

# KR MANGALAM UNIVERSITY

SOHNA ROAD, GURUGRAM



## OPERATING SYSTEM LAB FILE

COURSE CODE-ENCS351

**Submitted by**

Name: Aldrin Debbarma  
No.: 2401011482

**Submitted To**

Dr Satinder Pal Singh Roll

**Program:** BTech CSE  
**Section:** 6/F  
**:Session:** 2025/26  
**Date:** 02/12/2025

## Sub-Task 1: Initialize the logging configuration

**Objective:** Set up the logging system to log messages with timestamps and process names.

```
import logging
```

```
# Setup logger
```

```
logging.basicConfig(
```

```
filename='process_log.txt',
```

```
level=logging.INFO,
```

```
format='%(asctime)s - %(processName)s - %(message)s'
```

```
)
```

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file tree under 'PYTHON' containing 'main.py', 'process\_log.txt', 'README.md', 'service\_dependency\_log.txt', and 'system\_boot\_log.txt'. The main editor area shows the following Python code:

```
1 import os
2
3 # Configure log file
4 logging.basicConfig(
5     filename='process_log.txt',
6     level=logging.INFO,
7     format='%(asctime)s - %(processName)s - %(message)s'
8 )
9
10
11 def boot_step(step_name, delay=1):
12     """Simulates one step in system boot."""
13     print(f'[BOOT] {step_name}...')
14     logging.info(f'{step_name} started')
15     time.sleep(delay)
16     logging.info(f'{step_name} completed')
17     print(f'[DONE] {step_name} complete.\n')
18
19 def system_boot():
20     print("== System Boot Simulation ==\n")
21     logging.info("System boot sequence initiated")
22
23     steps = [
24         "Initializing Kernel",
25         "Loading Device Drivers",
26         "Mounting File System",
27         "Starting Network Services",
28         "Launching User Processes",
29         "Activating Scheduler"
30     ]
31
32     # Execute steps one by one (sequential boot)
33     for step in steps:
34         boot_step(step, delay=1)
35
36     logging.info("System Boot Completed Successfully")
37     print("[SYSTEM] Boot process completed successfully.\n")
38
39 if __name__ == "__main__":
40     system_boot()
```

The bottom terminal window shows the output of running the script:

```
[ashishyaduvanshi@ashish-MacBook-Air python % python3 main.py
== System Boot Simulation ==
[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.
[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.
[BOOT] Mounting File System...
[DONE] Mounting File System complete.
[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.
[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.
[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.
[SYSTEM] Boot process completed successfully.
ashishyaduvanshi@ashish-MacBook-Air python % ]
```

The status bar at the bottom indicates the code is at line 40, column 24, in Python mode, with BLACKBOX Autocomplete: ON and BLACKBOX: Open Chat.

## Sub-Task 2: Define a function that simulates a process task

**Objective:** Write a function that mimics the work of a system process.

```
import time
```

```
# Dummy function to simulate a task
```

```
def system_process(task_name):
```

```
    logging.info(f'{task_name} started')
```

```
    time.sleep(2) # Simulate task delay
```

```
    logging.info(f'{task_name} ended')
```

The screenshot shows a code editor interface with a dark theme. The left sidebar has an 'EXPLORER' section showing a 'PYTHON' folder with files: main.py, process\_log.txt, README.md, and service\_dependency\_log.txt. The main area displays the content of 'main.py'. The code uses the 'multiprocessing' module to simulate the start and completion of three services: Database\_Service, Web\_Server, and Application\_Service. It includes logging and time delays to mimic real-world behavior. The bottom right corner shows a terminal window with the output of running the script.

```
import multiprocessing
import logging
import time
import os

# Configure log file
logging.basicConfig(
    filename='service_dependency_log.txt',
    level=logging.INFO,
    format='[%(asctime)s - %(processName)s (PID: %(processId)d) - %(message)s'
)

def service_task(name, delay):
    """Simulates a dependent system service."""
    logging.info(f'{name} started')
    print(f'[{name}] PID={os.getpid()} started.')
    time.sleep(delay) # Simulate work
    logging.info(f'{name} ended')
    print(f'[{name}] PID={os.getpid()} completed.\n')

def main():
    print('*** System Service Dependency Simulation ***\n')

    # Define dependent services in order
    services = [
        ('Database_Service', 3),
        ('Web_Server', 2),
        ('Application_Service', 2)
    ]

    # Start services one after another (dependency order)
    for name, delay in services:
        print(f'[Parent] Starting {name} (depends on previous service)...')
        p = multiprocessing.Process(target=service_task, args=(name, delay), name=name)
        p.start()
        p.join() # Wait until current service finishes before starting next
        print(f'[Parent] {name} finished successfully.\n')

    logging.info('All dependent services started and completed successfully.')
    print(f'[Parent] All dependent services have been started in order.')

if __name__ == "__main__":
    main()
```

[Parent] Starting Database\_Service (depends on previous service)...  
[Database\_Service] PID=89584 started.  
[Database\_Service] PID=89584 completed.  
[Parent] Database\_Service finished successfully.  
  
