Operating Systems Project
Linux Process Management and Monitoring

Dr. Samar (Instructor)
Eng. Mohamed Hassan (Teaching Assistant)

Introduction

Dear Students,

This project is designed to give you hands-on experience with Linux process management and monitoring. You will implement multiple programs and scripts to explore processes, threads, CPU scheduling, signals, and process states. The project also includes creating a simple graphical interface to visualize process scheduling and CPU usage over time.

Project Objectives

By completing this project, you will:

- Create and manage processes using `fork()`.
- Implement multithreaded processes using POSIX threads (`pthreads`).
- Explore CPU scheduling using `nice` values and understand the Completely Fair Scheduler (CFS).
- Handle signals, understanding the difference between `SIGTERM` and `SIGKILL`.
- Create zombie and orphan processes and observe them using Linux monitoring tools.
- Monitor processes dynamically using `ps`, `top`, or `htop`.
- Automate process management with a Bash script that allows launching, monitoring, renicing, and terminating processes.
- Develop a GUI (or visualization interface) to display processes, their CPU usage, and scheduling over time, similar to a Gantt chart.

Project Requirements

CPU-Intensive Process

- Implement a process that consumes maximum CPU to observe scheduling behavior. - Monitor how CPU allocation changes when the nice value is adjusted.

Slow/Idle Process

- Implement a process that sleeps most of the time. - Handle signals gracefully, demonstrating the difference between `SIGTERM` (polite shutdown) and `SIGKILL` (forceful termination). - Observe how the scheduler treats low-CPU processes.

Zombie Process Creation

- Implement a parent process that spawns a child process. - Make the child terminate immediately while the parent sleeps. - Use process monitoring tools to verify that the child becomes a zombie.

Threaded Process

- Create a multithreaded process using POSIX threads. - Observe thread scheduling and CPU usage across multiple threads.

Process Manager Script

- Develop a Bash script to interactively manage your processes. - The script should allow:

- Viewing all running processes with their PID, PPID, state, CPU and memory usage, priority, and nice value.
- Launching CPU hog, slow, zombie, and threaded processes.
- Renicing processes to change their priorities dynamically.
- Sending signals (`SIGTERM` or `SIGKILL`) to running processes.

GUI / Gantt Chart Visualization

- Develop a simple graphical interface (using Java Swing, JavaFX, or any other framework) to visualize processes and CPU scheduling. - The interface should include:

- A table or list showing PID, command, CPU%, memory usage, state, and priority.
- A Gantt-like chart or bars displaying CPU usage of each process over time.
- Optional: Buttons to launch, terminate, or renice processes from the interface.

Tools and Environment

- Linux OS (Ubuntu recommended) - GCC compiler for C programs - Bash shell for scripting - Java JDK for GUI development - Linux monitoring tools: `ps`, `top`, `htop`

Submission Instructions

- Submit all source files for your C programs, Bash scripts, and GUI implementation. - Include a README explaining how to run each component. - Include screenshots of your GUI and monitoring outputs. - Make sure all programs and scripts run correctly on a standard Linux environment.

Conclusion

This project is a practical exercise to help you understand Linux process management, threads, CPU scheduling, signals, and monitoring. Be creative in implementing the GUI or visualization interface to demonstrate process scheduling and CPU usage dynamically. Anything you add beyond the basic requirements will be appreciated as part of your work.