Notes 6

Using Wildcards/ File Globbing

Description;

- Wildcards or file globing is a shell feature that, using special characters, allows us to rapidly specify groups of filenames. Because we work with files all the time, it is useful to be able to work with multiple files at the same time.

Why and when should I use a wildcard?

Using wildcards can save you a lot of time. Imagine you need to move all the text files in a folder. You could move them one by one—if you know each file's name. But what if there are 300 files? Moving them one at a time would take forever! Wildcards help you by allowing you to match all files that meet a certain pattern or criteria, so you can move them all at once with a single command.

Here are some examples:

- Copy all photos from a drive to a flash drive.
- Delete certain types of files from a folder.
- List all files that have a date in their name.

Wildcard definition example:

- (*) matches 0 to any number of characters ls ~/Downloads/*.png
- ? matches 1 character ls ~/Downloads/f?ll.sh
- [] matches 1 character from a set ls ~/Downloads/f[0-9]ll.sh

---

The * (star/asterisk) wildcard:

- Examples:
  - list all of the directories inside a given directory without listing their content
    - ls -1d wildcard_extra_practice/*/
  - move all the .sh (shell scripts) files
    - mv wildcard_extra_practice/*.sh wildcard_extra_practice/scripts/
  - list and then move all the image files in the wildcard_extra_practice directory. You will need to create a directory first.
    - mkdir wildcard_extra_practice/images/
    - ls wildcard_extra_practice/img-.
    - mv -v wildcard_extra_practice/img-*.*images/

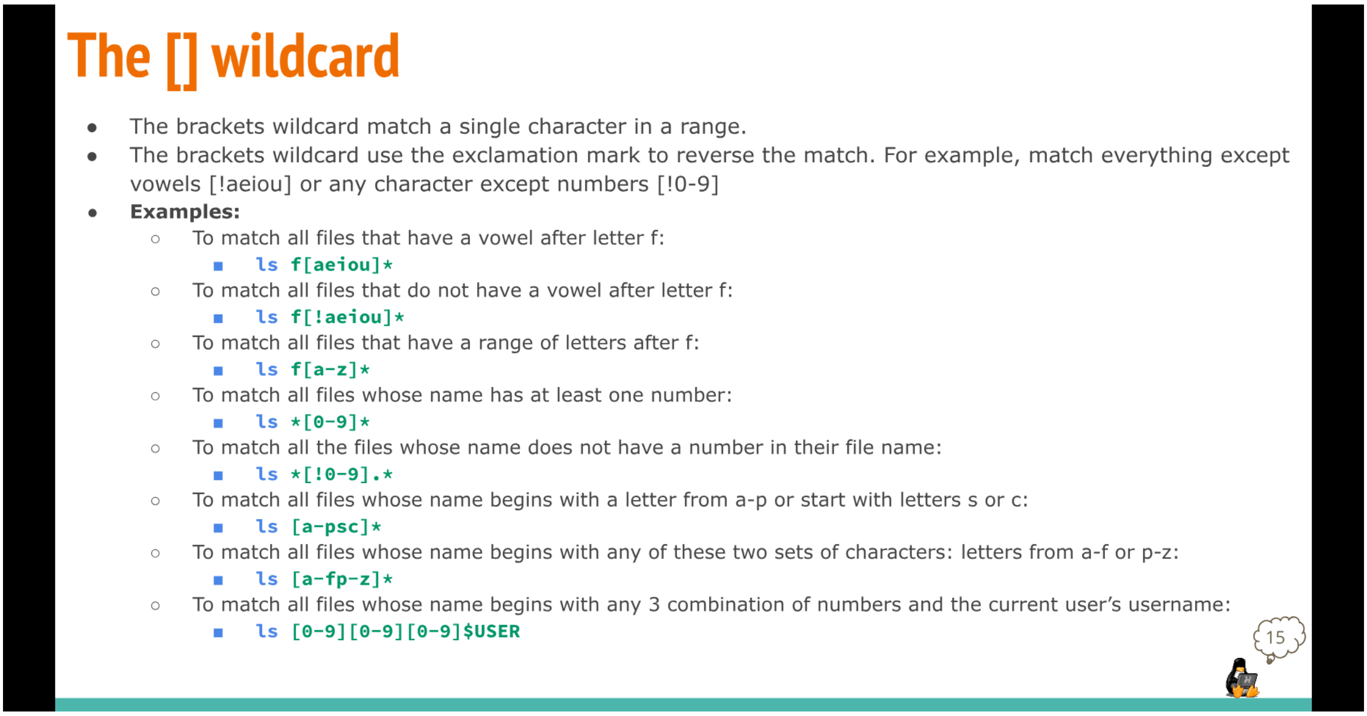
The ? (question mark) wildcard:

- Examples:
  - List all the files that contain a 4 letter file extension.
    - ls -1X wildcard_extra_practice/*.????
  - List all the files that contain a 4 letter file extension and start with letter i:
    - ls -1X wildcard_extra_practice/i*.????
  - List all th files Microsoft Office 365 files.
    - ls -1X wildcard_extra_practice/*.???x

The [] (square brackets) wildcard:

- Examples:
  - List all the files that start with a capital letter
    - ls wildcard_extra_practice/[A-Z]*
  - List all the files that contain a number in their name
    - `ls wildcard_extra_practice/[0-9]

