Operating System Week 3

Khaula Molapo

Year 2 Semester 2

1.

Imagine running multiple applications on Windows, such as a browser, music player, and Word document. Each application is a process, moving through different states as the operating system manages them. A process starts in the new state when created, then moves to ready when waiting for CPU allocation. When the CPU scheduler picks it, the process transitions to the running state, actively executing instructions. If it requests I/O, such as loading a webpage or saving a file, it moves to the waiting state until the I/O completes. Afterward, it returns to the ready queue, waiting for another chance to run. Once a process finishes, it enters the terminated state. These transitions ensure fairness and efficient multitasking. By managing states and switching processes quickly, Windows gives the illusion that all apps run simultaneously, though the CPU handles them one at a time. This scenario highlights the role of process management in real-world OS tasks.

2.

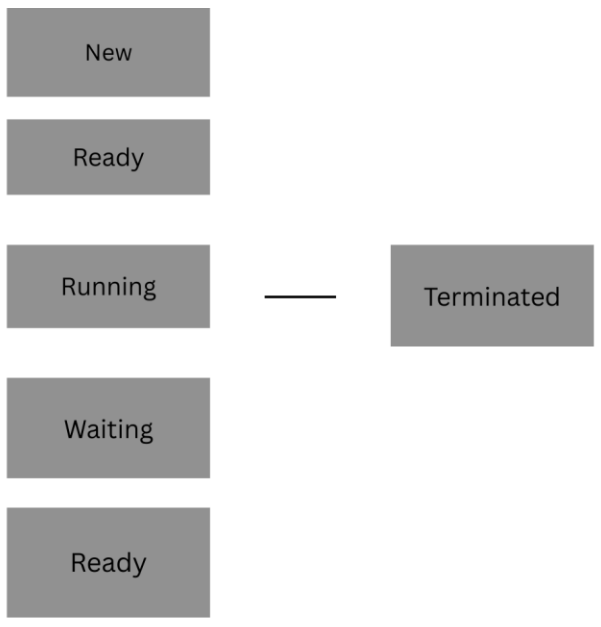
A Process Control Block (PCB) is a crucial data structure in an operating system that stores all the information about a process. It contains details such as process ID, current state (ready, waiting, running), CPU registers, scheduling information, memory management data, and open files. The PCB acts as the process's identity card, allowing the OS to pause, resume, or switch processes efficiently during context switching. Without the PCB, the OS could not track process execution. In practice, PCBs enable multitasking, allowing the CPU to save a process's context, load another, and later resume the first exactly where it left off.

3.

Using VirtualBox to run Ubuntu gave me hands-on experience with process lifecycle management. I opened the terminal and ran the top command, which displayed all active processes, their IDs, CPU usage, and memory consumption. Observing processes in real time showed how some are in the running state, while others are waiting or sleeping. I selected one process and terminated it using the kill command, immediately removing it from the list and demonstrating its transition to the terminated state. This exercise made the process lifecycle tangible, showing how the OS dynamically manages and schedules processes. It highlighted the importance of monitoring, resource allocation, and state transitions. Overall, working with real processes reinforced theoretical knowledge, bridging the gap between OS concepts and practical tasks.

4.

The diagram of process states shows the common transitions an operating system manages. A process starts in the new state, then moves to ready when prepared for execution. The scheduler selects it for the running state, allowing instructions to be executed. If the process requests I/O, it enters the waiting state until the operation completes, after which it returns to the ready queue. When execution finishes, the process reaches the terminated state. Arrows in the diagram illustrate how processes move between states based on events or system decisions. Visualizing these states clarifies how the OS ensures efficiency, fairness, and resource utilization, making multitasking possible. The diagram highlights the dynamic nature of process management and how the OS maintains responsiveness.


New
Ready
Running
Terminated
Waiting
Ready