Osicat Manual

draft version

Table of Contents

1 Osicat

`osicat` [Package]

Osicat is a lightweight operating system interface for Common Lisp on Unix-platforms. It is not a POSIX-style `api`, but rather a simple lispy accompaniment to the standard `ansi` facilities.

Osicat homepage:

http://www.common-lisp.net/project/osicat/

Concepts:

Designated directory

When a relative pathname designator is used as a directory designator it is first resolved against `*default-pathname-default*`, and then against the current directory. (With `merge-pathnames` in both cases.)

1.1 Conditions

`system-error` [Condition]

Class precedence list: `system-error`, `error`, `serious-condition`, `condition`, `t`

Base class for errors signalled by Osicat.

1.2 Environment

`environment-variable` *name* [Function]

`environment-variable` returns the environment variable identified by `name`, or `nil` if one does not exist. `name` can either be a symbol or a string.

`setf environment-variable` sets the environment variable identified by `name` to `value`. Both `name` and `value` can be either a symbols or strings. Signals an error on failure.

`environment` [Function]

`environment` returns the current environment as an assoc-list. `setf environment` modifies the environment its argument.

Often it is preferable to use `setf environment-variable` and `makunbound-environment-variable` to modify the environment instead of `setf environment`.

`makunbound-environment-variable` *name* [Function]

Removes the environment variable identified by `name` from the current environment. `name` can be either a string or a symbol. Returns the string designated by `name`. Signals an error on failure.

1.3 Directories

`current-directory` [Function]

`current-directory` returns the operating system's current directory, which may or may not correspond to `*default-pathname-defaults*`.

`setf current-directory` changes the operating system's current directory to the `pathspec`. An error is signalled if the `pathspec` is wild or does not designate a directory.

`delete-directory-and-files` *dirname* **&key** *if-does-not-exist* [Function]

Recursively deletes all files and directories within the directory designated by the non-wild pathname designator `dirname` including `dirname` itself. `if-does-not-exist` must be one of `:error` or `:ignore` where `:error` means that an error will be signaled if the directory `dirname` does not exist.

`delete-directory` *pathspec* [Function]

Deletes the directory designated by `pathspec`. Returns `t`. The directory must be empty. Symbolic links are not followed.

Signals an error if `pathspec` is wild, doesn't designate a directory, or if the directory could not be deleted.

`directory-exists-p` *pathspec* [Function]

Checks whether the file named by the pathname designator `pathspec` exists and is a directory. Returns its truename if this is the case, `nil` otherwise. Follows symbolic links.

`list-directory` *pathspec* **&key** *bare-pathnames* [Function]

Returns a fresh list of pathnames corresponding to all files within the directory named by the non-wild pathname designator `pathspec`. If `bare-pathnames` is non-NIL only the files's bare pathnames are returned (with an empty directory component), otherwise the files' pathnames are merged with `pathspec`.

`mapdir` *function pathspec* [Function]

Applies function to each entry in directory designated by `pathspec` in turn and returns a list of the results. Binds `*default-pathname-defaults*` to the directory designated by pathspec round to function call.

If `pathspec` designates a symbolic link, it is implicitly resolved.

Signals an error if `pathspec` is wild or doesn't designate a directory.

`walk-directory` *dirname fn* **&key** *directories if-does-not-exist test* [Function]

Recursively applies the function `fn` to all files within the directory named by the non-wild pathname designator `dirname` and all of its sub-directories. Returns `t` on success.

`fn` will only be applied to files for which the function `test` returns a true value. If `directories` is not `nil`, `fn` and `test` are applied to directories as well. If `directories` is `:depth-first`, `fn` will be applied to the directory's contents first. If `directories` is `:breadth-first` and `test` returns `nil`, the directory's content will be skipped. `if-does-not-exist` must be one of `:error` or `:ignore` where `:error` means that an error will be signaled if the directory `dirname` does not exist.

`with-directory-iterator` (*iterator pathspec*) **&body** *body* [Macro]

`pathspec` must be a valid directory designator: `*default-pathname-defaults*` is bound, and (CURRENT-DIRECTORY) is set to the designated directory for the dynamic scope of the body.

Within the lexical scope of the body, `iterator` is defined via macrolet such that successive invocations of (ITERATOR) return the directory entries, one by one. Both files and directories are returned, except '.' and '..'. The order of entries is not guaranteed. The entries are returned as relative pathnames against the designated directory. Entries that are symbolic links are not resolved, but links that point to directories are interpreted as directory designators. Once all entries have been returned, further invocations of (ITERATOR) will all return `nil`.

The value returned is the value of the last form evaluated in body. Signals an error if `pathspec` is wild or does not designate a directory.

