FACULTY OF COMPUTER SCIENCE & IT
UNIVERSITY OF MALAYA

OPERATING SYSTEMS

LAB ASSIGNMENT

Subject Code:	WIA 2004
Session:	2024/2025
Semester:	2

OPERATING SYSTEMS LAB SYLLABUS

(Practical Hours for every session: 02)

Implement the following programs using C language or any other language that is suitable.
Please follow your Lecturer instructions.

Lab No.	Topics	List of Assignment	Week
1	**CPU Scheduling Algorithms**	Write a program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time. a) FCFS	5
2		Write a program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time. a) SJF	6
3	**File Allocation Strategies**	Write a program to simulate the following file allocation strategies. a) Sequential	7
4	**Memory Management Techniques**	Write a program to simulate the following contiguous memory allocation techniques. a) Best fit	8
5		*****Write a program to simulate the following contiguous memory allocation techniques a) First fit	9
6	**Deadlock Management**	Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance.	10
7	**Page Replacement**	Write a program to simulate page replacement algorithms a) FIFO	11
8	**Synchroniza-tion**	*Write a program to simulate the concept of Dining-Philosophers problem.	12

OPERATING SYSTEMS LABORATORY

OBJECTIVE:

This lab complements the operating systems course. Students will 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.

OUTCOMES:

Upon the completion of Operating Systems practical course, the student will be able to:

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

LAB 1 & 2

1.1 OBJECTIVE

Write a program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time for the above problem.

a) FCFS b) SJF

1.2 DESCRIPTION

Assume all the processes arrive at the same time.

1.2.1 FCFS CPU SCHEDULING ALGORITHM

For FCFS scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. The scheduling is performed on the basis of **arrival time** of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. Calculate the waiting time and turnaround time of each of the processes accordingly.

1.2.2 SJF CPU SCHEDULING ALGORITHM

For SJF scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. Arrange all the jobs in order with respect to their **burst times**. There may be two jobs in queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly.

LAB 3

3.1 OBJECTIVE

Write a program to simulate the following file allocation strategies.
a) Indexed

3.2 DESCRIPTION

A file is a collection of data, usually stored on disk. As a logical entity, a file enables to divide data into meaningful groups. As a physical entity, a file should be considered in terms of its organization. The term "file organization" refers to the way in which data is stored in a file and, consequently, the method(s) by which it can be accessed.

3.2.1 INDEXED FILE ALLOCATION

Indexed file allocation strategy brings all the pointers together into one location: an index block. Each file has its own index block, which is an array of disk-block addresses. The i^{th} entry in the index block points to the i^{th} block of the file. The directory contains the address of the index block. To find and read the i^{th} block, the pointer in the i^{th} index-block entry is used.

LAB 4 & 5

5.1 OBJECTIVE

*Write a program to simulate the following contiguous memory allocation techniques
a) Best-fit b) First-fit

5.2 DESCRIPTION

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided. When it is time to load or swap a process into main memory, and if there is more than one free block of memory of sufficient size, then the operating system must decide which free block to allocate. Best-fit strategy chooses the block that is closest in size to the request. First-fit chooses the first available block that is large enough.

LAB 6

6.1 OBJECTIVE

Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance.

6.2 DESCRIPTION

In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock. Deadlock avoidance is one of the techniques for handling deadlocks. This approach requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this additional knowledge, it can decide for each request whether or not the process should wait. To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process. Banker's algorithm is a deadlock avoidance algorithm that is applicable to a system with multiple instances of each resource type.

LAB 7

7.1 OBJECTIVE

Write a program to simulate page replacement algorithms
a) FIFO

7.2 DESCRIPTION

Page replacement is basic to demand paging. It completes the separation between logical memory and physical memory. With this mechanism, an enormous virtual memory can be provided for programmers on a smaller physical memory. There are many different page-replacement algorithms. Every operating system probably has its own replacement scheme. A FIFO replacement algorithm associates with each page the time when that page was brought into memory.

LAB 8

8.1 OBJECTIVE

*Write a program to simulate the concept of Dining-Philosophers problem.

8.2 DESCRIPTION

The dining-philosophers problem is considered a classic synchronization problem because it is an example of a large class of concurrency-control problems. It is a simple representation of the need to allocate several resources among several processes in a deadlock-free and starvation-free manner. Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and the table is laid with five single chopsticks. When a philosopher thinks, she does not interact with her colleagues. From time to time, a philosopher gets hungry and tries to pick up the two chopsticks that are closest to her (the chopsticks that are between her and her left and right neighbors). A philosopher may pick up only one chopstick at a time. Obviously, she cam1ot pick up a chopstick that is already in the hand of a neighbor. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she is finished eating, she puts down both of her chopsticks and starts thinking again. The dining-philosophers problem may lead to a deadlock situation and hence some rules have to be framed to avoid the occurrence of deadlock.