CSE309 OPERATING SYSTEMS LAB

Jan 2024

SCHOOL OF COMPUTING

LIST OF EXPERIMENTS

Process Creation and Management

1) a). Creation of a child process using fork and communication between parent and child using pipe
   b). Creation of two children and communication between the siblings

Inter-Process Communication

2) a). IPC using Shared Memory
   b). IPC using Message Queues.

Processor Scheduling

3) a) Simulation of processor scheduling algorithms and analyzing their performances.
   b) Simulation of CPU scheduling with CPU and IO burst times
4) Simulation of multithreading using pthread

Concurrency and Synchronization

5) Implementing Peterson's algorithm

6) a)Implementing solution for producer-consumer problem
   b)Implementing solution for reader-writer problem

Deadlock handling

7) Implementing Banker's algorithm for deadlock avoidance

8) Implementation of deadlock detection algorithm

9) Implementing solution for dining philosopher's problem.

Paging

10)a)Simulate page replacement algorithms.
   b)Simulate address translation under paging

Disk Scheduling

11) Disk Scheduling Techniques

Additional Experiments

- Simulate dynamic partitioning and buddy system
- Demonstrate File allocation techniques

1a. Creation of a child process and communication

Objective:

To create a child process using fork system call and use pipe for interaction between parent and child

Pre-requisite:

Knowledge of parent-child process, fork and pipe commands.

Procedure:

- ❑ Develop the parent process with code for calls to fork and pipe
- ❑ Child process created as a result of fork()
- ❑ Write a message into pipe under parent part of the code
- ❑ Suspend parent process to invoke child
- ❑ Reads the message from the pipe under the child part of the code
- ❑Parent and child terminates

Pre-Lab:

Practice on getpid, getppid commands

Additional Exercises:

Creation of multiple children

1b. Creation of two children and communication

Objective:

To create two child processes using fork system call and use pipe for interaction between child1 and child2

Pre-requisite:

Knowledge of parent-child process, fork and pipe commands.

Procedure:

- Develop the parent process with two fork calls and a pipe
- Child processes created as a result of fork()
- Write a message into pipe under child1 part of the code
- Read the message from the pipe under the child2 part of the code
- Parent and child processes terminate

Pre-Lab:

Practice on getpid, getppid commands

Additional Exercises:

Creation of multiple children

2a. IPC using Shared Memory

Objective:

To implement IPC using shared memory concept with the help of the library functions available.

Prerequisite:

Knowledge of IPC, Shared memory functions, their syntaxes and functionalities

Procedure:

- Create the sender process and receiver process
- Create a shared memory making using the appropriate function
- Sender pushes its message into shared memory
- Receiver retrieves the message and displays it to the user

Pre-Lab

Practicing shmat, shmget

Additional Exercise:

IPC based on chatting application

2a. IPC using Message Queue

Objective:

To implement IPC using message queues with the help of the library functions available.

Prerequisite:

Knowledge of IPC, Message queues, syntax and functionalities

Procedure:

- Create the sender process and receiver process
- Create a message queue using the appropriate functions
- Sender pushes its message into message queue using msgsnd
- Receiver retrieves the message using msgrcv and show it to the user

Pre-Lab

Practicing shmat, shmget, Msgget, Msgsnd, Msgrcv

Additional Exercise:

Message queue based multi-process communication

3a. Simulation of CPU Scheduling algorithms

Objective:

Simulation of preemptive and non-preemptive CPU Scheduling algorithms

Prerequisite:

Knowledge of scheduling algorithms

Procedure:

- ❑ Input the number of processes to be scheduled
- ❑ Input the arrival time and CPU burst time of each process
- ❑ Calculate the turn around time and the waiting time of the processes based on the following scheduling methods:
  - First Come First Serve (FCFS), Shortest Job First (SJF),
  - Round Robin(RR), Preemptive SJF or Shortest Remaining Time(SRT)
- ❑Compare the mean turn around time and choose the algorithm providing the best result.

Pre-Lab:

Waiting Time, Burst Time and Average Waiting Time calculation

Additional Exercise: **Feedback, Priority , HRRN**

3b. Simulation of CPU Scheduling with IO

Objective:

Simulation of CPU Scheduling algorithm with IO burst time

Prerequisite:

Knowledge of scheduling algorithms

Procedure:

- Input the number of processes to be scheduled
- Input the arrival time, CPU burst time1, IO burst time and CPU burst time2 of each process
- Calculate the turn around time and the waiting time of the processes based on the FCFS

Pre-Lab:

Waiting Time, Burst Time and Average Waiting Time calculation

4. Simulation of Multi-processor Scheduling

Objective:

Simulation of multi-processor Scheduling algorithm

Prerequisite:

Knowledge of scheduling algorithms

Procedure:

- Input the number of processors available
- Input the number of processes to be scheduled
- Input the arrival time and CPU burst time of each process
- Schedule the processes on the free processors and calculate the turn around time using FCFS
- Compare the calculated TT with the TT of the same data scheduled on uni-processor and analyze the performance gain

Pre-Lab:

Waiting Time, Burst Time and Average Waiting Time calculation

5. Simulate Thread Scheduling

Objective:

To Simulate multi-threading using ptherad

Prerequisite:

Knowledge of thread concepts and ptherad functions

Procedure:

