Lisp File,
 - net.didierverna.clon.core/src/group.lisp.. 18
- Lisp File,
 - net.didierverna.clon.core/src/item.lisp... 13
- Lisp File,
 - net.didierverna.clon.core/src/options/enum-base.lisp..17
- Lisp File,
 - net.didierverna.clon.core/src/options/enum.lisp..17
- Lisp File,
 - net.didierverna.clon.core/src/options/flag.lisp..14
- Lisp File,
 - net.didierverna.clon.core/src/options/lispobj.lisp..16
- Lisp File,
 - net.didierverna.clon.core/src/options/negatable.lisp..15
- Lisp File,
 - net.didierverna.clon.core/src/options/option.lisp..13
- Lisp File,
 - net.didierverna.clon.core/src/options/path.lisp..16
- Lisp File,
 - net.didierverna.clon.core/src/options/stropt.lisp..16
- Lisp File,
 - net.didierverna.clon.core/src/options/switch-base.lisp..15
- Lisp File,
 - net.didierverna.clon.core/src/options/switch.lisp..15
- Lisp File,
 - net.didierverna.clon.core/src/options/valued.lisp..14
- Lisp File,
 - net.didierverna.clon.core/src/options/xswitch.lisp..17
- Lisp File,
 - net.didierverna.clon.core/src/output/face.lisp..20
- Lisp File,
 - net.didierverna.clon.core/src/output/sheet.lisp..21
- Lisp File,
 - net.didierverna.clon.core/src/retrieval/cmdline.lisp..18
- Lisp File,
 - net.didierverna.clon.core/src/retrieval/enviro... 19

Lisp File,		N	
net.didierverna.clon.core/src/synopsis.lisp... 20		net.didierverna.clon.asd..... 11	
Lisp File,		net.didierverna.clon.core.asd..... 11	
net.didierverna.clon.core/src/text.lisp... 13		net.didierverna.clon.core/package.lisp..... 12	
Lisp File,		net.didierverna.clon.core/src..... 9	
net.didierverna.clon.core/src/util.lisp... 12		net.didierverna.clon.core/src/container.lisp..18	
Lisp File, net.didierverna.clon.setup.asd..... 11		net.didierverna.clon.core/src/context.lisp..24	
Lisp File,		net.didierverna.clon.core/src/group.lisp.... 18	
net.didierverna.clon.setup/package.lisp... 25		net.didierverna.clon.core/src/item.lisp.... 13	
Lisp File,		net.didierverna.clon.core/src/options..... 9	
net.didierverna.clon.setup/src/configuration.lisp..26		net.didierverna.clon.core/src/options/enum-base.lisp..17	
Lisp File,		net.didierverna.clon.core/src/options/enum.lisp..17	
net.didierverna.clon.setup/src/readtable.lisp..26		net.didierverna.clon.core/src/options/flag.lisp..14	
Lisp File,		net.didierverna.clon.core/src/options/lispobj.lisp..16	
net.didierverna.clon.setup/src/termio.lisp..26		net.didierverna.clon.core/src/options/negatable.lisp..15	
Lisp File,		net.didierverna.clon.core/src/options/option.lisp..13	
net.didierverna.clon.setup/src/version.lisp..25		net.didierverna.clon.core/src/options/path.lisp..16	
Lisp File, net.didierverna.clon.termio.asd.... 11		net.didierverna.clon.core/src/options/stropt.lisp..16	
Lisp File,		net.didierverna.clon.core/src/options/switch-base.lisp..15	
net.didierverna.clon.termio/sbcl/constants.lisp		net.didierverna.clon.core/src/options/switch.lisp..15	
Lisp File,		net.didierverna.clon.core/src/options/valued.lisp..14	
net.didierverna.clon.termio/termio.lisp... 11		net.didierverna.clon.core/src/options/xswitch.lisp..17	
		net.didierverna.clon.core/src/output..... 10	
		net.didierverna.clon.core/src/output/face.lisp..20	
		net.didierverna.clon.core/src/output/sheet.lisp..21	
		net.didierverna.clon.core/src/retrieval.... 10	
		net.didierverna.clon.core/src/retrieval/cmdline.lisp..18	
		net.didierverna.clon.core/src/retrieval/environ.lisp..19	
		net.didierverna.clon.core/src/synopsis.lisp..20	
		net.didierverna.clon.core/src/text.lisp.... 13	
		net.didierverna.clon.core/src/util.lisp.... 12	
		net.didierverna.clon.setup.asd..... 11	
		net.didierverna.clon.setup/package.lisp.... 25	
		net.didierverna.clon.setup/src..... 10	
		net.didierverna.clon.setup/src/configuration.lisp..26	
		net.didierverna.clon.setup/src/readtable.lisp..26	
		net.didierverna.clon.setup/src/termio.lisp..26	
		net.didierverna.clon.setup/src/version.lisp..25	
		net.didierverna.clon.termio.asd..... 11	
		net.didierverna.clon.termio/sbcl/constants.lisp..11	
		net.didierverna.clon.termio/termio.lisp.... 11	
M			
Module, net.didierverna.clon.core/src..... 9			
Module,			
net.didierverna.clon.core/src/options..... 9			
Module,			
net.didierverna.clon.core/src/output..... 10			
Module,			
net.didierverna.clon.core/src/retrieval... 10			
Module, net.didierverna.clon.setup/src..... 10			

