Linux File System Operations

1. Introduction to Linux File System

- **File**: Stores data, settings, or commands.
- **Directory**: A structure that stores files and subdirectories.
- **File System (FS)**:
    - Method to organize and manage files on a disk.
    - Tracks files and maps physical storage to logical paths.
- Files are organized in a **tree structure**.

---

2. Linux Directory Structure

Linux files are classified into:

1. **User Files**: Created and used by system users.
2. **System Files**: Executables, configurations, and binaries.
3. **Device Files**: Interface files for hardware (e.g., sound card, NIC).

Mount Point:

- A directory where additional storage is logically connected.

---

3. Types of Files in Linux

1. **Regular Files (`-`)**: Normal data files.
2. **Directory Files (`d`)**: Contain files and other directories.
3. **Device Files**:
    - **Block Files (`b`)**: Read/write data in blocks (e.g., hard disk).
    - **Character Files (`c`)**: Read/write data character-by-character (e.g., keyboard, printer).
4. **Special Files**:
    - **Named Pipe Files (`p`)**: Used for inter-process communication.
    - **Symbolic Link (`l`)**: Shortcut to another file.
    - **Socket Files (`s`)**: Communication between applications.

---

4. Types of Linux Commands

- **External Commands**: Exist as separate files (e.g., `ls`, `cat`).
- **Internal Commands**: Built into the shell (e.g., `echo`, `cd`, `mkdir`).

---

5. Basic File Operations

Command	Description
`cp file1 file2`	Copy `file1` to `file2`.
`mv file1 file2`	Move/rename `file1` to `file2`.
`rm file`	Remove a file.
`rmdir directory`	Remove a directory.
`cat file`	Display a file's content.
`head file`	Show first lines of a file.
`tail file`	Show last lines of a file.
`grep 'keyword' file`	Search for a keyword in a file.
`wc file`	Count lines, words, and characters.

---

6. Advanced File Operations

- **Copying Files Remotely**:

  ```
  bash

  scp user1@host1:file1 user2@host2:file2
  ```

- **Finding Files**:

  ```
  bash

  find / -name "*.sh"
  ```

- **Deleting a File by Inode**:

  ```
  bash

  find /path -inum <inode_number> -exec rm {} \;
  ```

---

7. Links in Linux

1. **Symbolic (Soft) Link**:
   - Points to another file.
   - Created using `ln -s target link_name`.
   - Example:

     ```
     bash

     ln -s /usr/src/sys ~/sys
     ```

   - **Attributes**:

     ```
     bash

     ls -l file
     lrwxrwxrwx ... file -> target
     ```

2. **Hard Link**:

- Refers to the same inode (data block) as the original file.
- Created using `ln target link_name`.
- Example:

  ```
  bash

  ln file.txt hard_link.txt
  ```

Differences between Hard and Soft Links:

Feature	Soft Link	Hard Link
Reference	Points to the file's path.	Points to file's inode.
Cross-partition	Yes.	No.
Original Deletion	Link breaks.	Link remains.

---

8. Special File Types

1. **Block Device File**:
   - Handles bulk data.
   - Created using:

     ```
     bash

     sudo mknod block_file b <major> <minor>
     ```

2. **Character Device File**:
   - Reads/writes data one character at a time.
   - Created using:

     ```
     bash

     sudo mknod char_file c <major> <minor>
     ```

3. **Named Pipe File (FIFO)**:
   - Used for inter-process communication (IPC).
   - Created using:

     ```
     bash

     mkfifo pipe_name
     ```

   - Example:
     - **Sender** (`send.sh`):

       ```
       bash

       echo "Hello" > /tmp/pipe
       ```

     - **Receiver** (`receive.sh`):

       ```
       bash

       cat < /tmp/pipe
       ```

4. **Socket File**:
    - Enables communication between processes.
    - Example: `/run/cups/cups.sock`.
5. **Temporary File**:
    - Created using `mktemp`:

      ```
      bash

      mktemp
      ```

---

9. Inodes

- **Inode (Index Node)**:
    - Stores file metadata:
        - Owner (UID/GID).
        - Permissions.
        - File size.
        - Pointers to data blocks.
- **Directories**:
    - Are special files storing filenames and corresponding inode numbers.
    - Special entries:
        - `.`: Current directory.
        - `..`: Parent directory.

---

10. Logical vs Physical File System

- **Logical File System**: Represents a virtual structure.
- **Physical File System**: The actual data blocks on storage devices.
- **Mounting**:
    - File systems can be mounted to paths:

      ```
      bash

      mount /dev/sda1 /mnt/data
      ```

---

11. Virtual File System (VFS)

- Acts as an **interface** between applications and the actual file system.
- Key Objects in VFS:
    1. **Superblock**: File system properties.
    2. **Inode**: Represents files and metadata.
    3. **Dentry**: Directory entry (path components).

4. **File Object**: Represents open files.

---

12. EXT2 File System

1. **Components**:
    - **Superblock**: Central metadata (free blocks, state).
    - **Group Descriptor**: State of block groups.
    - **Inode Table**: Metadata of files.
    - **Data Blocks**: Store file content.
2. **Directory Entries**:
    - Contains inode number, name length, and name.
3. **File Lookup**:
    - Resolves pathnames (`/home/user/file`):
        1. Starts from root (`/`).
        2. Resolves each component (e.g., `home`, `user`, `file`).
        3. Retrieves file's inode.

---

13. Allocation of Data Blocks

- **EXT2 Block Allocation**:
    - Maps logical file blocks to physical storage.
    - Efficiently allocates sequential blocks to minimize fragmentation.