States of a Process:

When we double click on a program to run it, actually it goes through several stages before allowing us to use it. A process is actually a programming in execution using the computer resources such as memory, and CPU.

Actually, when a user opens a program, it becomes a process and is loaded into the main memory (RAM). In memory the process is divided into four parts:

a. **Stack**: This section stores the temporary information like function calls and local variables.
b. **Heap**: It is used for memory that the process requests while it is running. For example, if the program needs more memory to store extra data, then it gets it from the heap.
c. **Data Section**: It stores global variables and static variables, which are the data the program uses throughout its execution.
d. **Text Section**: It is where the actual instructions of the program are stored. It's the part of the memory where the program's instructions are loaded to be executed by the computer.

The Two-State Model of a Process:

It includes:

a. Running State: Means the process is using CPU to do its work.
b. Not Running State: That means process is not currently using the CPU. It could be waiting for something, like user input or data, or it might be paused.

When a program or process is created:

At first it is not using the CPU. The dispatcher checks if the CPU is free. Then if the CPU is free, then the dispatcher lets the process use the CPU and it moves it into the running state. When the CPU is available, the CPU scheduler decides which process gets its turn to run next.

There are five states model and seven states model as well.

How Does a Process Move From One State to Other State?

a. When a process is created, it's in a new state. It moves to the next state when the OS has allocated resources to it and ready to be executed.
b. When the CPU becomes available, the OS selects a process from the ready queue depending on the various scheduling algorithms and moves it to running state.
c. When a process needs to wait for something like user input from user, or an IO event to occur, it moves to the blocked state.
d. When a higher-priority process is being ready then the OS may pre-empt the running process and move it to the ready state.
e. When a process completes its execution, then it is terminated by the OS.

Multiprogramming

We have many processes ready to run. There are two types of multiprogramming:

- **Preemption:** Process is forcefully removed from CPU. Pre-emption is also called time sharing or multitasking.
- **Non-Preemption:** Processes are not removed until they complete the execution. Once control is given to the CPU for a process execution, till the CPU releases the control by itself, control cannot be taken back forcibly from the CPU.

Context Switching: When the operating system switches from executing one process to another, it must save the current process's context and load the context of the next process to execute. This is known as context switching.

Inter-Process Communication: Processes may need to communicate with each other to share data or coordinate actions. The operating system provides mechanisms for inter-process communication, such as shared memory, message passing, and synchronization primitives.

Process Synchronization: Multiple processes may need to access a shared resource or critical section of code simultaneously. The operating system provides synchronization mechanisms to ensure that only one process can access the resource or critical section at a time.

Features of The Process State

- A process can move from the running state to the waiting state if it needs to wait for a resource to become available.
- A process can move from the waiting state to the ready state when the resource it was waiting for becomes available.
- A process can move from the ready state to the running state when it is selected by the operating system for execution.
- The scheduling algorithm used by the operating system determines which process is selected to execute from the ready state.
- The operating system may also move a process from the running state to the ready state to allow other processes to execute.
- A process can move from the running state to the terminated state when it completes its execution.
- A process can move from the waiting state directly to the terminated state if it is aborted or killed by the operating system or another process.
- A process can go through ready, running and waiting state any number of times in its lifecycle but new and terminated happens only once.
- The process state includes information about the program counter, CPU registers, memory allocation, and other resources used by the process.
- The operating system maintains a process control block (PCB) for each process, which contains information about the process state, priority, scheduling information, and other process-related data.
- The process state diagram is used to represent the transitions between different states of a process and is an essential concept in process management in operating systems.

Frequently Asked Questions on States of a Process in Operating Systems – FAQs

When does a process enter the 'ready' state?

A process enters the 'ready' state when it is loaded into the main memory and is waiting to be assigned to a processor for execution. It is ready to run but is waiting for CPU time.

Can a process move directly from 'running' to 'ready' state?

Yes, a process can move from 'running' to 'ready' state if it is interrupted or preempted by a higher priority process, or if it voluntarily yields the CPU.