THE UNIVERSITY OF AZAD JAMMU AND KASHMIR, MUZAFFARABAD

Department of Software Engineering

Submitted to:	Engr. Sidra Rafique
Course Title:	Operating System
Course Code:	CS-2105
Session:	2024-2028
Lab No:	04
Roll No:	2024-SE-15
Submitted By:	Shahzad Ahmed Awan
Submission date:	November 1, 2025

Table of Contents

Table of Figures

Lab 04-Understanding Processes and Process Control Block (PCB)

1. Objective

The aim of this LAB is to understand how processes work in an operating system, how they are represented through the Process Control Block (PCB), and how to identify process IDs (PIDs) using both Command Prompt (cmd) and Task Manager.
This LAB also explores the different process states through a state transition diagram.

2. Background

A process is a program in execution — an active entity that includes the program code, the program counter, stack, data, and allocated system resources.
The Process Control Block (PCB) is a critical data structure that stores information about each process such as its ID, state, memory details, and CPU registers.

When multiple processes run, the operating system maintains each PCB to manage scheduling, execution, and switching between processes efficiently.

3. Task 1 – Process Creation and Observation

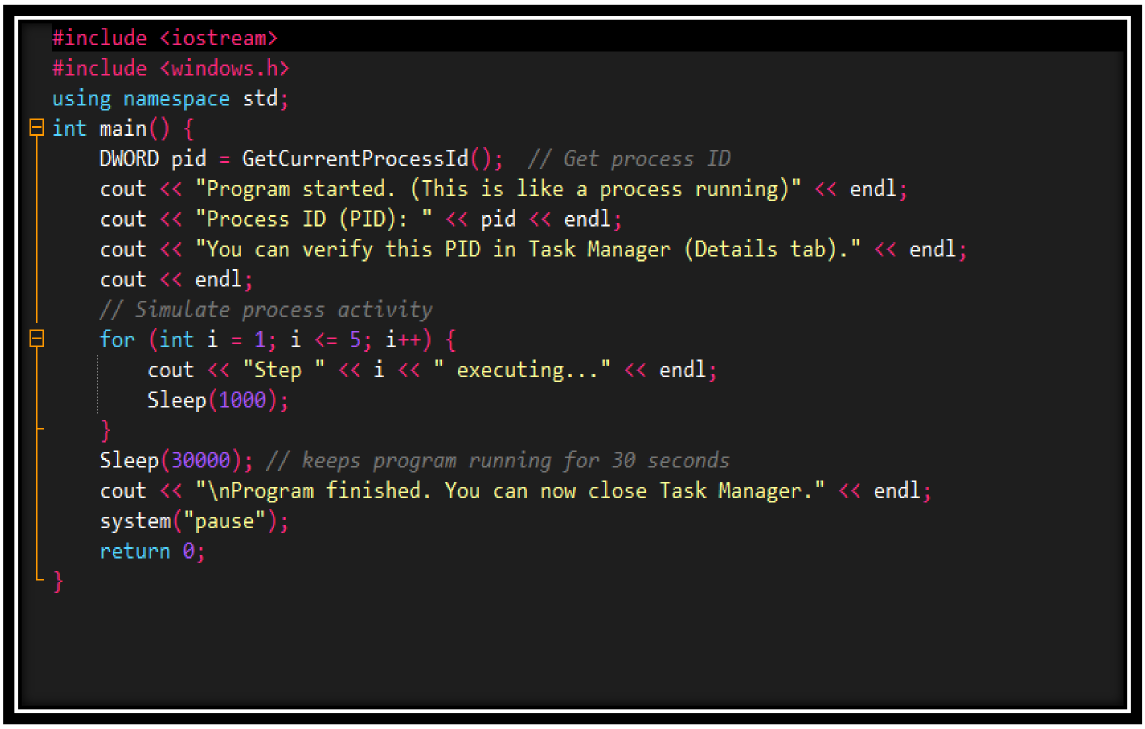
3.1 Aim

To execute a simple C++ program that simulates a process and to observe it in the Task Manager and Command Prompt.

3.2 Procedure

1. Open Dev-C++ or any IDE and write the following program.
2. Compile and run it to generate the `.exe` file.
3. While the program is running:
    - Open Task Manager → Details tab, locate your program using its Process ID (PID).
    - Or, open Command Prompt and type:
    - `tasklist | find "YourProgramName.exe"`
4. Observe the process entry and its PID appearing in the list.

4. Source Code



```cpp
#include <iostream>
#include <windows.h>
using namespace std;
int main() {
    DWORD pid = GetCurrentProcessId();  // Get process ID
    cout << "Program started. (This is like a process running)" << endl;
    cout << "Process ID (PID): " << pid << endl;
    cout << "You can verify this PID in Task Manager (Details tab)." << endl;
    cout << endl;
    // Simulate process activity
    for (int i = 1; i <= 5; i++) {
        cout << "Step " << i << " executing..." << endl;
        Sleep(1000);
    }
    Sleep(30000); // keeps program running for 30 seconds
    cout << "\nProgram finished. You can now close Task Manager." << endl;
    system("pause");
    return 0;
}
```

Figure 1: Source Code for Program with assigned PID

5. Task 2 – Process States and PCB

5.1 Objective

To understand and illustrate different process states in the operating system and their transitions.

