

***University of Central Punjab***

**Name:**

**Asfia 209**

**Iqra 064**

**Class:**

**BSCS**

**Section: B**

**Assignment No: 4**

**Submission Date:**

**27\_02\_2025**

**Submitted To:**

**Prof: Hassan Rauf**

**Process Scheduling using FCFS in Windows**

**1. Introduction**

This project implements First-Come, First-Served (FCFS) Scheduling using Windows API for process management. It simulates multiple tasks, records execution times, and calculates scheduling metrics such as turnaround time and waiting time.

**Purpose and Objectives:**

* To demonstrate process scheduling in an operating system.
* To implement FCFS scheduling, ensuring processes execute in the order they arrive.
* To explore process creation, execution, and termination using system-level programming.
* Importance in Real-World Applications:
* FCFS is widely used in batch processing and simple scheduling scenarios.
* It serves as a fundamental scheduling algorithm used in OS design.
* Helps in understanding how the OS handles process execution in a sequential manner.

**2. Project Scope**

**Features and Functionalities:**

* Accepts user input for the number of tasks and their execution times.
* Uses Windows API to simulate process creation.
* Implements FCFS scheduling, ensuring the first arriving task executes first.
* Displays a summary table with execution details.
* Target Users or Systems:
* Designed for students and OS enthusiasts learning about CPU scheduling.
* Runs specifically on Windows OS, as it relies on the Windows API for process management.

**3. Technology Stack**

* Programming Language: C++
* Operating System: Windows
* Development Tools: Any C++ compiler (e.g., MinGW, Visual Studio)
* Libraries & Frameworks Used:
* Windows API for process creation and management.
* Standard C++ libraries for time tracking and vector operations.

**4. Operating System Concepts Used**

**1. Process Management**

Process Creation: Uses Windows API to launch and manage new processes.

CPU Scheduling: Implements FCFS (First-Come, First-Served) scheduling.

Synchronization: Ensures proper task execution order using delays.

**2. Memory Management**

Each process executes in its own allocated memory space, managed by the OS.

**3. File System Management**

The program does not explicitly use file handling but could be extended to log results into a file.

**4. Device Management**

The project interacts with the Windows command-line interface (cmd.exe) for task execution.

**5. Security & Access Control**

Uses Windows process handling functions securely, ensuring controlled execution.

**5. Implementation Details**

**Core Algorithms and Logic:**

1. User Input: The user provides the number of tasks and their execution times.

2. Process Creation: Each task is launched using Windows API's CreateProcess function.

3. FCFS Scheduling:

Tasks are stored in a list and sorted by arrival time.

Processes execute in order of arrival to maintain FCFS discipline.

6. Conclusion

**Summary of Project Outcomes**

* Successfully implemented FCFS scheduling using Windows API for process management.
* Simulated process creation and execution using system-level programming.
* Computed and displayed scheduling metrics such as turnaround time and waiting time.

**Key Takeaways and Learnings**

* Understanding CPU scheduling and its impact on process execution.
* Practical application of Windows process creation APIs.
* Insights into task synchronization and execution management.
* Implementation of real-world OS scheduling techniques in C++.