A.2 Functions

%

%defgroup 50
%version 52

(

(setf argument-styles) 71
(setf cmdline-option-name) 54
(setf cmdline-option-option) 54
(setf cmdline-option-source) 54
(setf cmdline-option-value) 54
(setf cmdline-options) 73
(setf column) 73
(setf error-string) 76
(setf frame-left-margin) 56
(setf frame-right-margin) 56
(setf frame-sface) 56
(setf frames) 77
(setf highlight-frame-highlight-property-instances) 57
(setf highlight-frame-left-margin) 57
(setf highlight-frame-right-margin) 57
(setf highlight-frame-sface) 57
(setf highlight-property-instance-name) 57
(setf highlight-property-instance-value) 58
(setf no-values) 82
(setf traversedp) 90
(setf winsize-ws-col) 69
(setf winsize-ws-row) 69
(setf winsize-ws-xpixel) 69
(setf winsize-ws-ypixel) 70
(setf yes-values) 91

~

~-reader 70

A

accumulate 50
add-subface 52
allocate-winsize 53
argument 70
argument-name 70
argument-popable-p 53
argument-required-p 70
argument-style 70
argument-styles 71
attach-face-tree 53
available-right-margin 53

B

background 71
beginning-of-string-p 53
blink 71
bottom-padding 71

C

check 71, 72
check-name-clash 72, 73
clindent 53
clon-options-group 73
close-frame 73
close-line 53
close-sface 53
closest-match 54
cmdline 43
cmdline-convert 54
cmdline-option-name 54
cmdline-option-option 54
cmdline-option-p 54
cmdline-option-source 54
cmdline-option-value 54
cmdline-options 73
cmdline-options-p 43
cmdline-p 43
column 73
comment 74
complete-string 54
concealedp 74
configuration 43
configure 44
contents 74
convert 74, 75
copy-cmdline-option 55
copy-frame 55
copy-highlight-frame 55
copy-highlight-property-instance 55
copy-instance 75
crossed-out-p 75
current-frame 55
current-left-margin 55
current-right-margin 55
current-sface 55

D

declare-valid-superclass 50
defabstract 50
default-value 75
defgroup 42
defindent 51
defoption 51
defsynopsis 42
description 75
directory-pathname-p 55
do-cmdline-options 42
do-options 51
dump 42

E

econd	51
endpush	51
enum	75
env-val	76
env-var	76
environment-convert	55
error-handler	76
error-string	76
executablep	44
exit	44
exit-abnormally	56

F

face-highlight-property-set-p	56
face-highlight-property-value	56
fallback-value	76
find-sface	56
flush-sheet	56
foreground	76
frame-left-margin	56
frame-p	56
frame-right-margin	56
frame-sface	56
framedp	77
frames	77
Function, %version	52
Function, (setf cmdline-option-name)	54
Function, (setf cmdline-option-option)	54
Function, (setf cmdline-option-source)	54
Function, (setf cmdline-option-value)	54
Function, (setf frame-left-margin)	56
Function, (setf frame-right-margin)	56
Function, (setf frame-sface)	56
Function, (setf highlight-frame-highlight-property-instances)	57
Function, (setf highlight-frame-left-margin)	57
Function, (setf highlight-frame-right-margin)	57
Function, (setf highlight-frame-sface)	57
Function, (setf highlight-property-instance-name)	57
Function, (setf highlight-property-instance-value)	58
Function, (setf winsize-ws-col)	69
Function, (setf winsize-ws-row)	69
Function, (setf winsize-ws-xpixel)	69
Function, (setf winsize-ws-ypixel)	70
Function, ~-reader	70
Function, add-subface	52
Function, allocate-winsize	53
Function, argument-popable-p	53
Function, attach-face-tree	53
Function, available-right-margin	53
Function, beginning-of-string-p	53
Function, clindent	53
Function, close-line	53
Function, close-sface	53
Function, closest-match	54
Function, cmdline	43
Function, cmdline-convert	54
Function, cmdline-option-name	54

