LECTURE NOTES PREPARED BY PRANATI PATTNAIK

TH:1-OPERATING SYSTEMS

UNIT-1(over View of OS)

What is an Operating System?

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Operating System – Examples

Windows: This is one of the most popular and commercial operating systems developed and marketed by Microsoft. It has different versions in the market like Windows 8, Windows 10 etc and most of them are paid.

- Linux/Unix: This is a Unix based and the most loved operating system first released on September 17, 1991 by Linus Torvalds. Today, it has 30+ variants available like Fedora, OpenSUSE, CentOS, UBuntu etc. Most of them are available free of charges though you can have their enterprise versions by paying a nominal license fee.
- MacOS This is again a kind of Unix operating system developed and marketed by Apple Inc. since 2001.
- iOS This is a mobile operating system created and developed by Apple Inc. exclusively for its mobile devices like iPhone and iPad etc.
- Android This is a mobile Operating System based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

Operating System – Functions

- Process Management
- I/O Device Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- Security Management
- Command Interpreter System
- Control over system performance
- Job Accounting
- Error Detection and Correction
- Coordination between other software and users
- Many more other important tasks

What is the Unix Operating System?

Unix is an Operating System. Unix also acts as an interface between the user and the hardware, i.e. the user and the computer itself. Unix mainly focuses on the concept of Kernel and Shell division of an Operating System due to which it is so powerful.

Features of Unix Operating System

- Following are the features of the Unix operating system that date from the very first version to the latest versions of Unix:

- **Multitasking Operating System**: In Unix, a user can run multiple tasks/processes simultaneously. For example, a user can run a text editor and can also open a web browser at the same time. Multiple processes being run at the same time. This multitasking feature can also be seen in Windows OS.

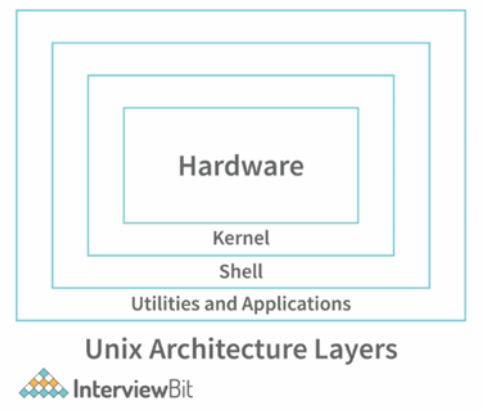
- **Multiuser Operating System**: In Unix, multiple users can run their own tasks/processes simultaneously. Hence, Unix is called a multiuser OS.

- **GUI :** Unix system has graphical user interface

- **The Powerful Unix Toolkit**: Unix has a toolkit, that is, it has multiple applications that help in performing multiple tasks. For example, Unix has compilers and interpreters, text manipulation tools, system administration tools, etc. Whenever a new version of Unix is released, some of the old programs/applications are either modified or removed from the toolkit and some new applications are added.

So, these were some of the topmost features of the Unix Operating System. Apart from this, the ease of programming in Unix because of the above features and the amazing Unix documentation helps increase its power more. Now that we have some basic idea about Unix and its features, let us learn about Unix Architecture.

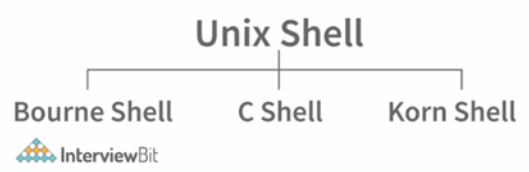
UNIX Architecture Ken Thompson, Dennis Ritchie, and their team developed an Operating System called Unix (Uniplexed Information Computing System) in 1970



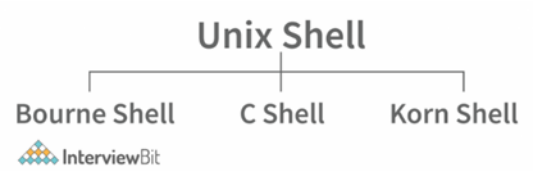
The Unix architecture has 4 layers.

