

Wildcards

What is a Wildcard?

A wildcard is a symbol used to replace or represent one or more characters in a file name.

There are 3 wildcards: *, ? and [set].

* wildcard

The asterisk (*) matches **zero or more characters** in a filename.

- Examples:
 - List all the files in a given directory:
 - `ls Downloads/*`
 - List all text files in a given directory:
 - `ls Downloads/*.txt`
 - List all the text files in a given directory that start with the letter f
 - `ls Downloads/f*.txt`
- Example breakdown:

When working with wildcards, keep in mind two things:

- **What do the files have in common?**
 - This is the part of the file name which is the same in all the files you want to match. For example, the file extension.
- **What part of the file name is irrelevant?**
 - This is the part of the file name that is irrelevant in the matching. In our example, this is the file name

Let's assume that you want to list only the **css** files inside this directory in a single column.

The command for achieving this would be `ls -l *.css`

The *** wildcard** replaces the name of the files because we do not care about the file name in this instance. In the command, we keep the extension because that is what the files must have in common.

```

1/1 + [ ] [ ]
1: Terminal
~/website via [ ]
→ ls -X1
assets
docs
scripts
main.css
media.css
reset.css
style.css
index.html
script.js
background.png
~/website via [ ]
→ ls -X1 *.css
main.css
media.css
reset.css
style.css
~/website via [ ]
→
  
```



5

? wildcard

The **? wildcard** metacharacter matches precisely one character. This wildcard is very useful when working with hidden files (also called dot files).

- Examples:
 - List all the hidden files in the current working directory

- `ls ./.*?*`
- List all hidden files in the parent directory
 - `ls ../. ??*`
- List all the files that have 2 characters in the file name between letters b and k
 - `ls b??k*`
- List all the files that have a single character between letters f and l.
 - `ls f?l*`
- List all the files with a 2 letter file extension
 - `ls *.*??`

- Example breakdown:

Lets Breakdown this command

ls -lX

This ls command will list the files in a single column sorted by file extension

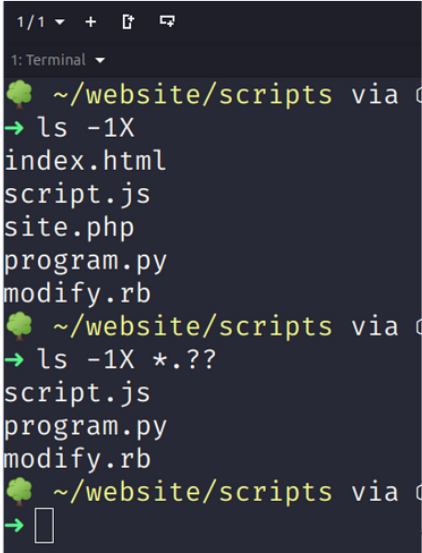
.??

The * will match any filename (any number of characters). The period is static as it is part of the file extension. The ?? indicates that the files must have only 2 characters in the file extension.

<

11

>



❑ wildcard

The brackets wildcard matches a single character in a range. The brackets wildcard use the exclamation mark to reverse the match.

- Examples:
 - To match all files that have a vowel after letter f:
 - `ls f[aeiou]`
 - To match all files that do not have a vowel after letter f:
 - `ls f[!aeiou]*`
 - To match all files that have a range of letters after f:
 - `ls f[a-z]`

- More Examples:

More Examples

- To match all files whose name has at least one number:
 - `ls *[0-9]*`
- To match all the files whose name does not have a number in their file name:
 - `ls *![0-9].*`
- To match all files whose name begins with a letter from a-p or start with letters s or c:
 - `ls [a-psc]*`
- To match all files whose name begins with any of these two sets of characters: letters from a-f or p-z:
 - `ls [a-fp-z]*`
- To match all files whose name begins with any 3 combination of numbers and the current user's username:
 - `ls [0-9][0-9][0-9]$USER`

1
5

Brace expansion

Brace expansion is a feature of the bash shell that generates argument strings. Those strings can be used by commands to operate on files. This feature does not make calls to the operating system like wildcards, they just generate file names based on a given pattern.

- **How to use:**

- Start with an open brace
- With no spaces, type your string separating entries by command
- Close the brace
 - Example:
 - `mkdir -pv example_site{assets/large,docs/share,scripts/js}`

- More examples:

- Create 3 different files with the same name but different file extensions
 - `touch file.{md,txt,rtf}`
- Create 10 files in a range from 0 to 9:
 - `touch file {0..9}.txt`
- Remove specific files that start with a given keyword
 - `rm image_{01..08}*camera.{png,jpg}`
- Create an entire directory tree in a single command (1 level deep)
 - `mkdir -pv project_saturn/{code,source,database}/new`
- Create an entire directory tree in a single command (2 levels deep)
 - `mkdir -pv`
 - `project_jupiter/site{old,new}/{code/{scripts/markup},assets/{imgs,mp3,mp4}}`

- Example breakdown:

More on Brace Expansion

How exactly can I properly write such a complicated string??!


```
mkdir -pv project_jupiter/site/{old,new}/{code/{scripts,markup},assets/{imgs,mp3,mp4}}
```

Start by identifying the top level directories
In this case we want to create "old" and "new" and inside each we want to create "code" and "assets"


```
project_jupiter/site/{old,new}/{code,assets}
```

Now that we have the general structure, we can just expand the sub directories.
So for "code", I need to subdirectories. {scripts,markup} Then I repeat the process for assets

```
project_jupiter/site/{old,new}/{code/{scripts,markup},assets/{imgs,mp3,mp4}}
```



23



2

3