ASSIGNMENT-2

NAME-ANJALI
ROLL NO.-2301420021
COURSE-BTech CSE(DS)

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.

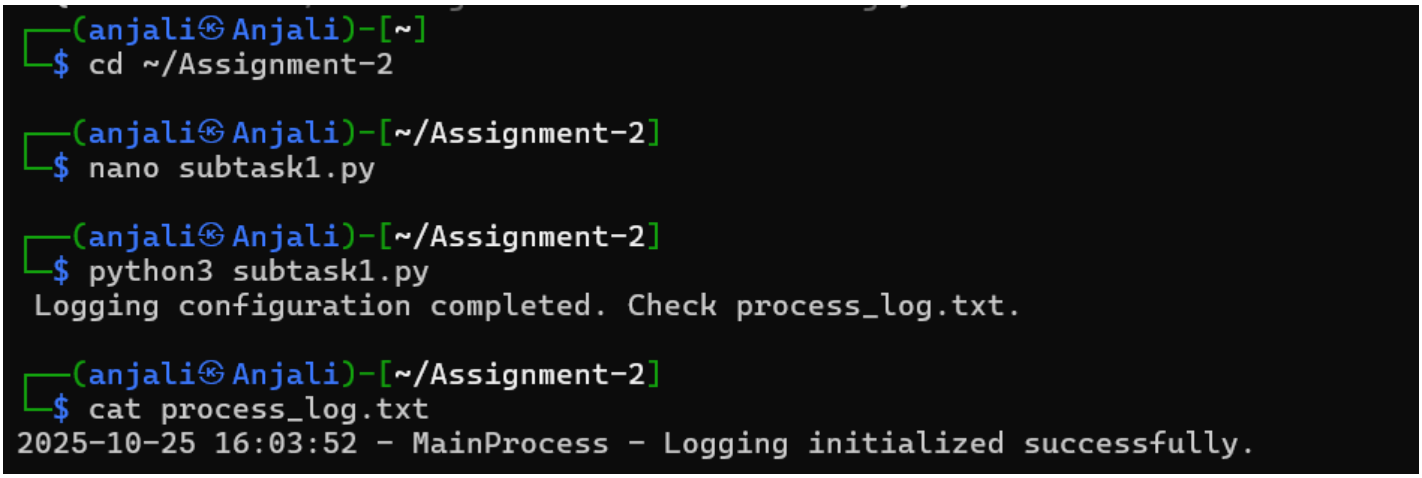
Code:

```
  GNU nano 8.4                                                    subtask1.py
"""
Sub-Task 1: Initialize logging configuration
"""
import logging

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging. INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

logging.info("Logging initialized successfully.")
print(" Logging configuration completed. Check process_log.txt.")
```

Output:


(anjali㉿Anjali)-[~]
$ cd ~/Assignment-2
(anjali㉿Anjali)-[~/Assignment-2]
$ nano subtask1.py
(anjali㉿Anjali)-[~/Assignment-2]
$ python3 subtask1.py
Logging configuration completed. Check process_log.txt.
(anjali㉿Anjali)-[~/Assignment-2]
$ cat process_log.txt
2025-10-25 16:03:52 - MainProcess - Logging initialized successfully.

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

Code:

```
  GNU nano 8.4                                                    subtask2.py
"""
Sub-Task 2: Define a process function and test it in the main process.
"""

import logging
import time

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging.INFO,
    format='%(asctime)s -%(processName)s -%(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

def system_process(task_name):
    logging. info(f"{task_name} started")
    time.sleep(2)
    logging.info(f"{task_name} ended")

if __name__ == "__main__":
    print("Running system_process() once ... ")
    system_process("Test-Process")
    print("Task completed. Check process_log.txt.")
```

Output:

```
┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ nano subtask2.py

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ python3 subtask2.py
Running system_process() once ...
Task completed. Check process_log.txt.

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ cat process_log.txt
2025-10-25 16:34:04 -MainProcess -Test-Process started
2025-10-25 16:34:06 -MainProcess -Test-Process ended
```

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

Code:

```
  GNU nano 8.4                                                  subtask3.py
"""
Sub-Task 3: Create and start multiple processes concurrently.
"""

import multiprocessing
import logging
import time

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging. INFO,
    format='%(asctime)s -%(processName)s -%(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

def system_process(task_name):
    logging. info(f"{task_name} started")
    time.sleep(2)
    logging. info(f"{task_name} ended")

if __name__ == "__main__":
    print("System Starting ... ")

    p1= multiprocessing.Process(target=system_process, args=("Process-1",))
    p2 = multiprocessing.Process(target=system_process, args=("Process-2", ))

    p1.start()
    p2.start()

    print("Processes started concurrently. Check process_log.txt.")
```

Output:

```
┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ nano subtask3.py

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ python3 subtask3.py
System Starting ...
Processes started concurrently. Check process_log.txt.

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ cat process_log.txt
2025-10-25 16:16:41 -Process-1 -Process-1 started
2025-10-25 16:16:41 -Process-2 -Process-2 started
2025-10-25 16:16:43 -Process-1 -Process-1 ended
2025-10-25 16:16:43 -Process-2 -Process-2 ended
```

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

Code:

```
  GNU nano 8.4                                                    subtask4.py
"""
Sub-Task 4: Join processes and confirm proper termination.
"""

import multiprocessing
import logging
import time

logging.basicConfig(
    filename='process_log.txt',
    filemode='w',
    level=logging. INFO,
    format='%(asctime)s - %(processName)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

def system_process(task_name):
    logging.info(f"{task_name} started")
    time.sleep(2)
    logging.info(f"{task_name} ended")

if __name__ == "__main__":
    print("System Starting ... ")

    p1 = multiprocessing. Process(target=system_process, args=("Process-1",))
    p2 = multiprocessing. Process(target=system_process, args=("Process-2",))

    p1.start()
    p2.start()

    p1. join()
    p2.join()

    print("System Shutdown.")
    logging. info("All processes completed. System shutting down.")
```

Output:

```
┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ nano subtask4.py

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ python3 subtask4.py
System Starting ...
System Shutdown.

┌──(anjali㉿Anjali)-[~/Assignment-2]
└─$ cat process_log.txt
2025-10-25 16:26:20 - Process-1 - Process-1 started
2025-10-25 16:26:20 - Process-2 - Process-2 started
2025-10-25 16:26:22 - Process-1 - Process-1 ended
2025-10-25 16:26:22 - Process-2 - Process-2 ended
2025-10-25 16:26:22 - MainProcess - All processes completed. System shutting down.
```