NAME

ExtUtils::Manifest - utilities to write and check a MANIFEST file

SYNOPSIS

```
use ExtUtils::Manifest qw(...funcs to import...);

mkmanifest();

my @missing_files    = manicheck;
my @skipped          = skipcheck;
my @extra_files      = filecheck;
my($missing, $extra) = fullcheck;

my $found    = manifind();

my $manifest = maniread();

manicopy($read,$target);

maniadd({$file => $comment, ...});
```

DESCRIPTION

Functions

ExtUtils::Manifest exports no functions by default. The following are exported on request

mkmanifest

```
mkmanifest();
```

Writes all files in and below the current directory to your *MANIFEST*. It works similar to the result of the Unix command

```
find . > MANIFEST
```

All files that match any regular expression in a file *MANIFEST.SKIP* (if it exists) are ignored.

Any existing *MANIFEST* file will be saved as *MANIFEST.bak*.

manifind

```
my $found = manifind();
```

returns a hash reference. The keys of the hash are the files found below the current directory.

manicheck

```
my @missing_files = manicheck();
```

checks if all the files within a `MANIFEST` in the current directory really do exist. If `MANIFEST` and the tree below the current directory are in sync it silently returns an empty list. Otherwise it returns a list of files which are listed in the `MANIFEST` but missing from the directory, and by default also outputs these names to STDERR.

filecheck

```
my @extra_files = filecheck();
```

finds files below the current directory that are not mentioned in the `MANIFEST` file. An optional

file MANIFEST.SKIP will be consulted. Any file matching a regular expression in such a file will not be reported as missing in the MANIFEST file. The list of any extraneous files found is returned, and by default also reported to STDERR.

fullcheck

```
my($missing, $extra) = fullcheck();
```

does both a manicheck() and a filecheck(), returning then as two array refs.

skipcheck

```
my @skipped = skipcheck();
```

lists all the files that are skipped due to your MANIFEST.SKIP file.

maniread

```
my $manifest = maniread();
my $manifest = maniread($manifest_file);
```

reads a named MANIFEST file (defaults to MANIFEST in the current directory) and returns a HASH reference with files being the keys and comments being the values of the HASH. Blank lines and lines which start with # in the MANIFEST file are discarded.

maniskip

```
my $skipchk = maniskip();
my $skipchk = maniskip($manifest_skip_file);

if ($skipchk->($file)) { .. }
```

reads a named MANIFEST.SKIP file (defaults to MANIFEST.SKIP in the current directory) and returns a CODE reference that tests whether a given filename should be skipped.

manicopy

```
manicopy(\%src, $dest_dir);
manicopy(\%src, $dest_dir, $how);
```

Copies the files that are the keys in %src to the $dest_dir. %src is typically returned by the maniread() function.

```
manicopy( maniread(), $dest_dir );
```

This function is useful for producing a directory tree identical to the intended distribution tree.

$how can be used to specify a different methods of "copying". Valid values are cp, which actually copies the files, ln which creates hard links, and best which mostly links the files but copies any symbolic link to make a tree without any symbolic link. cp is the default.

maniadd

```
maniadd({ $file => $comment, ...});
```

Adds an entry to an existing *MANIFEST* unless its already there.

$file will be normalized (ie. Unixified). **UNIMPLEMENTED**

MANIFEST

A list of files in the distribution, one file per line. The MANIFEST always uses Unix filepath conventions even if you're not on Unix. This means *foo/bar* style not *foo\bar*.

Anything between white space and an end of line within a MANIFEST file is considered to be a

comment. Any line beginning with # is also a comment. Beginning with ExtUtils::Manifest 1.52, a filename may contain whitespace characters if it is enclosed in single quotes; single quotes or backslashes in that filename must be backslash-escaped.

```
# this a comment
some/file
some/other/file           comment about some/file
'some/third file'         comment
```

MANIFEST.SKIP

The file MANIFEST.SKIP may contain regular expressions of files that should be ignored by mkmanifest() and filecheck(). The regular expressions should appear one on each line. Blank lines and lines which start with # are skipped. Use `\#` if you need a regular expression to start with a #.

For example:

```
# Version control files and dirs.
\bRCS\b
\bCVS\b
,v$
\B\.svn\b

# Makemaker generated files and dirs.
^MANIFEST\.
^Makefile$
^blib/
^MakeMaker-\d

# Temp, old and emacs backup files.
~$
\.old$
^#.*#$
^\.#
```

If no MANIFEST.SKIP file is found, a default set of skips will be used, similar to the example above. If you want nothing skipped, simply make an empty MANIFEST.SKIP file.

In one's own MANIFEST.SKIP file, certain directives can be used to include the contents of other MANIFEST.SKIP files. At present two such directives are recognized.

#!include_default

This inserts the contents of the default MANIFEST.SKIP file

#!include /Path/to/another/manifest.skip

This inserts the contents of the specified external file

The included contents will be inserted into the MANIFEST.SKIP file in between *#!start included /path/to/manifest.skip* and *#!end included /path/to/manifest.skip* markers. The original MANIFEST.SKIP is saved as MANIFEST.SKIP.bak.

EXPORT_OK

`&mkmanifest`, `&manicheck`, `&filecheck`, `&fullcheck`, `&maniread`, and `&manicopy` are exportable.

GLOBAL VARIABLES

`$ExtUtils::Manifest::MANIFEST` defaults to `MANIFEST`. Changing it results in both a different `MANIFEST` and a different `MANIFEST.SKIP` file. This is useful if you want to maintain different distributions for different audiences (say a user version and a developer version including RCS).

`$ExtUtils::Manifest::Quiet` defaults to 0. If set to a true value, all functions act silently.

`$ExtUtils::Manifest::Debug` defaults to 0. If set to a true value, or if PERL_MM_MANIFEST_DEBUG is true, debugging output will be produced.

DIAGNOSTICS

All diagnostic output is sent to `STDERR`.

`Not in MANIFEST:` *file*

is reported if a file is found which is not in `MANIFEST`.

`Skipping` *file*

is reported if a file is skipped due to an entry in `MANIFEST.SKIP`.

`No such file:` *file*

is reported if a file mentioned in a `MANIFEST` file does not exist.

`MANIFEST:` *$!*

is reported if `MANIFEST` could not be opened.

`Added to MANIFEST:` *file*

is reported by mkmanifest() if $Verbose is set and a file is added to MANIFEST. $Verbose is set to 1 by default.

ENVIRONMENT

PERL_MM_MANIFEST_DEBUG

Turns on debugging

SEE ALSO

ExtUtils::MakeMaker which has handy targets for most of the functionality.

AUTHOR

Andreas Koenig `andreas.koenig@anima.de`

Maintained by Michael G Schwern `schwern@pobox.com` within the ExtUtils-MakeMaker package and, as a separate CPAN package, by Randy Kobes `r.kobes@uwinnipeg.ca`.