Introduction to Operating Systems

What is an Operating System?

- Definition:
  An Operating System (OS) is system software that manages hardware, software resources, and provides services for computer programs.
- Acts as an interface between user and hardware.
- An operating system is a software program required to manage and operate a computing device like smartphones, tablets, computers, supercomputers, web servers, cars, network towers, smartwatches, etc.

Functions of an Operating System

- Process Management
- Memory Management
- File System Management
- Device Management
- Security and Access Control
- User Interface

Types of Operating Systems

- **Batch Operating System (IBM's z/OS**; Unisys MCP; and Burroughs MCP/BCS)
- **Time-Sharing (Multitasking) OS (**UNIX and LINUX/Modern Windows)
- **Distributed OS (**Solaris, OSF/1, and Mach, Google's Fuchsia OS)
- **Real-Time OS (**VxWorks, QNX, FreeRTOS, and LynxOS.)
- **Mobile OS (**BlackBerry OS, Windows Phone, Symbian, HarmonyOS, Tizen, and KaiOS.)

Popular Operating Systems

- **Windows** (by Microsoft)
- **Linux** (open-source)
- **macOS** (by Apple)
- **Android** (by Google)

- **iOS** (by Apple)

Multitasking, Multiprogramming & Multithreading

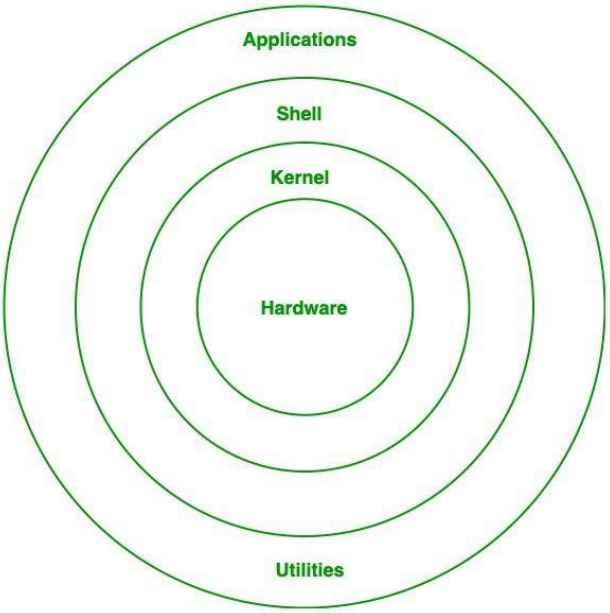
- **Multitasking:** Multiple tasks at the same time
- **Multiprogramming:** More than one program in memory
- **Multithreading:** Multiple threads in a process

Security in OS

- Authentication (Username & Password)
- Authorization
- Encryption
- Firewalls & Updates

User Interfaces

- **Command-Line Interface (CLI)**
- **Graphical User Interface (GUI)**
- Examples: Windows Explorer, Linux Terminal



1st Generation (1940s-1950s):

- **Technology:** Vacuum tubes and plugboards.
- **Operating System:** None.
- **Characteristics:** Computers were programmed manually, with users having direct access to the hardware. Tasks were performed sequentially, and there was no concept of an operating system as we know it today.

2nd Generation (1950s-1960s):

