glob

(PHP 4 >= 4.3.0, PHP 5, PHP 7, PHP 8)

glob — Find pathnames matching a pattern

Description ¶

glob(string `$pattern`, int `$flags` = 0): array|false

The **glob()** function searches for all the pathnames matching `pattern` according to the rules used by the libc glob() function, which is similar to the rules used by common shells.

Parameters ¶

`pattern`

The pattern. No tilde expansion or parameter substitution is done.

Special characters:

- `*` - Matches zero or more characters.
- `?` - Matches exactly one character (any character).
- `[...]` - Matches one character from a group of characters. If the first character is `!`, matches any character not in the group.
- `\` - Escapes the following character, except when the **`GLOB_NOESCAPE`** flag is used.

`flags`

Valid flags:

- **`GLOB_MARK`** - Adds a slash (a backslash on Windows) to each directory returned

- `GLOB_NOSORT` - Return files as they appear in the directory (no sorting). When this flag is not used, the pathnames are sorted alphabetically
- `GLOB_NOCHECK` - Return the search pattern if no files matching it were found
- `GLOB_NOESCAPE` - Backslashes do not quote metacharacters
- `GLOB_BRACE` - Expands {a,b,c} to match 'a', 'b', or 'c'
- `GLOB_ONLYDIR` - Return only directory entries which match the pattern
- `GLOB_ERR` - Stop on read errors (like unreadable directories), by default errors are ignored.

  Note: The `GLOB_BRACE` flag is not available on some non GNU systems, like Solaris or Alpine Linux.

Return Values ¶

Returns an array containing the matched files/directories, an empty array if no file matched or `false` on error.

> **Note**:
>
> On some systems it is impossible to distinguish between empty match and an error.

Examples ¶

Example #1 Convenient way how glob() can replace opendir() and friends.

```
<?php
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "\n";
}
?>
```

The above example will output something similar to:

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

Notes ¶

> **Note**: This function will not work on remote files as the file to be examined must be accessible via the server's filesystem.

> **Note**: This function isn't available on some systems (e.g. old Sun OS).

See Also ¶

- opendir() - Open directory handle
- readdir() - Read entry from directory handle
- closedir() - Close directory handle
- fnmatch() - Match filename against a pattern

+ add a note

User Contributed Notes 49 notes

up

down
385
crayonviolent at phpfreaks dot com ¶
13 years ago

```
Since I feel this is rather vague and non-helpful, I thought I'd make a post detailing the mechanics
of the glob regex.

glob uses two special symbols that act like sort of a blend between a meta-character and a
quantifier.  These two characters are the * and ?

The ? matches 1 of any character except a /
The * matches 0 or more of any character except a /

If it helps, think of the * as the pcre equivalent of .* and ? as the pcre equivalent of the dot (.)

Note: * and ? function independently from the previous character. For instance, if you do
glob("a*.php") on the following list of files, all of the files starting with an 'a' will be
returned, but * itself would match:

a.php // * matches nothing
aa.php // * matches the second 'a'
ab.php // * matches 'b'
abc.php // * matches 'bc'
b.php // * matches nothing, because the starting 'a' fails
bc.php // * matches nothing, because the starting 'a' fails
bcd.php // * matches nothing, because the starting 'a' fails

It does not match just a.php and aa.php as a 'normal' regex would, because it matches 0 or more of
any character, not the character/class/group before it.

Executing glob("a?.php") on the same list of files will only return aa.php and ab.php because as
mentioned, the ? is the equivalent of pcre's dot, and is NOT the same as pcre's ?, which would match
0 or 1 of the previous character.

glob's regex also supports character classes and negative character classes, using the syntax [] and
[^]. It will match any one character inside [] or match any one character that is not in [^].

With the same list above, executing

glob("[ab]*.php) will return (all of them):
a.php  // [ab] matches 'a', * matches nothing
aa.php // [ab] matches 'a', * matches 2nd 'a'
ab.php // [ab] matches 'a', * matches 'b'
abc.php // [ab] matches 'a', * matches 'bc'
b.php // [ab] matches 'b', * matches nothing
bc.php // [ab] matches 'b', * matches 'c'
bcd.php // [ab] matches 'b', * matches 'cd'

glob("[ab].php") will return a.php and b.php

glob("[^a]*.php") will return:
b.php // [^a] matches 'b', * matches nothing
bc.php // [^a] matches 'b', * matches 'c'
bcd.php // [^a] matches 'b', * matches 'cd'
```

glob("[^ab]*.php") will return nothing because the character class will fail to match on the first character.

