Operating Systems Lab

Course Objectives:

1. To understand and implement basic services and functionalities of the operating system using system calls.
2. To use modern operating system calls and synchronization libraries in software/ hardware interfaces.
3. To understand the benefits of thread over process and implement synchronized programs using multithreading concepts.
4. To Analyze and simulate CPU Scheduling Algorithms like FCFS, Round Robin, SJF, and Priority.
5. To implement memory management schemes and page replacement schemes.
6. To simulate file allocation and organization techniques.
7. To understand the concepts of deadlock in operating systems and implement them in multiprogramming system

Week	Name of the Program
Week 1	**Execute the following UNIX command** PWD, CD, MKDIR, RMDIR, LS, CP, MV, RM, CHMOD, MAN, CAT, WHO, MORE **Assignment:** Explain each command with examples and option for all above mentioned commands.
Week 2	**Execute the following UNIX command** TEE, PIPE, HEAD, TAIL, CUT, PASTE, DIFF, COMM, GREP, ECHO, ETC… **Assignment**: Explain each command with examples and option for all the week 2 commands.
Week 3	Write a C program to simulate the following non-preemptive CPU Scheduling algorithms to find turnaround time, waiting time, average turnaround time and average waiting time. a) FCFS b) SJF **Assignment**: Write a C program to simulate the following preemptive CPU Scheduling algorithms to find turnaround time, waiting time, average turnaround time and average waiting time. a) FCFS b) SRTF
Week 4	Write a C program to simulate the following non-preemptive CPU Scheduling algorithms to find turnaround time and waiting time, average turnaround time and average waiting time. c) Round Robin (pre-emptive) d) Priority

	Assignment: Write a C program to implement round robin CPU scheduling algorithm for the following given scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Consider the time quantum size for the system processes and user processes to be 5 msec and 2 msec respectively

Week 4	Write a C program to simulate the following non-preemptive CPU Scheduling algorithms to find turnaround time and waiting time, average turnaround time and average waiting time. c) Round Robin (pre-emptive) d) Priority **Assignment**: Write a C program to implement round robin CPU scheduling algorithm for the following given scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Consider the time quantum size for the system processes and user processes to be 5 msec and 2 msec respectively
Week 5	Write a C program to simulate multi level queue scheduling algorithm. Assignment: Write a C program to simulate MFT memory management Scheme with unequal sized partitions
Week 6	Write a C program to simulate the following contiguous memory allocation techniques a) Worst-fit b) Best-fit c) First-fit **Assignment**: Write a C program to implement compaction technique
Week 7	Write a C program to simulate page replacement algorithms a) FIFO b) LRU c) Optimal
Week 8	Write a C program to simulate the following file organization techniques a) Single level directory b) Two level directory c) Hierarchical **Assignment**: Write a C program to simulate a two-level index scheme for file allocation?

Week 9	Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance. Assignment: Write a C program to simulate readers-writers problem using monitors.
Week 10	Write a C program to simulate disk scheduling algorithms a) FCFS b) SCAN c) C-SCAN
Week 11	Write a C program to implement Dining philosophers problem

Course Outcomes:

- Upon the completion of Operating Systems practical course, the student will be able to gain practical experience with designing and implementing concepts of operating systems such as system calls, CPU scheduling, process management, memory management, file systems and deadlock handling, using C language in Linux environment.