SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Fourth Semester B. Tech. [Section B]

Course Name: Operating Systems Lab [CSP226]

LIST OF EXPERIMENTS [ACADEMIC YEAR 2023-2024]

SN	Experiment	COs
01	To study and execute basic Linux System Commands and write shell scripts to display the system particulars (processor, processes & memory).	CO1
02	To write and execute I/O system calls in Linux.	CO1
03	To write C programs to implement the process control system calls.	CO1, CO2
04	To write and execute C programs to demonstrate different CPU scheduling algorithms.	CO2
05	To write C programs to demonstrate inter process communication (IPC) using sockets, shared memory and/or pipes/message queues.	CO2, CO3
06	Write C programs to implement threads and semaphores for process synchronization.	CO3
07	Write C programs to demonstrate the Banker's Algorithm and recovery processes.	CO3
08	Write C programs to implement different disk scheduling algorithms and to demonstrate different memory management schemes.	CO4

COURSE OUTCOMES

01 | Implement system commands by making use of LINUX system calls.
02 | Implement processes and process schedulers.
03 | Design solutions to process synchronization and deadlock handling.
04 | Implement Memory management and File management solutions.

JOURNAL COMPILATION GUIDELINES

Each experiment compilation in Laboratory Journal must include

1. The experiment number, as **EXPERIMENT NO - NN**
2. **Date** of Experimentation
3. **Aim** of the Experiment
4. **Problem Statement**
5. **Theory**

   Elaboration on concepts (purpose, syntax & characteristics) handled by the experiment, including system calls/utilities /algorithms for the routines included in the solution.

   Refer **Table 1** : Theory Compilation Guidelines (Cycle-1).

   Refer **Table 2** : Theory Compilation Guidelines (Cycle-2).

6. **Program-Listing**

   Printouts on A4 paper presenting a well-documented source code compilation and outputs.

   Do not insert snapshot images of the executed code on the console.

   The print document must include details, viz.,

   - Experiment No., Author, Roll Number, Semester & Section, Date of Compilation/Execution, Source/Script File Name
   - Aim and Problem Statement
   - Command & Execution Trace on CLI or C Code / Shell Script with outputs.

7. Answers to **Viva-Voce questions** (if any of these are listed).
8. The **inferences** drawn from the experiment that facilitates learning.

Note: Arrange the compilation for each experiment strictly in the aforementioned sequence.

TABLE 1: THEORY COMPILATION GUIDELINES [CYCLE-1]

Experiment No.		Theory Compilation Details
01	**Aim**	To study and execute basic Linux System Commands and write shell scripts to display the system particulars (processor, processes & memory).]
	Theory	▪ **Linux Console Commands**: pwd, uname, who, whoami, which, ls, cat, touch, cp, rm, mv, mkdir, chdir, wc, grep, … ▪ **System Particulars**: proc/cpuinfo, proc/meminfo, proc/stat, lscpu, nproc, hwinfo, top, head, tail, ps, top, …
02	**Aim**	To write and execute I/O system calls in Linux.
	Theory	▪ **I/O System Calls**: open, close, read, write, creat, unlink, umask, chmod, lseek, link, system ▪ **System Flags** : O_CREAT, O_WRONLY, O_TRUNC ▪ **Mode Bits**: S_IRUSR, S_IWUSR, S_IRGRP, S_IROTH, ...
03	**Aim**	To write C programs to implement the process control system calls.
	Theory	▪ **Process Commands**: fork, exec family (execl, execlp, execv, execvp, execle, execve), getpid, getppid, vfork, wait, sleep, exit, _exit
04	**Aim**	To write and execute C programs to demonstrate different CPU scheduling algorithms.
	Theory	▪ **CPU Scheduling**: Preemptive & Non-preemptive scheduling, FCFS, SJF, Round Robin, Shortest Remaining Time First, Earliest Deadline First

Note:

- As a general practice, **use of global variables will be restricted** across the C codes in CSP226.
- All codes will follow **modular approach** and coding style [appropriate indentation].
- The programmer shall incorporate **contextual naming** [meaningful long names] for variables and use functions will be preferred in the program.

