



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

***Rocky Enterprise Linux 9.2 Manual Pages on command 'File::Spec.3perl'***

***\$ man File::Spec.3perl***

File::Spec(3perl) Perl Programmers Reference Guide File::Spec(3perl)

NAME

File::Spec - portably perform operations on file names

SYNOPSIS

```
use File::Spec;
```

```
$x=File::Spec->catfile('a', 'b', 'c');
```

which returns 'a/b/c' under Unix. Or:

```
use File::Spec::Functions;
```

```
$x = catfile('a', 'b', 'c');
```

DESCRIPTION

This module is designed to support operations commonly performed on file specifications (usually called "file names", but not to be confused with the contents of a file, or Perl's file handles), such as concatenating several directory and file names into a single path, or determining whether a path is rooted. It is based on code directly taken from MakeMaker 5.17, code written by Andreas K?nig, Andy Dougherty, Charles Bailey, Ilya Zakharevich, Paul Schinder, and others.

Since these functions are different for most operating systems, each set of OS specific routines is available in a separate module, including:

File::Spec::Unix

File::Spec::Mac

File::Spec::OS2

File::Spec::Win32

File::Spec::VMS

The module appropriate for the current OS is automatically loaded by File::Spec. Since some modules (like VMS) make use of facilities available only under that OS, it may not be possible to load all modules under all operating systems.

Since File::Spec is object oriented, subroutines should not be called directly, as in:

```
File::Spec::catfile('a','b');
```

but rather as class methods:

```
File::Spec->catfile('a','b');
```

For simple uses, File::Spec::Functions provides convenient functional forms of these methods.

## METHODS

### canonpath

No physical check on the filesystem, but a logical cleanup of a path.

```
$cpath = File::Spec->canonpath( $path );
```

Note that this does *not* collapse `x/../y` sections into `y`. This is by design. If `/foo` on your system is a symlink to `/bar/baz`, then `/foo/../quux` is actually `/bar/quux`, not `/quux` as a naive `../`-removal would give you. If you want to do this kind of processing, you probably want "Cwd"'s `realpath()` function to actually traverse the filesystem cleaning up paths like this.

### catdir

Concatenate two or more directory names to form a complete path ending with a directory.

But remove the trailing slash from the resulting string, because it doesn't look good, isn't necessary and confuses OS/2. Of course, if this is the root directory, don't cut off the trailing slash :-)

```
$path = File::Spec->catdir( @directories );
```

### catfile

Concatenate one or more directory names and a filename to form a complete path ending with a filename

```
$path = File::Spec->catfile( @directories, $filename );
```

### curdir

Returns a string representation of the current directory.

```
$curdir = File::Spec->curdir();
```

### devnull

Returns a string representation of the null device.

```
$devnull = File::Spec->devnull();
```

#### rootdir

Returns a string representation of the root directory.

```
$rootdir = File::Spec->rootdir();
```

#### tmpdir

Returns a string representation of the first writable directory from a list of possible temporary directories. Returns the current directory if no writable temporary directories are found. The list of directories checked depends on the platform; e.g.

File::Spec::Unix checks \$ENV{TMPDIR} (unless taint is on) and /tmp.

```
$tmpdir = File::Spec->tmpdir();
```

#### updir

Returns a string representation of the parent directory.

```
$updir = File::Spec->updir();
```

#### no\_upwards

Given a list of files in a directory (such as from "readdir()"), strip out '.' and '..'.

SECURITY NOTE: This does NOT filter paths containing '..', like

'../../../../etc/passwd', only literal matches to '.' and '..'.

```
@paths = File::Spec->no_upwards( readdir $dirhandle );
```

#### case\_tolerant

Returns a true or false value indicating, respectively, that alphabetic case is not or is significant when comparing file specifications. Cygwin and Win32 accept an optional drive argument.

```
$is_case_tolerant = File::Spec->case_tolerant();
```

#### file\_name\_is\_absolute

Takes as its argument a path, and returns true if it is an absolute path.

```
$is_absolute = File::Spec->file_name_is_absolute( $path );
```

This does not consult the local filesystem on Unix, Win32, OS/2, or Mac OS (Classic).

It does consult the working environment for VMS (see "file\_name\_is\_absolute" in

File::Spec::VMS).

#### path

Takes no argument. Returns the environment variable "PATH" (or the local platform's equivalent) as a list.

```
@PATH = File::Spec->path();
```

join

join is the same as catfile.

splitpath

Splits a path in to volume, directory, and filename portions. On systems with no concept of volume, returns "" for volume.

```
($volume,$directories,$file) =  
    File::Spec->splitpath( $path );  
($volume,$directories,$file) =  
    File::Spec->splitpath( $path, $no_file );
```

For systems with no syntax differentiating filenames from directories, assumes that the last file is a path unless \$no\_file is true or a trailing separator or /. or ../ is present. On Unix, this means that \$no\_file true makes this return ( "", \$path, "" ).

