Operating Systems Lab(ENCS351) Report

Name: Spandan Jain

Roll No: 2301010038

Course: Operating Systems Lab

Abstract

This report summarizes two Operating Systems laboratory experiments implemented using Python. Assignment 1 focuses on process creation and management using the os module, including fork, exec, zombie/orphan process demonstration, and priority management. Assignment 2 demonstrates system startup simulation using multiprocessing and logging to illustrate concurrent process execution and synchronization.

Assignment 1 – Process Creation and Management

Objective

To understand and implement process creation, execution, and management in Linux using Python's os module.

Tools Used

Python 3.8+, Ubuntu (or WSL), Terminal, ps command, /proc filesystem

Key Concepts

• Process creation using os.fork()

• Program execution using os.execvp()

• Zombie and Orphan process demonstration

• Process inspection using /proc/[pid]/status

• Priority control using os.nice()

Results and Observations

The experiment successfully demonstrated multiple process states. Child processes created via fork() executed independently. Using execvp(), each child replaced its memory space with a new program. When the parent delayed wait(), the child entered a zombie state.

When the parent exited early, the child became an orphan, adopted by init (PID 1). Priority changes were visible through CPU time differences recorded in logs.

Assignment 2 – System Startup Simulation

Objective

To simulate the startup of a system using Python's multiprocessing and logging modules, demonstrating concurrent process execution and synchronization.

Tools Used

Python 3.8+, Multiprocessing library, Logging library, Ubuntu/WSL

Key Concepts

• Process creation using multiprocessing.Process()

• Concurrent execution and synchronization

• Logging of process start/end events with timestamps

Results and Observations

The system startup simulation created multiple independent processes representing boot tasks. Each process logged its start and completion times in process_log.txt. The logs verified that processes executed concurrently and terminated successfully. This demonstrated real-world concepts of parallel startup tasks in modern operating systems.

Conclusion

Through these experiments, key operating system concepts such as process lifecycle, parent-child relationships, and concurrent execution were practically observed. Python provided a simple interface to system-level operations using the os and multiprocessing modules. The hands-on understanding of zombie/orphan processes, process inspection, and logging mechanisms deepened comprehension of OS internals.

References

• Python os module documentation – https://docs.python.org/3/library/os.html

• Python multiprocessing documentation – https://docs.python.org/3/library/multiprocessing.html

• Linux man pages for fork, exec, ps, and nice