- Create user level threads using pthreads
- Associate different function to each thread
- Assign priorities to the threads
- Schedule them on LWP by requesting the kernel

Pre-Lab:

Multi-threading concepts

6. Simulate Peterson's Algorithm

Objective:

To Simulate Peterson's algorithms for mutual exclusion

Prerequisite:

Knowledge of critical-sections, mutual exclusion, bounded waiting, Synchronization and producer-consumer problem

Procedure:

- Two Process Solution by sharing two variables
- Turn, Flag[2]
- Turn - Indicates whose turn is to enter the critical section (CS)
- Flag[] - Array used to indicate if a process is ready to enter CS
- flag[i]= true implies process Pi is ready

Pre-Lab:

Multi-threading, Multi-processors

7a. Simulating Producer-Consumer problem

Objective:

Implementation of Producer-Consumer problem using bounded and unbounded variations

Pre-requisite

Knowledge of Concurrency, Mutual exclusion, Synchronization and producer-consumer problem

Procedure:

- Implement producer-consumer program with producers and consumers simulated as threads.
- Employ necessary semaphores for bounded and unbounded implementations
- Run the program to allow the producer and consumer share the buffer by synchronizing themselves through mutual exclusion

Pre-Lab:

Simple programs using Semaphore functions

Additional Exercise:

Multiple producers and consumers

7b. Simulating Reader-Writer problem

Objective:

To write a code to solve the readers writers problem based on reader priority and writer priority solution

Pre-requisite

Knowledge of Concurrency, Mutual exclusion, Synchronization and Reader writer problem

Procedure:

- ❑ Create a reader process
- ❑ Create a writer process
- ❑ Implement necessary semaphores
- ❑ Implement the programs giving reader priority and writer priority

Pre-Lab :

Semaphore, multi-processes

Additional Exercise:

multiple readers and writers, solution based on message passing

8. Banker's Algorithm for DeadlockAvoidance

Objective:

Simulate bankers algorithm for dead lock avoidance

Procedure:

- Get the number of processes and resources
- Create the following data structures:
- **Available** – Number of available resources of each types.
- **Max** – Maximum demand of each process.
- **Allocation** – Number of resources of each type currently allocated to each process.
- **Need** – Remaining resource need of each process. (Max-Allocation)
- Use **Safety algorithm** and **Resource-Request algorithm**.

Pre-Lab:

Prior knowledge of deadlocks and all deadlock avoidance methods.

Additional Exercise:

Deadlock prevention - circular wait, Deadlock recovery, Finding cycle in resource allocation graph

9. Deadlock Detection Algorithm

Objective:

To implement the deadlock detection algorithm

Procedure

- ❑ Construct the Allocation and Available matrices
- ❑ Follow the following steps
- ✓ Mark each process that has a row in the allocation matrix of all zeros
- ✓ Initialize a temporary vector W to equal to the available vector
- ✓ Find an index i such that process i is currently unmarked and the ith row of Q is less than or equal to W. If no such row found terminate algorithm
- ✓ If row found, mark process i and add the corresponding row of the allocation matrix to W.
- ✓ Return to step 3
- ❑ A deadlock exist is there are unmarked processes

Pre-Lab:

Prior knowledge of deadlocks and all three deadlock strategies.

10. Implementing solution for Dining Philosopher's problem

Objective:

To write a code to solve the Dining philosopher problem.

Prerequisite

Knowledge of Concurrency, Deadlock and Starvation

Procedure

- ❑ Create philosopher process
- ❑ Declare semaphore for mutual exclusion and left & right forks
- ❑ Implement function for obtaining fork - takefork
- ❑ Implement function for releasing fork - putfork.
- ❑Implement function for testing blocked philosophers

Pre-Lab:

Semaphore, multi-processes

Additional Exercise:

Solution using monitors

11a. Simulate Page Replacement algorithms

Objective:

Simulate page replacement algorithms.

Prerequisite:

Knowledge of Paging concepts, replacement algorithms

Procedure:

- Input the number of memory frames and the page reference string
- Select a page to replace based on the following approaches and calculate the page faults under each approach:
  - FIFO
  - Least Recently Used
  - Optimal page replacement

Pre-Lab :

Paging

Additional Exercise:

FIFO, MRU, MFU, Second-chance

11b. Simulate Address Translation in Paging

Objective:

To simulate the address translation from logical to physical address under paging

Prerequisite:

Knowledge of Pages, Frames, Memory partitioning

Procedure:

- Get the range of physical and logical addresses.
- Get the page size.
- Get the page number of the data.
- Construct page table by mapping logical address to physical address.
- Search page number in page table and locate the base address.
- Calculate the physical address of the data.

Pre-Lab:

Paging, Page replacement, Address calculation methods

Additional Exercise:

Simulation of thrashing

12. Disk Scheduling Techniques

Objective:

Simulation of disk scheduling techniques.

Prerequisite:

Knowledge of disk scheduling algorithms.

Procedure:

- Input the number of tracks on disk
- Input the list of requests
- Input the current head position
- Calculate the seek time as per the algorithms listed below:
  - First-in-First-Out
  - Shortest Seek Time First
  - Scan

Pre-Lab:

Calculation of Seek time, transfer time etc.

Additional Exercise:

C-SCAN, Look, C-Look