1st lab. - FUNDAMENTAL COMMANDS FOR MANAGING FILES AND DIRECTORIES AND GETTING HELP IN A LINUX SYSTEM:

*This structured approach should make it easier for students to understand and experiment with these bash commands for file management, directory navigation, file type and timestamps, file copying, moving files, file removal, directory creation, directory removal, locating files using **locate** and **find**, and obtaining command location and information using **whereis**, **which**, and **type**.*

BASIC LINUX COMMANDS SUMMARY:

File and Directory Operations:
- Copy files and directories using **cp**.
- Identify file types with **file**.
- List files and directories with **ls**.
- Create directories using **mkdir**.
- Remove files with **rm** and empty directories with **rmdir**.
- Change file timestamps and create empty files with **touch**.

File and Text Searching:
- Search for files with **find**.
- Quickly locate files by name using **locate**.
- Search for text patterns in files using **grep**.

System and Directory Navigation:
- Display system information with **uname**.
- Find the current working directory using **pwd**.
- Navigate directories with **cd**.

Command Location and Information:
- Determine command interpretation and location using **type**.
- Find the executable path of a command with **which**.
- Locate command-related files and sources with **whereis**.

Accessing Command Documentation:
- Access detailed manual pages with **man**.
- Get built-in command help with **help**.
- Search for commands and descriptions using **apropos**.

File and Directory Operations (10 minutes)

1. **Copying Files and Directories (cp):**
   - **Explanation:** Introduce the cp command for copying files and directories.
   - **Examples:**
     - Copy a file: **cp source_file destination_file**
     - Example: **cp file1.txt file2.txt** - Copies file1.txt to file2.txt.
     - Copy a directory and its contents: **cp -r source_directory destination_directory**
     - Example: **cp -r folder1 folder2** - Recursively copies folder1 to folder2.
2. **Determining File Type (file):**
   - **Explanation:** Explain how the file command identifies the type of a file by analyzing its content.
   - **Examples:**

        - Determine file type: **file filename**
        - Example: **file image.png** - Identifies image.png as a PNG image file.

3. **Listing Files and Directories (ls):**
    - **Explanation:** The ls command lists information about files and directories in the current directory by default.
    - **Examples:**
        - List current directory: **ls**
        - Example: **ls** - Lists files and directories in the current directory.
        - List files in a specific directory: **ls /var/log**
        - Example: **ls /var/log** - Lists files and directories in the /var/log directory.
4. **Creating Directories (mkdir):**
    - **Explanation:** The mkdir command is used to create new directories.
    - **Examples:**
        - Create a directory: **mkdir directory_name**
        - Example: **mkdir new_folder** - Creates a directory named new_folder.
5. **Removing Files and Directories (rm and rmdir):**
    - **Explanation:**
        - **rm (Remove):** The rm command is used to remove (delete) files.
        - **rmdir (Remove Directory):** The rmdir command is used to remove empty directories.
    - **Examples:**
        - Remove a file: **rm filename**
        - Example: **rm old_file.txt** - Deletes the file old_file.txt.
        - Remove an empty directory: **rmdir empty_directory**
        - Example: **rmdir empty_folder** - Removes the empty directory empty_folder.
6. **Changing File Timestamp (touch):**
    - **Explanation:** The touch command allows you to update the access and modification times of a file to the current time and create new empty files.
    - **Examples:**
        - Update file timestamp: **touch filename**
        - Example: **touch myfile.txt** - Updates the timestamp of myfile.txt.
        - Create a new empty file: **touch newfile**
        - Example: **touch newfile** - Creates a new empty file named newfile.

<u>File and Text Searching (10 minutes)</u>

7. **Searching for Files (find):**

- **Explanation:** Introduce the find command for searching for files in a directory hierarchy based on various criteria.
- **Examples:**
    - Search for files based on criteria: **find starting_directory -options expression**
    - Example: **find /home/user/ -name document.txt** - Searches for document.txt in the /home/user/ directory.

