Overview of Process Management in Operating Systems

Y. Judith Petrizia[1], R. Sujitha[2], Sree Saradha M[3]
Assistant Professor, Department of MCA[1,2,3]
T J Institute of Technology, Chennai, India

Abstract: *Process management is a fundamental aspect of operating systems that ensures efficient execution, scheduling, and coordination of multiple processes. It involves creating, executing, suspending, and terminating processes while managing system resources such as CPU time and memory. The operating system maintains various process states—New, Ready, Running, Waiting, and Terminated—to track process execution. Key components of process management include process scheduling, context switching, and inter-process communication (IPC), all of which contribute to system responsiveness and multitasking. Efficient process management enhances overall system performance, optimizing resource utilization and enabling smooth concurrent execution of applications.*

Keywords: operating systems

I. INTRODUCTION

What is a Process?
A **process** refers to an instance of a program that is being executed by the computer. It is created when a program is run, either manually by a user or programmatically by another process.
In UNIX and similar operating systems, this concept is fundamental to how the system handles multitasking.
Processes can be created in different ways:

1. **User Initiated**: The user can run a program manually (e.g., by typing a command in the terminal).
2. **Program Initiated**: One process can start another process, often referred to as a "child process."

Process Management:
It is an important part of the operating system. It allows you to control the way your computer runs by managing the currently active processes.
This includes ending processes that are no longer needed, setting process priorities, and more. You can do it on your computer also.
There are a few ways to manage your processes. The first is through the use of Task Manager. This allows you to see all of the processes currently running on your computer and their current status and CPU/memory usage.
You can end any process that you no longer need, set a process priority, or start or stop a service

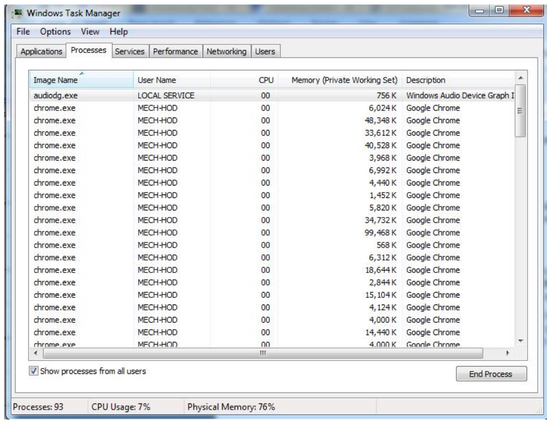

Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
End Process

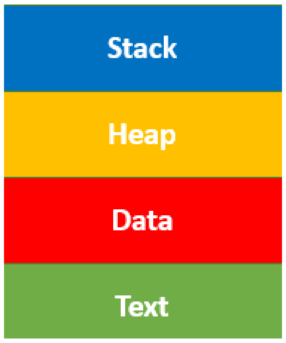
Image Name	User Name	CPU	Memory (Private Working Set)	Description
audiodg.exe	LOCAL SERVICE	00	756 K	Windows Audio Device Graph I
chrome.exe	MECH-HOD	00	6,024 K	Google Chrome
chrome.exe	MECH-HOD	00	48,348 K	Google Chrome
chrome.exe	MECH-HOD	00	33,612 K	Google Chrome
chrome.exe	MECH-HOD	00	40,528 K	Google Chrome
chrome.exe	MECH-HOD	00	3,968 K	Google Chrome
chrome.exe	MECH-HOD	00	6,992 K	Google Chrome
chrome.exe	MECH-HOD	00	4,440 K	Google Chrome
chrome.exe	MECH-HOD	00	1,452 K	Google Chrome
chrome.exe	MECH-HOD	00	5,820 K	Google Chrome
chrome.exe	MECH-HOD	00	34,732 K	Google Chrome
chrome.exe	MECH-HOD	00	99,468 K	Google Chrome
chrome.exe	MECH-HOD	00	568 K	Google Chrome
chrome.exe	MECH-HOD	00	6,312 K	Google Chrome
chrome.exe	MECH-HOD	00	18,644 K	Google Chrome
chrome.exe	MECH-HOD	00	2,844 K	Google Chrome
chrome.exe	MECH-HOD	00	15,104 K	Google Chrome
chrome.exe	MECH-HOD	00	4,124 K	Google Chrome
chrome.exe	MECH-HOD	00	4,000 K	Google Chrome
chrome.exe	MECH-HOD	00	14,440 K	Google Chrome
chrome.exe	MECH-HOD	00	4,000 K	Google Chrome

