Task 1: Process Creation Utility

```python
# process_management.py
# Author: jai
# Assignment: OS Lab - Process Creation and Management

import os
import time
import subprocess

# ---------- Task 1: Process Creation ----------
def task1_process_creation(n=3):
    print("\n--- Task 1: Process Creation (by jai) ---")
    children = []
    for i in range(n):
        pid = os.fork()
        if pid == 0:  # Child
            print(f"[Child] PID={os.getpid()}, PPID={os.getppid()}, Message=Hello from jai's child {i}")
            os._exit(0)
        else:
            children.append(pid)
    for cpid in children:
        os.waitpid(cpid, 0)
```

Task 2: Command Execution Using exec()

```python
# ---------- Task 2: Command Execution ----------
def task2_exec_commands(commands=["date", "ls", "ps"]):
    print("\n--- Task 2: Command Execution using execvp/subprocess (by jai) ---")
    for cmd in commands:
        pid = os.fork()
        if pid == 0:  # Child
            print(f"[Child] Executing command: {cmd} (by jai)")
            os.execvp(cmd, [cmd])
        else:
            os.waitpid(pid, 0)
```

Task 3: Zombie & Orphan Processes

```python
# ---------- Task 3: Zombie & Orphan ----------
def task3_zombie_orphan():
    print("\n--- Task 3: Zombie & Orphan Processes (by jai) ---")

    # Zombie
    pid = os.fork()
    if pid == 0:  # Child
        print(f"[Zombie Child] PID={os.getpid()}, PPID={os.getppid()} - exiting immediately (jai)")
        os._exit(0)
    else:
        print(f"[Parent] Created zombie process with PID={pid}, not waiting (jai)")
        time.sleep(5)  # Observe with: ps -el | grep defunct

    # Orphan
    pid = os.fork()
    if pid == 0:  # Child
        time.sleep(5)
        print(f"[Orphan Child] PID={os.getpid()}, PPID={os.getppid()} (jai)")
        os._exit(0)
    else:
        print(f"[Parent] Exiting before child finishes, making orphan (jai)")
        os._exit(0)
```

Task 4: Inspecting Process Info from /proc

```python
# ---------- Task 4: Inspect /proc ----------
def task4_proc_inspection(pid):
    print(f"\n--- Task 4: Inspecting /proc/{pid} (by jai) ---")
    try:
        with open(f"/proc/{pid}/status") as f:
            status = f.read()
        exe_path = os.readlink(f"/proc/{pid}/exe")
        fds = os.listdir(f"/proc/{pid}/fd")

        print(f"Process {pid} Info:")
        print("---- STATUS ----")
        print("\n".join(status.splitlines()[:10]))  # print first 10 lines
        print(f"---- Executable Path: {exe_path}")
        print(f"---- Open FDs: {fds}")

    except Exception as e:
        print(f"Error reading /proc/{pid}: {e}")
```

Task 5: Process Prioritizationp

```python
# ---------- Task 5: Process Prioritization ----------
def cpu_task():
    total = 0
    for i in range(10**7):
        total += i
    print(f"[Child {os.getpid()} by jai] Finished CPU task")


def task5_prioritization():
    print("\n--- Task 5: Process Prioritization with nice() (by jai) ---")
    for i, priority in enumerate([0, 5, 10]):
        pid = os.fork()
        if pid == 0:  # Child
            os.nice(priority)
            print(f"[Child] PID={os.getpid()}, Priority={priority} (jai)")
            cpu_task()
            os._exit(0)
    # Wait for all children
    for _ in range(3):
        os.wait()
```