Day 3-Linux Core Concepts Part 3

The **Linux kernel** is the core (or "brain") of the Linux operating system. It manages communication between hardware and software, ensuring that system resources are used efficiently.

It plays a central role in managing processes, memory, devices, and system security.

Responsibilities of the Kernel

The Linux kernel is responsible for four major tasks:

- **Memory Management**
  Keeps track of how much memory is used, by whom, and where it is located.
- **Process Management**
  Determines which processes can use the CPU, when, and for how long.
- **Device Drivers**
  Acts as a mediator between hardware and processes, allowing software to interact with hardware.
- **System Calls and Security**
  Receives requests for low-level services from user-space applications (via system calls) and ensures secure operations.

Architecture and Design

- **Monolithic Design**
  The Linux kernel is **monolithic**, meaning it handles CPU scheduling, memory management, file systems, and more *within the kernel itself*.
- **Modular Capabilities**
  It is also **modular**, allowing for dynamically loaded components (kernel modules) that extend functionality without rebooting.
  For example:
  - `lsmod  # Lists currently loaded modules`
  - `modprobe <module_name>  # Loads a module`

Checking Kernel Information

Use the `uname` command to view kernel version and system information:

- `uname # Prints system name`
- `uname -r # Prints kernel release`
- `uname -a # Prints all available system info`

Kernel Space vs User Space

In Linux, memory is divided into two main areas:

- **Kernel Space**
  This area is reserved for code that runs in kernel mode. It has unrestricted access to hardware and is responsible for low-level operations and services.
- **User Space (Userland)**
  This contains all user applications. These apps interact with the kernel through **system calls**. For example:
  `open()`, `close()`, `getpid()`, `readdir()`, `strlen()`, `closedir()`