[✓] Show processes from all users

Processes: 93 CPU Usage: 7% Physical Memory: 76%

The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

Process Architecture



- **Stack:** The Stack stores temporary data like function parameters, returns addresses, and local variables.
- **Heap** Allocates memory, which may be processed during its run time.
- **Data:** It contains the variable.
- **Text:**
  Text Section includes the current activity, which is represented by the value of the Program Counter.

The Role of the Operating System:
The operating system is the backbone of your computer. It’s responsible for managing all of your computer’s processes and making it possible for you to interact with your device. When you boot up your computer, the operating system is the first thing that starts running.
It loads into memory and starts managing all of the other programs and processes running on your computer. It also controls files and devices, allocates system resources, and handles communications between applications and users.

Process Control Block:

❖ A **process control block** (**PCB**), also sometimes called a **process descriptor**, is a data structure used by a computer operating system to store all the information about a process.

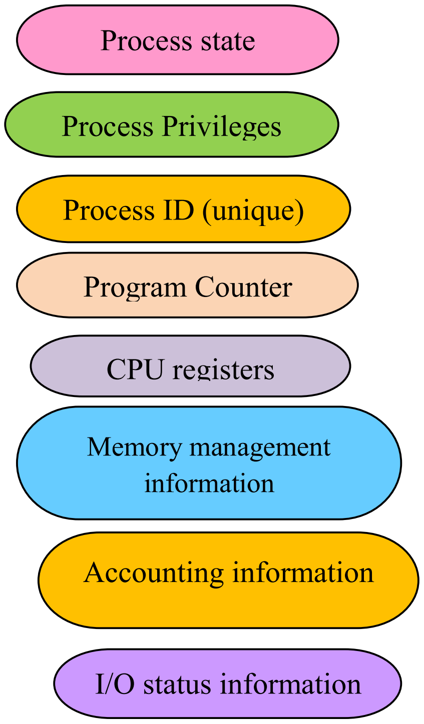
❖ When a process is created (initialized or installed), the operating system creates a corresponding process control block, which specifies and tracks the process state (i.e. new, ready, running, waiting or terminated). Since it is used to track process information, the PCB plays a key role in context switching.

❖ An operating system kernel stores PCBs in a process table.

❖ The current working directory of a process is one of the properties that the kernel stores in the process's PCB.

Role
The role of the PCBs is central in process management: they are accessed and/or modified by most utilities, particularly those involved with scheduling and resource management.

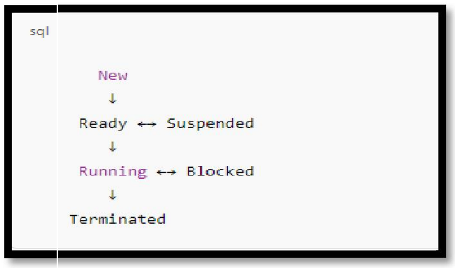
Structure
In multitasking operating systems, the PCB stores data needed for correct and efficient process management. Though the details of these structures are system-dependent, common elements fall in three main categories:



Attributes of Process:

Process State :

The Process State represents the current status of a process in an operating system . The Process states are new, ready, running, blocked- waiting, Terminated - Exit , Suspended –(Swapped → Optional this state)


```sql
   New
    ↓
  Ready ↔ Suspended
    ↓
  Running ↔ Blocked
    ↓
Terminated
```

Process Privileges:

A **process privilege** refers to the level of access and control a process has over system resources. Operating systems enforce privilege levels to ensure security, stability, and proper resource management.

