Process states, Process Control Block (PCB)

Process States

A process in an operating system goes through several states during its lifecycle. These states represent the current status of the process, and common ones include:

1. New:

- A process enters the "New" state when it is created but hasn't started executing.
- At this stage, the operating system sets up everything needed for the process, such as allocating resources (memory, input/output settings, etc.).
- Example: You double-click a program icon, and the operating system begins preparing it to run.

2. Ready:

- After all preparations are complete, the process is ready to execute but has to wait its turn for the CPU.
- Processes in this state are stored in a "Ready Queue," a list of all tasks that are waiting for the CPU.
- Example: If multiple apps are open, the OS decides which one gets CPU time first. Others stay in the ready state.

3. Running:

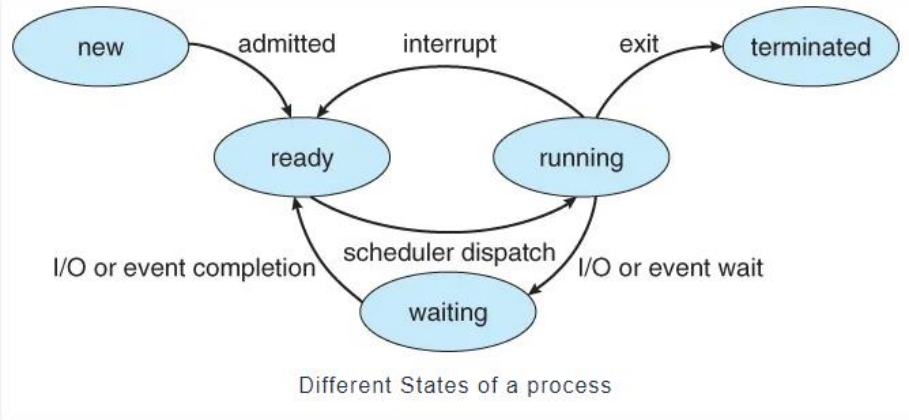
- The process moves from "Ready" to "Running" when the CPU starts working on it.
- During this time, the instructions of the process are actively executed step by step.
- This state lasts until:
  - The process finishes its job.
  - The CPU is needed by another higher-priority process.
  - The process needs to wait for something (e.g., input).
- Example: When you're typing in a word processor, its process is actively "Running."

4. Waiting (Blocked):

- If a process cannot continue running because it needs some event or resource to proceed (like waiting for data from a hard drive or a user's mouse click), it goes into the "Waiting" state.
- Once the required resource becomes available, the process moves back to the "Ready" state.
- Example: You save a file, and the app waits for confirmation that the file was saved successfully.

5. Terminated:

- After the process finishes its task, it enters the "Terminated" state. The operating system cleans up by freeing all resources the process was using (like memory).
- Example: Closing an app shuts down its processes, moving them to the terminated state.



Different States of a process

Process Control Block (PCB)

The **Process Control Block (PCB)** is a data structure used by the operating system to store all information related to a specific process. It acts as a repository for process-related data so the OS can track and manage processes efficiently. Key elements typically found in a PCB include:

1. **Process ID (PID)**:
    - Every process gets a unique ID to distinguish it from other processes.
    - The OS uses this ID to track the process and its activities.
    - Example: Imagine a group of students—each has a roll number to identify them.

2. **Process State**:
    - The PCB keeps a record of the process's current state (New, Ready, Running, Waiting, Terminated).
    - This helps the OS know what each process is currently doing.
3. **Program Counter**:
    - The program counter stores the location (or address) of the next instruction that the process is supposed to execute.
    - This ensures the process can resume correctly after being paused or interrupted.
4. **CPU Registers**:
    - These are small storage areas inside the CPU where the process temporarily keeps data while it's running.
    - When the CPU switches to another process, it saves the current process's registers in the PCB and restores the registers of the next process.
5. **Memory Information**:
    - Every process uses some memory, and the PCB stores details about its allocated memory (e.g., the start and end addresses of the memory block it occupies).
    - This ensures processes don't overwrite each other's data.
6. **I/O Status**:
    - If the process is interacting with input/output devices (like a keyboard or hard disk), the PCB keeps track of the devices and their status.
7. **Priority**:
    - Some processes are more urgent than others. The PCB stores the priority level to help the OS decide which process should run first.
8. **Accounting Information**:
    - This includes statistics like:
        - How much CPU time the process has used.
        - How long it has been running.
        - When it started, etc.
9. **Pointers**:
    - The PCB contains pointers to other related data structures, like queues or tables, that the operating system uses for scheduling and managing resources.

How it all Comes Together

Imagine the PCB as a file folder for each process. Inside the folder, you have everything about the process—where it’s at in the program, how much memory it’s using, what it’s waiting for, and more. When the OS switches between tasks, it grabs the PCB for the next process, checks all its info, and resumes it from where it left off. This allows multitasking, where multiple processes run smoothly without interfering with each other.

By managing **Process States** and storing everything in the **PCB**, the operating system ensures that your computer can work on many tasks at once—without crashing or losing track of what it's doing.