An operating system is a software that manages computer hardware, software resources, and provides a foundation for running applications. Here are some of the key operating system concepts:

Kernel:

- The kernel is the core component of the operating system. It is responsible for managing system resources, such as CPU, memory, and I/O devices, and for providing essential services to applications. The kernel runs in privileged mode and has direct access to hardware
- E.g # ls /boot => vmlinuz-3.10.0-1160.59.1.el7.x86_64

User Space:

- In most operating systems, there is a division between user space and kernel space. User space is where user-level applications run. They don't have direct access to hardware resources and must make system calls to request services from the kernel.
- E.g bash shell or Windows command line
  /bin/bash, /bin/sh

System Calls:

- System Calls: These are interfaces that allow programs to request services from the OS, such as reading a file or creating a new process.
- System calls are interfaces that allow user-level processes to interact with the kernel. They provide a way for applications to request services like I/O operations, process management, and memory allocation. Examples of system calls include open(), read(), and fork().

Device Management:

- Device Drivers: OS communicates with hardware devices through device drivers, which are specialized software components. This involves managing input and output devices.
- Device drivers are responsible for controlling and managing hardware devices. They act as intermediaries between user-level processes and the hardware. The kernel provides a standard interface for device drivers, allowing them to be loaded and unloaded dynamically

Interrupt Handling:

- Hardware and software events trigger interrupts, which are signals that prompt the CPU to temporarily suspend its current execution and switch to a different routine. The kernel handles interrupts, directing them to the appropriate interrupt service routines (ISRs) to manage various hardware events

Process Management:

- Processes: A process is an independent program running in memory. The OS manages processes by scheduling them, providing them with resources, and ensuring they don't interfere with each other.
- Thread: A thread is a smaller unit of a process, representing the smallest unit of execution. Multithreading allows for concurrent execution within a single process.
- Process Synchronization: OS provides mechanisms for processes to communicate and synchronize their actions. This is essential to avoid issues like race conditions and deadlocks.
- The operating system manages processes and threads. This includes process creation, scheduling, and termination, as well as thread creation, synchronization, and communication. These activities are crucial for achieving multitasking and parallelism.

Memory Management:

- Memory Allocation: The OS manages the allocation of physical memory to various processes and ensures they do not interfere with each other's memory.
- Virtual Memory: This allows processes to use more memory than is physically available. It involves paging and swapping data in and out of disk storage.
- This component is responsible for managing the computer's memory, including allocation and deallocation of memory for processes and managing virtual memory. Virtual memory allows processes to have the illusion of accessing more memory than is physically available

File System Management:

- File Systems: OS manages files on storage devices and provides a hierarchy of directories and files. It handles file creation, deletion, modification, and access control.

- I/O Operations: It facilitates input and output operations, providing a way for processes to read from and write to files.
- The file system component manages file storage, organization, and access. It is responsible for maintaining the directory structure, handling file operations, and ensuring data integrity.
- I/O management controls input and output operations for devices and files. It includes buffering, caching, and scheduling I/O requests to optimize data transfer.
- Security and Access Control: Security mechanisms, such as user authentication, permissions, and encryption, are integrated into the architecture to protect system resources and data from unauthorized access and tampering.

User Interface:

- User Interface: OS provides a user interface that allows users to interact with the computer system. This could be a command-line interface or a graphical user interface (GUI).
- The user interface component provides a way for users to interact with the operating system. It can include command-line interfaces (CLIs) or graphical user interfaces (GUIs).

Security and Authentication:

- User Authentication: OS enforces security by requiring users to log in with valid credentials.
- Access Control: It controls which users and processes have access to various system resources.

Networking and Communication:

- Networking: OS supports network communication and protocols, allowing devices to connect and communicate with each other.
- In networked operating systems, networking services and protocols are integrated into the architecture to support communication and data exchange across networks.

File Permissions:

- Permissions and Access Control: OS regulates access to files and directories, allowing for the definition of read, write, and execute permissions for users and groups.

Error Handling and Logging:

- Error Handling: OS manages and reports errors that occur in both hardware and software components.

- Logging: It keeps logs of system activities for troubleshooting and security analysis.

Process Scheduling:

- Process Scheduling: OS schedules processes to run on the CPU efficiently. Scheduling algorithms ensure fair access to CPU time and optimize system performance.

Interprocess Communication (IPC):

- IPC: OS provides mechanisms for processes to communicate and share data with each other. This can be achieved through shared memory, message passing, or other means.
- IPC mechanisms enable processes to communicate and share data. Common methods include message passing, shared memory, and pipes.