5.2 Explanation of Process States

- **New:** Process creation and resource allocation.
- **Ready:** Process waiting in the queue for CPU time.
- **Running:** CPU actively executing the process instructions.
- **Waiting/Blocked:** Process temporarily halted due to I/O or event wait.
- **Terminated:** Process has completed execution and its PCB is destroyed.

5.3 Flowchart Diagram

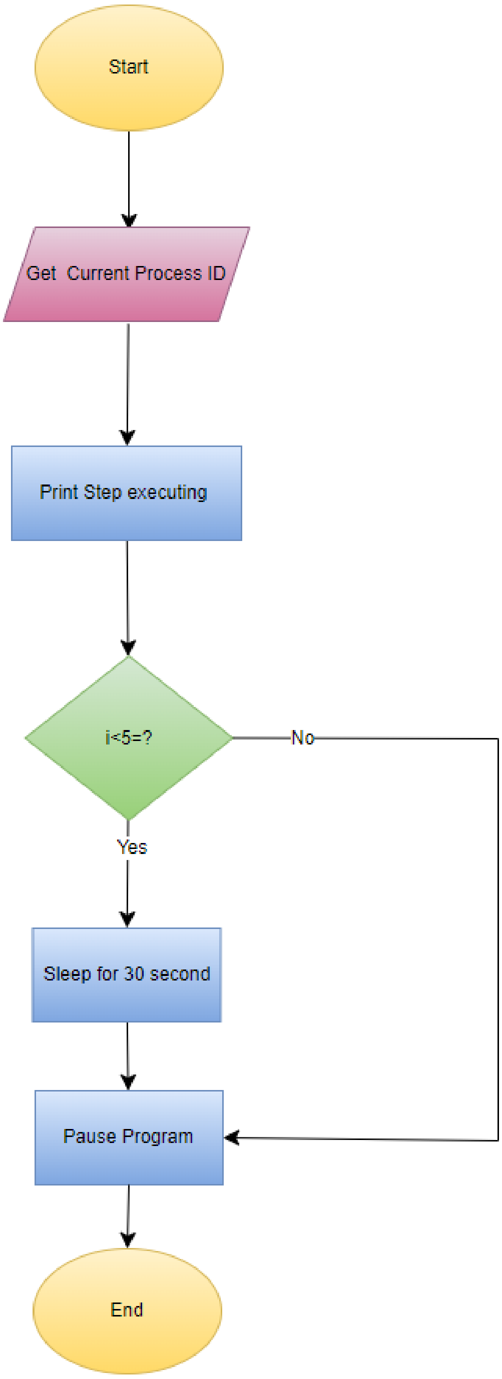
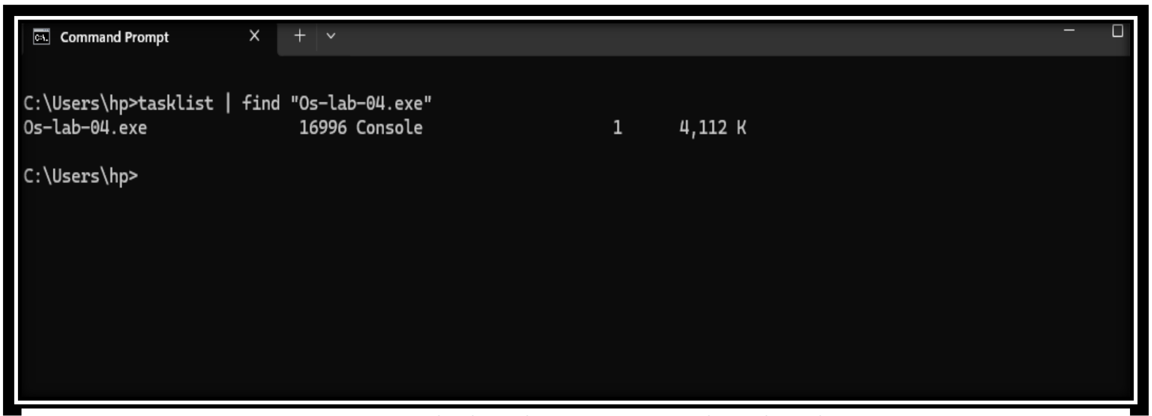


Figure 2: Flowchart for the Program

6. Observations

- Each running application corresponds to a process with a unique Process ID (PID).
- The Task Manager and CMD provide real-time information about process activity.

