New Summit College

Department of Computer Science and Information Technology

Operating System Lab Manual
BSc CSIT - 4th Semester

General Procedure for Lab Work

- Read the lab objective and problem statement carefully.
- Understand the theory and the required system calls or synchronization tools (e.g., fork, pthread, semaphore).

Lab 1: Process and Thread Lifecycle Simulation.

a. Process Creation and Termination

b. Thread Creation and Termination

Lab Statement

- Simulate and observe the lifecycle of a process and a thread using C programming.

Objective

- To implement the creation and termination of a process and a thread and understand their behaviour.

a. Process Creation and Termination

Procedure

- Use fork() to create a child process.
- Use exit() to terminate the child.
- Use wait() to synchronize with the parent.

b. Thread Creation and Termination

Procedure

- - Use pthread_create() to launch a thread.
- - Use pthread_exit() to end the thread.

Lab 2: Bounded Buffer Problem (Producer-Consumer)

Lab Statement

- Implement the producer-consumer problem using semaphores.

Objective

- To simulate synchronization between producer and consumer using semaphores and shared buffer.

Assumptions

- Buffer size = 5
- Semaphore values: empty = 5, full = 0, mutex = 1

Procedure

- Initialize the semaphores.
- Use one thread as producer, another as consumer.
- Synchronize access to the buffer using semaphores.

Lab 3: Classical IPC Problems Using Semaphores

a. Dining Philosopher Problem

Lab Statement

- To simulate the dining philosopher problem using semaphores to avoid deadlock.

Objective

- To synchronize five philosophers such that no two neighbours eat at the same time.

Assumptions

- Five philosophers
- Semaphore chopstick[5] = 1, mutex = 1

Procedure

- Each philosopher picks left and right chopstick.
- Use semaphores to ensure atomic pickup and putdown.

b. Reader Writer Problem

Lab Statement

- Implement reader-writer synchronization using semaphores.

Objective

- Allow multiple readers or one writer to access shared data without conflict.

Assumptions

- mutex = 1, rw_mutex = 1, readcount = 0

Procedure

- Reader increments count on entry.
- First reader locks the writer.
- Last reader releases the writer.

c. Sleeping Barber Problem

Lab Statement

- To simulate the sleeping barber problem using semaphores.

Objective

- To coordinate barber and customers such that barber sleeps when no customer is available.

Assumptions

- customers = 0, barbers = 0, mutex = 1
- Waiting chairs = 5

Procedure

- Customers wait if chairs available.
- Wake barber when they arrive.
- Barber sleeps when no customer.

Lab 4: CPU Scheduling Algorithms

Objective

- To simulate various CPU scheduling algorithms and analyse their behaviour with different sets of processes.

Lab 4.1: First Come First Serve (FCFS) Scheduling

Lab Statement

- To Implement the First Come First Serve scheduling algorithm.

Theory/Concept

- Non-pre-emptive Algorithm-Processes are executed in the order of their arrival.

Procedure

1. Accept arrival time and burst time of each process.
2. Sort the processes based on arrival time.
3. Execute each process in order of arrival.
4. Calculate average waiting time and turnaround time.

Lab 4.2: Shortest Job First (SJF) Scheduling – Non-pre-emptive

Lab Statement

- To Implement the SJF scheduling algorithm.

Theory/Concept

- Non-pre-emptive algorithm.
- Selects the process with the smallest burst time from the ready queue.

Procedure

1. Accept arrival and burst time.
2. At every step, select the shortest job available at current time.
3. Calculate Average waiting and turnaround time.

Lab 4.3: Shortest Remaining Time First (SRTF) – Preemptive SJF

Lab Statement

- To Implement the SRTF scheduling algorithm.

Theory/Concept

- Pre-emptive version of SJF.
- Continuously selects the process with the smallest remaining time.

Procedure

1. Accept arrival and burst time.
2. At each unit of time, select the job with the least remaining burst time.
3. Pre-empt current job if a new shorter one arrives.

Lab 4.4: Round Robin (RR) Scheduling

Lab Statement

- To Implement Round Robin CPU scheduling with a time quantum.

Theory/Concept

- Preemptive algorithm.
- Each process is assigned a fixed time quantum.
- Mention time quantum.

Procedure

- Accept arrival and burst times, and the time quantum.
- Schedule processes in circular queue fashion.
- Re-add unfinished processes to the end of the queue.

Lab 4.5: Priority Scheduling (Non-preemptive)

Lab Statement

- To Implement non-preemptive priority scheduling.

Theory/Concept

- Each process has a priority; lower number = higher priority.
- CPU is assigned to the process with the highest priority available at arrival.

Procedure

- Accept arrival, burst time, and priority.
- Sort based on priority at current time.
- No preemption until current job completes.

Lab 4.6: Priority Scheduling (Preemptive)

Lab Statement

- To Implement preemptive priority scheduling.

Theory/Concept

- A higher priority process can preempt a running lower priority process.
- Accept arrival time, burst time, and priority.
- Continuously check for higher-priority processes.
- Preempt current process if higher-priority process arrives.

Report Format (Handwritten)

1. Title of the Lab
2. Objective
3. Lab Statement
4. Theory/Concept
5. Algorithm/Procedure
6. Source Code(Comment if required)
7. Output Screenshot 8. Conclusion (What you learned)

Prepared by: Apil Adhikari

Operating System

For: CSIT 4th Semester Student