Function, cmdline-option-option	54
Function, cmdline-option-p	54
Function, cmdline-option-source	54
Function, cmdline-option-value	54
Function, cmdline-options-p	43
Function, cmdline-p	43
Function, complete-string	54
Function, configuration	43
Function, configure	44
Function, copy-cmdline-option	55
Function, copy-frame	55
Function, copy-highlight-frame	55
Function, copy-highlight-property-instance	55
Function, current-frame	55
Function, current-left-margin	55
Function, current-right-margin	55
Function, current-sface	55
Function, directory-pathname-p	55
Function, environment-convert	55
Function, executablep	44
Function, exit	44
Function, exit-abnormally	56
Function, face-highlight-property-set-p	56
Function, face-highlight-property-value	56
Function, find-sface	56
Function, flush-sheet	56
Function, frame-left-margin	56
Function, frame-p	56
Function, frame-right-margin	56
Function, frame-sface	56
Function, get-top-padding	57
Function, getenv	57
Function, getopt	44
Function, getopt-cmdline	44
Function, help	44
Function, help-spec-items-will-print	57
Function, highlight-frame-highlight-property-instances	57
Function, highlight-frame-left-margin	57
Function, highlight-frame-p	57
Function, highlight-frame-right-margin	57
Function, highlight-frame-sface	57
Function, highlight-property-instance-escape-sequence	57
Function, highlight-property-instance-name	57
Function, highlight-property-instance-p	58
Function, highlight-property-instance-value	58
Function, home-directory	58
Function, i-reader	58
Function, list-to-string	58
Function, macosp	58
Function, make-cmdline-option	58
Function, make-context	45
Function, make-enum	45
Function, make-flag	45
Function, make-frame	58
Function, make-group	46
Function, make-highlight-frame	58
Function, make-highlight-property-instance	59
Function, make-internal-enum	59
Function, make-internal-flag	59
Function, make-internal-lispobj	59
Function, make-internal-path	60
Function, make-internal-stropt	60
Function, make-internal-switch	61

- Function, `make-internal-text` 61
 - Function, `make-internal-xswitch` 61
 - Function, `make-lispobj` 46
 - Function, `make-path` 46
 - Function, `make-raw-face-tree` 61
 - Function, `make-raw-sface` 62
 - Function, `make-sheet` 62
 - Function, `make-stropt` 47
 - Function, `make-switch` 47
 - Function, `make-synopsis` 47
 - Function, `make-text` 48
 - Function, `make-xswitch` 48
 - Function, `match-option` 62
 - Function, `nickname-package` 48
 - Function, `open-line` 62
 - Function, `open-next-line` 62
 - Function, `open-sface` 62
 - Function, `option-abbreviation-distance` 62
 - Function, `option-call-p` 62
 - Function, `parent-generation` 63
 - Function, `pathname-component-null-p` 63
 - Function, `pop-frame` 63
 - Function, `potential-pack-char` 63
 - Function, `princ-char` 63
 - Function,
 - `princ-highlight-property-instances` 63
 - Function, `princ-spaces` 63
 - Function, `princ-string` 63
 - Function, `print-error` 64
 - Function, `print-faced-help-spec` 64
 - Function, `print-help` 64
 - Function, `print-string` 64
 - Function, `progname` 48
 - Function, `push-frame` 64
 - Function, `putenv` 64
 - Function, `reach-column` 64
 - Function, `read-argument` 64
 - Function, `read-call` 65
 - Function, `read-env-val` 65
 - Function, `read-long-name` 65
 - Function, `read-sface-tree` 65
 - Function, `read-value` 65
 - Function, `release-status-number` 65
 - Function, `remainder` 48
 - Function, `remove-keys` 65
 - Function, `replace-key` 65
 - Function, `replace-keys` 66
 - Function, `restart-on-error` 66
 - Function, `restartable-check` 66
 - Function, `restartable-cmdline-convert` 66
 - Function, `restartable-cmdline-junk-error` 66
 - Function, `restartable-convert` 67
 - Function, `restartable-environment-convert` 67
 - Function, `restrict-because` 67
 - Function, `safe-left-margin` 67
 - Function, `safe-right-margin` 67
 - Function, `search-branch` 67
 - Function, `search-face` 68
 - Function, `search-option` 68
 - Function, `search-option-by-abbreviation` 68
 - Function, `search-option-by-name` 68
 - Function, `search-sticky-option` 68
 - Function, `select-keys` 69
 - Function, `setup-termio` 48
 - Function, `split-path` 69
 - Function, `stream-line-width` 69
 - Function, `try-read-sface-tree` 69
 - Function, `try-read-theme` 69
 - Function, `version` 49
 - Function, `winsize-ws-col` 69
 - Function, `winsize-ws-row` 69
 - Function, `winsize-ws-xpixel` 69
 - Function, `winsize-ws-ypixel` 70
- ## G
- Generic Function, `(setf argument-styles)` 71
 - Generic Function, `(setf cmdline-options)` 73
 - Generic Function, `(setf column)` 73
 - Generic Function, `(setf error-string)` 76
 - Generic Function, `(setf frames)` 77
 - Generic Function, `(setf no-values)` 82
 - Generic Function, `(setf traversedp)` 90
 - Generic Function, `(setf yes-values)` 91
 - Generic Function, `argument` 70
 - Generic Function, `argument-name` 70
 - Generic Function, `argument-required-p` 70
 - Generic Function, `argument-style` 70
 - Generic Function, `argument-styles` 71
 - Generic Function, `background` 71
 - Generic Function, `blink` 71
 - Generic Function, `bottom-padding` 71
 - Generic Function, `check` 71
 - Generic Function, `check-name-clash` 72
 - Generic Function, `clon-options-group` 73
 - Generic Function, `close-frame` 73
 - Generic Function, `cmdline-options` 73
 - Generic Function, `column` 73
 - Generic Function, `comment` 74
 - Generic Function, `concealedp` 74
 - Generic Function, `contents` 74
 - Generic Function, `convert` 74
 - Generic Function, `copy-instance` 75
 - Generic Function, `crossed-out-p` 75
 - Generic Function, `default-value` 75
 - Generic Function, `description` 75
 - Generic Function, `enum` 75
 - Generic Function, `env-val` 76
 - Generic Function, `env-var` 76
 - Generic Function, `error-handler` 76
 - Generic Function, `error-string` 76
 - Generic Function, `fallback-value` 76
 - Generic Function, `foreground` 76
 - Generic Function, `framedp` 77
 - Generic Function, `frames` 77
 - Generic Function, `get-bottom-padding` 77
 - Generic Function, `header` 77
 - Generic Function, `help-spec` 77
 - Generic Function, `help-spec-will-print` 78
 - Generic Function, `hiddenp` 78
 - Generic Function, `highlight` 78
 - Generic Function, `highlightp` 79
 - Generic Function, `intensity` 79
 - Generic Function, `inversep` 79
 - Generic Function, `italicp` 79
 - Generic Function, `item` 79
 - Generic Function, `item-separator` 79
 - Generic Function, `items` 79
 - Generic Function, `junk` 80

