Lecture Notes: Operating Systems - Simplified : Lec 3

# Process

A process is an instance of a program that is being executed. It consists of program code and activity. A process can have multiple threads of execution that share process resources. Each process operates independently and is managed by the operating system's scheduler.

# Program

A program is a set of instructions written in a specific language to perform a particular task. It is a static file on disk, and when it is executed, it becomes a process

# Threads

A thread is the smallest unit of a process that can be scheduled for execution by the operating system. A process can have multiple threads, all sharing the same memory space but running independently. Threads within a process allow parallel execution of tasks within the same program and are often used to improve the efficiency and responsiveness of programs.

# Multitasking v/s Multiprogramming

## 1. Multitasking

Refers to the ability of an operating system to execute more than one task simultaneously. This is typically achieved by time-sharing, where the CPU switches rapidly between tasks.

## 2. Multiprogramming

Refers to the technique where multiple programs are loaded into memory simultaneously, and the CPU executes them in turns. In multiprogramming, one program runs while another is waiting for I/O operations to complete.

# Context Swicthing

Context switching refers to the process by which the operating system saves the current state of a running process or thread and restores the state of another one. It allows the CPU to switch between multiple processes or threads. This involves saving the register values, memory state, and other CPU state of the currently running task and restoring the same for the next task.

## 1. Context switching in Processes

Context switching between processes is more costly compared to threads because the operating system must not only save and restore CPU registers and program counters but also change memory mappings. Each process operates in its own memory space, and the switch requires updating the memory management unit (MMU) for the new process.

## 2. Context switching in Threads

Context switching between threads is typically lighter and faster than switching between processes because threads share the same memory and resources. The CPU only needs to switch the execution state, such as the program counter and register values. There is no need to change memory space, making thread context switching more efficient.