

Pathname expansion (globbing)

General

Unlike on other platforms you may have seen, on UNIX®, the shell is responsible for interpreting and expanding globs ("filename wildcards"). A called program will never see the glob itself; it will only see the expanded filenames as its arguments (here, all filenames matching `*.log`):

```
grep "changes:" *.log
```

The base syntax for the pathname expansion is the pattern matching syntax. The pattern you describe is matched against all existing filenames and the matching ones are substituted. Since this substitution happens **after word splitting**, all resulting filenames are literal and treated as separate words, no matter how many spaces or other IFS - characters they contain.

Normal behaviour

- with the set command (`-f` , `noglob`) you can entirely disable pathname expansion
- when matching a pathname, the slash-character (`/`) always needs to be matched explicitly
- the dot at the beginning of a filename must be matched explicitly (also one following a `/` in the glob)
- a glob that doesn't match a filename is unchanged and remains what it is

Customization

- when the shell option `nullglob` is set, non-matching globs are removed, rather than preserved
- when the shell option `failglob` is set, non-matching globs produce an error message and the current command is not executed
- when the shell option `nocaseglob` is set, the match is performed case-insensitive
- when the shell option `dotglob` is set, wildcard-characters can match a dot at the beginning of a filename
- when the shell option `dirspell` is set, Bash performs spelling corrections when matching directory names
- when the shell option `globstar` is set, the glob `**` will recursively match all files and directories. This glob isn't "configurable", i.e. you **can't** do something like `** .c`

to recursively get all `*.c` filenames.

- when the shell option `globasciiranges` is set, the bracket-range globs (e.g. `[A-Z]`) use C locale order rather than the configured locale's order (i.e. `ABC...abc...` instead of e.g. `AaBbCc...`) - since 4.3-alpha
- the variable `GLOBIGNORE` can be set to a colon-separated list of patterns to be removed from the list before it is returned

nullglob

Normally, when no glob specified matches an existing filename, no pathname expansion is performed, and the globs are **not** removed:

```
$ echo "Textfiles here:" *.txt
Textfiles here: *.txt
```

In this example, no files matched the pattern, so the glob was left intact (a literal asterisk, followed by dot-txt).

This can be very annoying, for example when you drive a for-loop using the pathname expansion:

```
for filename in *.txt; do
  echo "=== BEGIN: $filename ==="
  cat "$filename"
  echo "=== END: $filename ==="
done
```

When no file name matches the glob, the loop will not only output stupid text (" BEGIN: *.txt "), but also will make the `cat` -command fail with an error, since no file named `*.txt` exists.

Now, when the shell option `nullglob` is set, Bash will remove the entire glob from the command line. In case of the for-loop here, not even one iteration will be done. It just won't run.

So in our first example:

```
$ shopt -s nullglob
$ echo "Textfiles here:" *.txt
Textfiles here:
```

and the glob is gone.

Glob characters

- `*` - means 'match any number of characters'. `/` is not matched (and depending on your settings, things like `.` may or may not be matched, see above)
- `?` - means 'match any single character'
- `[abc]` - match any of the characters listed. This syntax also supports ranges, like `[0-9]`

For example, to match something beginning with either 'S' or 'K' followed by two numbers, followed by at least 3 more characters:

```
[SK][0-9][0-9]???
```

See also

- Introduction to expansion and substitution
- pattern matching syntax
- the set builtin command
- the shopt builtin command
- list of shell options



Discussion

Altair IV, [2011/12/27 16:12 \(\)](#), [2011/12/28 00:55 \(\)](#)

*globstar isn't "configurable", i.e. you can't do something like `**.*` to recursively get all `*.*` filenames.*

The globstar itself isn't configurable, but you can use it in conjunction with other globs to do this.

```
echo ./**/*.c
```

 [syntax/expansion/globs.txt](#)  Last modified: 2013/04/14 12:34 by thebonsai

This site is supported by Performing Databases - your experts for database administration

Bash Hackers Wiki



Except where otherwise noted, content on this wiki is licensed under the following license:
GNU Free Documentation License 1.3