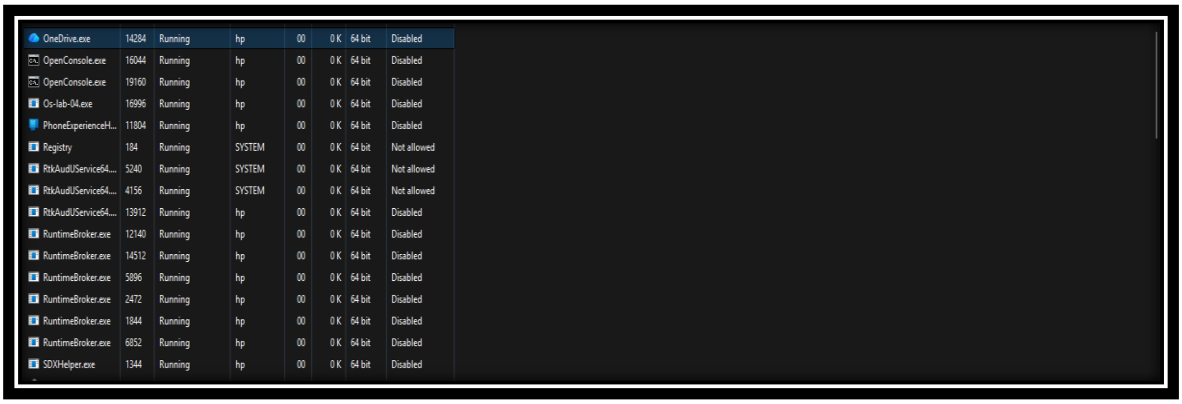
- During `Sleep()` calls, CPU usage drops, showing how the scheduler handles idle processes.
- The PCB maintains vital information allowing the OS to resume or switch between processes efficiently.



```
C:\Users\hp>tasklist | find "Os-lab-04.exe"
Os-lab-04.exe                16996 Console                    1      4,112 K

C:\Users\hp>
```

Figure 4: Checking the Process Status through cmd



OneDrive.exe	14284	Running	hp	00	0 K	64 bit	Disabled
OpenConsole.exe	16044	Running	hp	00	0 K	64 bit	Disabled
OpenConsole.exe	19160	Running	hp	00	0 K	64 bit	Disabled
Os-lab-04.exe	16996	Running	hp	00	0 K	64 bit	Disabled
PhoneExperienceH...	11804	Running	hp	00	0 K	64 bit	Disabled
Registry	184	Running	SYSTEM	00	0 K	64 bit	Not allowed
RtkAudUService64....	5240	Running	SYSTEM	00	0 K	64 bit	Not allowed
RtkAudUService64....	4156	Running	SYSTEM	00	0 K	64 bit	Not allowed
RtkAudUService64....	13912	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	12140	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	14512	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	5896	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	2472	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	1844	Running	hp	00	0 K	64 bit	Disabled
RuntimeBroker.exe	6852	Running	hp	00	0 K	64 bit	Disabled
SDXHelper.exe	1344	Running	hp	00	0 K	64 bit	Disabled

Figure 3: Finding the Process in the Task Manager

7. Conclusion

This LAB demonstrated how processes are created, executed, and managed in an operating system.
The experiment provided hands-on experience in identifying process IDs and understanding

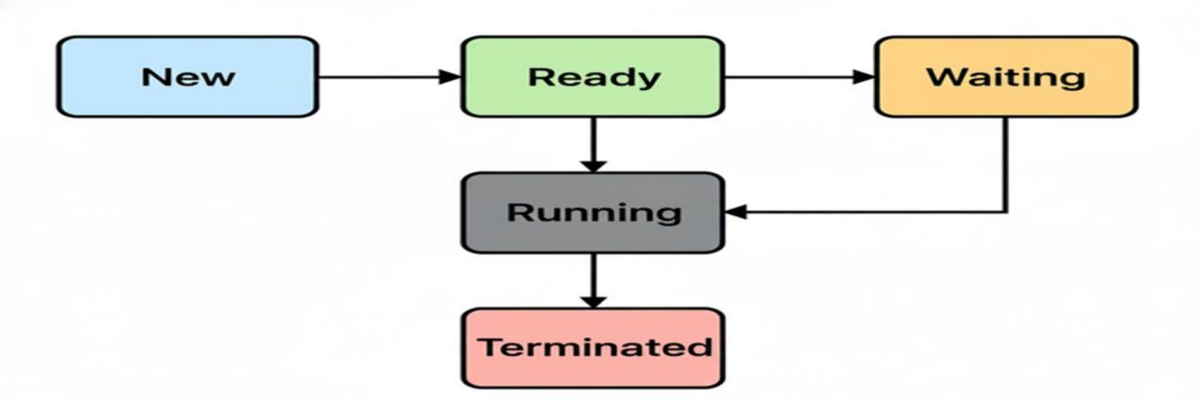


Figure 5: Diagram demonstrating different status a process can have

their behavior in Task Manager.
It also reinforced theoretical knowledge about PCB structure and process state transitions, linking system-level concepts with practical visualization.

8. Reflection

Through this LAB, I learned and practiced:

- How the operating system assigns and manages Process IDs.
- How to use Command Prompt and Task Manager to monitor running processes.
- The significance of Process Control Blocks (PCB) in maintaining process information.
- How a process moves through different states (New, Ready, Running, Waiting, Terminated).
- How process scheduling and waiting are represented practically using simple code simulation.

This LAB deepened my understanding of real-time process management and how the OS ensures coordination between multiple executing programs.