14. Finding Things

table of commands
01. What actually matters
02. The locate command
    options with locate
03. The find command
04. More Find
    size
05. How Timestamps work
06 . Find by Time
07. Logical Operators
08. Find w_Exec & User Defined Actions
09. xargs

table of commands

commands	applications
`locate <keyword>`	look for file name containing the passed keyword
`locate -e <keyword>`	look only existing files only for filenames containing the passed keyword
`locate -i <keyword>`	look for file name containing the passed keyword ignoring the casing
`locate -l<numbers> <keyword>`	will limit the number of entries that locate retrieves
`find <directory>`	print all the files and directories inside the passed directory including nested directories
`find -type f`	will limit the search to files
`find -type d`	will limit the search to directories
`find <path> -name "<pattern>"`	will provide specific patterns to find to use matching file names and directories
`find -empty`	will list down all empty files and folders

`<directory>`	
`find -size 20k`	to find files of size exactly 20kilobytes
`find -size -50M`	to find all files smaller than 1 megabyte
`find -size +1G`	to find all files larger than 1 gigabyte
`find -type -f -not -name "*.html"`	will print files not ending with `.html`

01. What actually matters

- `important`: find command basics
- `useful`: locate command, understanding timestamps, finding by time, find w/exec xargs command

02. The locate command

- `locate` command performs a search of pathnames across our machine that match a given substring and then prints out any matching names.
- example: `locate chick` will perform a search for all files that contain `chick` in their name
- It works for the whole system since it uses pre-loaded databases. Hence, the path is not relative

options with locate

- `-e` option will only print entries that actually exist at the time locate is run. *(there could be cases when files are modified and the entries may be is same)*
- `-i` option tells locate to ignore casing
- `-l<number> or --limi<number>` will limit the number of entries that locate retrieves

- `sudo updatedb` will update the database for the locate command manually. It updates by self after certain times.

03. The find command

- `find` doesn't use a database file to locate the files
- By default, find on its own will list every single file and directory nested in our current working directory.
- `find <directory>` will print all the files and directories inside the passed directory including nested directories
- `find -type f` will limit the search to files
- `find -type d` will limit the search to directories
- `find <path> -name "<pattern>"` will provide specific pattern to find to use matching file names and directories
    - example: `find ~ -type f -name "*.txt"` will print all files inside the home directory of type file and each file's name is ending with `.txt` extension

04. More Find

size

- `-size` can be used to find files according to the file sizes
- `-empty` will list down all empty files and folders
- example:
    1. `find -size +1G`: to find all files larger than 1 gigabyte
    2. `find -size -50M`: to find all files smaller than 1 megabyte
    3. `find -size 20k`: to find files of size exactly 20kilobytes

05. How Timestamps work

- Timestamps have mainly 3 parameters to keep track of files based on the time they are
    1. `mtime`: for modification time, is when a file was last modified
        - example: `ls -l` uses last modified time by default
    2. `ctime`: for change time, is when a file was last changed. It occurs anytime `mtime` changes but also when we rename a file, move it, or alter permissions.
        - example: `ls -lc` gives the last changed time
    3. `atime`: for access time, is updated when a file is read by an application or command like cat
        - example: `ls -lu` gives last accessed time

06 . Find by Time

- we can use various options with timestamps in order to change the dates modifications of the files.
- `-d` option is used to manage the date of respective files
- example:
    1. `touch two_days_ago -d "2 days ago"`
    2. `touch two_months_ago -d "2 months ago"`
    3. `touch 30_mins_ago -d "30 mins ago"`
    4. `touch right_now`
- using timestamps to find files
- example:
    1. `find -mmin +30`: finds files modified greater than 30 mins ago

2. `find -amin -30`: finds files accessed less than 30 mins ago
3. `find -cmin +20`: finds files changed more than 20 mins ago.
4. `find -mtime -5`: finds files modified less than 5 days ago *(it works as 5*24 hours)*

07. Logical Operators

- we can use `-and`, `-or`, `-not` operators to create more complex queries.
- example:
  1. `find -name "*chick*" -or -name "*kitty*"`: will print files containing either `chick` or `kitty` on it's name
  2. `find -type -f -not -name "*.html"`: will print files not ending with `.html`
  3. `find -cmin -60 -not -name "*.log"`: will print files which were changed less than 60 mins ago and doesn't end with `.log` extension

08. Find w_Exec & User Defined Actions

- we can provide `find` with our own action to perform using each matching pathname.
- syntax: `find -exec command {};`
  - `{}` are placeholder for the current pathname*(each match),* and each semicolon is required to indicate the end of the command
- example:
  1. `find -name "*broken*" -exec rm '{}' ';'`: deletes every file that contains `“broken”` in it's file name,

- note:

  `{}` is a kind of placeholder for each of the files matched with the condition

  `;` is used to denote the end of the command for the matched file.

09. xargs

- `xargs` build and execute command lines from standard input
- when we provide a command via `-exec`, that command is executed separately for every single element.
- we can use `xargs` to build up the input into a bundle that will be provided as an argument list to the next command
- example:

  `find -name "*.txt" -exec ls '{}' ';'` VS `find -name "*.txt" | xargs ls`