Generic Function, <code>left-padding</code>	80
Generic Function, <code>line-width</code>	80
Generic Function, <code>long-name</code>	80
Generic Function, <code>make-face-tree</code>	80
Generic Function, <code>mapoptions</code>	81
Generic Function, <code>name</code>	81
Generic Function, <code>negated-call</code>	81
Generic Function, <code>negated-pack</code>	82
Generic Function, <code>negated-pack-char</code>	82
Generic Function, <code>no-values</code>	82
Generic Function, <code>open-frame</code>	82
Generic Function, <code>option</code>	83
Generic Function, <code>option-sticky-distance</code>	83
Generic Function, <code>output-stream</code>	83
Generic Function, <code>parent</code>	83
Generic Function, <code>path-type</code>	83
Generic Function, <code>postfix</code>	84
Generic Function, <code>potential-pack</code>	84
Generic Function, <code>potential-pack-p</code>	84
Generic Function, <code>print-help-spec</code>	84
Generic Function, <code>retrieve-from-environment</code> ...	85
Generic Function, <code>retrieve-from-long-call</code>	85
Generic Function, <code>retrieve-from-negated-call</code>	85
Generic Function, <code>retrieve-from-short-call</code>	86
Generic Function, <code>right-padding</code>	86
Generic Function, <code>search-path</code>	86
Generic Function, <code>sface-tree</code>	86
Generic Function, <code>short-call</code>	87
Generic Function, <code>short-name</code>	87
Generic Function, <code>short-pack</code>	87
Generic Function, <code>short-pack-char</code>	87
Generic Function, <code>short-syntax-help-spec-prefix</code>	87
Generic Function, <code>sibling</code>	88
Generic Function, <code>stream-ioctl-output-handle</code>	88
Generic Function, <code>stringify</code>	88
Generic Function, <code>subface</code>	89
Generic Function, <code>subfaces</code>	89
Generic Function, <code>synopsis</code>	89
Generic Function, <code>theme</code>	89
Generic Function, <code>top-padding</code>	89
Generic Function, <code>traversedp</code>	90
Generic Function, <code>typespec</code>	90
Generic Function, <code>underline</code>	90
Generic Function, <code>untraverse</code>	90
Generic Function, <code>value</code>	91
Generic Function, <code>visiblep</code>	91
Generic Function, <code>yes-values</code>	91
<code>get-bottom-padding</code>	77
<code>get-top-padding</code>	57
<code>getenv</code>	57
<code>getopt</code>	44
<code>getopt-cmdline</code>	44

