Lab Experiment Sheet-1

School of Engineering and Technology
Course Code & Name: ENCS351 Operating System
Program Name: B.Tech CSE, AI ML, Data Science, Cyber, FSD, UX/UI

Submission Guidelines

General Requirements

- **Submission Deadline:** Assignments must be submitted within one week of the assignment's release date.
- **Submission Platform:** All assignments are to be submitted via the Learning Management System (LMS).
- **GitHub Link:** You must provide a link to your GitHub repository with your submission.
- **Individual Submission:** Assignments are to be completed and submitted by each individual student.
- **Formatting:** All assignments must adhere to the specific format shared in class.

Evaluation

- **Total Marks:** This assignment is worth a total of 5 marks.
- **Evaluation Metrics:** Assignments will be evaluated based on the following criteria:
    - **Originality:** The uniqueness and independent thought demonstrated in the work.
    - **Correctness:** The accuracy and validity of the solutions or content.
    - **Completeness:** The extent to which all parts of the assignment have been addressed.

Experiment Title: Process Creation and Management Using Python OS Module

Experiment Objectives:

In this assignment, students will simulate Linux process management operations using Python. The experiment focuses on replicating the behaviors of fork(), exec(), and process state inspections using the os and subprocess modules in Python. It provides an understanding of process creation, child-parent relationship, and zombie/orphan process scenarios.

Learning Outcomes:

- Understand the lifecycle of processes in Linux.
- Create child processes and execute system commands using Python.
- Simulate zombie and orphan processes.
- Inspect running processes using /proc.
- Demonstrate priority setting via nice values.

Concepts Used:

- os.fork(), os.getpid(), os.getppid()
- os._exit(), os.wait(), os.nice()
- subprocess.run(), os.execvp()
- Reading /proc/[pid]/status, /exe, and /fd

Detailed Instructions:

Task 1: Process Creation Utility

Write a Python program that creates N child processes using os.fork(). Each child prints:
- Its PID
- Its Parent PID
- A custom message

The parent should wait for all children using os.wait().

Task 2: Command Execution Using exec()

Modify Task 1 so that each child process executes a Linux command (ls, date, ps, etc.) using os.execvp() or subprocess.run().

Task 3: Zombie & Orphan Processes

Zombie: Fork a child and skip wait() in the parent.
Orphan: Parent exits before the child finishes.
Use ps -el | grep defunct to identify zombies.

Task 4: Inspecting Process Info from /proc

Take a PID as input. Read and print:
- Process name, state, memory usage from /proc/[pid]/status
- Executable path from /proc/[pid]/exe
- Open file descriptors from /proc/[pid]/fd

Task 5: Process Prioritization

Create multiple CPU-intensive child processes. Assign different nice() values. Observe and log execution order to show scheduler impact.

Expected Output:

- Child-parent process tree

- Executed system commands from child processes

- Verified zombie/orphan states using ps

- Process details from /proc

- Impact of priority using different nice values

Complexity Analysis:
Time Complexity: O(n) for n processes.

Space Complexity: O(n) due to maintaining process IDs and logs.

Practical Applications:

- Operating system kernel development
- Performance tuning via scheduling
- Real-time and embedded system programming
- Debugging tools and monitoring systems

Submission Format:

- process_management.py: Python script with all tasks.
- output.txt: Sample output for each task.
- report.pdf: Summary of objectives, code snapshots, and results.
- README.md: Instructions to run and requirements.

Rubrics for Evaluation

Criteria	Max Score	Description
Conduct of Experiment	5	Execution, understanding, minimal supervision
Result Demonstration	5	Correct output, command execution, process details
Viva-Voce	10	Clear concepts: fork, exec, zombie/orphan, /proc, scheduling

Evaluator's Signature: _____