

## **Programming Language—Common Lisp**

### **20. Files**

Version 15.17R, X3J13/94-101R.  
Fri 12-Aug-1994 6:35pm EDT

---

---

## 20.1 File System Concepts

This section describes the Common Lisp interface to file systems. The model used by this interface assumes that **files** are named by **filenames**, that a *filename* can be represented by a *pathname object*, and that given a *pathname* a **stream** can be constructed that connects to a *file* whose *filename* it represents.

For information about opening and closing *files*, and manipulating their contents, see Chapter 21 (Streams).

Figure 20–1 lists some *operators* that are applicable to *files* and directories.

|                     |                        |                       |
|---------------------|------------------------|-----------------------|
| <b>compile-file</b> | <b>file-length</b>     | <b>open</b>           |
| <b>delete-file</b>  | <b>file-position</b>   | <b>probe-file</b>     |
| <b>directory</b>    | <b>file-write-date</b> | <b>rename-file</b>    |
| <b>file-author</b>  | <b>load</b>            | <b>with-open-file</b> |

Figure 20–1. File and Directory Operations

### 20.1.1 Coercion of Streams to Pathnames

A **stream associated with a file** is either a *file stream* or a *synonym stream* whose target is a **stream associated with a file**. Such streams can be used as *pathname designators*.

Normally, when a *stream associated with a file* is used as a *pathname designator*, it denotes the *pathname* used to open the *file*; this may be, but is not required to be, the actual name of the *file*.

Some functions, such as **truename** and **delete-file**, coerce *streams* to *pathnames* in a different way that involves referring to the actual *file* that is open, which might or might not be the file whose name was opened originally. Such special situations are always notated specifically and are not the default.

### 20.1.2 File Operations on Open and Closed Streams

Many *functions* that perform *file* operations accept either *open* or *closed streams* as *arguments*; see Section 21.1.3 (Stream Arguments to Standardized Functions).

Of these, the *functions* in Figure 20–2 treat *open* and *closed streams* differently.

|                    |                        |                   |
|--------------------|------------------------|-------------------|
| <b>delete-file</b> | <b>file-author</b>     | <b>probe-file</b> |
| <b>directory</b>   | <b>file-write-date</b> | <b>truename</b>   |

Figure 20–2. File Functions that Treat Open and Closed Streams Differently

---

Since treatment of *open streams* by the *file system* may vary considerably between *implementations*, however, a *closed stream* might be the most reliable kind of *argument* for some of these functions—in particular, those in Figure 20–3. For example, in some *file systems*, *open files* are written under temporary names and not renamed until *closed* and/or are held invisible until *closed*. In general, any code that is intended to be portable should use such *functions* carefully.

| directory | probe-file | truename |
|-----------|------------|----------|
|-----------|------------|----------|

Figure 20–3. File Functions where Closed Streams Might Work Best

## 20.1.3 Truenames

Many *file systems* permit more than one *filename* to designate a particular *file*.

Even where multiple names are possible, most *file systems* have a convention for generating a canonical *filename* in such situations. Such a canonical *filename* (or the *pathname* representing such a *filename*) is called a **truename**.

The *truename* of a *file* may differ from other *filenames* for the file because of symbolic links, version numbers, logical device translations in the *file system*, *logical pathname* translations within Common Lisp, or other artifacts of the *file system*.

The *truename* for a *file* is often, but not necessarily, unique for each *file*. For instance, a Unix *file* with multiple hard links could have several *truenames*.

### 20.1.3.1 Examples of Truenames

For example, a DEC TOPS-20 system with *files* PS:<JOE>F00.TXT.1 and PS:<JOE>F00.TXT.2 might permit the second *file* to be referred to as PS:<JOE>F00.TXT.0, since the “.0” notation denotes “newest” version of several *files*. In the same *file system*, a “logical device” “JOE:” might be taken to refer to PS:<JOE>” and so the names JOE:F00.TXT.2 or JOE:F00.TXT.0 might refer to PS:<JOE>F00.TXT.2. In all of these cases, the *truename* of the file would probably be PS:<JOE>F00.TXT.2.

If a *file* is a symbolic link to another *file* (in a *file system* permitting such a thing), it is conventional for the *truename* to be the canonical name of the *file* after any symbolic links have been followed; that is, it is the canonical name of the *file* whose contents would become available if an *input stream* to that *file* were opened.

