**Linux Architecture & Basics**

**What is Linux Architecture?**

Linux is a **multi-user, multitasking operating system** with a layered design. It is built to:

- Manage hardware resources (CPU, memory, disk, network).
- Provide an interface for users to interact with the system.
- Run applications efficiently and securely.

At a high level, Linux has **three main layers**:

1. **Kernel** (Core of the OS)
2. **Shell** (User Interface)
3. **File System & User Space**

**1. Kernel (Core of Linux)**

The **Kernel** is the **heart of Linux**. It acts as the bridge between hardware and software.

**Responsibilities:**

- Manages CPU scheduling and memory.
- Controls input/output devices like keyboard, mouse, and storage.
- Manages file systems and network communication.
- Provides a secure environment for applications to run.

**Types of Kernels:**

- **Monolithic Kernel (Linux):** All major functions (device drivers, file system, networking) run inside the kernel. This makes it fast but large.
- **Microkernel:** Only minimal core services run in the kernel. Other services run in user space, making it modular and secure, but sometimes slower.

Example: When you open a file, the **kernel** handles the request by interacting with the **disk hardware** and providing the file to your application.

**2. Shell (User Interface)**

The **Shell** is the **command interpreter** between the user and the kernel.

**Functions:**

- Takes commands from the user.
- Interprets them and passes them to the kernel.
- Displays the results/output back to the user.

**Types of Shells:**

- **Command-Line Shell:** Text-based (e.g., Bash, Zsh, Csh).
- **Graphical Shell:** GUI-based environments (e.g., GNOME, KDE).

Without the shell, users would need to write **machine code** to communicate with hardware — which is impossible for practical use.

**3. File System & User Space**

Linux organizes everything (files, programs, devices, processes) into a **hierarchical tree structure**, starting at / (root).

**Key Directories:**

- /bin → essential commands
- /etc → configuration files
- /var → logs
- /home → personal files for each user
- /tmp → temporary files

**User Space** is where all applications and user processes run, separated from the kernel for security.

**Types of Shells in Linux**

**1. Bourne Shell (sh)**

- Original Unix shell by Stephen Bourne.
- Located at /bin/sh.
- Lightweight but limited features.
- Still used for compatibility in scripts.

**2. C Shell (csh)**

- Developed at Berkeley.
- Syntax is similar to **C language**.
- Provides features like **aliases** and **command history**.

**3. Korn Shell (ksh)**

- Created at AT&T.
- Combines features of **sh** and **csh**.
- More powerful scripting capabilities.

**4. Bourne Again Shell (bash)**

- Default in most Linux systems.

- Located at /bin/bash.
- Features: **command history, tab completion, scripting flexibility**.
- Most commonly used in practice.

**5. Z Shell (zsh)**

- Modern shell with advanced features.
- Supports **themes, plugins, and better autocompletion**.
- Popular among developers.

**Navigating the Linux Filesystem**

Everything in Linux is treated as a **file** – including devices, processes, and directories.

**Key Navigation Commands:**

- pwd → print working directory
- ls → list contents of directory
- cd → change directory
- tree → show directory structure in a tree view

**Common Shortcuts:**

- . → current directory
- .. → parent directory
- ~ → home directory
- / → root directory

Example:

```
cd /etc
ls -l
```

This takes you to the configuration directory and lists all files with details.

**File & Directory Management**

**Important Commands:**

- touch file1.txt → create an empty file
- mkdir project → create a new directory
- rm file1.txt → remove a file
- cp file1.txt backup.txt → copy file

- mv old.txt new.txt → rename or move file

**Viewing File Contents:**

- cat filename → display file
- more filename → page by page view
- less filename → scroll both directions

**File Permissions & Ownership**

Since Linux is multi-user, files and directories must have controlled access.

**Permission Types:**

- **Read (r):** view contents
- **Write (w):** modify or delete
- **Execute (x):** run program or script

**Example:**

```
ls -l file.txt

-rw-r--r--  1 student student 1234 Aug 29 14:00 file.txt
```

- Owner: read & write
- Group: read only
- Others: read only

**Changing Permissions:**

- chmod u+x script.sh → add execute for owner
- chmod 644 file.txt → read-write for owner, read-only for others

**Changing Ownership:**

- chown newuser:newgroup file.txt

**Task**

1. Create a directory called project_day1.
2. Inside it, create three folders: src/, logs/, and config/.
3. Add two files: app.sh in src/ and app.log in logs/.
4. Give execute permission to app.sh.
5. Copy app.sh into config/ as app_backup.sh.
6. Check permissions of all files using ls -l.