1.4 Files and Symbolic Links

`file-exists-p` *pathspec* **&optional** *file-kind* [Function]

Checks whether the file named by the pathname designator `pathspec` exists, if this is the case and `file-kind` is specified it also checks the file kind. If the tests succeed, return two values: truename and file kind of `pathspec`, `nil` otherwise. Follows symbolic links.

`regular-file-exists-p` *pathspec* [Function]

Checks whether the file named by the pathname designator `pathspec` exists and is a regular file. Returns its truename if this is the case, `nil` otherwise. Follows symbolic links.

`file-kind` *pathspec* **&key** *follow-symlinks* [Function]

Returns a keyword indicating the kind of file designated by `pathspec`, or `nil` if the file does not exist. Does not follow symbolic links by default.

Possible file-kinds in addition to `nil` are: `:regular-file`, `:symbolic-link`, `:directory`, `:pipe`, `:socket`, `:character-device`, and `:block-device`. If `follow-symlinks` is non-NIL and `pathspec` designates a broken symlink returns `:broken` as second value.

Signals an error if `pathspec` is wild.

`file-permissions` *pathspec* [Function]

`file-permissions` returns a list of keywords identifying the permissions of `pathspec`.

`setf file-permissions` sets the permissions of `pathspec` as identified by the symbols in list.

If `pathspec` designates a symbolic link, that link is implicitly resolved.

Permission symbols consist of `:user-read`, `:user-write`, `:user-exec`, `:group-read`, `:group-write`, `:group-exec`, `:other-read`, `:other-write`, `:other-exec`, `:set-user-id`, `:set-group-id`, and `:sticky`.

Both signal an error if `pathspec` is wild, or doesn't designate an existing file.

`open-temporary-file` **&key** *pathspec element-type external-format* [Function]

Creates a temporary file setup for input and output, and returns a stream connected to that file.

`pathspec` serves as template for the file to be created: a certain number of random characters will be concatenated to the file component of `pathspec`. If `pathspec` has no directory component, the file will be created inside `*temporary-directory*`. The file itself is unlinked once it has been opened.

`element-type` specifies the unit of transaction of the stream. Consider using `with-temporary-file` instead of this function.

On failure, a `file-error` may be signalled.

`with-temporary-file` (*stream* **&key** *pathspec element-type external-format*) **&body** *body* [Macro]

Within the lexical scope of the body, `stream` is connected to a temporary file as created by `open-temporary-file`. The file is closed automatically once `body` exits.

`make-link` *link* **&key** *target hard* [Function]

Creates `link` that points to `target`. Defaults to a symbolic link, but giving a non-NIL value to the keyword argument `:hard` creates a hard link. Returns the pathname of the link.

Relative targets are resolved against the link. Relative links are resolved against `*default-pathname-defaults*`.

Signals an error if either target or link is wild, target does not exist, or link exists already.

`read-link` *pathspec* [Function]

Returns the pathname pointed to by the symbolic link designated by `pathspec`. If the link is relative, then the returned pathname is relative to the link, not `*default-pathname-defaults*`.

Signals an error if `pathspec` is wild, or does not designate a symbolic link.

1.5 Users

`user-info` *id* [Function]

`user-info` returns the password entry for the given name or numerical user `id`, as an assoc-list.

1.6 Time

`get-monotonic-time` [Function]

Gets current time in seconds from a system's monotonic clock.

1.7 Pathname Utilities

`absolute-pathname-p` *pathspec* [Function]

Returns `t` if the `pathspec` designates an absolute pathname, `nil` otherwise.

`absolute-pathname` *pathspec* **&optional** *default* [Function]

Returns an absolute pathname corresponding to `pathspec` by merging it with `default`, and (CURRENT-DIRECTORY) if necessary.

`directory-pathname-p` *pathspec* [Function]

Returns `nil` if `pathspec` (a pathname designator) does not designate a directory, `pathspec` otherwise. It is irrelevant whether file or directory designated by `pathspec` does actually exist.

`pathname-as-directory` *pathspec* [Function]

Converts the non-wild pathname designator `pathspec` to directory form.

`pathname-as-file` *pathspec* [Function]

Converts the non-wild pathname designator `pathspec` to file form.

`pathname-directory-pathname` *pathspec* [Function]

Returns the directory part of `pathspec` as a pathname.

`relative-pathname-p` *pathspec* [Function]

Returns `t` if the `pathspec` designates a relative pathname, `nil` otherwise.

`unmerge-pathnames` *pathspec* **&optional** *default* [Function]

Removes those leading directory components from `pathspec` that are shared with `default`.

Index