DevOps Shack

Linux Architecture

Click Here To Enrol To Batch-6 | DevOps & Cloud DevOps

Linux architecture is structured in layers that help manage system resources and services efficiently. The primary components include the hardware, kernel, system libraries, system utilities, and user applications. Below is a detailed explanation with a textual diagram.

1. Hardware Layer

The hardware layer is the physical layer comprising all physical devices (CPU, memory, I/O devices, etc.) that are necessary for the system's operation.

2. Kernel Layer

The kernel is the core part of the Linux operating system. It interacts directly with the hardware and manages system resources. The kernel has several important sub-components:

- **Process Management:** Handles process creation, scheduling, and termination. Example: The fork() system call to create new processes.
- **Memory Management:** Manages the allocation and deallocation of memory. Example: The malloc() and free() functions in C.
- **Device Drivers:** Facilitate communication between the kernel and hardware devices. Example: Network drivers, USB drivers.
- **File System Management:** Manages file operations and storage. Example: The ext4 file system.
- **Network Management:** Manages networking tasks. Example: TCP/IP stack.

3. System Libraries

System libraries provide a set of functions that can be used by applications to perform tasks without interacting with the kernel directly. The most commonly used library is the GNU C Library (glibc), which provides system calls and basic functions.

Example: The printf() function in C to print output.

4. System Utilities

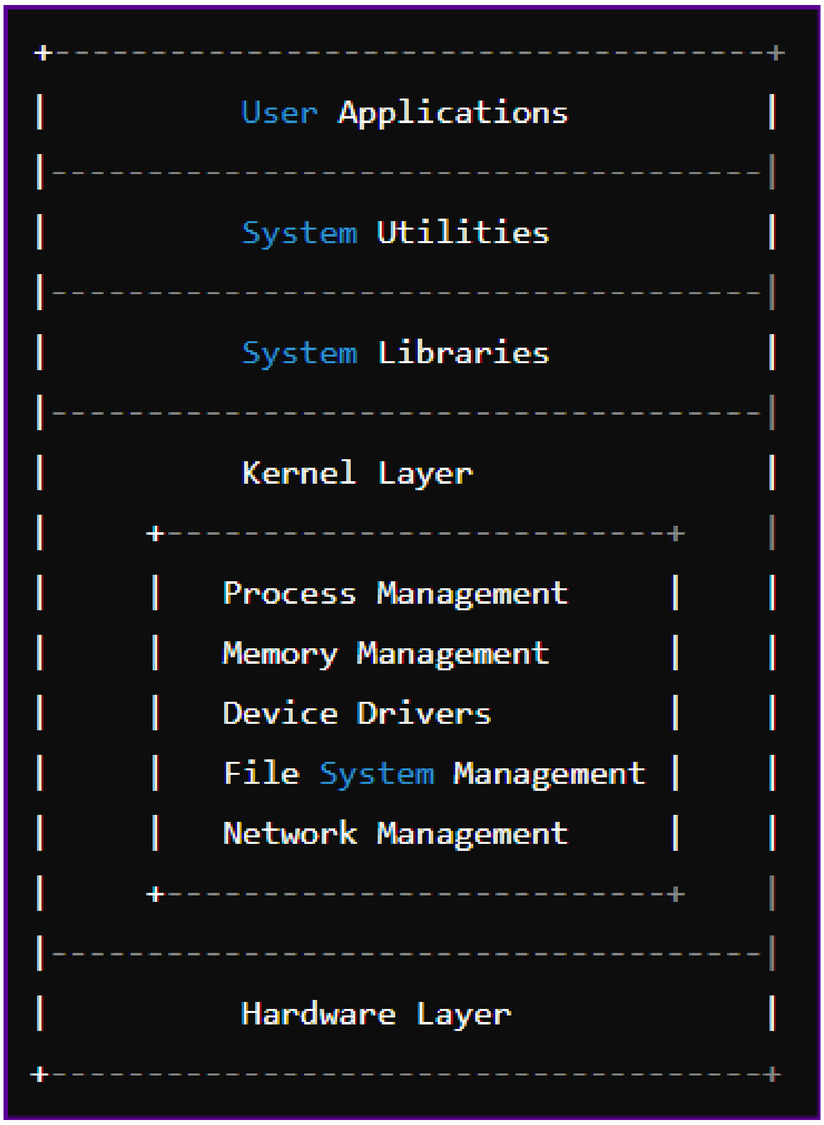
System utilities are programs that perform specific tasks related to system administration and management. These utilities can be categorized into two types:

