Processes and Threads in Operating Systems

Key Concepts and Mechanisms

Abdullah

233003

Introduction to Processes

- **Definition:**
- A process is a program in execution.
- It consists of program code, data, and system resources.
- **Importance:**
- Allows multitasking and efficient resource utilization.
- Each process has its own memory space and execution context.

Process Relationship

- **Parent Process:** The original process that creates another process.
- **Child Process:** A new process created by the parent.
- **Process Creation Methods:**
  - **Fork() (**UNIX-Based System)
  - **CreateProcess()** (Windows)
- **Example:**
  - A web browser (parent) launches multiple tabs (child processes).

Different States of a Process

- **New:** Process is created but not yet executed.
- **Ready:** Process is waiting for CPU allocation.
- **Running:** Process is executing instructions on the CPU.
- **Waiting:** Process is waiting for an event (e.g., I/O completion).
- **Terminated:** Process has completed execution or been stopped.

Process State Transitions

- **New → Ready:** Process is admitted into the system.
- **Ready → Running:** CPU scheduler selects a process to execute.
- **Running → Waiting:** Process waits for an event (e.g., I/O operation).
- **Waiting → Ready:** Event occurs, process moves to ready state.
- **Running → Terminated:** Process finishes execution or is forcefully stopped.

Process Control Block (PCB)

- The **PCB** stores process-related information, including:
  - **Process ID (PID)** – Unique identifier.
  - **Process State** – Ready, running, waiting, etc.
  - **CPU Registers** – Stores execution details.
  - **Memory Information** – Tracks allocated memory.
  - **I/O Status** – Keeps input/output details.
- Role of PCB:
  - OS uses PCB to track process execution.
  - Essential for context switching.

Context Switching

- **Definition:**
- The process of saving the state of one process and loading another.

Steps:

1.Save the current process’s PCB.

2.Load the next process’s PCB.

3.Resume execution of the new process.

- **Importance:**
- Enables multitasking.
- Ensures fair CPU utilization.

Introduction to Threads

- Definition:
  - A thread is the smallest execution unit within a process.
  - Threads share process resources (memory, files).
- **Difference between Process and Thread:**
  - A process has independent resources, while threads share resources.
- **Example:**
  - A web browser runs multiple threads for different tasks (loading pages, handling user input).

Various States of a Thread

- **New:** Thread is created but not yet started.
- **Runnable:** Thread is ready to run.
- **Running:** Thread is executing instructions.
- **Waiting:** Thread is paused, waiting for an event.
- **Terminated:** Thread has completed execution.

Benefits of Threads

- **Faster Context Switching:** Threads are lightweight and switch quickly.
- **Efficient Resource Sharing:** Threads share memory and data of the process.
- **Parallel Execution:** Enables multitasking within the same process.
- **Improved Performance:** Reduces execution time by running multiple tasks concurrently.

Types of Threads

- **User-Level Threads:**
  - Managed by user-level libraries.
  - Faster but lack OS support.
  - Example: Java Threads.

- **Kernel-Level Threads:**
  - Managed by the OS kernel.
  - More powerful but slower.
  - Example: Linux Kernel Threads.

Multithreading in OS

- **Definition:** Running multiple threads within a single process.
- **Types:**
  - **Multithreaded User Applications:** Web browsers, video players.
  - **Kernel Multithreading:** OS-level parallel execution.

- **Example:**
- A video player runs separate threads for decoding, audio processing, and video rendering.

Conclusion

- Processes and threads improve multitasking and system performance.
- Understanding states and transitions helps in OS optimization.
- Multithreading enhances efficiency and responsiveness.
- **Quote:** *“Efficient process and thread management is the backbone of modern operating systems.”*