H

<code>header</code>	77
<code>help</code>	44
<code>help-spec</code>	77, 78
<code>help-spec-items-will-print</code>	57
<code>help-spec-will-print</code>	78
<code>hiddenp</code>	78
<code>highlight</code>	78
<code>highlight-frame-highlight-</code> <code>property-instances</code>	57
<code>highlight-frame-left-margin</code>	57
<code>highlight-frame-p</code>	57
<code>highlight-frame-right-margin</code>	57
<code>highlight-frame-sface</code>	57
<code>highlight-property-ecase</code>	51
<code>highlight-property-instance-</code> <code>escape-sequence</code>	57
<code>highlight-property-instance-name</code>	57
<code>highlight-property-instance-p</code>	58
<code>highlight-property-instance-value</code>	58
<code>highlightp</code>	79
<code>home-directory</code>	58

I

<code>i-reader</code>	58
<code>intensity</code>	79
<code>inversep</code>	79
<code>italicp</code>	79
<code>item</code>	79
<code>item-separator</code>	79
<code>items</code>	79, 80

J

<code>junk</code>	80
-------------------------	----

L

<code>left-padding</code>	80
<code>line-width</code>	80
<code>list-to-string</code>	58
<code>long-name</code>	80

M

<code>macosp</code>	58
Macro, <code>%defgroup</code>	50
Macro, <code>accumulate</code>	50
Macro, <code>declare-valid-superclass</code>	50
Macro, <code>defabstract</code>	50
Macro, <code>defgroup</code>	42
Macro, <code>defindent</code>	51
Macro, <code>defoption</code>	51
Macro, <code>defsynopsis</code>	42
Macro, <code>do-cmdline-options</code>	42
Macro, <code>do-options</code>	51
Macro, <code>dump</code>	42
Macro, <code>econd</code>	51
Macro, <code>endpush</code>	51
Macro, <code>highlight-property-ecase</code>	51
Macro, <code>map-frames</code>	51
Macro, <code>maybe-pop-argument</code>	51

- Macro, maybe-push 52
- Macro, multiple-value-getopt-cmdline 43
- Macro, replace-in-keys 52
- Macro,
 - restartable-invalid-negated-syntax-error .. 52
- Macro, restartable-spurious-
cmdline-argument-error 52
- Macro, with-context 43
- Macro, with-context-error-handler 52
- Macro, with-winsize 52
- make-cmdline-option 58
- make-context 45
- make-enum 45
- make-face-tree 80, 81
- make-flag 45
- make-frame 58
- make-group 46
- make-highlight-frame 58
- make-highlight-property-instance 59
- make-internal-enum 59
- make-internal-flag 59
- make-internal-lispobj 59
- make-internal-path 60
- make-internal-stropt 60
- make-internal-switch 61
- make-internal-text 61
- make-internal-xswitch 61
- make-lispobj 46
- make-path 46
- make-raw-face-tree 61
- make-raw-sface 62
- make-sheet 62
- make-stropt 47
- make-switch 47
- make-synopsis 47
- make-text 48
- make-xswitch 48
- map-frames 51
- mapoptions 81
- match-option 62
- maybe-pop-argument 51
- maybe-push 52
- Method, (setf argument-styles) 71
- Method, (setf cmdline-options) 73
- Method, (setf column) 73
- Method, (setf error-string) 76
- Method, (setf frames) 77
- Method, (setf no-values) 82
- Method, (setf traversedp) 90
- Method, (setf yes-values) 91
- Method, argument 70
- Method, argument-name 70
- Method, argument-required-p 70
- Method, argument-style 70
- Method, argument-styles 71
- Method, background 71
- Method, blink 71
- Method, bottom-padding 71
- Method, check 72
- Method, check-name-clash 72, 73
- Method, clon-options-group 73
- Method, close-frame 73
- Method, cmdline-options 73
- Method, column 73
- Method, comment 74
- Method, concealedp 74
- Method, contents 74
- Method, convert 74, 75
- Method, copy-instance 75
- Method, crossed-out-p 75
- Method, default-value 75
- Method, description 75
- Method, enum 75
- Method, env-val 76
- Method, env-var 76
- Method, error-handler 76
- Method, error-string 76
- Method, fallback-value 76
- Method, foreground 76
- Method, framedp 77
- Method, frames 77
- Method, get-bottom-padding 77
- Method, header 77
- Method, help-spec 77, 78
- Method, help-spec-will-print 78
- Method, hiddenp 78
- Method, highlight 78
- Method, highlightp 79
- Method, intensity 79
- Method, inversep 79
- Method, italicp 79
- Method, item 79
- Method, item-separator 79
- Method, items 80
- Method, junk 80
- Method, left-padding 80
- Method, line-width 80
- Method, long-name 80
- Method, make-face-tree 81
- Method, mapoptions 81
- Method, name 81
- Method, negated-call 81
- Method, negated-pack 82
- Method, negated-pack-char 82
- Method, no-values 82
- Method, open-frame 83
- Method, option 83
- Method, option-sticky-distance 83
- Method, output-stream 83
- Method, parent 83
- Method, path-type 84
- Method, postfix 84
- Method, potential-pack 84
- Method, potential-pack-p 84
- Method, print-help-spec 84, 85
- Method, retrieve-from-environment 85
- Method, retrieve-from-long-call 85
- Method, retrieve-from-negated-call 85
- Method, retrieve-from-short-call 86
- Method, right-padding 86
- Method, search-path 86
- Method, sface-tree 86
- Method, short-call 87
- Method, short-name 87
- Method, short-pack 87
- Method, short-pack-char 87
- Method, short-syntax-help-spec-prefix 88
- Method, sibling 88
- Method, stream-ioctl-output-handle 88
- Method, stringify 88, 89

