COPYFILES

CopyFiles makes it easy to copy or move groups of files from one place to another.

Installation

Load `COPYFILES.LCOM` from the library.

Function

`(COPYFILES` *SOURCE DESTINATION OPTIONS*`)` [Function]

Copies the files designated by *SOURCE* to the place designated by *DESTINATION*.

SOURCE is a pattern such as given to `DIRECTORY` or `DIR`; it can also be a list of file names.

DESTINATION is either a directory name or a file-name pattern, with a one-to-one match of the wild card characters (*s) in *DESTINATION* to *s in *SOURCE*. The number of *s in each source pattern needs to match the number of *s in each destination pattern. (See examples below.)

Note: You must use *s if you want to name files. If you do not, the `COPYFILES` code assumes that any *DESTINATION* specified without *s in it is a destination directory.

OPTIONS is a list (if you have only one and it is a symbol, you can supply it as a symbol) that may include one or more of the options specified below.

Note: If the destination is a non-existent NS subdirectory, `COPYFILES` asks whether it should create it. If you answer `YES`, then it creates the subdirectory. If you answer `NO`, it aborts without processing any files.

OPTION: Conversation Mode

You can specify how verbose CopyFiles is about what it is doing:

`QUIET`	Don't print anything while working.
`(OUTPUT` *LISTFILE*`)`	Print the name of each file that gets copied on *LISTFILE*. `(OUTPUT T)` is the default.
`TERSE`	Only print a period (`.`) for each file moved/copied.

OPTION: Query Mode

You can specify whether CopyFiles should ask for confirmation before each transfer.

`ASK`	Ask each time before moving/copying a file (default is to not ask).
`(ASK N)`	Ask, with default to `No` after `DWIMWAIT` seconds.
`(ASK Y)`	Ask, with default to `Yes` after `DWIMWAIT` seconds.

OPTION: Version Control

CopyFiles normally uses the Lisp function COPYFILE to create a new file. It also usually copies only the highest version, and creates a new version at the destination. Alternatively, you can specify any of the following:

RENAME or MOVE	Use RENAMEFILE instead of COPYFILE; i.e., the source is deleted afterwards.
ALLVERSIONS	Copy all versions and preserve version numbers.
REPLACE	If a file by the same name exists on the destination, overwrite it (don’t create a new version).

Note: When * is used as the source version number, be sure to specify ALLVERSIONS. This is important because some devices list files by version number from highest to lowest, while by default the version numbers at the destination are assigned in ascending order. Hence, if ALLVERSIONS is not specified, the versions may be reversed, as can be verified by looking at the creation dates.

OPTION: When To Copy

CopyFiles normally compares the creation dates of the file on the source and any matching file on the destination to determine whether it is necessary to copy. The following options are mutually exclusive:

ALWAYS	Always copy the file.
>	Copy only when a file by the same name but an earlier creation date exists on the destination.
>A	Similar to >, but also copy if the file doesn’t exist on the destination; i.e., > ALWAYS.
#	Copy only when a file by the same name but a different creation date exists on the destination.
#A	Similar to #, but also copy if the file doesn’t exist on the destination, i.e., # ALWAYS.
=A	Copy only if there isn’t a file of the same name on the destination.

Not all combinations of options make sense; for example, ALLVERSIONS probably doesn’t work right with any date comparison algorithms.

The default setting is (>A); that is, copy the highest version if it doesn’t exist on the destination or if an older creation date exists, and print out messages about all files considered.

OPTION: Clean-Up After Copying Files

CopyFiles can be instructed to delete some files after it has finished copying.

PURGE	This involves a separate pass (afterwards): any file on the destination which doesn’t have a counterpart on the source is deleted.

`PURGESOURCE`	Converse of `PURGE` (and used by it): if the file is on the source and not on the destination, delete it.

Limitations

The creation date comparison does not work when either the source or the destination does not support creation dates. For example, the TCP-IP protocol doesn't support any way to find out the creation date of a remote file. For this reason, `COPYFILES` can only be used in `ALWAYS` mode when using a TCP-IP protocol.

Examples

```
(COPYFILES '{ERIS}<USER>*.MAIL '{PHYLUM}<USER>OLD-*.MAIL)
```

Copies any mail file on `{ERIS}<USER>` to `{PHYLUM}<USER>`, copying `FOO.MAIL` to `OLD-FOO.MAIL`.

```
(COPYFILES '{ERIS}<USER>*.MAIL '{PHYLUM}<USER>OLD-*.MAIL 'RENAME)
```

Uses `RENAMEFILE` instead.

```
(COPYFILES '({DSK}TEST {DSK}WEST) '{PHYLUM}<MYDIR>)
```

Copies the files `TEST` and `WEST` from `{DSK}` to `{PHYLUM}<MYDIR>`.

```
(COPYFILES '{PHYLUM}<USER>*.AR '{PHYLEX:}<USER> '=A)
```

Copies all ARs on `{PHYLUM}<USER>` to the `PHYLEX` NS file server; if any are already there, it won't bother copying them.

```
(COPYFILES '{PHYLUM}<USER>AR.INDEX '{DSK}AR.INDEX '(>A REPLACE))
```

Copies the AR index to `{DSK}`, replacing any older version that is already there.

```
COPYFILES({DSK}*.; {FLOPPY})
```

Copies all files on `{DSK}` that have no file name extensions to `{FLOPPY}`.

```
(COPYFILES '{ERIS}<USER> '{PHYLUM}<USER> '(#A PURGE))
```

Makes `{PHYLUM}<USER>` look like `{ERIS}<USER>`, bringing over any file that isn't already on `{PHYLUM}` and then deleting the ones that were on `{PHYLUM}` and aren't on `{ERIS}` any more.

[This page intentionally left blank]