SEMESTER-IV

COURSE 8: OPERATING SYSTEMS

Theory **Credits: 3** **3 hrs/week**

Course Objectives:

1. **Understand the evolution and core functions** of operating systems, including resource management and system types.
2. **Analyze process and thread management**, focusing on system calls, kernel modes, scheduling algorithms, and threading models.
3. **Evaluate process synchronization and deadlock handling**, using classical problems and inter-process communication methods.
4. **Apply memory management techniques**, including paging, segmentation, and virtual memory implementation.
5. **Examine file, I/O, and device management strategies**, along with basic OS-level security features.

Course Outcomes:

At the end of the course, students will be able to:

1. Explain the foundational principles and evolution of operating systems, including resource abstraction and core management functions.
2. Classify and compare different types of operating systems, such as multiprogramming, batch, time-sharing, real-time, and personal device-based systems.
3. Analyze process and thread management techniques, including processor modes, system calls, kernel functions, and scheduling algorithms.
4. Evaluate process synchronization and deadlock handling approaches, applying classical concurrency solutions and inter-process communication methods.
5. Apply memory, file, and I/O management strategies, incorporating allocation techniques, virtual memory models, disk scheduling, and OS-level security features.

Unit 1. Operating System Fundamentals:

Operating System Definition, History and Evolution of OS, Basic OS functions, Types of Operating Systems– Multiprogramming Systems, Batch Systems, Time Sharing Systems; Operating Systems for Personal Computers, Workstations and Hand-held Devices, Process Control & Real time Systems.

Unit 2. Process & Thread:

Processor and User Modes, Kernels, System Calls and System Programs, System View of the Process and Resources, Process Abstraction, Process Hierarchy, Threads, Threading Issues, Thread Libraries; Process Scheduling- Non-Preemptive and Preemptive Scheduling Algorithms.

Unit 3. Process Management:
Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Inter process Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery.

Unit 4. Memory Management:
Physical and Virtual Address Space; Memory Allocation Strategies–Fixed and -Variable Partitions, Paging, Segmentation, Virtual Memory.

Unit 5. File and I/O Management:
Directory Structure, File Operations, File Allocation Methods, Device Management, Pipes, Buffer, Shared Memory, Disk Scheduling algorithms.

Text Books:

1. Operating System Principles, Abraham Silberschatz, Peter Baer Galvin and GregGagne, 7th Edition, Wiley India Edition.
2. Operating Systems: Internals and Design Principles, Stallings , Pearson edition

Reference Books:

1. Operating Systems Design and Implementation, Andrew S. Tanenbaum, 3rd Edition, Pearson
2. A Text Book of Operating Systems, Singh, Kaur and Gupta, Khanna Publishers

Activities:

Outcome: Explain the foundational principles and evolution of operating systems, including resource abstraction and core management functions.

Activity: Timeline Creation & Concept Mapping - Students will collaboratively build a visual timeline showing the evolution of operating systems from early batch systems to modern distributed and mobile OS. They will also create a concept map illustrating core management functions (e.g., process, memory, file, device management).

Evaluation Method: Rubric-based assessment of timeline and concept map:

- Historical accuracy
- Inclusion of core OS functions
- Presentation and teamwork

Outcome: Analyze process and thread management, focusing on system calls, kernel modes, scheduling algorithms, and threading models.

Activity: Comparative Analysis Table & Case Study Discussion - Students will research and fill out a comparison table for OS types (multiprogramming, batch, time-sharing, real-time, personal device-based), focusing on architecture, scheduling, responsiveness, and use cases. Followed by a group discussion using real-world examples (e.g., RTOS in pacemakers vs. Android in smartphones).

Evaluation Method: Students will be evaluated on a 10-point scale based on

- Individual submission of comparison table
- Group participation score in discussion

Outcome: Analyze process and thread management techniques, including processor modes, system calls, kernel functions, and scheduling algorithms.

Activity: Simulation & Code Walkthrough - Students will simulate process scheduling using tools or pseudocode (e.g., Round Robin, Priority Scheduling). They will also analyze thread creation and system calls using a simple multithreaded program in Java.

Evaluation Method: Students will be assessed on a 10-point scale based on

- Scheduling algorithm simulation results
- Explanation of processor modes and system calls
- Code annotations for kernel functions

Outcome: Evaluate process synchronization and deadlock handling approaches, applying classical concurrency solutions and inter-process communication methods.

Activity: Role-play & Coding Challenge - Students will role-play classical problems (e.g., Dining Philosophers, Producer-Consumer) to understand synchronization. Then, they'll implement semaphore or monitor-based solutions in code and simulate deadlock detection or avoidance.

Evaluation Method: Students will be evaluated for 10 marks based on

- Code review for correctness and use of synchronization primitives
- Peer evaluation of role-play clarity and engagement
- Written test with deadlock scenarios and solution strategies

Outcome: Apply memory, file, and I/O management strategies, incorporating allocation techniques, virtual memory models, disk scheduling, and OS-level security features.

Activity: Interactive Lab & Design Task - Students will use OS simulators to experiment with memory allocation (paging, segmentation), disk scheduling (FCFS, SSTF, SCAN), and file system operations.

Evaluation Method: Students will be evaluated for 10 points based on

- Correct use of allocation techniques
- Disk scheduling output analysis
- Virtual memory behaviour observation

SEMESTER-IV

COURSE 8: OPERATING SYSTEMS

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Illustrate the LINUX commands
    a) pwd
    b) mkdir
    c) rmdir
    d) grep
    e) chmod
    f) ls
    g) rm
    h) cp

2. Write a program to calculate average waiting time and turn around time of each process using the following CPU Scheduling algorithm for the given process schedules.
    a) FCFS
    b) SJF
    c) Priority
    d) Round Robin

3. Simulate MVT and MFT memory management techniques

4. Write a program for Bankers Algorithm for Dead Lock Avoidance

5. Implement Bankers Algorithm Dead Lock Prevention.

6. Write a program to simulate Producer-Consumer problem.

7. Simulate all Page replacement algorithms.
    a) FIFO
    b) LRU
    c) LFU
    d) Optimal

8. Simulate Paging Techniques of memory management

9. Simulate the following disk scheduling algorithms
    a) FCFS
    b) SSTF
    c) SCAN
    d) CSCAN