The directory portion may or may not be returned with a trailing '/'.

The results can be passed to "catpath()" to get back a path equivalent to (usually identical to) the original path.

splitdir

The opposite of "catdir".

```
@dirs = File::Spec->splitdir( $directories );
```

\$directories must be only the directory portion of the path on systems that have the concept of a volume or that have path syntax that differentiates files from directories.

Unlike just splitting the directories on the separator, empty directory names ("" ) can be returned, because these are significant on some OSes.

catpath()

Takes volume, directory and file portions and returns an entire path. Under Unix, \$volume is ignored, and directory and file are concatenated. A '/' is inserted if need be. On other OSes, \$volume is significant.

```
$full_path = File::Spec->catpath( $volume, $directory, $file );
```

abs2rel

Takes a destination path and an optional base path returns a relative path from the base path to the destination path:

```
$rel_path = File::Spec->abs2rel( $path ) ;  
$rel_path = File::Spec->abs2rel( $path, $base ) ;
```

If \$base is not present or "", then Cwd::cwd() is used. If \$base is relative, then it is

converted to absolute form using "rel2abs()". This means that it is taken to be relative to Cwd::cwd().

On systems with the concept of volume, if \$path and \$base appear to be on two different volumes, we will not attempt to resolve the two paths, and we will instead simply return \$path. Note that previous versions of this module ignored the volume of \$base, which resulted in garbage results part of the time.

On systems that have a grammar that indicates filenames, this ignores the \$base filename as well. Otherwise all path components are assumed to be directories.

If \$path is relative, it is converted to absolute form using "rel2abs()". This means that it is taken to be relative to Cwd::cwd().

No checks against the filesystem are made. On VMS, there is interaction with the working environment, as logicals and macros are expanded.

Based on code written by Shigio Yamaguchi.

rel2abs()

Converts a relative path to an absolute path.

```
$abs_path = File::Spec->rel2abs( $path ) ;
```

```
$abs_path = File::Spec->rel2abs( $path, $base ) ;
```

If \$base is not present or "", then Cwd::cwd() is used. If \$base is relative, then it is converted to absolute form using "rel2abs()". This means that it is taken to be relative to Cwd::cwd().

On systems with the concept of volume, if \$path and \$base appear to be on two different volumes, we will not attempt to resolve the two paths, and we will instead simply return \$path. Note that previous versions of this module ignored the volume of \$base, which resulted in garbage results part of the time.

On systems that have a grammar that indicates filenames, this ignores the \$base filename as well. Otherwise all path components are assumed to be directories.

If \$path is absolute, it is cleaned up and returned using "canonpath".

No checks against the filesystem are made. On VMS, there is interaction with the working environment, as logicals and macros are expanded.

Based on code written by Shigio Yamaguchi.

For further information, please see File::Spec::Unix, File::Spec::Mac, File::Spec::OS2, File::Spec::Win32, or File::Spec::VMS.

File::Spec::Unix, File::Spec::Mac, File::Spec::OS2, File::Spec::Win32, File::Spec::VMS,  
File::Spec::Functions, ExtUtils::MakeMaker

## AUTHOR

Currently maintained by Ken Williams "<KWILLIAMS@cpan.org>".

The vast majority of the code was written by Kenneth Albanowski "<kjahds@kjahds.com>",

Andy Dougherty "<doughera@lafayette.edu>", Andreas K?nig

"<A.Koenig@franz.ww.TU-Berlin.DE>", Tim Bunce "<Tim.Bunce@ig.co.uk>". VMS support by

Charles Bailey "<bailey@newman.upenn.edu>". OS/2 support by Ilya Zakharevich

"<ilya@math.ohio-state.edu>". Mac support by Paul Schinder "<schinder@pobox.com>", and

Thomas Wegner "<wegner\_thomas@yahoo.com>". abs2rel() and rel2abs() written by Shigio

Yamaguchi "<shigio@tamacom.com>", modified by Barrie Slaymaker "<barries@slaysys.com>".

splitpath(), splitdir(), catpath() and catdir() by Barrie Slaymaker.

## COPYRIGHT

Copyright (c) 2004-2013 by the Perl 5 Porters. All rights reserved.

This program is free software; you can redistribute it and/or modify it under the same  
terms as Perl itself.

perl v5.34.0

2023-11-23

File::Spec(3perl)