Method, subface	89
Method, subfaces	89
Method, synopsis	89
Method, theme	89
Method, top-padding	90
Method, traversedp	90
Method, typespec	90
Method, underline	90
Method, untraverse	90, 91
Method, value	91
Method, visiblep	91
Method, yes-values	91
multiple-value-getopt-cmdline	43

N

name	81
negated-call	81
negated-pack	82
negated-pack-char	82
nickname-package	48
no-values	82

O

open-frame	82, 83
open-line	62
open-next-line	62
open-sface	62
option	83
option-abbreviation-distance	62
option-call-p	62
option-sticky-distance	83
output-stream	83

P

parent	83
parent-generation	63
path-type	83, 84
pathname-component-null-p	63
pop-frame	83
postfix	84
potential-pack	84
potential-pack-char	63
potential-pack-p	84
princ-char	63
princ-highlight-property-instances	63
princ-spaces	63
princ-string	63
print-error	64
print-faced-help-spec	64
print-help	64
print-help-spec	84, 85
print-string	64
progname	48
push-frame	64
putenv	64

R

reach-column	64
read-argument	64
read-call	65
read-env-val	65
read-long-name	65
read-sface-tree	65
read-value	65
release-status-number	65
remainder	48
remove-keys	65
replace-in-keys	52
replace-key	65
replace-keys	66
restart-on-error	66
restartable-check	66
restartable-cmdline-convert	66
restartable-cmdline-junk-error	66
restartable-convert	67
restartable-environment-convert	67
restartable-invalid-negated-syntax-error	52
restartable-spurious-cmdline- argument-error	52
restrict-because	67
retrieve-from-environment	85
retrieve-from-long-call	85
retrieve-from-negated-call	85
retrieve-from-short-call	86
right-padding	86

S

safe-left-margin	67
safe-right-margin	67
search-branch	67
search-face	68
search-option	68
search-option-by-abbreviation	68
search-option-by-name	68
search-path	86
search-sticky-option	68
select-keys	69
setup-termio	48
sface-tree	86
short-call	87
short-name	87
short-pack	87
short-pack-char	87
short-syntax-help-spec-prefix	87, 88
sibling	88
split-path	69
stream-ioctl-output-handle	88
stream-line-width	69
stringify	88, 89
subface	89
subfaces	89
synopsis	89

T

theme 89
top-padding 89, 90
traversedp 90
try-read-sface-tree 69
try-read-theme 69
typespec 90

U

underline 90
untraverse 90, 91

V

value 91
version 49
visiblep 91

W

winsize-ws-col 69
winsize-ws-row 69
winsize-ws-xpixel 69
winsize-ws-ypixel 70
with-context 43
with-context-error-handler 52
with-winsize 52

Y

yes-values 91

A.3 Variables

*

<code>*configuration*</code>	50
<code>*context*</code>	41
<code>*copyright-years*</code>	41
<code>*executable*</code>	41
<code>*highlight-properties*</code>	50
<code>*item-names*</code>	50
<code>*release-major-level*</code>	41
<code>*release-minor-level*</code>	41
<code>*release-name*</code>	41
<code>*release-status*</code>	41
<code>*release-status-level*</code>	42
<code>*synopsis*</code>	42