You can also use ranges of characters inside the character class by having a starting and ending character with a hyphen in between. For example, [a-z] will match any letter between a and z, [0-9] will match any (one) number, etc..

glob also supports limited alternation with {n1, n2, etc..}. You have to specify GLOB_BRACE as the 2nd argument for glob in order for it to work. So for example, if you executed glob("{a,b,c}.php", GLOB_BRACE) on the following list of files:

```
a.php
b.php
c.php
```

all 3 of them would return. Note: using alternation with single characters like that is the same thing as just doing glob("[abc].php"). A more interesting example would be glob("te{xt,nse}.php", GLOB_BRACE) on:

```
tent.php
text.php
test.php
tense.php
```

text.php and tense.php would be returned from that glob.

glob's regex does not offer any kind of quantification of a specified character or character class or alternation. For instance, if you have the following files:

```
a.php
aa.php
aaa.php
ab.php
abc.php
b.php
bc.php
```

with pcre regex you can do ~^a+\.php$~ to return

```
a.php
aa.php
aaa.php
```

This is not possible with glob. If you are trying to do something like this, you can first narrow it down with glob, and then get exact matches with a full flavored regex engine. For example, if you wanted all of the php files in the previous list that only have one or more 'a' in it, you can do this:

```
<?php
   $list = glob("a*.php");
   foreach ($list as $l) {
      if (preg_match("~^a+\.php$~",$file))
         $files[] = $l;
```

```
    }
?>
```

glob also does not support lookbehinds, lookaheads, atomic groupings, capturing, or any of the 'higher level' regex functions.

glob does not support 'shortkey' meta-characters like \w or \d.

up
down
72
uramihsayibok, gmail, com ¶
13 years ago

Those of you with PHP 5 don't have to come up with these wild functions to scan a directory recursively: the SPL can do it.

```
<?php

$dir_iterator = new RecursiveDirectoryIterator("/path");
$iterator = new RecursiveIteratorIterator($dir_iterator, RecursiveIteratorIterator::SELF_FIRST);
// could use CHILD_FIRST if you so wish

foreach ($iterator as $file) {
    echo $file, "\n";
}

?>
```

Not to mention the fact that $file will be an SplFileInfo class, so you can do powerful stuff really easily:

```
<?php

$size = 0;
foreach ($iterator as $file) {
    if ($file->isFile()) {
        echo substr($file->getPathname(), 27) . ": " . $file->getSize() . " B; modified " . date("Y-
m-d", $file->getMTime()) . "\n";
        $size += $file->getSize();
    }
}

echo "\nTotal file size: ", $size, " bytes\n";

?>
```

```
\Luna\luna.msstyles: 4190352 B; modified 2008-04-13
\Luna\Shell\Homestead\shellstyle.dll: 362496 B; modified 2006-02-28
\Luna\Shell\Metallic\shellstyle.dll: 362496 B; modified 2006-02-28
\Luna\Shell\NormalColor\shellstyle.dll: 361472 B; modified 2006-02-28
\Luna.theme: 1222 B; modified 2006-02-28
\Windows Classic.theme: 3025 B; modified 2006-02-28

Total file size: 5281063 bytes
```

up

down
33
redcube at gmx dot de ¶
16 years ago
```
Please note that glob('*') ignores all 'hidden' files by default. This means it does not return files
that start with a dot (e.g. ".file").
If you want to match those files too, you can use "{,.}*" as the pattern with the GLOB_BRACE flag.

<?php
// Search for all files that match .* or *
$files = glob('{,.}*', GLOB_BRACE);
?>

Note: This also returns the directory special entries . and ..
```
up
down
25
Ultimater at gmail dot com ¶
11 years ago
```
glob() isn't limited to one directory:

<?php
$results=glob("{includes/*.php,core/*.php}",GLOB_BRACE);
echo '<pre>',print_r($results,true),'</pre>';
?>

Just be careful when using GLOB_BRACE regarding spaces around the comma:
{includes/*.php,core/*.php} works as expected, but
{includes/*.php, core/*.php} with a leading space, will only match the former as expected but not the
latter
unless you have a directory named " core" on your machine with a leading space.
PHP can create such directories quite easily like so:
mkdir(" core");
```
up
down
5
Anonymous ¶
1 year ago
```
Include dotfiles excluding . and .. special dirs with .[!.]*

<?php
$all_files = array_merge(glob('.[!.]*'), glob('*'));
// or
$all_files = glob('{.[!.],}*', GLOB_BRACE);
?>
```
up
down
25
ni dot pineau at gmail dot com ¶
10 years ago
```
Note that in case you are using braces with glob you might retrieve duplicated entries for files that
matche more than one item :

<?php
```

```
$a = glob('/path/*{foo,bar}.dat',GLOB_BRACE);
print_r($a);

?>

Result :
Array
(
    [0] => /path/file_foo.dat
    [1] => /path/file_foobar.dat
    [2] => /path/file_foobar.dat
)
```

up
down
32
Sam Bryan ¶
10 years ago

```
glob is case sensitive, even on Windows systems.

It does support character classes though, so a case insensitive version of
<?php glob('my/dir/*.csv') ?>

could be written as
<?php glob('my/dir/*.[cC][sS][vV]') ?>
```

