13. Expansion

commands table

commands	application
`ls *.html`	match any files that end with `.html`
`?`	character represents any single character.
`[]`	can specify a range of characters to match.
`^`	specify a range of characters to NOT match1. What actually matters in this section
`{}`	to generate arbitrary strings. It can generate multiple strings based on a pattern.
`$((expression))`	perform arithmetic via expansion

- no new commands will be learned but we will learn a few things that shell

02. Pathname Expansion Basics

echo

- It's particularly useful in shell scripts when we want to output something to the screen from within a file

wildcard characters

- we can use special wildcard patterns that can match multiple filenames at once.
- wildcard characters can be used with multiple commands
- the asterisk `*` character represents zero or more characters in a filename
    - example:
        1. `ls *.html` will match any files that end with `.html` like `index.html` and `navbar.html`
        2. `cat blue*` will match any files that start with `blue` like `blue.html` or `bluesteel.js`

03. More pathname Expansion

The `?` wildcard

- the question mark `?` the character represents any single character.
    - example:
        1. `ls app.??` will match any files named `app` that end with two character file extensions like `app.js` or `app.py` but not `app.css`

2. `ls pic?.png` will match `pic1.png`, `pic2.png` or anything just like that pattern

The `[]` wildcard (Range Wildcards)

- the square brackets `[]` can specify a range of characters to match.
    - example:
        1. `ls pic[123].png` will only match `pic1.png, pic2.png, and pic3.png`
        2. `ls file[0-9]` will match `file1`, `file2` up to `file9`
        3. `ls [A-F]*` will match any files that begin with a `A, B, C, D, E, F`

The `^` wildcard (Negating Ranges)

- Inside of square brackets, we can specify a range of characters to NOT match, using a caret `^`
- example:
    1. `ls [^a]` will match any files that do NOT start with `a`
    2. `ls [^aApP]` will match any files that do NOT start with `a, A, p, P`
    3. `ls [^0-9]` will match any files that do NOT start with a numeric digit `0-9`

04. Tilde Expansion

- if we use `~username` if the user exists then move to that user's home directory

05. The magic of Brace Expansion

- Brace expansion is used to generate arbitrary strings. It can generate multiple strings based on a pattern.
- We provide a set of strings inside of curly braces `{}` as well as optional surrounding prefixes and suffixes.
- The specified strings are used to generate all possible combinations with the optional prefixes and suffixes.
- example:
    1. `touch page{1,2,3}.txt` will generate 3 new files: `page1.txt, page2.txt, page3.txt`
- we can also nest the braces
- example:
    1. `echo {x,y{1..5},z}` will print `x, y1, y2, y3, y4, y5, z`

06. Arithmetic Expansion

- The shell will perform arithmetic via expansion using the `$((expression))` syntax
- `note:` calculations are based upon the integer values only
- inside the parenthesis, we can with arithmetic expressions using
    1. addition `+`
    2. subtraction `-`
    3. multiplication `*`
    4. division `/`
    5. exponentiation `**`
    6. modulo or remainder operator `%`

- example:
    1. `echo $((10+10))` will give `20`

07. Quoting Double vs Single

- while running an `echo` command, even though if we part multiple spaces, it will split them into single while giving output. We need to use quotes to prevent spaces like those
- example:
    1. `echo hello there how are you?` will give `hello there how are you?`
- also if we use `$` it will consider the word attached as a variable instead of a word to print on the screen. If no such variable is found then it will render blank
- example:
    1. `echo hello $sup` will most likely give `hello`

Double quote vs single quote

- The double quote will ignore spacing, except for dollar sign `$`, backslash `\` , and backtick `` ` ``
- Pathname expansion, brace expansion, and word splitting will be ignored. However, command substitution and arithmetic expansion is still performed because dollar sign still have meaning inside double quotes which is for variable
- single quotes suppress all forms of substitutions and works including for dollar sign `$`, backslash `\` , and backtick `` ` ``

08. Command Substitution

- we can use the `$(command)` syntax to display the output of another command.
- `ntoe`: it will not replace the command output but actually run the command on the go
- example:
    1. `echo today is $(date)` will give `today is <output_of_date_command>`