POSIX class	Represents	Means	Example using `ls`
`[:upper:]`	`[A-Z]`	Uppercase letters	`ls *[[:upper:]]*` - List files with uppercase letters
`[:lower:]`	`[a-z]`	Lowercase letters	`ls *[[:lower:]]*` - List files with lowercase letters
`[:digit:]`	`[0-9]`	Digits	`ls *[[:digit:]]*` - List files containing digits
`[:alpha:]`	`[A-Za-z]`	Alphabetic characters	`ls *[[:alpha:]]*` - List files with alphabetic characters
`[:alnum:]`	`[A-Za-z0-9]`	Alphanumeric characters	`ls *[[:alnum:]]*` - List files with alphanumeric characters
`[:space:]`	`[ \t\n\r\f\v]`	Whitespace characters	`ls *[[:space:]]*` - List files with spaces in names
`[:punct:]`	Punctuation	Punctuation characters	`ls *[[:punct:]]*` - List files with punctuation characters
`[:blank:]`	`[ \t]`	Space and tab	`ls *[[:blank:]]*` - List files with spaces or tabs
`[:xdigit:]`	`[0-9A-Fa-f]`	Hexadecimal digits	`ls *[[:xdigit:]]*` - List files with hexadecimal characters
`[:cntrl:]`	Control characters	Control characters	`ls *[[:cntrl:]]*` - List files with control characters
`[:print:]`	Printable	Printable characters	`ls *[[:print:]]*` - List files with printable characters
`[:graph:]`	Graphical	Visible characters (not spaces)	`ls *[[:graph:]]*` - List files with graphical characters
`[:word:]`	`[A-Za-z0-9_]`	Word characters (alphanumeric + underscore)	`ls *[[:word:]]*` - List files containing word characters
`[:ascii:]`	ASCII characters	All ASCII characters (0-127)	`ls *[[:ascii:]]*` - List files containing ASCII characters



The [] wildcard

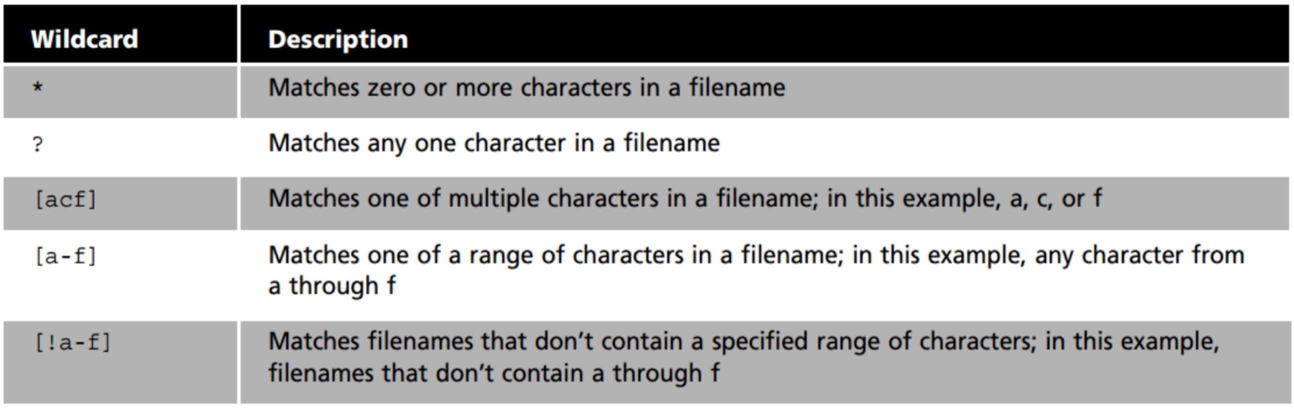
- The brackets wildcard match a single character in a range.
- The brackets wildcard use the exclamation mark to reverse the match. For example, match everything except vowels [!aeiou] or any character except numbers [!0-9]
- **Examples:**
  - To match all files that have a vowel after letter f:
    - `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]*`
  - 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`

Note:

- Wildcards also have their limitations.
  - For instance, "List all the files that do not contain a digit in the file name" You may assume that the approach would be ls *[!0-9]*. However, this does not work. This wildcard translates to: "all files and directories in the current directory whose names contain at least one character that is not a digit (0-9)" However, if you ever need this, here are two approaches:
    - Use ls + grep:
      - ls *[!0-9]* | grep -v '[0-9]'
    - Use find:
    - find ./ -not -name '*[0-9]*'
  - Find is preferred for scripts because ls is not suited for shell scripts. ls is intended to be a human friendly tool and the output is difficult to parse properly.

Extra Screenshots

Using Wildcards / File Globbing (quick reference)



Wildcard	Description
*	Matches zero or more characters in a filename
?	Matches any one character in a filename
[acf]	Matches one of multiple characters in a filename; in this example, a, c, or f
[a-f]	Matches one of a range of characters in a filename; in this example, any character from a through f
[!a-f]	Matches filenames that don't contain a specified range of characters; in this example, filenames that don't contain a through f

Using Brace Expansion

- Brace expansion {} is not a wildcard but another feature of bash that allows you to generate arbitrary strings to use with commands.
- For example,
  - To create a whole directory structure in a single command:
    - `mkdir -p music/{jazz,rock}/{mp3files,vidoes,oggfiles}/new{1..3}`
  - To create a N number of files use:
    - `touch website{1..5}.html`
    - `touch file{A..Z}.txt`
    - `touch file{001..10}.py`
    - `touch file{{a..z},{0..10}}.js`
  - Remove multiple files in a single directory
    - `rm -r {dir1,dir2,dir3,file.txt,file.py}`