find

- it helps you search for files and directories in real time, based on name, size, type, time, permissions,and more.

  ```
  find [path] [conditions] [actions]
  ```

  - Examples:
    - find . -name file1.txt
      - Finds File named exactly file1.txt in current dir (.) and subdirs.
    - find . -type f
      - Finds all files (type f) in current dir and subdirs.
    - find . -type d
      - Finds all directories in the current directory and below.
    - find . -type l
      - Finds all symbolic links
    - find /home/user -name "*.log"
      - Finds all .log files in /home/user and subdirs.
    - Also find by size:
      - Size units:
        - k → kilobytes ,M → megabytes ,G → gigabytes
          - find . -size 4k
            - Finds files exactly 4 kilobytes in size.
          - find . -size +4k
            - Finds files larger than 4KB
          - find . -size -4k
            - Finds files smaller than 4KB

Command Nesting

1. cmd1 | cmd2
   - two commands are executed seperately
   - output of first command(cmd1) is given to the second command(cmd2)
2. cmd1 && cmd2
   - two commands are executed seperately
   - e.g. => mkdir newdir && cd newdir
     - if first command (cmd1) is failed, then second command (cmd2) is not executed
3. cmd1 || cmd2
   - two commands are executed seperately
   - if first command (cmd1) is successful, then second command (cmd2) is not executed
   - e.g. => mkdir mydir || echo "Already exists"
4. cmd1 ; cmd2
   - two commands are executed seperately

   - e.g => echo "Start";ls; echo "End"
   - here all commands are executed one by one regardless of success or failure of any command
5. command &
   - command is executed in background
   - also called as asynchronous execution because the shell prompt returns immediately for next command without waiting for current command to complete
   - e.g => sleep 30 &

Link

Hard Link

- A hard link is like a second name for a file.
- It points directly to the same inode as the original file.
- Changes to either file affect the same data

  ```
   ln file.txt hardlink.txt # create hard link
  ```

  - Both files now point to the same content.
  - Deleting one doesn’t delete the data as long as the other exists.
  - Hard links cannot be created for directories or across different filesystems

Symbolic link

- A symbolic link is like a shortcut or alias.
- It points to the filename, not the content.
- If the original is deleted, the symlink breaks.

  ```
  ln -s file.txt symlink.txt  # create symbolic link
  ```

  - Can link to files or directories
  - Points to the path (name), not the data .
  - it Breaks if the original file is deleted
  - symlinks creates for files and directories and across different filesystems.