EXPERIMENT NO – 01

Aim

To study and execute basic Linux System Commands and write shell scripts to display the system particulars (processor, processes & memory).

Problem Statement

(A) Execute the following Linux Commands to (preferably embedded in a shell script) –

[1] Display the current directory
[2] Display the OS kernel information
[3] Display the username and other particulars
[4] Display contents of a directory (various options for structuring the output)
[5] Display contents of a file
[6] Create an empty file
[7] Create a directory
[8] Switch to a directory
[9] Display number of lines, words, & characters in a file
[10] Locate & display records with matching text

(B) Write shell scripts to –

[1] Display the number of processors and cores on the host machine
[2] Display the speed (frequency) of each processor.
[3] Display the total, available and free memory on the host machine
[4] Display the total number processes spawned and the number of context switches performed since the system boot.

Write two different ways in which the above set of tasks could be performed.

Viva-Voce Questions

1. Which all file system formats are supported on Linux?
2. What are different shells in Linux? Name the default shell for Linux.
3. Enlist the advantages of command line interfaces.
4. Enlist the limitations of graphical user interfaces.
5. What is swap space? How it is implemented in Linux?

EXPERIMENT NO – 02

Aim

To write and execute I/O system calls in Linux.

Problem Statement

(A) Write a menu-driven program in C that uses different Linux I/O system commands to execute following tasks to –

[1] Create a file on the disk drive. Check if the said file already exists.
[2] Open a disk file for reading data content.
[3] Open a disk file for writing data content.
[4] Close a file.
[5] Remove the file from the disk.
[6] Read and Write a file starting a specified position.
[7] Change file permissions and masking certain privileges.
[8] Display pre-state & post-state for appropriate I/O commands.

Note: Ensure that a proper conditional verification (using return values) is carried out while implementing a specific operation. The states with respect to the operation being executed – before (pre-state) & after (post-state) should be adequately included using **system().**

Viva-Voce Questions

1. What are file descriptors? Enlist the standard file descriptors.
2. Enlist the specific header files needed for activating the read(), close(), open()?
3. Explain the consequence of creating a file using open() without specifying the mode field?
4. How can a creat() be implemented using open()?

EXPERIMENT NO – 03

Aim

To write C programs to implement the process control system calls.

Problem Statement

(A) Write a menu-driven program in C that uses different process control commands to execute following tasks to –

[1] Create processes that execute the address space of parent.
[2] Create processes that do not execute the address space of parent.
[3] Display IDs for parent and child processes
[4] Execute processes swapping running process with another one when – (a) absolute path is requires, (b) absolute path is optional. Include all variations.
[5] Create zombie and orphan processes.
[6] Terminate the processes – (a) with cleanup, (b) without cleanup.

Note: Ensure that a proper conditional verification (using return values) is carried out while implementing a specific operation. The states with respect to the operation being executed – before (pre-state) & after (post-state) should be adequately included using **system().**

Viva-Voce Questions

1. How do fork() and vfork() differ?
2. Enlist and explain the different Linux process states?
3. Explain the daemon process and the orphan process.
4. How do system() and execl() differ?
5. Contrast between wait() and sleep().

EXPERIMENT NO – 04

Aim

To write and execute C programs to demonstrate different CPU scheduling algorithms.

Problem Statement

(A) Consider a scenario wherein N processes ($4 \leq N \leq 7$) are to be scheduled on a single processor. Write a menu-driven program in C to implement the following CPU scheduling algorithms –

[1] First Come First Serve (common arrival & distinct arrival)
[2] Shortest Job First (non-preemptive)
[3] Round-Robin
[4] Shortest Remaining Time First

Note: Appropriately display the contents of data structures and process state parameters performing step-by-step process.

Viva-Voce Questions

1. How are short-term and log-term schedulers characterized?
2. What is cooperative scheduling? List the OSs employing it.
3. List and explain the different measures of evaluating CPU scheduling algorithms.
4. Explain the convoy effect.
5. State the factors that decide the effectiveness of round-robin approach.