Types of Process Privileges:

1. User Mode vs. Kernel Mode
2. Privilege Levels in Process Execution
3. Process Permissions and Access Control

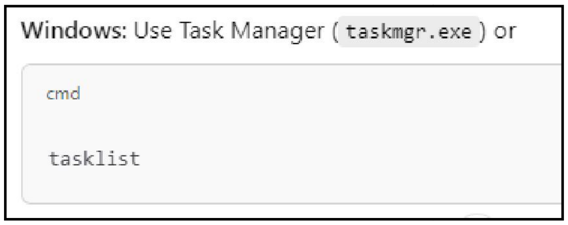
Process Number (PID)

It represents a unique numerical ID assigned by the operating system to each running process. It helps the OS manage and track processes efficiently.

How Process Numbers Are Assigned

- When a process is created (via fork() in Linux or Create Process() in Windows), the OS assigns it a unique PID.
- PIDs are typically assigned sequentially, but they can wrap around when the maximum limit is reached.

Finding the Process Number (PID)



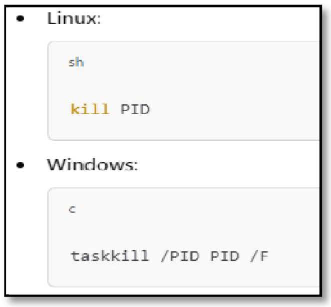
Windows: Use Task Manager (`taskmgr.exe`) or

```cmd
tasklist
```

Special Process Numbers

- **PID 0**: The scheduler process (also known as the "swapper" or "idle task").
- **PID 1**: The first process, typically init (Linux) or system, responsible for starting all other processes.
- **PID 2+**: Other system and user processes.

Killing a Process Using Its Number:



- Linux:

```sh
kill PID
```

- Windows:

```c
taskkill /PID PID /F
```

Program Counter (PC):

The **Program Counter (PC)** is a special-purpose CPU register that holds the **memory address of the next instruction** to be fetched and executed. It plays a crucial role in the execution cycle of a process.

Functions of the Program Counter:

1. **Stores the Next Instruction Address**
   - The PC keeps track of the location of the next instruction in the program's memory.
2. **Increments Automatically**
   - After fetching an instruction, the PC automatically increases (usually by 1 or the instruction size in bytes).
   - Example: In a system with a 4-byte instruction size, the PC increments by 4.
3. **Controls Sequential Execution**
   - Ensures instructions are executed in the correct order, unless altered by branching, loops, or jumps.
4. **Handles Branching & Jump Instructions**
   - If a **jump, call, or return** instruction is executed, the PC updates to the new address instead of incrementing sequentially.
   - Example: A conditional **if-else** statement may alter the PC value based on a condition.

5. **Interrupt Handling**
   - When an interrupt occurs (e.g., from hardware or software), the PC saves the address of the next instruction before jumping to the interrupt service routine (ISR).
   - After handling, execution resumes from the saved address.

Role of the Program Counter in the Instruction Cycle:
Fetch -- The instruction at the address stored in the PC is fetched from memory.
Decode -- The instruction is decoded by the CPU.
Execute -- The CPU executes the instruction.
Update PC -- The PC moves to the next instruction, or jumps if a branch occurs.

CPU Registers:
CPU **registers** are small, high-speed memory locations inside the processor used to store data, instructions, and addresses for quick access. They help improve execution speed by reducing the need to fetch data from slower main memory (RAM).

Types of CPU Registers:
General-Purpose Registers (GPRs) :
Used to store temporary data and intermediate results during execution.

