Introduction to Operating Systems

1. What is an Operating System (OS)?

- Definition: An OS is system software that manages hardware resources and provides services to applications.

- Functions of an OS:

  - Manages hardware (CPU, memory, I/O devices).

  - Provides a user interface (CLI, GUI).

  - Manages files, processes, and memory.

  - Ensures system security and stability.

2. Common Operating Systems

Apart from this there are around 585+ Operating Systems in the world (Redhat, IBM, Blackberry, Solaris, Sun microsystems, etc..)

3. Types of Operating Systems

- Single-User, Single-Tasking OS: Only one user and one task can run at a time (e.g., MS-DOS).

- Single-User, Multi-Tasking OS: Allows multiple tasks but only for a single user (e.g., Windows, macOS).

- Multi-User OS: Multiple users can access the system simultaneously (e.g., UNIX, Linux).

- Real-Time OS: Processes data as it comes in, typically used in embedded systems (e.g., VxWorks, RTLinux).

- Distributed OS: Manages a group of computers and makes them work as a single system (e.g., Amoeba).

- Embedded OS: Specialized OS for devices like routers and IoT devices (e.g., Android for phones, Windows IoT).

Examples:

Comparison of File Systems: Windows (NTFS) vs. Linux (ext4)

1. NTFS (Windows File System)

- Full Name: New Technology File System

- Introduced: 1993 (with Windows NT)

- Features:

- Metadata and Journaling: NTFS uses a journaling system to keep track of changes, making it more reliable in case of crashes.

- Access Control Lists (ACLs): NTFS has a built-in mechanism for defining user and group permissions on files and directories.

- File Compression and Encryption: Supports built-in file-level compression and encryption (EFS - Encrypting File System).

- File Size: Supports file sizes up to 16 exabytes, with theoretical volume size limits of 16 EB.

- Resilience: NTFS uses a Master File Table (MFT) to store metadata, ensuring quick recovery of file system integrity after a crash.

- Compatibility: NTFS is primarily compatible with Windows systems. Linux can read and write to NTFS using external drivers like `ntfs-3g`.

2. ext4 (Linux File System)

- Full Name: Fourth Extended File System

- Introduced: 2008

- Features:

- Metadata and Journaling: Like NTFS, ext4 supports journaling, but it allows three journaling modes (writeback, ordered, and journal) to balance between performance and data safety.

- File Size: Supports file sizes up to 16 TB with a volume size limit of 1 exabyte.

- Delayed Allocation: Improves performance by delaying the allocation of blocks until the last possible moment.

- Extents: Uses a more efficient system called "extents" to manage large files, reducing fragmentation.

- Backward Compatibility: Ext4 is backward compatible with ext3 and ext2 file systems.

- Compatibility: Ext4 is native to Linux but can be read and written on Windows with third-party tools.

Key Differences:

- Performance: ext4 generally provides better performance for read/write operations on Linux due to its delayed allocation and extents, whereas NTFS is optimized for Windows with strong focus on reliability and security.

- Security: NTFS has more advanced security features, like ACLs and file encryption. ext4 focuses more on performance and efficiency.

- Compatibility: NTFS is highly integrated into the Windows ecosystem, while ext4 is native to Linux.

 # Multitasking in macOS

macOS provides multitasking through a combination of several underlying technologies and system processes that allow users to run multiple applications simultaneously without performance degradation.

1. Preemptive Multitasking:

- Definition: macOS uses preemptive multitasking, where the system's kernel decides which process gets CPU time and for how long. This ensures that multiple applications can run simultaneously by giving each process a share of CPU time based on priority.

- Benefit: It prevents a single application from monopolizing system resources, maintaining smooth performance.

2. Memory Management:

- macOS uses virtual memory to allow more applications to run than the actual physical memory (RAM) supports. Unused portions of applications are swapped out to disk, freeing up RAM for active tasks.

- It includes advanced memory compression techniques to optimize available RAM usage.

3. Task Scheduling:

- Scheduling algorithms determine which tasks are more important and assign them more CPU time. Less important tasks receive lower priority but are still managed to ensure system stability.

- macOS dynamically adjusts the priority of tasks based on user activity. For example, background tasks like file indexing are deprioritized when the user is actively interacting with an app.