+

<code>+tiocgwinz+</code>	49
--------------------------------	----

A

<code>argument</code>	94, 95, 96, 97
<code>argument-name</code>	102, 108, 110, 112, 116
<code>argument-required-p</code>	116
<code>argument-style</code>	113
<code>argument-styles</code>	113

B

<code>background</code>	106
<code>blink</code>	105
<code>bottom-padding</code>	105

C

<code>clon-options-group</code>	115
<code>cmdline-options</code>	101
<code>column</code>	111
<code>comment</code>	94, 96
<code>concealedp</code>	106
<code>Constant, +tiocgwinz+</code>	49
<code>Constant, offset-of-winsize-ws-col</code>	49
<code>Constant, offset-of-winsize-ws-row</code>	49
<code>Constant, offset-of-winsize-ws-xpixel</code>	49
<code>Constant, offset-of-winsize-ws-ypixel</code>	49
<code>Constant, size-of-winsize</code>	49
<code>contents</code>	115
<code>crossed-out-p</code>	106

D

<code>default-value</code>	116
<code>description</code>	109

E

<code>enum</code>	103, 117
<code>env-var</code>	93, 110
<code>error-handler</code>	102
<code>error-string</code>	94

F

<code>fallback-value</code>	116
<code>foreground</code>	106
<code>framedp</code>	106
<code>frames</code>	112

H

<code>header</code>	107
<code>hiddenp</code>	108
<code>highlight</code>	102
<code>highlight-property-instances</code>	99
<code>highlightp</code>	111

I

<code>intensity</code>	105
<code>inversep</code>	105
<code>italicp</code>	105
<code>item</code>	92, 93, 97, 98
<code>item-separator</code>	105
<code>items</code>	100

L

<code>left-margin</code>	99
<code>left-padding</code>	104
<code>line-width</code>	102, 111
<code>long-name</code>	109

N

<code>name</code>	98, 99, 104
<code>negated-pack</code>	114
<code>no-values</code>	113

O

<code>offset-of-winsize-ws-col</code>	49
<code>offset-of-winsize-ws-row</code>	49
<code>offset-of-winsize-ws-xpixel</code>	49
<code>offset-of-winsize-ws-ypixel</code>	49
<code>option</code>	96, 98
<code>output-stream</code>	111

P

<code>parent</code>	106
<code>path-type</code>	110
<code>postfix</code>	114
<code>potential-pack</code>	115
<code>progname</code>	101

R

<code>remainder</code>	102
<code>right-margin</code>	99
<code>right-padding</code>	104

S

- search-path 102
 - sface 98
 - sface-tree 111
 - short-name 109
 - short-pack 114
 - sibling 110
 - size-of-winsize 49
 - Slot, argument 94, 95, 96, 97
 - Slot, argument-name 102, 108, 110, 112, 116
 - Slot, argument-required-p 116
 - Slot, argument-style 113
 - Slot, argument-styles 113
 - Slot, background 106
 - Slot, blink 105
 - Slot, bottom-padding 105
 - Slot, clon-options-group 115
 - Slot, cmdline-options 101
 - Slot, column 111
 - Slot, comment 94, 96
 - Slot, concealedp 106
 - Slot, contents 115
 - Slot, crossed-out-p 106
 - Slot, default-value 116
 - Slot, description 109
 - Slot, enum 103, 117
 - Slot, env-var 93, 110
 - Slot, error-handler 102
 - Slot, error-string 94
 - Slot, fallback-value 116
 - Slot, foreground 106
 - Slot, framedp 106
 - Slot, frames 112
 - Slot, header 107
 - Slot, hiddenp 108
 - Slot, highlight 102
 - Slot, highlight-property-instances 99
 - Slot, highlightp 111
 - Slot, intensity 105
 - Slot, inversep 105
 - Slot, italicp 105
 - Slot, item 92, 93, 97, 98
 - Slot, item-separator 105
 - Slot, items 100
 - Slot, left-margin 99
 - Slot, left-padding 104
 - Slot, line-width 102, 111
 - Slot, long-name 109
 - Slot, name 98, 99, 104
 - Slot, negated-pack 114
 - Slot, no-values 113
 - Slot, option 96, 98
 - Slot, output-stream 111
 - Slot, parent 106
 - Slot, path-type 110
 - Slot, postfix 114
 - Slot, potential-pack 115
 - Slot, progname 101
 - Slot, remainder 102
 - Slot, right-margin 99
 - Slot, right-padding 104
 - Slot, search-path 102
 - Slot, sface 98
 - Slot, sface-tree 111
 - Slot, short-name 109
 - Slot, short-pack 114
 - Slot, sibling 110
 - Slot, source 98
 - Slot, subfaces 106
 - Slot, synopsis 101
 - Slot, theme 102
 - Slot, top-padding 105
 - Slot, traversedp 107
 - Slot, typespec 108
 - Slot, underline 105
 - Slot, value 95, 98, 99
 - Slot, visiblep 104
 - Slot, yes-values 113
 - source 98
 - Special Variable, **configuration** 50
 - Special Variable, **context** 41
 - Special Variable, **copyright-years** 41
 - Special Variable, **executable** 41
 - Special Variable, **highlight-properties** 50
 - Special Variable, **item-names** 50
 - Special Variable, **release-major-level** 41
 - Special Variable, **release-minor-level** 41
 - Special Variable, **release-name** 41
 - Special Variable, **release-status** 41
 - Special Variable, **release-status-level** 42
 - Special Variable, **synopsis** 42
 - subfaces 106
 - synopsis 101
- T**
- theme 102
 - top-padding 105
 - traversedp 107
 - typespec 108
- U**
- underline 105
- V**
- value 95, 98, 99
 - visiblep 104
- Y**
- yes-values 113

