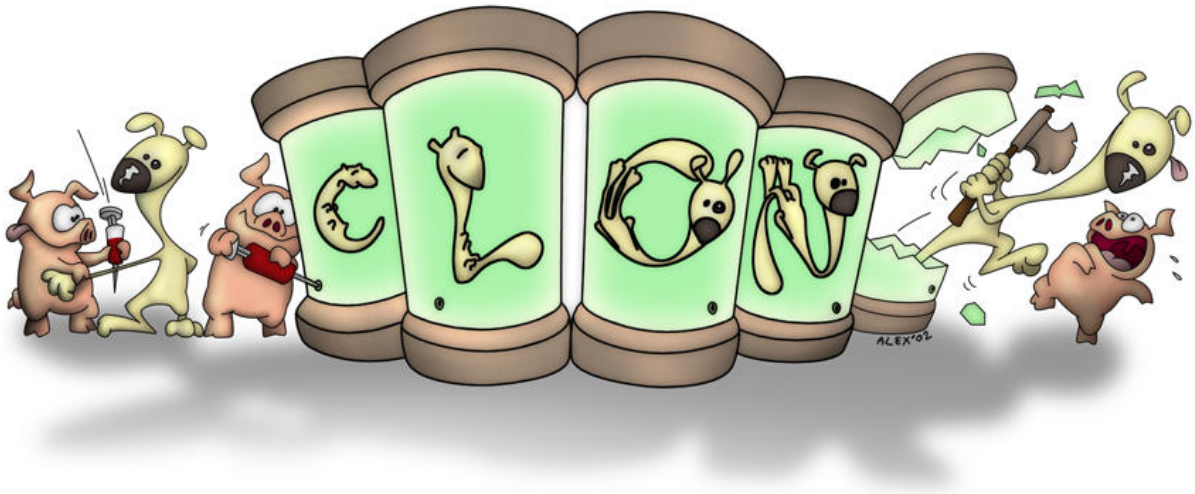
The Clon Reference Manual

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.

Table of Contents

Copying

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 clonified[1] 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)

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-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 initargs 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 *FORMS* [Macro]
Define a new synopsis.

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

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

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

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 OPTION* [Function]
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 THEME LINE-WIDTH HIGHLIGHT* [Function]
Print CONTEXT's help.

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

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

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

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 DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

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 DESCRIPTION ENV-VAR HIDDEN* [Function]

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 DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR TYPESPEC FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

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 DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE TYPE HIDDEN* [Function]

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 DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE HIDDEN* [Function]

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 DESCRIPTION ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

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 DESCRIPTION ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

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)

`progname` &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 major*10000 + minor*100 + 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/didie`
`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/didie`
`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/didie`
`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/didie`
`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/didie`
`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/didie`
`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 *FORMS* [Macro]
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/didie`
`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/didie`
`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 INDENTation 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 *MATCH-LENGTH SHORTEST-DISTANCE CLOSEST-MATCH* [Function]

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 CMDLINE-ARGUMENT* [Function]

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 explicitely 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 INSTANCE* [Function]

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 (*SFACE* **SFACE**) (*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 (*SFACE* **SFACE**) (*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 *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR FALLBACK-VALUE DEFAULT-VALUE TYPE HIDDEN* [Function]

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* &key *ARGUMENT-NAME ARGUMENT-TYPE ENV-VAR 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.
- 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* &key *ARGUMENT-STYLE ARGUMENT-TYPE ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

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* &key *ARGUMENT-NAME ARGUMENT-TYPE ENUM ENV-VAR DEFAULT-VALUE HIDDEN* [Function]

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 THEME LINE-WIDTH HIGHLIGHT* [Function]
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* *LONG-NAME PARTIAL-NAME* [Function]
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 wo 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* *LONG-NAME* [Function]
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/didie`
`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/didie`
`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/didie`
`Lisp/clon/termio/sbcl/constants.lisp-temp`

`winsize-ws-ypixel` *STRUCT* [Function]
`(setf winsize-ws-ypixel)` *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/didie`
`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`) (*CONTAINER2* `container`) [Method]
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* &optional *CMDLINE-ARGUMENT CMDLINE* [Generic Function]
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 *CMDLINE-ARGUMENT CMDLINE* [Generic Function]

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 *CMDLINE-ARGUMENT CMDLINE* [Method]

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

`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'subface named NAME, or nil.