Special-Purpose Registers :
These registers have dedicated functions:
- **Accumulator Register (ACC) →** Stores arithmetic and logical operation results.
- **Program Counter (PC) →** Holds the address of the next instruction to be executed. Automatically increments after each instruction fetch.
- **Instruction Register (IR)** ->Stores the current instruction being executed. The CPU decodes and executes instructions from this register.
- **Stack Pointer (SP) →**Points to the top of the stack in memory. Used for function calls, storing return addresses, and managing local variables.
- **Base Pointer (BP) / Frame Pointer (FP) →**Points to the base of the current stack frame in memory. Helps in accessing function parameters and local variables.
- **Memory Address Register (MAR)→**Holds the address of memory from which data is to be fetched or where data is to be stored.
- **Memory Data Register (MDR)→**Holds the actual data being transferred between the CPU and memory.
- **Status Register / Flag Register→**Stores condition flags that indicate the result of an operation.

Memory Management Information :
This includes the memory information like information related to the Segment table, memory limits, and page table.
Functions of Memory Management
1. **Allocation & De-allocation**
   - Assigns memory to processes when needed.
   - Frees memory when a process terminates.
2. **Tracking Memory Usage**
   - Keeps records of which memory blocks are free or in use.
3. **Process Isolation**
   - Prevents one process from accessing another process's memory, ensuring security and stability.
4. **Swapping & Paging**
   - Moves processes between RAM and disk (virtual memory) to manage limited physical memory.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 2, February 2025

Types of Memory Management Techniques

1) **1. Contiguous Memory Allocation**
   - **Single Partition**: The entire memory is given to one process.
   - **Fixed Partitioning**: Memory is divided into fixed-size blocks.
   - **Dynamic Partitioning**: Partitions are created dynamically based on process needs.
   - **Issues**: Internal and external fragmentation.
2) **2. Paging:**
   - Divides memory into fixed-size pages and loads them into frames in RAM.
   - Eliminates external fragmentation but may have internal fragmentation.
   - Uses a **page table** to map logical addresses to physical addresses.
3) **3. Segmentation:**
   - Divides memory into variable-sized segments based on logical divisions (code, data, stack).
   - Provides flexibility but can cause external fragmentation.
4) **4. Virtual Memory:**
   - Uses **paging and swapping** to simulate more memory than physically available.
   - Helps run large applications on limited RAM.
   - Uses a **page replacement algorithm** (e.g., FIFO, LRU).

Accounting information:
This includes the information about the amount of CPU used for process execution, execution time, time limits, etc.

Purpose of Accounting Information
- **Resource Usage Tracking** – Monitors CPU time, memory, disk usage, and network activity.
- **Billing & Cost Allocation** – Used in cloud computing and multi-user environments to charge users based on usage.
- **Performance Analysis** – Helps administrators optimize system performance.
- **Security & Auditing** – Detects unauthorized access and potential security threats.
- **Process Monitoring** – Tracks active and completed processes for debugging and troubleshooting.

I/O Status Information:
I/O (Input/output) status information refers to the data maintained by the operating system about the state and performance of input/output operations.

Importance of I/O Status Information
- **Tracks Device Activity** – Determines if a device is active, idle, or malfunctioning.
- **Optimizes Performance** – Helps in efficient scheduling of I/O requests.
- **Detects Errors** – Logs hardware failures, access issues, and communication problems.
- **Ensures Fair Resource Allocation** – Prevents one process from monopolizing I/O resources.
- **Enhances Security** – Logs I/O activities for auditing and troubleshooting.

Key Components of I/O Status Information
a) Device Status
- Indicates whether the device is **ready, busy, or in an error state**.

b) I/O Queue Status
- Shows **pending I/O requests** waiting to be processed and helps optimize **I/O scheduling algorithms** (FCFS, SSTF, SCAN, etc.).

c) Data Transfer Rate

- Speed at which data is read/written (e.g., HDD = **100 MB/s**, SSD = **500 MB/s**).