8. **Searching for Files by Name (locate):**
    - **Explanation:** Introduce the locate command for fast file searching by name using a pre-built database.
    - **Examples:**
        - Locate files by name: **locate pattern**
        - Example: **locate image.png** - Quickly finds files named image.png on the system.
9. **Searching Text Using grep:**
    - **Explanation:** Explain how the grep command is used to search for text patterns in files.
    - **Examples:**
        - Search for lines starting with "while" in a file: **grep '^while' filelist**
        - Example: **grep '^while' myfile.txt** - Finds lines starting with "while."
        - Search for lines containing "case" or "esac" in a file: **grep 'case\|esac' filelist**
        - Example: **grep 'case\|esac' myfile.txt** - Finds lines containing either "case" or "esac."

<u>**System and Directory Navigation (10 minutes)**</u>

10. **Getting System Information (uname):** -
    - **Explanation:** The uname command is used to display system information about the operating system. –
    - **Examples:** -
        - Display system information: **uname** –
            - Example: **uname** - Shows basic information about the operating system. –
        - Display all system information: **uname -a** –
            - Example: **uname -a** - Provides detailed system information, including the kernel version, hostname, etc.
11. **Printing Working Directory (pwd):**
    - **Explanation:** The pwd command displays the current working directory's absolute path.
    - **Examples:**
        - Show the current directory path: **pwd**
        - Example: **pwd** - Shows the current directory's absolute path.
12. **Changing Directory (cd):**
    - **Explanation:** The cd command is used to change the current directory.
    - **Examples:**
        - Move up one directory: **cd ..**

        - Example: **cd ..** - Moves up one level in the directory hierarchy.
        - Move to the current directory (no change): **cd .**
        - Example: **cd .** - Stays in the same directory.
        - Change to a specific directory: **cd directory_name**
        - Example: **cd profesor** - Moves to the "profesor" directory.
        - Change to the root directory: **cd /**
        - Example: **cd /** - Moves to the root directory.
        - Change to the home directory: **cd ~**
        - Example: **cd ~** - Moves to the user's home directory.

<u>**Command Location and Information (5 minutes)**</u>

13. **Checking Command Interpretation (type):** -
    - **Explanation:** The type command indicates how a command is interpreted and where it's located. –
    - **Examples:** - Check the type and location of a command: **type command_name** –
        - Example: **type ls** - Shows that ls is an alias for ls --color=auto.
14. **Locating Command Location (which and whereis):**
    - **which:**
        - **Explanation:** The which command finds the path of the executable file for a given command.
        - **Examples:**
            - Find the path of a command: **which command_name**
            - Example: **which grep** - Displays the path to the grep executable.
    - **whereis:**
        - **Explanation:** The whereis command locates the binary, source, and manual pages for a given command.
        - **Examples:**
            - Locate command-related files: **whereis command_name**
            - Example: **whereis nano** - Shows the locations of the nano binary, source, and manual pages.

<u>**Additional Commands (5 minutes)**</u>

15. **Accessing Command Documentation (man, help, apropos):** -
    - **Explanation:**
        - - **man (Manual Pages):** The man command is used to access the manual pages for commands and programs, providing detailed documentation.
        - - **help:** The help command provides built-in help for shell commands and is specific to the shell you are using.
        - - **apropos:** The apropos command is used to search for commands and their descriptions. –
    - **Examples:** -
        - Access command manual: **man command_name** - Example: **man ls** - Opens the manual page for the ls command. –
        - Get built-in help: **help command_name** - Example: **help cd** - Provides help for the cd command. –

- Search for commands: **apropos keyword** - Example: **apropos text** - Searches for commands related to "text."

FILE GLOBBING (MAIN BASH COMMANDS AND WILDCARD CHARACTERS USED IN FILE GLOBBING)

<u>Introduction to File Globbing (5 minutes)</u>

- Explain the concept of file globbing and its importance in Linux.
- Mention that wildcards are used to match files and paths.
- Briefly introduce the asterisk (*) and question mark (?) wildcards.

<u>Displaying and Matching Files (10 minutes)</u>

1. **Display All Files in Current Directory:**
    - **Explanation:** Use the echo command with an asterisk * wildcard to display all files in the current working directory.
    - **Example: echo *** - Displays all files and directories in the current directory.
2. **View Files Starting with a Specific Letter:**
    - **Explanation:** Use the echo command with a letter followed by an asterisk * to view files and directories starting with that letter.
    - **Example: echo D*** - Displays files and directories starting with "D."
3. **Wildcard Position and Multiple Characters:**
    - **Explanation:** Use asterisk * at any position within the pattern and search for files containing specific letters.
    - **Example: echo D*n*** - Displays files and directories starting with "D" and containing "n."
4. **Case Sensitivity and Hidden Files:**
    - **Explanation:** Note about case sensitivity in Linux and hidden files (starting with a dot .).
    - **Example:** Mention that Dn and DN can produce different results.