`subface` *FACE* (*NAMES* `list`) &aux *BRANCH* [Method]
Return the leaf face from FACE'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)

`left-margin` [Slot]

Readers [`frame-left-margin`], page 56, (function)

Writers [`(setf frame-left-margin)`], page 56, (function)

`right-margin` [Slot]

Readers [`frame-right-margin`], page 56, (function)

Writers [`(setf frame-right-margin)`], page 56, (function)

`highlight-frame` () [Structure]

The HIGHLIGHT-FRAME structure.
This structure holds both layout and highlight properties used for printing.

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

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

Direct superclasses

[`frame`], page 98, (structure)

Direct methods

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

Direct slots

`highlight-property-instances` [Slot]

Readers [`highlight-frame-highlight-property-instances`], page 57, (function)

Writers [`(setf highlight-frame-highlight-property-instances)`], page 57, (function)

`highlight-property-instance` () [Structure]

The HIGHLIGHT-PROPERTY-INSTANCE structure.

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

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

Direct superclasses

`structure-object` (structure)

Direct slots

`name` [Slot]

Readers [`highlight-property-instance-name`], page 57, (function)

Writers [`(setf highlight-property-instance-name)`], page 57, (function)

`value` [Slot]

Readers [`highlight-property-instance-value`], page 58, (function)

Writers [`(setf highlight-property-instance-value)`], 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 associatiion 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)

`progname` [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 `list`

`search-path` [Slot]
The search path for Clon files.

Readers [`search-path`], page 86, (generic function)

`theme` [Slot]
The theme filename.

Readers [`theme`], page 89, (generic function)

`line-width` [Slot]
The line width for help display.

Readers [`line-width`], page 80, (generic function)

`highlight` [Slot]
Clon's output highlight mode.

Readers [`highlight`], page 78, (generic function)

`error-handler` [Slot]
The behavior to adopt on option retrieval errors.

Type `symbol`

Initform `:quit`

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

Direct Default Initargs

Initarg	Value
`:cmdline`	`(net.didierverna.clon:cmdline)`

`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 `"type"`

`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 relatively 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, - (<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.

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)

`concealedp` [Slot]
The face's concealed status.

Initargs `:concealed`

Readers [`concealedp`], page 74, (generic function)

`crossed-out-p` [Slot]
The face's crossed out status.

Initargs `:crossed-out`

Readers [`crossed-out-p`], page 75, (generic function)

`framedp` [Slot]
The face's framed status.

Initargs `:framed`

Readers [`framedp`], page 77, (generic function)

`foreground` [Slot]
The face foreground.

Initargs `:foreground`

Readers [`foreground`], page 76, (generic function)

`background` [Slot]
The face background.

Initargs `:background`

Readers [`background`], page 71, (generic function)

`subfaces` [Slot]
The face children.

Initargs `:subfaces`

Readers [`subfaces`], page 89, (generic function)

`parent` [Slot]
The face parent.

Readers [`parent`], page 83, (generic function)

`flag ()` [Class]
The FLAG class.
This class implements options that don't take any argument.

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

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

Direct superclasses
[`option`], page 108, (class)

Direct methods
[`retrieve-from-environment`], 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

- `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)

`potential-pack` [Slot]
The potential pack string.

Type `(or null string)`

Readers [`potential-pack`], page 84, (generic function)

`clon-options-group` [Slot]
The Clon options group.

Type `net.didierverna.clon::group`

Initargs `:clon-options-group`

Readers [`clon-options-group`], page 73, (generic function)

`text ()` [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

`contents` [Slot]
The actual text string.

Type `string`

Initargs `:contents`

Readers [`contents`], page 74, (generic function)

`valued-option ()` [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

argument-name [Slot]
The option's argument display name.

Initargs :argument-name

Initform "arg"

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

argument-required-p [Slot]
Whether the option's argument is required.

Readers [argument-required-p], page 70, (generic function)

fallback-value [Slot]
The option's fallback value.

Initargs :fallback-value

Readers [fallback-value], page 76, (generic function)

default-value [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` [Slot]

The set of possible non-boolean values.

Appendix A Indexes

A.1 Concepts

F

L

M

N

A.2 Functions

%

(

~

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Y

A.3 Variables

*

+

A

B

C

D

E

F

H

I

L

N

O

P

R

S

T

U

V

Y

A.4 Data types

A

C

E

F

G

H

I

L

M

N

O

P

S

T

U

V

X