- **Basic Utilities:** Include commands like ls, cp, mv, and rm.
- **Advanced Utilities:** Include commands like ps, top, df, and iptables.

5. User Applications

User applications are programs that run in the user space. These include software like web browsers, text editors, and custom applications developed by users.

Textual Diagram



```
+-------------------------------------+
|          User Applications          |
|-------------------------------------|
|          System Utilities           |
|-------------------------------------|
|          System Libraries           |
|-------------------------------------|
|          Kernel Layer               |
|     +--------------------------+    |
|     |   Process Management     |    |
|     |   Memory Management      |    |
|     |   Device Drivers         |    |
|     |   File System Management |    |
|     |   Network Management     |    |
|     +--------------------------+    |
|-------------------------------------|
|          Hardware Layer             |
+-------------------------------------+
```

Example Explanation

1. **Hardware Layer:** The CPU, memory, disk drives, and other peripheral devices form the hardware layer.
2. **Kernel Layer:** Suppose a user wants to copy a file from one location to another. The request goes through the kernel:
    - **Process Management:** The cp command initiates a process.
    - **Memory Management:** Memory is allocated for the process.
    - **Device Drivers:** The kernel communicates with the disk drivers to read and write data.
    - **File System Management:** The kernel handles the file system operations to copy the file.
    - **Network Management:** If the file is copied over the network, the kernel handles network communication.
3. **System Libraries:** The cp command uses standard library functions like open(), read(), write(), and close() to perform file operations.
4. **System Utilities:** The cp command itself is a system utility that a user can invoke from the command line.
5. **User Applications:** A file manager application might provide a graphical interface for the user to copy files, which ultimately uses the cp command behind the scenes.

This layered architecture allows Linux to be flexible, secure, and efficient in managing system resources and executing user applications.

Example Scenario: Copying a File

Step 1: User Input

A user enters the command to copy a file:

cp /home/user/source.txt /home/user/destination.txt

Step 2: Shell Interpretation

The shell (e.g., Bash) interprets the command and initiates the cp process.

Step 3: System Call Interface

The cp utility makes system calls to perform the copy operation. Some of the key system calls involved are:

- open(): Opens the source file for reading.
- read(): Reads data from the source file.
- open(): Opens or creates the destination file for writing.
- write(): Writes data to the destination file.
- close(): Closes both the source and destination files.

Step 4: Kernel Operations

These system calls interact with the kernel to perform the necessary operations.

1. **Process Management**
   - The kernel schedules and manages the cp process. If multiple processes are running, the kernel allocates CPU time to the cp process.
2. **Memory Management**
   - The kernel allocates memory for the cp process to store data read from the source file before writing it to the destination file.
3. **File System Management**
   - The kernel handles file operations. It ensures that the source file is correctly read and the destination file is correctly written.
   - The kernel updates the file system metadata to reflect the new file in the destination directory.
4. **Device Drivers**
   - The kernel communicates with the disk drivers to read data from the source file and write data to the destination file. If the files are on different physical drives, the kernel manages data transfer between the drives.

Step 5: System Libraries

The cp utility uses functions provided by system libraries (e.g., GNU C Library) to make the necessary system calls.