<u>Advanced Wildcard Usage (15 minutes)</u>

5. **Wildcard for Exactly One Character:**
    - **Explanation:** Use the question mark ? wildcard to match exactly one character.
    - **Example: echo /usr/bin/??** - Lists files in /usr/bin with two-character names.
6. **Wildcard for Zero or More Characters:**
    - **Explanation:** Combining asterisk * (zero or more characters) and question mark ? (exactly one character) wildcards.
    - **Example: echo D?*** - Lists files starting with "D" and having at least one more character.

7. **Search for Specific Set of Characters:**

    - **Explanation:** Use square brackets [] to specify supported ASCII characters for matching.

    - **Example: echo [PMT]*** - Lists files and directories starting with "P," "M," or "T."

8. **Specify a Range of Characters:**

    - **Explanation:** Use square brackets [3-6] to specify a range of characters (e.g., 3 to 6).

    - **Example: echo /etc/rc[3-6]*** - Lists files in /etc/rc3.d, /etc/rc4.d, /etc/rc5.d, and /etc/rc6.d.

9. **Search for Characters Within a Range:**

    - **Explanation:** Use square brackets [A-D] to specify a range of characters (e.g., A to D).

    - **Example: echo [A-D]??*** - Lists files starting with characters from A to D and containing at least three characters.

<u>Complex Wildcard Patterns (5 minutes)</u>

10. **Search for Specific Characters in a Position:** -
    - **Explanation:** Use [p-u] to match characters from p to u in a specific position. –
    - **Example: echo M[p-u]???** - Lists files starting with "M," having characters from p to u in the 2nd position, and followed by exactly three more characters.

11. **Combining Multiple Wildcards:**

    - **Explanation:** Use multiple wildcards in a pattern.

    - **Example: echo /usr/bin/p*t*[2-6]** - Lists files in /usr/bin starting with "p," containing "t," and ending with numbers from 2 to 6.

12. **Wildcard Negation:**

    - **Explanation:** Use the ! or ^ character to negate a pattern.

    - **Example: echo /etc/rc[^3-6]*** or **echo /etc/rc[!3-6]*** - Lists files in /etc/rc excluding those containing numbers 3 to 6.

<u>Special Considerations (5 minutes)</u>

13. **Special Meaning Inside Square Brackets:** -
    - **Explanation:** Mention that wildcard characters lose their special meaning inside square brackets and are treated as regular characters. –
    - **Example:** Specifying **[*]*** will try to match filenames starting with the asterisk * character, which is rare.

<u>Conclusion (5 minutes)</u>

- Summarize key points and concepts covered during the session.

- Emphasize the practicality and flexibility of wildcard patterns in file globbing.

- Encourage students to practice and experiment with these wildcards on their own.

- Provide additional resources or references for further learning if needed.

FILE GLOBBING AND WILDCARD USAGE SUMMARY:

I. Basics of File Globbing

- File globbing in Linux is essential for matching files and directories using wildcards.
- The primary wildcards are the asterisk (*) and question mark (?).

II. Displaying and Matching Files

- Display all files in the current directory with **echo ***.
- View files starting with a specific letter using **echo [Letter]***.
- Use wildcards at different positions for advanced matching.
- Be aware of case sensitivity and hidden files.

III. Advanced Wildcard Usage

- Match files with exactly one character using **[?]**.
- Combine ***** and **?** for versatile matching.
- Search for files starting with specific characters with **[Characters]***.
- Specify a range of characters like **[3-6]***.
- Search for characters within a range with **[A-D]??***.

IV. Complex Wildcard Patterns

- Find files with specific characters in a particular position using **[Letter][p-u]???**.
- Experiment with combining multiple wildcards.
- Utilize negation with **!** or **^** to exclude specific files.

V. Special Considerations

- Be aware that wildcards inside square brackets lose their special meaning and are treated as regular characters.

VI. Conclusion

- File globbing and wildcards are powerful tools in Linux for effective file and path matching.