NAME

fileparse - split a pathname into pieces

basename - extract just the filename from a path

dirname - extract just the directory from a path

SYNOPSIS

```
use File::Basename;

($name,$path,$suffix) = fileparse($fullname,@suffixlist);
$name = fileparse($fullname,@suffixlist);
fileparse_set_fstype($os_string);
$basename = basename($fullname,@suffixlist);
$dirname = dirname($fullname);

($name,$path,$suffix) = fileparse("lib/File/Basename.pm",qr{\.pm});
fileparse_set_fstype("VMS");
$basename = basename("lib/File/Basename.pm",".pm");
$dirname = dirname("lib/File/Basename.pm");
```

DESCRIPTION

These routines allow you to parse file specifications into useful pieces using the syntax of different operating systems.

fileparse_set_fstype

You select the syntax via the routine fileparse_set_fstype().

If the argument passed to it contains one of the substrings "VMS", "MSDOS", "MacOS", "AmigaOS" or "MSWin32", the file specification syntax of that operating system is used in future calls to fileparse(), basename(), and dirname(). If it contains none of these substrings, Unix syntax is used. This pattern matching is case-insensitive. If you've selected VMS syntax, and the file specification you pass to one of these routines contains a "/", they assume you are using Unix emulation and apply the Unix syntax rules instead, for that function call only.

If the argument passed to it contains one of the substrings "VMS", "MSDOS", "MacOS", "AmigaOS", "os2", "MSWin32" or "RISCOS", then the pattern matching for suffix removal is performed without regard for case, since those systems are not case-sensitive when opening existing files (though some of them preserve case on file creation).

If you haven't called fileparse_set_fstype(), the syntax is chosen by examining the builtin variable $^O according to these rules.

fileparse

The fileparse() routine divides a file specification into three parts: a leading **path**, a file **name**, and a **suffix**. The **path** contains everything up to and including the last directory separator in the input file specification. The remainder of the input file specification is then divided into **name** and **suffix** based on the optional patterns you specify in `@suffixlist`. Each element of this list can be a qr-quoted pattern (or a string which is interpreted as a regular expression), and is matched against the end of **name**. If this succeeds, the matching portion of **name** is removed and prepended to **suffix**. By proper use of `@suffixlist`, you can remove file types or versions for examination.

You are guaranteed that if you concatenate **path**, **name**, and **suffix** together in that order, the result will denote the same file as the input file specification.

In scalar context, fileparse() returns only the **name** part of the filename.

EXAMPLES

Using Unix file syntax:

```
($base,$path,$type) = fileparse('/virgil/aeneid/draft.book7',
    qr{\.book\d+});
```

would yield

```
$base eq 'draft'
$path eq '/virgil/aeneid/',
$type eq '.book7'
```

Similarly, using VMS syntax:

```
($name,$dir,$type) = fileparse('Doc_Root:[Help]Rhetoric.Rnh',
   qr{\..*});
```

would yield

```
$name eq 'Rhetoric'
$dir  eq 'Doc_Root:[Help]'
$type eq '.Rnh'
```

`basename`

The basename() routine returns the first element of the list produced by calling fileparse() with the same arguments, except that it always quotes metacharacters in the given suffixes. It is provided for programmer compatibility with the Unix shell command basename(1).

`dirname`

The dirname() routine returns the directory portion of the input file specification. When using VMS or MacOS syntax, this is identical to the second element of the list produced by calling fileparse() with the same input file specification. (Under VMS, if there is no directory information in the input file specification, then the current default device and directory are returned.) When using Unix or MSDOS syntax, the return value conforms to the behavior of the Unix shell command dirname(1). This is usually the same as the behavior of fileparse(), but differs in some cases. For example, for the input file specification *lib/*, fileparse() considers the directory name to be *lib/*, while dirname() considers the directory name to be *.*).