1.Hardware: The physical base of the system, including the CPU, RAM (memory), and hard disks.
2.Kernel: The "heart" of the operating system. It is the only layer that interacts directly with the hardware.
3. The Shell

- The Shell is a collection of UNIX Commands.
- The Shell acts as an interface between the user and the kernel.
- The Shell is a Command Line Interpreter(CLI)–Translates the commands provided by the user and converts it into a language that is understood by the Kernel.
- Only one Kernel running on the system, but several shells in action-one for each user who is logged in.
  Eg: C Shell, Bourne Shell, Korn Shell etc.



- **4.Applications & Utilities:** The outermost layer where users perform actual work. This includes standard Unix commands (like ls to list files or cp to copy them) and external software like text editors or web browsers.
- **Resource Management:** It allocates memory and CPU time to different tasks.
- **Process Control:** It creates, schedules, and stops programs (processes).
- **File Management:** It handles how data is saved and retrieved from storage.
- **The Shell**



Features of UNIX

1) **Simple design, organization and functioning**

The architecture of the UNIX is simple, organized and functional.

2) **Portability**

The code can be changed and compiled on a new machine.

3) UNIX Shell

The user interface UNIX Shell provides the services that the user wants.

4) **Hierarchical file system**

UNIX uses a hierarchical file structure to store information.

5) **Multi user**

UNIX allows more than one user to share the same computer system at the same time.

6) **Multi-tasking**

More than one program can be run at a time.

7) **Security**

UNIX provides high security level(System level security and File level security)

8) **Pipes and Filters**

In UNIX, we can create complex programs from simple programs.

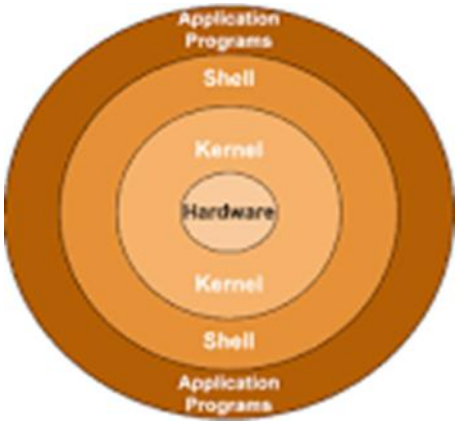
9) **Utilities**

UNIX has over 200 utility programs for various functions.

10) **Machine Independence**

The system hides the machine architecture from the user, making it easier to write applications that can run any computer system.

What is the kernel in Unix?



In Unix, the kernel is the core component of the operating system, acting as the essential bridge between software applications and the computer's hardware, managing all system resources like CPU, memory, and devices, handling process scheduling, file management, and responding to software requests (system calls) to ensure smooth, secure, and efficient operation. Key Functions

- **Hardware Management:**

Directly controls hardware (disks, keyboard, memory) and communicates with peripherals via device drivers.

- **Process Management:**

Schedules processes, allocates CPU time, and manages process creation/termination for multitasking.

- **Memory Management:**

Decides which parts of memory processes can use and handles situations where memory is scarce.

- **System Calls:**

Provides an interface (like read(), write()) for applications to request services from the kernel without direct hardware access, ensuring safety.

- **Resource Allocation:**

Manages all system resources, ensuring fair and efficient access for different programs.

Features of Kernel

1) **Concurrency**

Many processes run concurrently to improve the performance of the system.

2) **Virtual Memory(VM)**

Memory management subsystem implements the virtual memory and users need not worry about the executable program size and RAM size.

3) **Paging**

It is a technique to minimize the internal and external fragmentation in the physical memory.

