Name: Sanchi Shukla

Roll No: 2301410028

Course: BTech CSE (Cyber Security)

ASSIGNMENT 2

Problem Statement:

Modern operating systems are responsible for initializing system components, creating processes, managing execution, and gracefully shutting down. This lab aims to simulate these core concepts using Python, helping students visualize how processes are handled at the OS level. The focus is on creating a simplified startup mechanism that spawns multiple processes and logs their lifecycle using the multiprocessing and logging modules. This hands-on simulation enhances conceptual clarity and promotes coding proficiency in scripting real-world OS behavior.

Sub-Task 1: Initialize the logging configuration

CODE

File Actions Edit View Help

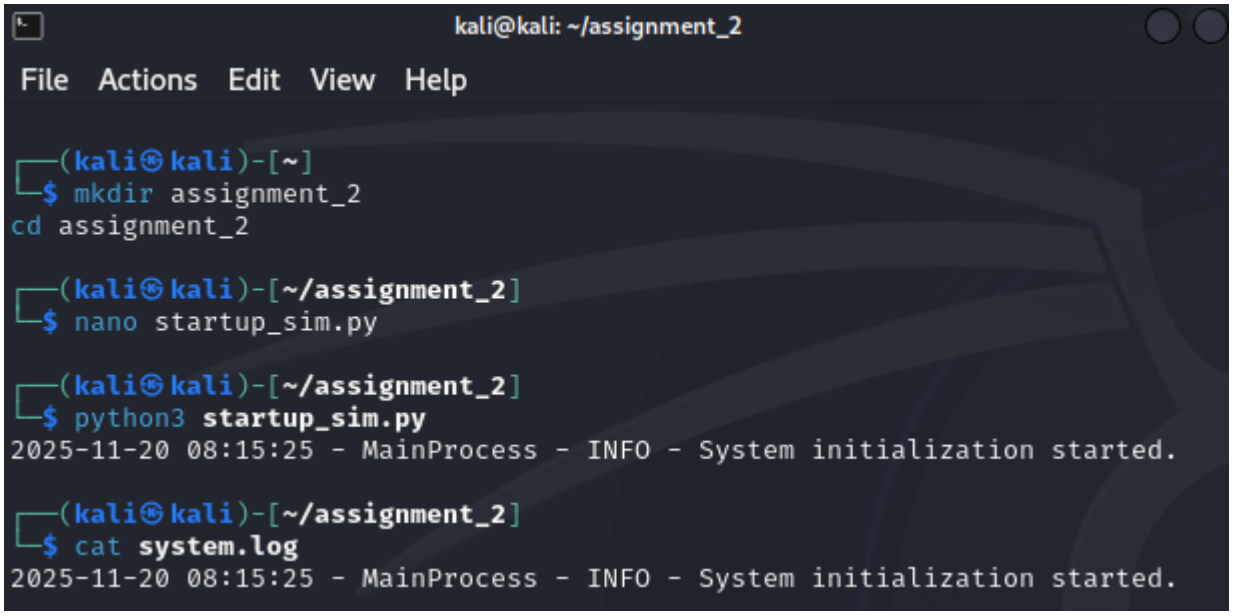
GNU nano 8.0

```
import logging

def init_logging():
    logging.basicConfig(
        level=logging.INFO,                    # Log INFO and above
        format="%(asctime)s - %(processName)s - %(levelname)s - %(message)s",
        datefmt="%Y-%m-%d %H:%M:%S",
        handlers=[
            logging.FileHandler("system.log"),    # Save logs to file
            logging.StreamHandler()               # Also print to terminal
        ]
    )

if __name__ == "__main__":
    init_logging()
    logging.info("System initialization started.")
```

OUTPUT


kali@kali: ~/assignment_2
File Actions Edit View Help
(kali㉿kali)-[~]
$ mkdir assignment_2
cd assignment_2
(kali㉿kali)-[~/assignment_2]
$ nano startup_sim.py
(kali㉿kali)-[~/assignment_2]
$ python3 startup_sim.py
2025-11-20 08:15:25 - MainProcess - INFO - System initialization started.
(kali㉿kali)-[~/assignment_2]
$ cat system.log
2025-11-20 08:15:25 - MainProcess - INFO - System initialization started.

Sub-Task 2: Define a function that simulates a process task (e.g., sleep for 2 seconds).

CODE

```
File  Actions  Edit  View  Help
  GNU nano 8.0
import time

def process_task():
    print("Process started.")
    time.sleep(2)
    print("Process finished.")

if __name__ == "__main__":
    process_task()
import time

def process_task():
    # This function simulates some work done by a process
    print("Process started.")
    time.sleep(2)  # simulate a 2-second task
    print("Process finished.")
```

OUTPUT

```
┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ nano process_task.py


┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ python3 process_task.py
Process started.
Process finished.
```

Sub-Task 3: Create at least two processes and start them concurrently.

CODE

```
File  Actions  Edit  View  Help
  GNU nano 8.0
from multiprocessing import Process
import time

def process_task():
    print("Process started.")
    time.sleep(2)
    print("Process finished.")

if __name__ == "__main__":
    # Create two processes
    p1 = Process(target=process_task)
    p2 = Process(target=process_task)

    # Start both processes
    p1.start()
    p2.start()

    # Wait for both processes to complete
    p1.join()
    p2.join()

    print("Both processes completed.")
```

OUTPUT

```
┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ nano multitask.py

┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ python3 multitask.py
Process started.
Process started.
Process finished.
Process finished.
Both processes completed.
```

Sub-Task 4: Ensure proper termination and joining of processes, and verify the output in the log file.

CODE

```
File  Actions  Edit  View  Help
  GNU nano 8.0
import logging
from multiprocessing import Process
import time

# ———— Initialize Logging ————
def init_logging():
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(processName)s - %(levelname)s - %(message)s",
        datefmt="%Y-%m-%d %H:%M:%S",
        handlers=[
            logging.FileHandler("system.log"),
            logging.StreamHandler()
        ]
    )

# ———— Process Task ————
def process_task():
    logging.info("Process started its task.")
    time.sleep(2)
    logging.info("Process finished its task.")

# ———— Main Program ————
if __name__ == "__main__":
    init_logging()
    logging.info("System startup simulation beginning...")

    # Create two processes
    p1 = Process(target=process_task, name="Process-1")
    p2 = Process(target=process_task, name="Process-2")

    # Start processes
    p1.start()
    p2.start()

    # Wait for them to finish
    p1.join()
    p2.join()

    logging.info("All processes have terminated. System shutdown complete.")
```

OUTPUT

```
┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ nano process_logging.py

┌──(kali㉿kali)-[~/os_simulation/assignment_2]
└─$ python3 process_logging.py
2025-11-20 08:37:50 - MainProcess - INFO - System startup simulation beginning...
2025-11-20 08:37:50 - Process-1 - INFO - Process started its task.
2025-11-20 08:37:50 - Process-2 - INFO - Process started its task.
2025-11-20 08:37:52 - Process-1 - INFO - Process finished its task.
2025-11-20 08:37:52 - Process-2 - INFO - Process finished its task.
2025-11-20 08:37:52 - MainProcess - INFO - All processes have terminated. System shutdown complete.
```