TABLE 2: THEORY COMPILATION GUIDELINES [CYCLE-2]

Experiment No.		Theory Compilation Details
05	**Aim**	To write C programs to demonstrate inter process communication (IPC) using sockets, shared memory and/or pipes/message queues.
	Theory	▪ **Sockets**: Brief discussion on Types, Sockets Calls ▪ **Shared Memory & Pipes**: Brief Discussion, Their Necessity, Related Unix Calls
06	**Aim**	Write C programs to implement threads and semaphores for process synchronization.
	Theory	▪ **Threads**: Brief discussion on Types, Unix System Calls ▪ **Semaphores**: Categorization & Merits, Unix System Calls
07	**Aim**	Write C programs to demonstrate the Banker's Algorithm and recovery processes.
	Theory	▪ **Process Deadlock**: Characterization ▪ **Banker's Algorithm:** Data Structures, Procedures
08	**Aim**	Write C programs to implement different disk scheduling algorithms and to demonstrate different memory management schemes.
	Theory	▪ **Disk Scheduling:** Necessity & Types, OS terminologies ▪ **Memory Management**: Basic Strategies, Fragmentation

Note:

- As a general practice, **use of global variables will be restricted** across the C codes in CSP226.
- All codes will follow **modular approach** and coding style [appropriate indentation].
- The programmer shall incorporate **contextual naming** [meaningful long names] for variables and use functions will be preferred in the program.

EXPERIMENT NO – 05

Aim

To write C programs to demonstrate inter process communication (IPC) using sockets, shared memory and/or pipes/message queues.

Problem Statement

[1] Write a C program to implement a Chat Service between two users. The Client and the Server Programs may execute on different machines over a local area network (for simplicity you may run them on the same machine). Use TCP sockets.

[2] Write a C program to implement shared memory. The server will receive a number from the client and return the sum-of-its digits / reversed number back to the client.

[3] Write a C program to create bi-directional communication using pipes.

Viva-Voce Questions

1. What do you understand by blocking operation?
2. Why is close() call necessary in socket implementation?
3. Differentiate between shared memory and pipe.
4. How do a TCP and a UDP socket differ?
5. Compare shared memory and sockets.

EXPERIMENT NO – 06

Aim

Write C programs to implement threads and semaphores for process synchronization.

Problem Statement

Write a program to demonstrate threads. The demonstration may create 2 or more threads, where each thread will proceed with a distinct computation using a function or a separate program. The main flow will continue as normal.

Write a program to implement producer consumer problem using semaphores.

Viva-Voce Questions

1. Compare and contrast between a thread and a process.
2. What are advantages and limitations of kernel level threads?
3. List advantages of semaphores.
4. Compare and contrast mutex and semaphore.
5. What a POSIX threads? List associated system calls.

EXPERIMENT NO – 07

Aim

Write C programs to demonstrate the Banker's Algorithm and recovery processes.

Problem Statement

Consider a 5-6 process snapshot accessing 3-5 resources. Each resource can have multiple instances.

1. Apply Banker's Algorithm to check whether the process snapshot is in Safe State (not deadlocked).
2. Consider a new request from the queued process and determine whether the new request can be granted immediately.

Use appropriate AVAILABLE [ ], MAX [ ][ ] and ALLOCATION [ ][ ] data.

Viva-Voce Questions

1. Enlist and explain in brief the conditions characterizing a deadlock.
2. How does a deadlock differ from starvation?
3. Differentiate between deadlock avoidance and deadlock prevention.
4. Which approach(s) does contemporary OSs employ to deal with deadlock?

EXPERIMENT NO – 08

Aim

Write C programs to demonstrate different memory management schemes ~~and to implement different disk scheduling algorithms~~.

Problem Statement

Consider a partitioned memory with fixed sized (static) partitioning. Apply the following contiguous memory allocation techniques (any two) for a *specific memory map* with partitions.

1. First Fit
2. Best Fit
3. Worst Fit

Viva-Voce Questions

1. What are the relative merits of Worst Fit Techniques?
2. Elaborate on the relative demerits of Best Fit.
3. Explain the significance of the valid/invalid bit and the dirty bit.
4. Discuss the following terms in connection to disk scheduling –
    a. Seek time
    b. Rotational latency
    c. Disk access time