Operations on Process

Operations on processes refer to the actions or activities that an operating system performs to manage processes during their lifecycle. Processes require management for efficient multitasking, resource allocation, and overall system performance. Here are the main **operations on processes** explained in detail:

1. Process Creation

- A new process is created when a program is launched. This process could be started by the user or by another process.
- When creating a process, the operating system:
  - Allocates resources (CPU time, memory, etc.).
  - Initializes the process control block (PCB) with information about the new process (like its process ID, state, etc.).
  - Places the new process into the "Ready" state.

Parent and Child Processes:

- The process that creates another process is called the **parent process**, and the newly created process is the **child process**.
- Example: A web browser (parent) can launch tabs or plugins (child processes).

2. Process Scheduling

- Since multiple processes compete for limited CPU time, the operating system decides **which process should run next** using scheduling algorithms (e.g., First-Come-First-Served, Round Robin).
- Process scheduling ensures fairness and efficient utilization of the CPU.

3. Process Termination

- A process is terminated when it completes its task or is explicitly stopped (e.g., by the user or the OS).
- When terminating a process, the OS:

- Frees up resources allocated to the process (memory, I/O devices, etc.).
- Removes the process from process tables and queues.

4. Process Suspension and Resumption

- Sometimes, a process needs to be temporarily paused (suspended) to free up resources or let other high-priority processes run.
- When the suspended process is ready to resume, the OS restores its PCB data and execution continues.

5. Context Switching

- A **context switch** occurs when the CPU switches from running one process to another.
- The OS saves the current process's state (e.g., program counter, register values) in its PCB and loads the next process's state from its PCB.
- Context switching ensures multitasking but comes with overhead (time required to save/load process states).

6. Inter-process Communication (IPC)

- Processes often need to **share information** or **communicate** with each other.
- IPC mechanisms include:
  - **Message Passing**: Processes exchange messages.
  - **Shared Memory**: A memory region is shared between processes.

7. Process Synchronization

- Synchronization ensures processes do not interfere with each other, especially when sharing resources.
- Example: If two processes need access to a shared file, synchronization prevents conflicts or data corruption.

8. Process Priority Changes

- The OS can change a process's priority to ensure important tasks get completed quickly.

- Example: A process responsible for handling user input might be given higher priority than a background update process.

These operations allow the operating system to manage processes efficiently, ensuring smooth multitasking and system stability.