up
down
17
david dot schueler at tel-billig dot de ¶
11 years ago

```
Don't use glob() if you try to list files in a directory where very much files are stored (>100.000).
You get an "Allowed memory size of XYZ bytes exhausted ..." error.
You may try to increase the memory_limit variable in php.ini. Mine has 128MB set and the script will
still reach this limit while glob()ing over 500.000 files.

The more stable way is to use readdir() on very large numbers of files:
<?php
// code snippet
if ($handle = opendir($path)) {
    while (false !== ($file = readdir($handle))) {
        // do something with the file
        // note that '.' and '..' is returned even
    }
    closedir($handle);
}
?>
```

up
down
3
simon at paragi dot dk ¶
6 years ago

```
This is a simple and versatile function that returns an array tree of files, matching wildcards:

<?php
```

```
// List files in tree, matching wildcards * and ?
function tree($path){
  static $match;

  // Find the real directory part of the path, and set the match parameter
  $last=strrpos($path,"/");
  if(!is_dir($path)){
    $match=substr($path,$last);
    while(!is_dir($path=substr($path,0,$last)) && $last!==false)
      $last=strrpos($path,"/",-1);
  }
  if(empty($match)) $match="/*";
  if(!$path=realpath($path)) return;

  // List files
  foreach(glob($path.$match) as $file){
    $list[]=substr($file,strrpos($file,"/")+1);
  }

  // Process sub directories
  foreach(glob("$path/*", GLOB_ONLYDIR) as $dir){
    $list[substr($dir,strrpos($dir,"/",-1)+1)]=tree($dir);
  }

  return @$list;
}
?>
```

up
down
10
r dot hartung at roberthartung dot de ¶
13 years ago

```
You can use multiple asterisks with the glob() - function.

Example:

<?php
  $paths = glob('my/*/dir/*.php');
?>

$paths will contains paths as following examples:

- my/1/dir/xyz.php
- my/bar/dir/bar.php
- my/bar/dir/foo.php
```

up
down
3
heavyraptor at gmail dot com ¶
14 years ago

```
glob() (array_sum() and array_map() in fact too) can be very useful if you want to calculate the sum
of all the files' sizes located in a directory:

<?php
```

```
$bytes = array_sum(array_map('filesize',glob('*')));
?>

Unfortunately there's no way to do this recursively, using glob() (as far as I know).
```

up
down
4
nuntius ¶
13 years ago

```
First off, it's nice to see all of the different takes on this. Thanks for all of the great examples.

Fascinated by the foreach usage I was curious how it might work with a for loop. I found that glob
was well suited for this, especially compared to opendir.  The for loop is always efficient when you
want to protect against a potential endless loop.

$dir=$_SERVER['DOCUMENT_ROOT']."/test/directory_listing/test";
    echo $dir;
    $filesArray=glob($dir."/*.*");

    $line.="<pre>";
    $line.=print_r($filesArray, true);
    $line.="</pre>";
    $line.="<hr>";

    for($i=0;$i<count($filesArray);$i++) {
        $line.=key($filesArray)." - ";
        $line.=$filesArray[$i]."<br/>";
         next($filesArray);
    }

    echo $line;

Note that I pulled the glob array keys if you should need them.

Also you can tweak it for searches with something like this... (case sensitive)

$search_names="Somedocname";
$filesArray=glob($dir."/".$search_names."*.*");

Enjoy!
```

up
down
9
carlos dot lage at nospam at foo dot bar at gmail dot com ¶
14 years ago

```
I lost hours looking for the solution for this problem.
glob() wasn't eating up my directory names (stuff like "foobar[]"), and I searched online for some
hours, I tried preg_quote to no avail.

I finally found the proper way to escape stuff in glob() in an obscure Python mailing list:

<?php
preg_replace('/(\*|\?|\[)/', '[$1]', $dir_path);
?>
```

```
If you want to add a directory path before your pattern, you should do it like this:

<?php
glob(preg_replace('/(\*|\?|\[)/', '[$1]', $dir_path).'*.txt');
?>
preg_quote WILL NOT work in all cases (if any).
```

up
down
4
eric at muyser dot com ¶
14 years ago

```
As a follow up to recursively determining all paths (by viajy at yoyo dot org) and opendir being
faster than glob (by Sam Yong - hellclanner at live [dot] com).

The list all dirs code didn't seem to work, at least on my server (provided by parazuce [at] gmail
[dot] com).

I needed a function to create an unlimited multidimensional array, with the names of the
folders/files intact (no realpath's, although that is easily possible). This is so I can simply loop
through the array, create an expandable link on the folder name, with all the files inside it.

This is the correct way to recurse I believe (no static, return small arrays to build up the
multidimensional array), and includes a check for files/folders beginning with dots.

// may need modifications

function list_files($path)
{
    $files = array();

    if(is_dir($path))
    {
        if($handle = opendir($path))
        {
            while(($name = readdir($handle)) !== false)
            {
                if(!preg_match("#^\.#", $name))
                if(is_dir($path . "/" . $name))
                {
                    $files[$name] = list_files($path . "/" . $name);
                }
                else
                {
                    $files[] = $name;
                }
            }

            closedir($handle);
        }
    }

    return $files;
}
```