NAME

File::Spec::Unix - File::Spec for Unix, base for other File::Spec modules

SYNOPSIS

```
require File::Spec::Unix; # Done automatically by File::Spec
```

DESCRIPTION

Methods for manipulating file specifications. Other File::Spec modules, such as File::Spec::Mac, inherit from File::Spec::Unix and override specific methods.

METHODS

canonpath()

No physical check on the filesystem, but a logical cleanup of a path. On UNIX eliminates successive slashes and successive "/.".

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

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 OS2. Of course, if this is the root directory, don't cut off the trailing slash :-)

catfile

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

curdir

Returns a string representation of the current directory. "." on UNIX.

devnull

Returns a string representation of the null device. "/dev/null" on UNIX.

rootdir

Returns a string representation of the root directory. "/" on UNIX.

tmpdir

Returns a string representation of the first writable directory from the following list or the current directory if none from the list are writable:

```
$ENV{TMPDIR}
/tmp
```

Since perl 5.8.0, if running under taint mode, and if $ENV{TMPDIR} is tainted, it is not used.

updir

Returns a string representation of the parent directory. ".." on UNIX.

no_upwards

Given a list of file names, strip out those that refer to a parent directory. (Does not strip symlinks, only '.', '..', and equivalents.)

case_tolerant

Returns a true or false value indicating, respectively, that alphabetic is not or is significant when comparing file specifications.

file_name_is_absolute

Takes as argument a path and returns true if it is an 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 as an array.

join

join is the same as catfile.

splitpath

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

Splits a path into volume, directory, and filename portions. On systems with no concept of volume, returns '' for volume.

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 OSs.

On Unix,

```
    File::Spec->splitdir( "/a/b//c/" );
```

Yields:

```
    ( '', 'a', 'b', '', 'c', '' )
```

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 needed (though if the directory portion doesn't start with '/' it is not added). On other OSs, $volume is significant.

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()* 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()*.

On systems that have a grammar that indicates filenames, this ignores the $base filename. 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()*.

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()* 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()*.

On systems that have a grammar that indicates filenames, this ignores the $base filename. 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.

SEE ALSO

File::Spec