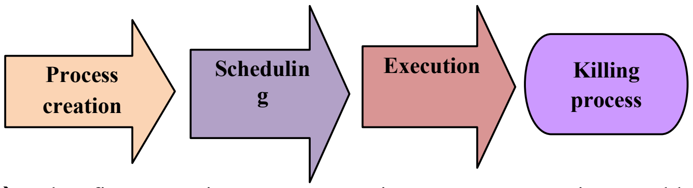
d) Error Logs & Interrupts

Detects issues like **disk bad sectors, printer errors, or network failures** and **interrupts** to signal when an operation is complete or an error occurs.

e) Buffer and Cache Status

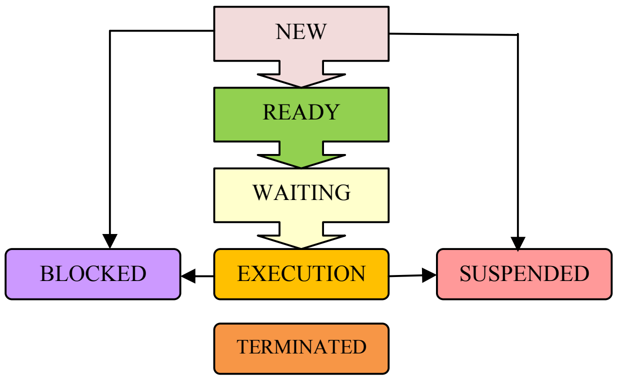
- Monitors temporary storage of data during I/O operations to **smooth data flow**.

Process Operations:



1. **Process creation** → The first step is process creation. Process creation could be from a user request (using fork()), a system call by a running process, or system initialization.
2. **Scheduling** → If the process is ready to get executed, then it will be in the ready queue, and now it's the job of the scheduler to choose a process from the ready queue and starts its execution
3. **Execution** → The process has started executing, it can go into a waiting queue or blocked state. Maybe the process wants to make an I/O request, or some high-priority process comes in.
4. **Killing the process** → After process execution, the operating system terminates the Process control block(PCB).

States of the process:



New – The process is being created.
Ready – The process is waiting for CPU allocation.
Running – The process is currently executing.
Blocked (Waiting) – The process is waiting for I/O or an event to complete.
Suspended: the time when a process is ready for execution but has not been placed in the ready queue by OS.

Content Switching in an Operating System
Definition:
Content switching is the process of saving and restoring the state (or context) of a CPU so that multiple processes can share a single CPU efficiently.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.67

Volume 5, Issue 2, February 2025

Why is Context Switching Needed?

1. **Multitasking** – The OS needs to switch between multiple processes to provide concurrent execution.
2. **Process Scheduling** – When a higher-priority process arrives, the CPU must switch to it.
3. **Interrupt Handling** – The CPU may need to pause a process to handle interrupts (like I/O operations).
4. **Time-Sharing Systems** – In a multi-user environment, the CPU switches between processes to ensure fair resource distribution.

Steps in Context Switching

1. **Save the Current Process State**
   - The OS saves the CPU registers, program counter, and other process-specific data.
2. **Select the Next Process**
   - The OS scheduler picks the next process to execute based on scheduling algorithms (e.g., Round Robin, Priority Scheduling).
3. **Load the New Process State**
   - The saved state of the selected process is restored, and execution resumes.

Overhead of Context Switching

- Context switching is **time-consuming** because saving and restoring process states takes CPU cycles.
- Excessive switching can lead to **reduced performance** due to frequent state saving and loading.
- Minimizing unnecessary switches is important for system efficiency.

II. CONCLUSION

Process management plays a crucial role in the efficient operation of an operating system by handling multiple processes, and ensuring resource allocation, and maintaining system stability. It includes aspects of execution, and PCB with diagrams, as well as we have explained about process of CPU and its types well explained. Effective process management improves system performance, responsiveness, and resource utilization, making it essential for modern computing environments. As technology advances, process management continues to evolve, integrating more efficient scheduling algorithms and optimizations to enhance user experience and system efficiency.

REFERENCES

[1]. www.google.com
[2]. www.scribd.com
[3]. www.studysmarter.co.uk
[4]. www.guru99.com
[5]. Operating Systems: A Concept-Based Approach, D M Dhamdhere
[6]. Operating Systems: Internals and Design Principles, William Stallings