

Operating System Laboratory (OSL) Practicals – Detailed Explanation

Address Book

Problem Statement: Design and implement a program to create and maintain an address book that stores names, phone numbers, and email addresses. The program should allow users to add, search, modify, and delete contact details.

Algorithm / Logic Used: Uses file handling or structure arrays to store data. Basic CRUD operations (Create, Read, Update, Delete).

Advantages:

- Easy data management.
- Simple to implement and understand.

Disadvantages:

- Not efficient for large datasets.
- No concurrency or database support.

Fork 2A

Problem Statement: Write a program that demonstrates the use of fork() system call to create a child process and show parent-child execution.

Algorithm / Logic Used: Use fork() in Linux to duplicate the current process. Parent and child run independently after fork. getpid() and getppid() are used to show process IDs.

Advantages:

- Demonstrates process creation.
- Helps understand multitasking.

Disadvantages:

- Increases system overhead if many processes are created.
- Managing synchronization between processes is complex.

Fork 2B

Problem Statement: Implement a program using fork() to show execution order and demonstrate use of wait() or exec() system calls.

Algorithm / Logic Used: Parent uses `wait()` to pause until the child finishes. `exec()` can replace the current process image with a new program.

Advantages:

- Useful for understanding process control.
- Helps differentiate `fork()` and `exec()` usage.

Disadvantages:

- Resource intensive.
- Difficult to debug due to multiple processes.

CPU Scheduling – SJF (Shortest Job First)

Problem Statement: Simulate Shortest Job First scheduling to find waiting time and turnaround time for each process.

Algorithm / Logic Used: Select process with the smallest burst time first. Non-preemptive scheduling.

Advantages:

- Minimizes average waiting time.
- Efficient for batch systems.

Disadvantages:

- Causes starvation for long processes.
- Requires prior knowledge of burst time.

CPU Scheduling – Round Robin (RR)

Problem Statement: Implement Round Robin Scheduling with different arrival times. Each process gets an equal time slice (quantum) cyclically.

Algorithm / Logic Used: Use a ready queue and assign CPU to each process for a fixed time. If unfinished, reinsert it into the queue.

Advantages:

- Fair allocation of CPU time.
- Best suited for time-sharing systems.

Disadvantages:

- Higher context-switching overhead.
- Performance depends on chosen time quantum.

Process Synchronization – Reader Writer Problem

Problem Statement: Simulate Reader-Writer synchronization using semaphores to allow multiple readers but only one writer at a time.

Algorithm / Logic Used: Use mutex and semaphore for coordination. Readers can read simultaneously if no writer is active.

Advantages:

- Ensures data consistency.
- Efficient for read-heavy operations.

Disadvantages:

- Writers may starve if many readers exist.
- Complex implementation with semaphores.

Process Synchronization – Producer Consumer Problem

Problem Statement: Implement a Producer–Consumer system using semaphores to synchronize data sharing via a buffer.

Algorithm / Logic Used: Use mutex, empty, and full semaphores. Producer adds items, consumer removes items from the buffer.

Advantages:

- Avoids data corruption and race conditions.
- Demonstrates synchronization mechanisms.

Disadvantages:

- Requires careful semaphore handling.
- Deadlocks can occur if not managed properly.

Banker's Algorithm

Problem Statement: Implement Banker's Algorithm for deadlock avoidance in a system with multiple processes and resources.

Algorithm / Logic Used: Checks whether resource allocation leads to a safe state. If not, delays the allocation.

Advantages:

- Prevents deadlocks proactively.
- Ensures system safety.

Disadvantages:

- Requires advance knowledge of maximum resource needs.
- Computationally expensive for large systems.

Page Replacement – FCFS (First Come First Serve)

Problem Statement: Implement FCFS Page Replacement to manage memory pages when a page fault occurs.

Algorithm / Logic Used: Replace the page that arrived first in memory.

Advantages:

- Simple and easy to implement.
- FIFO queue-based.

Disadvantages:

- Poor performance (Belady's Anomaly).
- Doesn't consider page usage frequency.

Page Replacement – LRU (Least Recently Used)

Problem Statement: Implement LRU Page Replacement which replaces the least recently used page when a new page needs to be loaded.

Algorithm / Logic Used: Track the order of page usage using stack or counters.

Advantages:

- Better performance than FCFS.
- Reflects actual memory usage patterns.

Disadvantages:

- Requires hardware or software support to track usage.
- Slightly complex implementation.

Page Replacement – Optimal

Problem Statement: Implement Optimal Page Replacement, which replaces the page that will not be used for the longest time in future.

Algorithm / Logic Used: Simulate future page references to choose victim page.

Advantages:

- Produces minimal number of page faults.
- Theoretically best algorithm.

Disadvantages:

- Not possible to know future references in practice.
- Used mainly for comparison.

Inter Process Communication – FIFO

Problem Statement: Implement communication between two processes using FIFO (Named Pipes).

Algorithm / Logic Used: Use mkfifo() and standard read/write functions. Data flows in first-in, first-out order.

Advantages:

- Simple and reliable for one-way communication.
- Works between unrelated processes.

Disadvantages:

- Unidirectional.
- Data not persistent after reading.

Inter Process Communication – PIPE

Problem Statement: Implement communication between parent and child process using unnamed pipe.

Algorithm / Logic Used: Use pipe() system call. One end for writing, one for reading.

Advantages:

- Fast communication between related processes.

- Easy to implement.

Disadvantages:

- Only works for parent-child processes.
- Unidirectional by default.

Disk Scheduling – SCAN

Problem Statement: Implement SCAN Disk Scheduling Algorithm to reduce seek time by moving the head in one direction servicing all requests.

Algorithm / Logic Used: Disk arm moves like an elevator (forward, then backward).

Advantages:

- Reduces variance in response time.
- Prevents starvation.

Disadvantages:

- Long waiting time for requests at the end of disk.
- Complex head movement management.

Disk Scheduling – CLOOK

Problem Statement: Implement CLOOK (Circular LOOK) scheduling algorithm.

Algorithm / Logic Used: Head moves only in one direction and jumps back to the first request without servicing in reverse.

Advantages:

- More uniform wait time.
- Avoids unnecessary traversal.

Disadvantages:

- Slightly complex compared to FCFS.
- Longer average seek time than LOOK for small workloads.

Disk Scheduling – SSTF

Problem Statement: Implement SSTF Disk Scheduling, which selects the request nearest to the current head position.

Algorithm / Logic Used: Choose the request with minimum seek distance.

Advantages:

- Reduces total head movement.
- Better performance than FCFS.

Disadvantages:

- Causes starvation for far-away requests.
- Complex to implement dynamically.