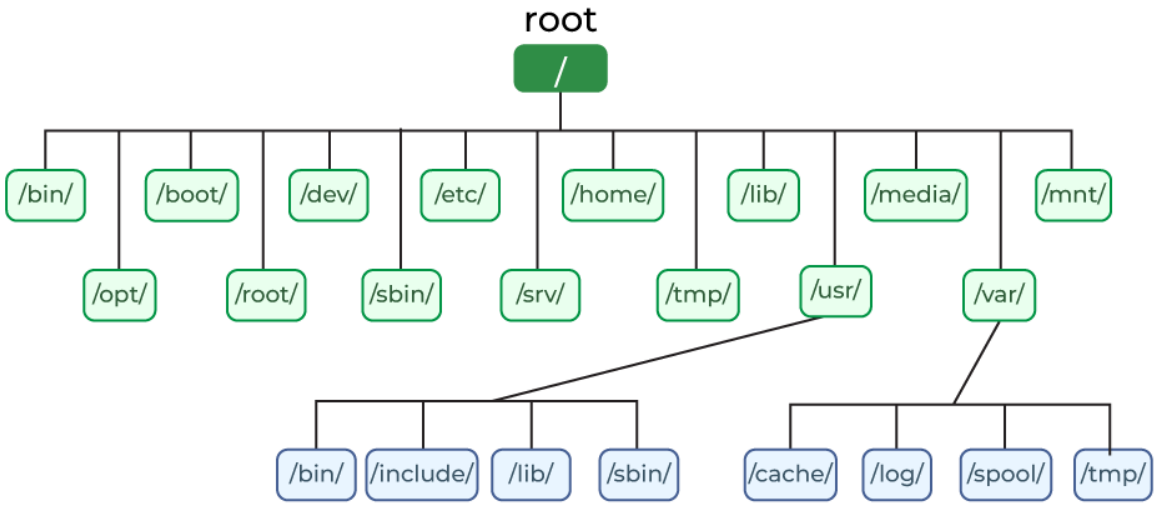
- **Technology:** Transistors.
- **Operating System:** Batch processing systems like GMOS.
- **Characteristics:** Batch processing was introduced to improve efficiency by grouping similar jobs together and running them sequentially. Users submitted jobs on punch cards, and the system would process them without direct interaction.

3rd Generation (1960s-1970s):

- **Technology:** Integrated circuits.
- **Operating System:** Multiprogramming systems.
- **Characteristics:** Multiprogramming allowed multiple jobs to reside in memory simultaneously, enabling the CPU to switch between them and improve resource utilization. This led to the development of time-sharing systems where multiple users could interact with the system concurrently.

4th Generation (1980s-Present):

- **Technology:** Microprocessors and personal computers.
- **Operating System:** Personal computer operating systems like Windows, macOS, and Linux, as well as distributed systems.
- **Characteristics:** This era saw the rise of personal computers and graphical user interfaces (GUIs). Operating systems became more user-friendly and offered features like multitasking, networking, and support for a wide range of applications.



Unix / Linux - File System

Sr.No.	Directory & Description
1	**/** This is the root directory which should contain only the directories needed at the top level of the file structure
2	**/bin** This is where the executable files are located. These files are available to all users
3	**/dev** These are device drivers
4	**/etc** Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages
5	**/lib** Contains shared library files and sometimes other kernel-related files
6	**/boot** Contains files for booting the system
7	**/home**

	Contains the home directory for users and other accounts
8	**/mnt** Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively
9	**/proc** Contains all processes marked as a file by **process number** or other information that is dynamic to the system
10	**/tmp** Holds temporary files used between system boots
11	**/usr** Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others
12	**/var** Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
13	**/sbin** Contains binary (executable) files, usually for system administration. For example, ***fdisk*** and ***ifconfig*** utlities
14	**/kernel** Contains kernel files

Write a C program to simulate producer-consumer problem.

DESCRIPTION

Producer-consumer problem, is a common paradigm for cooperating processes. A producer process produces information that is consumed by a consumer process. One solution to the producer-consumer problem uses shared memory. To allow producer and consumer processes to run concurrently, there must be available a buffer of items that can be filled by the producer and emptied by the consumer. This buffer will reside in a region of memory that is shared by the producer and consumer processes. A producer can produce one item while the consumer is consuming another item. The producer and consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced.

- **Producer:** Generates data and places it into the shared buffer.
- **Consumer:** Retrieves data from the shared buffer and processes it.
- **Shared Buffer:** A fixed-size memory area used for communication between the producer and consumer.
- **Synchronization:** Mechanisms like semaphores are used to control access to the buffer and prevent race conditions.

OUTPUT

```
1. Produce 2. Consume 3. Exit
Enter your choice: 2
Buffer is Empty
1. Produce 2. Consume 3. Exit
Enter your choice: 1
Enter the value: 100
1. Produce 2. Consume 3. Exit
Enter your choice: 2
The consumed value is 100
1. Produce 2. Consume 3. Exit
Enter your choice: 3
```