4. App Nap:

- App Nap is a feature that conserves system resources by reducing the CPU time and energy consumption of background applications when they are not visible or being actively used. This improves multitasking efficiency and system performance.

5. Grand Central Dispatch (GCD):

- GCD is macOS's technology for managing concurrent operations . It simplifies running tasks concurrently, distributing work across multiple CPU cores, which helps applications perform better in multitasking scenarios.

6. Graphics and Window Management:

- macOS uses Quartz Compositor for rendering the graphical user interface. It allows each application to draw its window independently, improving the visual smoothness when switching between apps.

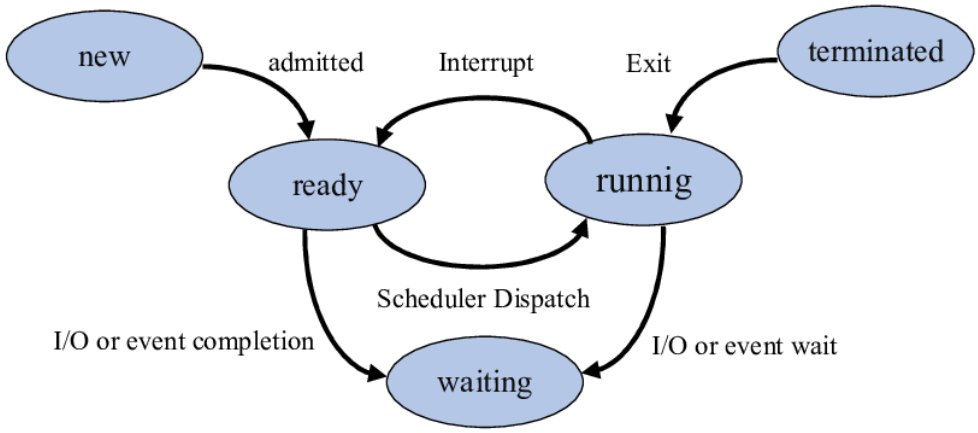
- Features like Mission Control and Spaces provide a graphical overview and efficient management of running applications, making multitasking more intuitive for users.

These mechanisms together ensure that macOS can efficiently handle multiple apps at once while optimizing performance and conserving resources.

Process Creation and Termination

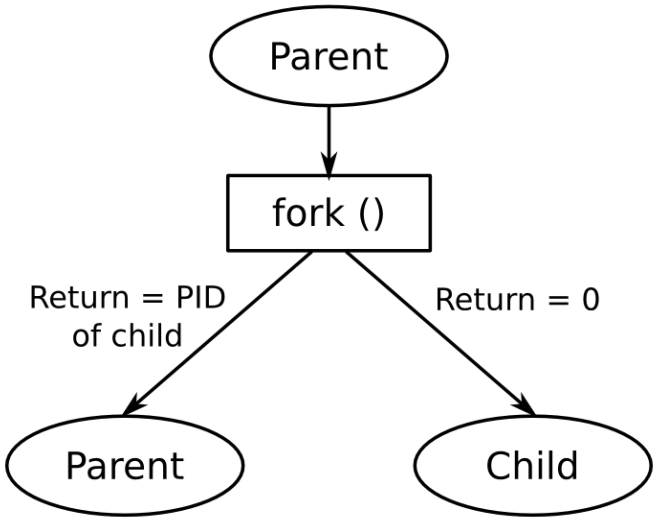
1. What is a Process?

- Definition: A process is an instance of a program in execution.



- Process States:
  - New: The process is being created.
  - Ready: The process is waiting to be assigned to a CPU.
  - Running: Instructions are being executed.
  - Waiting: The process is waiting for some event (e.g., I/O).
  - Terminated: The process has finished execution.

2. Process Creation



- Steps in Process Creation:

- System Call (e.g., `fork()` in UNIX): OS creates a new process.

- Parent-Child Processes: The parent process creates the child process. Both can run concurrently.

- Resource Allocation: OS allocates memory, CPU, and I/O to the new process.

- Example: In UNIX-like systems, a process creates a child using the `fork()` system call, followed by `exec()` to replace the process memory.