4) **Virtual File System(VFS)**

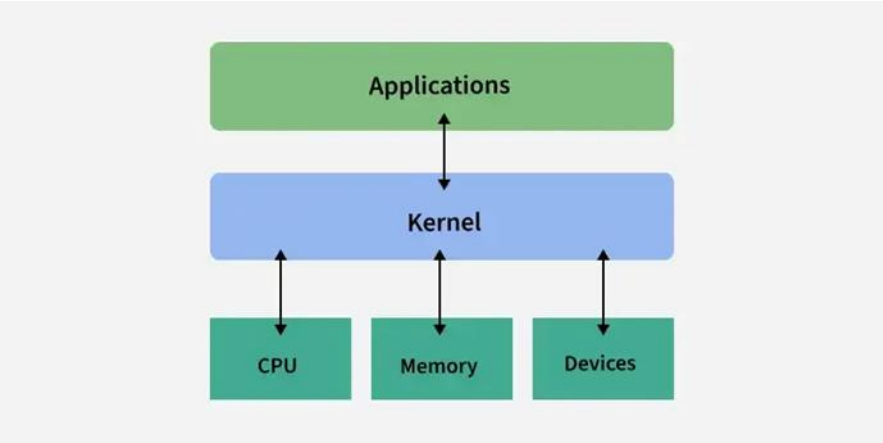
A VFS is a file system used to help the user to hide the different file systems complexities.

5) **Inter process communication**

It ranges from asynchronous signaling of events to synchronous transmission of messages between processes.

Working of Kernel

- The kernel is the first part of the OS loaded into memory during boot, and it stays resident while the system is running.
- It operates in a privileged mode (kernel mode), separate from user mode for applications; user apps can't directly access hardware or critical resources.
- Applications make requests to the kernel via system calls (or software interrupts). The kernel handles these by switching from user mode to kernel mode.
- Kernel executes the requested operation (e.g. file I/O, process creation, memory allocation).
- On completion, kernel returns result (or error) to user space.
- Kernel does context switching as needed (scheduler picks next process/thread) to allow multitasking.
- interact with the system



System calls in Unix User programs cannot directly access hardware or critical OS resources because it would make the system unstable and insecure. To maintain safety, the operating system provides system calls — controlled interfaces that allow user programs to request services from the kernel. These calls act as a gateway between user mode and kernel mode. System Calls are,

- A way for programs to interact with the operating system.
- Provide the services of the operating system to the user programs.
  Only entry points into the kernel are executed in kernel mode.

How do System Calls Work?

A system call is a controlled entry point that allows a user program to request a service from the operating system. Here's how it works:

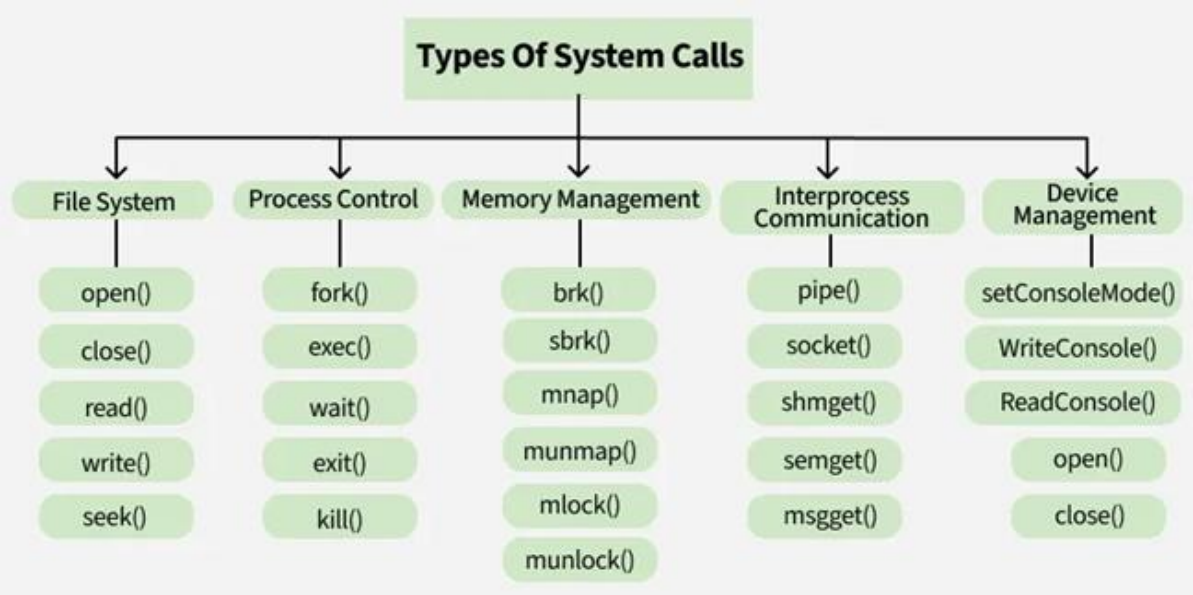
- The user program executes a system call instruction (e.g., using syscall or int 0x80).
- The CPU switches from user mode → kernel mode for safe execution.
- The kernel identifies the system call number and performs the requested operation (file access, process creation, memory allocation, etc.).
- After completing the task, the kernel switches back to user mode.
- The result (success/failure/data) is returned to the program.
- Without system calls, every program would need its own way to access hardware, leading to inconsistent and insecure systems.