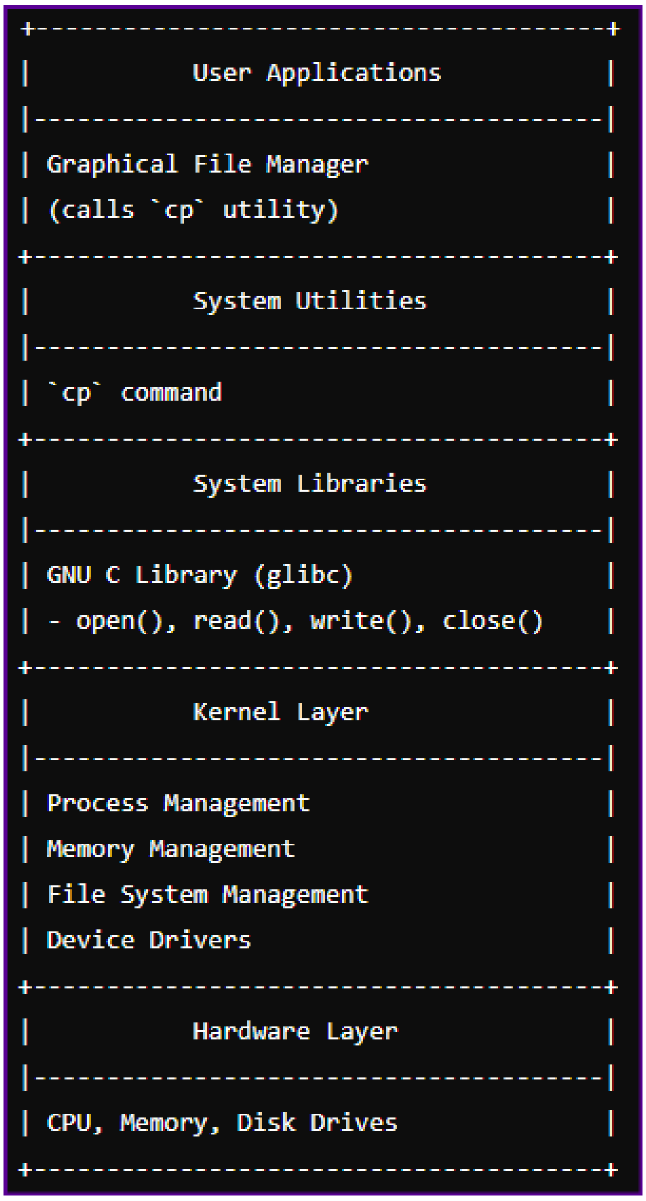
Step 6: System Utilities

The cp command itself is a system utility. It simplifies the process of copying files by encapsulating the necessary system calls and error handling in a user-friendly command.

Step 7: User Applications

If a user uses a file manager (e.g., Nautilus on GNOME), the file manager provides a graphical interface for copying files. When the user drags and drops a file to copy it, the file manager internally calls the cp utility or uses similar system calls to perform the copy operation.

Textual Diagram of the Example



```
+---------------------------------------+
|           User Applications           |
|---------------------------------------|
| Graphical File Manager                |
| (calls `cp` utility)                  |
+---------------------------------------+
|           System Utilities            |
|---------------------------------------|
| `cp` command                          |
+---------------------------------------+
|           System Libraries            |
|---------------------------------------|
| GNU C Library (glibc)                 |
| - open(), read(), write(), close()    |
+---------------------------------------+
|           Kernel Layer                |
|---------------------------------------|
| Process Management                    |
| Memory Management                     |
| File System Management                |
| Device Drivers                        |
+---------------------------------------+
|           Hardware Layer              |
|---------------------------------------|
| CPU, Memory, Disk Drives              |
+---------------------------------------+
```

Explanation of the Example

1. **User Input:** The user enters the cp command in the terminal or uses a graphical file manager.
2. **Shell Interpretation:** The shell interprets the command and initiates the cp process.
3. **System Call Interface:** The cp command makes system calls to perform the copy operation.
4. **Kernel Operations:** The kernel manages process scheduling, memory allocation, file operations, and communication with hardware.
5. **System Libraries:** The cp command uses system libraries to interact with the kernel.
6. **System Utilities:** The cp command is a system utility that encapsulates the copy operation.
7. **User Applications:** A graphical file manager may provide a user interface for copying files, internally using the cp command or similar system calls.

This example illustrates how the various layers of Linux architecture work together to perform a simple task like copying a file.