3. Process Termination

- Steps in Process Termination:

- Normal Termination: When a process completes execution.

- Abnormal Termination: When terminated due to errors or system calls like `kill`.

- Example: A web browser process terminates normally when you close the application, while a process like an unresponsive program can be terminated by the user via the Task Manager.

Linux/UNIX uses command-line tools to create processes (`fork()`), observe the process table (`ps`), and manually terminate processes (`kill`).

Preemptive Scheduling Techniques

1. What is Scheduling?

- Definition: CPU scheduling is the process by which the OS decides which process runs at any given time.

- Types of Scheduling:

- Preemptive: The OS can interrupt a running process to assign the CPU to another process.

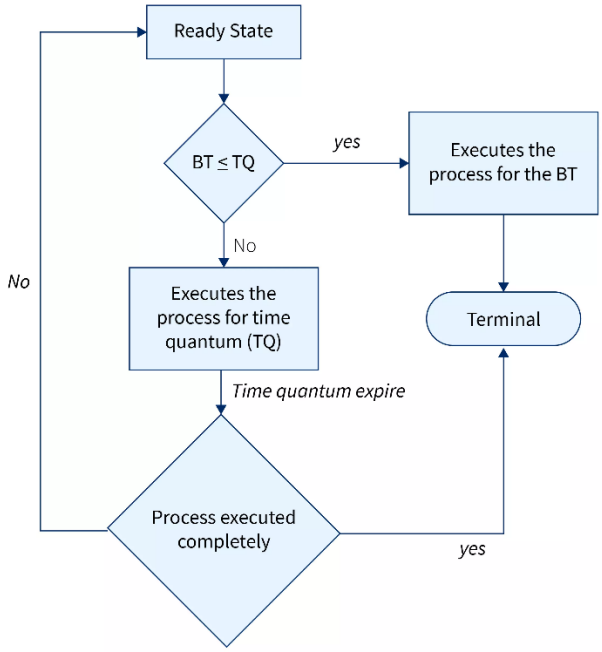
- Non-Preemptive: Once a process starts running, it keeps the CPU until it finishes or voluntarily yields it.

2. Preemptive Scheduling Techniques

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

Preemptive Scheduling

- Round Robin (RR):



- Each process is assigned a fixed time slice (quantum), and after that time, the process is preempted and placed at the back of the queue.

- Example: A time-sharing system where every running process gets 2 seconds of CPU time.

- Priority Scheduling:

- Processes are assigned a priority, and the CPU is allocated to the highest-priority process. Preemption occurs when a higher-priority process arrives.

- Example: In a real-time system, critical processes (e.g., emergency response systems) get priority over non-urgent tasks.

- Shortest Job Next (SJN):

- The CPU is assigned to the process that has the smallest execution time. Preemptive versions like Shortest Remaining Time First (SRTF) allocate the CPU to processes with the least remaining time.

- Example: Batch processing systems prioritize short jobs over long ones.

- Multilevel Queue Scheduling:

- Processes are grouped based on priority or type and allocated to different queues. The OS schedules between the queues with predefined policies.

- Example: OS might prioritize system processes over user applications.

3. Advantages and Disadvantages of Preemptive Scheduling

- Advantages:

- Ensures fair CPU time distribution.

- Suitable for real-time systems where tasks must be handled quickly.

- Reduces the risk of a single process monopolizing CPU resources.

- Disadvantages:

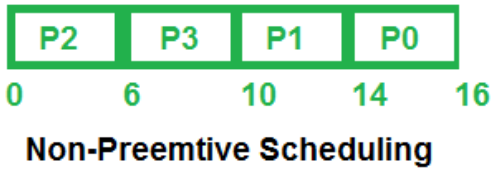
- Increased overhead due to frequent context switching.

- Potential for starvation if a low-priority process never gets CPU time.

Non-Preemptive Scheduling

In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4



Non-Preemtive Scheduling

Advantages of Non-Preemptive Scheduling

- It has a minimal scheduling burden.
- It is a very easy procedure.
- Less computational resources are used.

- It has a high throughput rate.

Disadvantages of Non-Preemptive Scheduling

- Its response time to the process is super.
- Bugs can cause a computer to freeze up.