In the case of a *file* still being created (that is, of an *output stream* open to such a *file*), the exact *truename* of the file might not be known until the *stream* is closed. In this case, the *function* **truename** might return different values for such a *stream* before and after it was closed. In fact, before it is closed, the name returned might not even be a valid name in the *file system*—for example, while a file is being written, it might have version :newest and might only take on a specific numeric value later when the file is closed even in a *file system* where all files have numeric versions.

---

## directory

---

*Function*

### Syntax:

`directory pathspec &key → pathnames`

### Arguments and Values:

*pathspec*—a *pathname designator*, which may contain *wild* components.

*pathnames*—a *list of physical pathnames*.

### Description:

Determines which, if any, *files* that are present in the file system have names matching *pathspec*, and returns a *fresh list of pathnames* corresponding to the *truenames* of those *files*.

An *implementation* may be extended to accept *implementation-defined* keyword arguments to **directory**.

### Affected By:

The host computer's file system.

### Exceptional Situations:

If the attempt to obtain a directory listing is not successful, an error of *type* **file-error** is signaled.

### See Also:

**pathname**, **logical-pathname**, **ensure-directories-exist**, Section 20.1 (File System Concepts), Section 21.1.1.1.2 (Open and Closed Streams), Section 19.1.2 (Pathnames as Filenames)

### Notes:

If the *pathspec* is not *wild*, the resulting list will contain either zero or one elements.

Common Lisp specifies “&key” in the argument list to **directory** even though no *standardized* keyword arguments to **directory** are defined. “:allow-other-keys t” may be used in *conforming programs* in order to quietly ignore any additional keywords which are passed by the program but not supported by the *implementation*.

---

## probe-file

---

*Function*

### Syntax:

`probe-file pathspec → truename`

### Arguments and Values:

*pathspec*—a *pathname designator*.

*truename*—a *physical pathname* or `nil`.

### Description:

**probe-file** tests whether a file exists.

**probe-file** returns *false* if there is no file named *pathspec*, and otherwise returns the *truename* of *pathspec*.

If the *pathspec designator* is an open *stream*, then **probe-file** produces the *truename* of its associated *file*. If *pathspec* is a *stream*, whether open or closed, it is coerced to a *pathname* as if by the function **pathname**.

### Affected By:

The host computer's file system.

### Exceptional Situations:

An error of type **file-error** is signaled if *pathspec* is *wild*.

An error of type **file-error** is signaled if the *file system* cannot perform the requested operation.

### See Also:

**truename**, **open**, **ensure-directories-exist**, **pathname**, **logical-pathname**, Section 20.1 (File System Concepts), Section 21.1.1.1.2 (Open and Closed Streams), Section 19.1.2 (Pathnames as Filenames)

---

## ensure-directories-exist

*Function*

---

### Syntax:

**ensure-directories-exist** *pathspec* &key *verbose* → *pathspec*, *created*

### Arguments and Values:

*pathspec*—a *pathname designator*.

*verbose*—a *generalized boolean*.

*created*—a *generalized boolean*.

### Description:

Tests whether the directories containing the specified *file* actually exist, and attempts to create them if they do not.

---

If the containing directories do not exist and if *verbose* is *true*, then the *implementation* is permitted (but not required) to perform output to *standard output* saying what directories were created. If the containing directories exist, or if *verbose* is *false*, this function performs no output.

The *primary value* is the given *pathspec* so that this operation can be straightforwardly composed with other file manipulation expressions. The *secondary value*, *created*, is *true* if any directories were created.

### Affected By:

The host computer's file system.

### Exceptional Situations:

An error of *type* **file-error** is signaled if the host, device, or directory part of *pathspec* is *wild*.

If the directory creation attempt is not successful, an error of *type* **file-error** is signaled; if this occurs, it might be the case that none, some, or all of the requested creations have actually occurred within the *file system*.

### See Also:

**probe-file**, **open**, Section 19.1.2 (Pathnames as Filenames)

---

## truename

*Function*

---

### Syntax:

`truename filespec → truename`

### Arguments and Values:

*filespec*—a *pathname designator*.

*truename*—a *physical pathname*.

### Description:

**truename** tries to find the *file* indicated by *filespec* and returns its *truename*. If the *filespec designator* is an open *stream*, its associated *file* is used. If *filespec* is a *stream*, **truename** can be used whether the *stream* is open or closed. It is permissible for **truename** to return more specific information after the *stream* is closed than when the *stream* was open. If *filespec* is a *pathname* it represents the name used to open the file. This may be, but is not required to be, the actual name of the file.