A.4 Data types

A

abstract-class 100

C

Class, abstract-class 100
 Class, container 100
 Class, context 101
 Class, enum 102
 Class, enum-base 103
 Class, face 103
 Class, flag 106
 Class, group 107
 Class, item 107
 Class, lisplib 108
 Class, negatable 108
 Class, option 108
 Class, path 110
 Class, sface 110
 Class, sheet 111
 Class, stropt 112
 Class, switch 112
 Class, switch-base 112
 Class, synopsis 114
 Class, text 115
 Class, valued-option 115
 Class, xswitch 116
 cmdline-error 91
 cmdline-junk-error 92
 cmdline-option 98
 cmdline-option-error 92
 Condition, cmdline-error 91
 Condition, cmdline-junk-error 92
 Condition, cmdline-option-error 92
 Condition, environment-error 93
 Condition, environmental-option-error 93
 Condition, home-directory 93
 Condition, invalid-argument 94
 Condition, invalid-cmdline-argument 94
 Condition, invalid-environment-value 94
 Condition, invalid-negated-equal-syntax 95
 Condition, invalid-negated-syntax 95
 Condition, invalid-short-equal-syntax 95
 Condition, invalid-value 95
 Condition, missing-cmdline-argument 96
 Condition, option-error 96
 Condition, spurious-cmdline-argument 96
 Condition, unknown-cmdline-option-error 97
 Condition, unrecognized-negated-call-error 97
 Condition, unrecognized-short-call-error 97
 container 100
 context 101

E

enum 102
 enum-base 103
 environment-error 93
 environmental-option-error 93

F

face 103
 flag 106
 frame 98

G

group 107

H

highlight-frame 99
 highlight-property-instance 99
 home-directory 93

I

invalid-argument 94
 invalid-cmdline-argument 94
 invalid-environment-value 94
 invalid-negated-equal-syntax 95
 invalid-negated-syntax 95
 invalid-short-equal-syntax 95
 invalid-value 95
 item 107

L

lisplib 108

M

missing-cmdline-argument 96

N

negatable 108
 net.didierverna.clon 5, 27
 net.didierverna.clon.core 7
 net.didierverna.clon.setup 7, 39
 net.didierverna.clon.setup/termio 6
 net.didierverna.clon.termio 5

O

option 108
 option-error 96

P

Package, net.didierverna.clon 27
 Package, net.didierverna.clon.setup 39
 path 110

S

<code>sface</code>	110
<code>sheet</code>	111
<code>spurious-cmdline-argument</code>	96
<code>stropt</code>	112
Structure, <code>cmdline-option</code>	98
Structure, <code>frame</code>	98
Structure, <code>highlight-frame</code>	99
Structure, <code>highlight-property-instance</code>	99
<code>switch</code>	112
<code>switch-base</code>	112
<code>synopsis</code>	114
System, <code>net.didierverna.clon</code>	5
System, <code>net.didierverna.clon.core</code>	7
System, <code>net.didierverna.clon.setup</code>	7
System, <code>net.didierverna.clon.setup/termio</code>	6
System, <code>net.didierverna.clon.termio</code>	5

T

<code>text</code>	115
-------------------------	-----

U

<code>unknown-cmdline-option-error</code>	97
<code>unrecognized-negated-call-error</code>	97
<code>unrecognized-short-call-error</code>	97

V

<code>valued-option</code>	115
----------------------------------	-----

X

<code>xswitch</code>	116
----------------------------	-----