Globbing

> *How to match paths using 'glob' patterns*

Globbing, also known as wildcard matching, is a technique used in computer systems and programming languages to match one or more files or paths based on a pattern containing wildcard characters. It is commonly used in command-line interfaces, file managers, and programming languages to perform operations on multiple files or directories at once.

Glob patterns involve the use of wildcards and special characters to match and then filter. They patterns are similar to regex patterns, but simpler and limited in scope.

Note, this guide is based on Bash and ZSH usage.

Resources

- GNU/Linux Command-Line Tools Summary: Wildcards
- Bash: globbing
- Wikipedia: glob (programming)
- Linux Programmer's Manual: GLOB(7)
- Globs Bash guide.

Some links copied from begin/globbing

Use cases

- **Command-line interfaces**: In Unix-like operating systems (e.g., Linux, macOS), globbing is used in the shell to perform operations on multiple files or directories matching a pattern. Example: `rm *.txt` removes all files with the `.txt` extension in the current directory.
- **File managers**: Many graphical file managers support globbing to filter and select files based on patterns.
- **Programming languages**: Many programming languages, such as Python, JavaScript, Ruby, and Bash, provide built-in support for globbing or have libraries that implement globbing functionality. Example (Python): `import glob; files = glob.glob('*.py')` retrieves a list of all Python files in the current directory.
- **Regular expressions**: While globbing and regular expressions serve different purposes, some programming languages and tools allow the use of globbing patterns within regular expressions for

more advanced pattern matching.

Common wildcard characters

Based on a Globbing cheatsheet

Wildcard matches are used alongside literals.

Basic wildcards:

- `*` (asterisk) - Match any character zero or more times.
- `**` (double asterisk) - Match any character zero or more times, including `/` unlike the others.
- `?` (question mark) - Match any single character exactly one time.
- `[...]` (character range). Match one of any of the characters. e.g. `[abc]`, `[123]`
- `[^...]` (negated character range): Matches any single character not within the specified range or set.

Basic examples

- `*.txt` matches all files with the `.txt` extension.
- `file?.txt` matches `file1.txt`, `file2.txt`, but not `file10.txt`.
- `file[0-9].txt` matches `file0.txt`, `file1.txt`, ..., `file9.txt`.
- `[0-9a-z]` will match 0 to 9 and a to z (lowercase only).
- `[\w]` will match any word characters (alphanumeric).
- `file[^0-9].txt` matches `fileA.txt`, `file_.txt`, but not `file1.txt`.
- `*/*` - will match `foo/bar`.
- `*` will match all files and directories in the current directory.

```
echo *

ls *

for P in *; do echo $P; done
```

Advanced

Wildcard expansion can be done. This can be previewed with `echo` but might be used with `ls`.

```
echo foo/{bar,baz}
# => foo/bar foo/baz
```

POSIX character classes

You can also use named ranges, such as:

- `[::alpha::]` for any letter.
- `[::alnum::]` for any letter or number.
- `[::lower::]` for any lowercase character.

e.g.

Match `a1` but not `aa` with:

`[[:alpha:][:digit:]]`

Match hidden characters

Bash 4

```
shopt -s dotglob
```

ZSH

```
setopt dotglob
```
