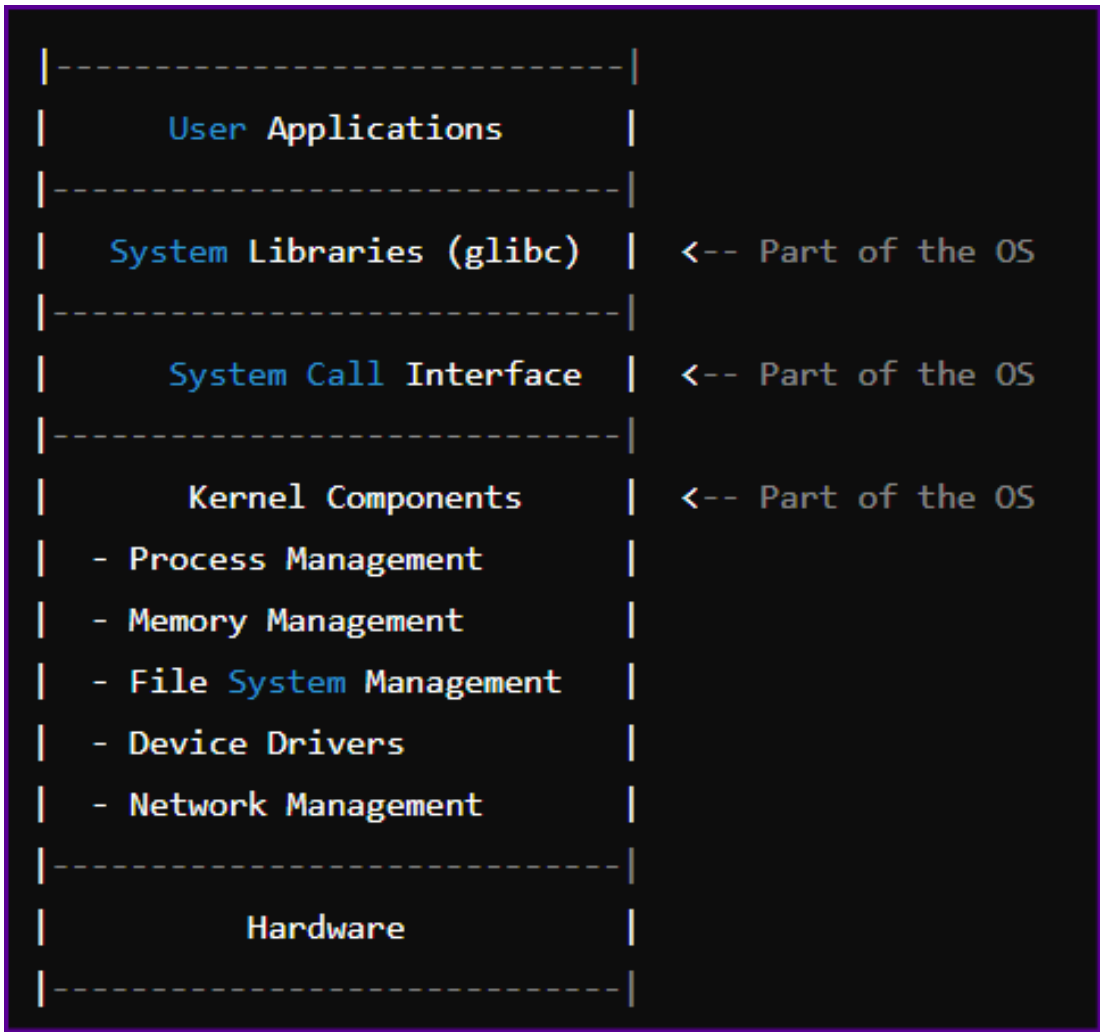
DevOps Shack

LINUX ARCHITECTURE

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

The operating system (OS) in the provided diagram is represented by several components, primarily the **Kernel Layer** and **System Call Interface (SCI)**, but also extending to include the **System Libraries**. Here's an updated textual diagram highlighting the OS:



Detailed Explanation with OS Highlight

1. **Hardware Layer**
    - This layer consists of the physical components of the system such as the CPU, memory, storage devices, and network interfaces. The OS directly interacts with this layer through device drivers.
2. **Kernel Layer** (Part of the OS)
    - **Process Management:** The kernel handles the creation, scheduling, and termination of processes, ensuring fair CPU time allocation.
    - **Memory Management:** It manages memory allocation and deallocation, including virtual memory, paging, and swapping.
    - **File System Management:** The kernel manages data storage, retrieval, and updates on storage devices, supporting various file systems.
    - **Device Drivers:** These are part of the kernel that interact with hardware devices, providing the necessary interface for communication between hardware and the OS.
    - **Network Management:** The kernel handles network interfaces and protocols, managing network communication and services.
3. **System Call Interface (SCI)** (Part of the OS)
    - This interface acts as a bridge between user space and kernel space, providing standardized calls that user applications use to interact with the kernel. System calls include operations like reading/writing files, creating processes, and managing memory.
4. **User Space**
    - **System Libraries (glibc):** These libraries provide standard functions for user applications to interact with the kernel through the system call interface. They offer higher-level functionality than direct system calls.
    - **User Applications:** Programs and utilities that users interact with, such as the shell, text editors, web browsers, and custom applications. These are not part of the OS but rely on it to function.

Example: Creating and Managing a File

To illustrate the OS's role in creating and managing a file:

1. **User Application:** A user runs a text editor (e.g., Vim) to create a new text file.
2. **System Libraries:** The text editor uses standard library functions (e.g., fopen(), fwrite()) to create and write to the file.
3. **System Call Interface:** These library functions make system calls (e.g., open(), write()) to the kernel to perform the actual file operations.
4. **Kernel Components:**

- **File System Management:** The kernel's file system management component processes the system calls to create the file on the storage device.
- **Memory Management:** Manages the memory used by the text editor and the buffers used for writing data.
- **Process Management:** Schedules the text editor process to run and allocates CPU time for it.

5. **Device Drivers:** The file system management component interacts with the appropriate device drivers to write the file data to the storage device.
6. **Hardware Layer:** The storage device (e.g., HDD or SSD) performs the physical writing of the file data.

Conclusion

The operating system in Linux is represented primarily by the kernel layer and the system call interface, extending to include the system libraries. These components work together to manage hardware resources, provide essential services, and enable user applications to perform their tasks efficiently. Understanding this architecture helps in appreciating how the OS facilitates interaction between hardware and software.