004

Process Creation and Management

Operating Systems

Lab004 Process Creation and Management

(Windows and Linux)

Objective

The primary objective of this lab is to help students understand **process creation, management, and inter-process communication (IPC)** in both **Linux and Windows** environments (You will work on the Operating System you have, do not need to implement the solution for both Operating Systems).

By working with process-related system calls and functions, you will:

- Learn how to **create and manage processes** using system calls (fork(), exec(), CreateProcess()).
- Understand **process states** and how they transition.
- Experiment with **process synchronization** (waiting for a process to finish).
- Explore **Inter-Process Communication (IPC)** using pipes and shared memory.

Overview

In this lab, you will write **small programs** that demonstrate process creation and management. The lab consists of multiple exercises that introduce you to different aspects of **process handling**.

You will:

1. **Create a child process** using fork() (Linux) or CreateProcess() (Windows).
2. **Observe parent-child process relationships** and how process execution order works.
3. **Use process synchronization** (wait() in Linux and WaitForSingleObject() in Windows).
4. **Demonstrate IPC using pipes** to send messages between processes.
5. **Implement shared memory** (optional, advanced task) to exchange data between processes.

Assignment Tasks

1. Creating a New Process

Goal: Create a program that spawns a child process, and both parent and child print messages.

Linux:

- Use fork() to create a new process.
- Use getpid() and getppid() to print **Process IDs**.
- Use sleep() to observe execution order.

Windows:

- Use CreateProcess() to start a new process.
- Print **Process ID** using GetProcessId().

Expected Output:

Parent Process: PID=1234
Child Process: PID=5678, Parent PID=1234

2. Synchronizing Parent and Child Process

Goal: Modify the program so the parent waits for the child to finish execution.

Linux:

- Use waitpid() to **wait** for the child process.

Windows:

- Use WaitForSingleObject() to **wait** for the child process.

Expected Output:

Child Process: PID=5678, Parent PID=1234
Parent Process: Child has finished execution.

3. Inter-Process Communication Using Pipes

Goal: Establish a **one-way communication channel** between parent and child using a pipe.

Linux:

- Use pipe() to create a **pipe**.
- The **parent process** writes a message, and the **child reads** it.

Windows:

- Use CreatePipe() to set up a pipe.
- The **parent process** writes, and the **child reads**.

Expected Output:
Parent Process: Writing "Hello from Parent"
Child Process: Received "Hello from Parent"

4. **Creating Multiple Child Processes**

Goal: Modify the program to spawn **multiple child processes**.
Linux:

- Use a **loop with fork()** to create 3 child processes.

Windows:

- Use a **loop with CreateProcess()** to create 3 child processes.

Expected Output:
Parent Process: PID=1000
Child 1: PID=1001, Parent PID=1000
Child 2: PID=1002, Parent PID=1000
Child 3: PID=1003, Parent PID=1000

5. **Shared Memory (Advanced Task)**

Goal: Implement **shared memory** for communication between processes.
Linux:

- Use shmget() and shmat() to create shared memory.
- The **parent writes** data, and the **child reads** it.

Windows:

- Use CreateFileMapping() and MapViewOfFile().

Expected Output:
Parent Process: Writing "Shared Memory Example"
Child Process: Read "Shared Memory Example"

Deliverables
You must submit:

1. **Source Code** for each task.
2. **Execution Logs or Screenshots** showing output.
3. **A Short Report**, explaining:
    - The **process creation steps**.
    - How synchronization works.
    - How IPC mechanisms function.

Tips for Success

- **Test Each Step Separately:** Start with simple process creation before adding IPC.
- **Use Debugging Prints:** Print PIDs and message exchanges to understand execution order.
- **Ensure Proper Cleanup:** Avoid zombie processes by always waiting for child processes.