### Examples:

```
;; An example involving version numbers. Note that the precise nature of
;; the truename is implementation-dependent while the file is still open.
(with-open-file (stream ">vistor>test.text.newest")
```

---

```
(values (pathname stream)
        (truename stream)))
→ #P"S:>vistor>test.text.newest", #P"S:>vistor>test.text.1"
or
→ #P"S:>vistor>test.text.newest", #P"S:>vistor>test.text.newest"
or
→ #P"S:>vistor>test.text.newest", #P"S:>vistor>_temp.._temp..1"

;; In this case, the file is closed when the truename is tried, so the
;; truename information is reliable.
(with-open-file (stream ">vistor>test.text.newest")
  (close stream)
  (values (pathname stream)
          (truename stream)))
→ #P"S:>vistor>test.text.newest", #P"S:>vistor>test.text.1"

;; An example involving TOP-20's implementation-dependent concept
;; of logical devices -- in this case, "DOC:" is shorthand for
;; "PS:<DOCUMENTATION>" ...
(with-open-file (stream "CMUC::DOC:DUMPER.HLP")
  (values (pathname stream)
          (truename stream)))
→ #P"CMUC::DOC:DUMPER.HLP", #P"CMUC::PS:<DOCUMENTATION>DUMPER.HLP.13"
```

### Exceptional Situations:

An error of *type* **file-error** is signaled if an appropriate *file* cannot be located within the *file system* for the given *filespec*, or if the *file system* cannot perform the requested operation.

An error of *type* **file-error** is signaled if *pathname* is *wild*.

### See Also:

**pathname**, **logical-pathname**, Section 20.1 (File System Concepts), Section 19.1.2 (Pathnames as Filenames)

### Notes:

**truename** may be used to account for any *filename* translations performed by the *file system*.

---

## file-author

*Function*

---

### Syntax:

**file-author** *pathspec* → *author*

### Arguments and Values:

*pathspec*—a *pathname designator*.



---

*author*—a *string* or **nil**.

**Description:**

Returns a *string* naming the author of the *file* specified by *pathspec*, or **nil** if the author's name cannot be determined.

**Examples:**

```
(with-open-file (stream ">relativity>general.text")
  (file-author s))
→ "albert"
```

**Affected By:**

The host computer's file system.

Other users of the *file* named by *pathspec*.

**Exceptional Situations:**

An error of *type* **file-error** is signaled if *pathspec* is *wild*.

An error of *type* **file-error** is signaled if the *file system* cannot perform the requested operation.

**See Also:**

**pathname**, **logical-pathname**, Section 20.1 (File System Concepts), Section 19.1.2 (Pathnames as Filenames)

---

## file-write-date

*Function*

---

**Syntax:**

**file-write-date** *pathspec* → *date*

**Arguments and Values:**

*pathspec*—a *pathname designator*.

*date*—a *universal time* or **nil**.

**Description:**

Returns a *universal time* representing the time at which the *file* specified by *pathspec* was last written (or created), or returns **nil** if such a time cannot be determined.

**Examples:**

```
(with-open-file (s "noel.text"
  :direction :output :if-exists :error)
```

---

```
(format s "~&Dear Santa,~2%I was good this year.  ~
          Please leave lots of toys.~2%Love, Sue~
          ~2%attachments: milk, cookies~%")
(truename s))
→ #P"CUPID:/susan/noel.text"
(with-open-file (s "noel.text")
  (file-write-date s))
→ 2902600800
```

### Affected By:

The host computer's file system.

### Exceptional Situations:

An error of *type* **file-error** is signaled if *pathspec* is *wild*.

An error of *type* **file-error** is signaled if the *file system* cannot perform the requested operation.

### See Also:

Section 25.1.4.2 (Universal Time), Section 19.1.2 (Pathnames as Filenames)

---

## rename-file

*Function*

---

### Syntax:

**rename-file** *filespec new-name* → *defaulted-new-name, old-truename, new-truename*

### Arguments and Values:

*filespec*—a *pathname designator*.

*new-name*—a *pathname designator* other than a *stream*.

*defaulted-new-name*—a *pathname*

*old-truename*—a *physical pathname*.

*new-truename*—a *physical pathname*.

### Description:

**rename-file** modifies the file system in such a way that the file indicated by *filespec* is renamed to *defaulted-new-name*.

