NAME

File::DosGlob - DOS like globbing and then some

SYNOPSIS

```
require 5.004;

# override CORE::glob in current package
use File::DosGlob 'glob';

# override CORE::glob in ALL packages (use with extreme caution!)
use File::DosGlob 'GLOBAL_glob';

@perlfiles = glob  "..\\pe?l/*.p?";
print <..\\pe?l/*.p?>;

# from the command line (overrides only in main::)
> perl -MFile::DosGlob=glob -e "print <../pe*/*p?>"
```

DESCRIPTION

A module that implements DOS-like globbing with a few enhancements. It is largely compatible with perlglob.exe (the M$ setargv.obj version) in all but one respect--it understands wildcards in directory components.

For example, `<..\\l*b\\file/*glob.p?>` will work as expected (in that it will find something like '..\lib\File/DosGlob.pm' alright). Note that all path components are case-insensitive, and that backslashes and forward slashes are both accepted, and preserved. You may have to double the backslashes if you are putting them in literally, due to double-quotish parsing of the pattern by perl.

Spaces in the argument delimit distinct patterns, so `glob('*.exe *.dll')` globs all filenames that end in `.exe` or `.dll`. If you want to put in literal spaces in the glob pattern, you can escape them with either double quotes, or backslashes. e.g. `glob('c:/"Program Files"/*/*.dll')`, or `glob('c:/Program\ Files/*/*.dll')`. The argument is tokenized using `Text::ParseWords::parse_line()`, so see *Text::ParseWords* for details of the quoting rules used.

Extending it to csh patterns is left as an exercise to the reader.

NOTES

- Mac OS (Classic) users should note a few differences. The specification of pathnames in glob patterns adheres to the usual Mac OS conventions: The path separator is a colon ':', not a slash '/' or backslash '\'. A full path always begins with a volume name. A relative pathname on Mac OS must always begin with a ':', except when specifying a file or directory name in the current working directory, where the leading colon is optional. If specifying a volume name only, a trailing ':' is required. Due to these rules, a glob like <*:> will find all mounted volumes, while a glob like <*> or <:*> will find all files and directories in the current directory.

  Note that updirs in the glob pattern are resolved before the matching begins, i.e. a pattern like "*HD:t?p::a*" will be matched as "*HD:a*". Note also, that a single trailing ':' in the pattern is ignored (unless it's a volume name pattern like "*HD:"), i.e. a glob like <:*:> will find both directories *and* files (and not, as one might expect, only directories).

  The metachars '*', '?' and the escape char '\' are valid characters in volume, directory and file names on Mac OS. Hence, if you want to match a '*', '?' or '\' literally, you have to escape these characters. Due to perl's quoting rules, things may get a bit complicated, when you want to match a string like '*' literally, or when you want to match '\' literally, but treat the immediately following character '*' as metachar. So, here's a rule of thumb (applies to both

single- and double-quoted strings): escape each '*' or '?' or '\' with a backslash, if you want to treat them literally, and then double each backslash and your are done. E.g.

- Match '*' literally

```
escape both '\' and '*'  : '\\\*'
double the backslashes   : '\\\\\\*'
```

(Internally, the glob routine sees a '*', which means that both '\' and '*' are escaped.)

- Match '\' literally, treat '*' as metachar

```
escape '\' but not '*'   : '\\*'
double the backslashes   : '\\\\*'
```

(Internally, the glob routine sees a '*', which means that '\' is escaped and '*' is not.)

Note that you also have to quote literal spaces in the glob pattern, as described above.

EXPORTS (by request only)

glob()

BUGS

Should probably be built into the core, and needs to stop pandering to DOS habits. Needs a dose of optimizium too.

AUTHOR

Gurusamy Sarathy <gsar@activestate.com>

HISTORY

- Support for globally overriding glob() (GSAR 3-JUN-98)
- Scalar context, independent iterator context fixes (GSAR 15-SEP-97)
- A few dir-vs-file optimizations result in glob importation being 10 times faster than using perlglob.exe, and using perlglob.bat is only twice as slow as perlglob.exe (GSAR 28-MAY-97)
- Several cleanups prompted by lack of compatible perlglob.exe under Borland (GSAR 27-MAY-97)
- Initial version (GSAR 20-FEB-97)

SEE ALSO

perl

perlglob.bat

Text::ParseWords