System calls do not always cause context switching. They primarily involve a mode switch from user mode to kernel mode. A context switch happens only when the calling process is blocked, not during every system call.

Types of System Calls

Services provided by an OS are typically related to any kind of operation that a user program can perform like creation, termination, forking, moving, communication, etc. Similar types

of operations are grouped into one single system call category. System calls are classified into the following categories.



- **File System**: Used to create, open, read, write, and manage files and directories.
- **Process Control**: Used to create, execute, synchronize, and terminate processes.
- **Memory Management**: Used to allocate, de allocate, and manage memory for processes.
- **Interprocess Communication (IPC)**: Used for data exchange and communication between different processes.
- **Device Management**: Used to request and release devices, and to perform read/write operations on them

System Program

Definition : System programs in an operating system are software tools that help users manage files, run programs, and control system resources. They include file managers, program loaders, compilers, and system utilities, making it easier to operate the computer efficiently and keep it running smoothly.

The examples of System Programs :

File Management : A file is a collection of specific information stored in the memory of a computer system. File management is defined as the process of manipulating files in the computer system, its management includes the process of creating, modifying and deleting files.

Command Line Interface(CLI's) : CLIs is the essential tool for user . It provide user facility to write commands directly to the system for performing any operation . It is a text-based way to interact with operating system. CLIs can perform many tasks like file manipulation, system configuration and etc.

Device drivers : Device drivers work as a simple translator for OS and devices . Basically it act as an intermediatory between the OS and devices and provide facility to both OS and devices to understand each other's language so that they can work together efficiently without interrupt.

Status Information : Information like date, time amount of available memory, or disk space is asked by some users. Others providing detailed performance, logging, and debugging information which is more complex. All this information is formatted and displayed on output devices or printed. Terminal or other output devices or files or a window of GUI is used for showing the output of programs.

File Modification : This is used for modifying the content of files. Files stored on disks or other storage devices, we use different types of editors. For searching contents of files or perform transformations of files we use special commands.
Programming-Language support : For common programming languages, we use Compilers, Assemblers, Debuggers, and interpreters which are already provided to users. It provides all support to users. We can run any programming language. All important languages are provided.
Program Loading and Execution : When the program is ready after Assembling and compilation, it must be loaded into memory for execution. A loader is part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages for starting a program. Loaders, relocatable loaders, linkage editors, and Overlay loaders are provided by the system.
Communications : Connections among processes, users, and computer systems are provided by programs. Users can send messages to another user on their screen, User can send e-mail, browsing on web pages, remote login, the transformation of files from one user to another.