It is an error to specify a filename containing a *wild* component, for *filespec* to contain a **nil** component where the file system does not permit a **nil** component, or for the result of defaulting missing components of *new-name* from *filespec* to contain a **nil** component where the file system does not permit a **nil** component.

---

If *new-name* is a *logical pathname*, **rename-file** returns a *logical pathname* as its *primary value*.

**rename-file** returns three values if successful. The *primary value*, *defaulted-new-name*, is the resulting name which is composed of *new-name* with any missing components filled in by performing a **merge-pathnames** operation using *filespec* as the defaults. The *secondary value*, *old-truename*, is the *truename* of the *file* before it was renamed. The *tertiary value*, *new-truename*, is the *truename* of the *file* after it was renamed.

If the *filespec designator* is an open *stream*, then the *stream* itself and the file associated with it are affected (if the *file system* permits).

### Examples:

```
;; An example involving logical pathnames.
(with-open-file (stream "sys:chemistry;lead.text"
                    :direction :output :if-exists :error)
  (princ "eureka" stream)
  (values (pathname stream) (truename stream)))
→ #P"SYS:CHEMISTRY;LEAD.TEXT.NEWEST", #P"Q:>sys>chem>lead.text.1"
(rename-file "sys:chemistry;lead.text" "gold.text")
→ #P"SYS:CHEMISTRY;GOLD.TEXT.NEWEST",
   #P"Q:>sys>chem>lead.text.1",
   #P"Q:>sys>chem>gold.text.1"
```

### Exceptional Situations:

If the renaming operation is not successful, an error of *type* **file-error** is signaled.

An error of *type* **file-error** might be signaled if *filespec* is *wild*.

### See Also:

**truename**, **pathname**, **logical-pathname**, Section 20.1 (File System Concepts), Section 19.1.2 (Pathnames as Filenames)

---

## delete-file

*Function*

---

### Syntax:

**delete-file** *filespec* → *t*

### Arguments and Values:

*filespec*—a *pathname designator*.

### Description:

Deletes the *file* specified by *filespec*.

If the *filespec designator* is an open *stream*, then *filespec* and the file associated with it are affected (if the file system permits), in which case *filespec* might be closed immediately, and the deletion might be immediate or delayed until *filespec* is explicitly closed, depending on the requirements of the file system.

It is *implementation-dependent* whether an attempt to delete a nonexistent file is considered to be successful.

**delete-file** returns *true* if it succeeds, or signals an error of *type* **file-error** if it does not.

The consequences are undefined if *filespec* has a *wild* component, or if *filespec* has a **nil** component and the file system does not permit a **nil** component.

### Examples:

```
(with-open-file (s "delete-me.text" :direction :output :if-exists :error))  
→ NIL  
(setq p (probe-file "delete-me.text")) → #P"R:>fred>delete-me.text.1"  
(delete-file p) → T  
(probe-file "delete-me.text") → false  
(with-open-file (s "delete-me.text" :direction :output :if-exists :error)  
  (delete-file s))  
→ T  
(probe-file "delete-me.text") → false
```

### Exceptional Situations:

If the deletion operation is not successful, an error of *type* **file-error** is signaled.

An error of *type* **file-error** might be signaled if *filespec* is *wild*.

### See Also:

**pathname**, **logical-pathname**, Section 20.1 (File System Concepts), Section 19.1.2 (Pathnames as Filenames)

---

## file-error

*Condition Type*

---

### Class Precedence List:

**file-error**, **error**, **serious-condition**, **condition**, **t**

### Description:

The *type* **file-error** consists of error conditions that occur during an attempt to open or close a file, or during some low-level transactions with a file system. The “offending pathname” is initialized by the **:pathname** initialization argument to **make-condition**, and is *accessed* by the *function* **file-error-pathname**.

---

**See Also:**

`file-error-pathname`, `open`, `probe-file`, `directory`, `ensure-directories-exist`

---

## **file-error-pathname**

*Function*

---

**Syntax:**

`file-error-pathname condition`  $\rightarrow$  *pathspec*

**Arguments and Values:**

*condition*—a *condition* of type **file-error**.

*pathspec*—a *pathname designator*.

**Description:**

Returns the “offending pathname” of a *condition* of type **file-error**.

**Exceptional Situations:**

**See Also:**

**file-error**, Chapter 9 (Conditions)

---

Version 15.17R, X3J13/94-101R.  
Fri 12-Aug-1994 6:35pm EDT

---