005

Process States

Operating Systems

Lab005 Process States

(Windows and Linux)

Objective
The goal of this exercise is to help students understand **process states** in an operating system. Students will explore how processes transition between different states (e.g., running, waiting, sleeping, stopped) by writing programs that create and manage processes while observing their states using system commands.

Assignment Tasks

Task 1: Creating and Observing Process States
Students will write a program that:

- Creates multiple child processes.
- Forces the child processes into different states (running, sleeping, stopped, zombie).
- Uses system commands to observe and confirm the process states.

Task 2: Experimenting with Process State Transitions
Students will modify their program to:

- Change process states dynamically.
- Use signals (kill, SIGSTOP, SIGCONT, etc.) to control state transitions.
- Observe and document changes using system commands like ps, top, and procfs (Linux) or Task Manager & tasklist (Windows).

You will start using the files: process_states_linux.c or process_states_windows.c

Lab Instructions

1. **Compile and run the program** on your system.
    - **Linux:** Use ps aux and top to observe the process states.
    - **Windows:** Use **Task Manager** or tasklist to observe process states.
2. **Experiment with state transitions**
    - Use kill -SIGSTOP <PID> and kill -SIGCONT <PID> (Linux).
    - Use SuspendThread() and ResumeThread() (Windows).
3. **Modify the code** to:
    - Create multiple child processes.
    - Implement additional **signal handling** for termination.
4. **Document your observations.**
    - Which states were observed?
    - How did signals or function calls affect state changes?

Expected Learning Outcomes

- Understanding **process lifecycle** and **state transitions**.
- Using **system commands** to track process states.
- Managing **processes dynamically** in **Linux and Windows**.
- Using **signals and threading functions** for process control.