[Parent] Starting Web\_Server (depends on previous service)...  
[Web\_Server] PID=89589 started.  
[Web\_Server] PID=89589 completed.  
[Parent] Web\_Server finished successfully.  
  
[Parent] Starting Application\_Service (depends on previous service)...  
[Application\_Service] PID=89593 started.  
[Application\_Service] PID=89593 completed.  
[Parent] Application\_Service finished successfully.  
  
[Parent] All dependent services have been started in order.

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

**Objective:** Use the multiprocessing module to initiate parallel tasks.

```
import multiprocessing
```

```
if __name__ == '__main__':
```

```
    print("System Starting...")
```

```
# Create processes
```

```
p1 = multiprocessing.Process(target=system_process, args=('Process-1',))
```

```
p2 = multiprocessing.Process(target=system_process, args=('Process-2',))
```

```
# Start processes
```

```
p1.start()
```

```
p2.start()
```

```
main.py
1 import os
2
3 # Configure log file
4 logging.basicConfig(
5     filename='system_boot_log.txt',
6     level=logging.INFO,
7     format='%(asctime)s - %(message)s'
8 )
9
10 def boot_step(step_name, delay=1):
11     """Imitates one step in system boot."""
12     print(f'[BOOT] {step_name}...')
13     logging.info(f'{step_name} started')
14     time.sleep(delay)
15     logging.info(f'{step_name} completed')
16     print(f'[DONE] {step_name} complete.\n')
17
18 def system_boot():
19     print('==> System Boot Simulation ==\n')
20     logging.info('System Boot sequence initiated')
21
22     steps = [
23         "Initializing Kernel",
24         "Loading Device Drivers",
25         "Mounting File System",
26         "Starting Network Services",
27         "Launching User Processes",
28         "Activating Scheduler"
29     ]
30
31     # Execute steps one by one (sequential boot)
32     for step in steps:
33         boot_step(step, delay=1)
34
35     logging.info("System Boot Completed Successfully")
36     print("[SYSTEM] Boot process completed successfully.\n")
37
38 if __name__ == "__main__":
39     system_boot()
```

```
ashishyaduvanshi@ashish-MacBook-Air python % python3 main.py
==> System Boot Simulation ==
[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.
[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.
[BOOT] Mounting File System...
[DONE] Mounting File System complete.
[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.
[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.
[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.
[SYSTEM] Boot process completed successfully.

ashishyaduvanshi@ashish-MacBook-Air python %
```

## Sub-Task 4: Ensure proper termination and verify logs

**Objective:** Wait for processes to complete and confirm the shutdown.

```
# Wait for processes to complete
```

```
p1.join()
```

```
p2.join()
```

```
print("System Shutdown.")
```

```
main.py
1 import multiprocessing
2 import logging
3 import time
4 import os
5
6 # Configure logging
7 logging.basicConfig(
8     filename='system_shutdown_log.txt',
9     level=logging.INFO,
10    format='%(asctime)s - %(processName)s (PID: %(processId)d) - %(message)s'
11 )
12
13 def shutdown_service(name, delay):
14     """Simulates stopping of a system service."""
15     print(f"({name}) PID={os.getpid()} shutting down...")
16     logging.info(f"({name}) shutdown initiated")
17     time.sleep(delay)
18     logging.info(f"({name}) shutdown completed")
19     print(f"({name}) PID={os.getpid()} shutdown completed.\n")
20
21 def main():
22     print("== System Shutdown Simulation ==\n")
23     logging.info("System shutdown sequence initiated")
24
25     services = [
26         ("Application_Service", 2),
27         ("Web_Server", 2),
28         ("Database_Service", 3)
29     ]
30
31     process_list = []
32
33     # Start all shutdown processes
34     for name, delay in services:
35         p = multiprocessing.Process(target=shutdown_service, args=(name, delay), name=name)
36         process_list.append(p)
37         p.start()
38
39     # Wait for all to finish
40     for p in process_list:
41         p.join()
42
43     print("[SYSTEM] All services have been shut down successfully.")
44     logging.info("System shutdown completed successfully.\n")
45
46 if __name__ == "__main__":
47     main()

ashishyaduvanshi@ashishyaduvanshi-MacBook-Air: ~ python3 main.py
== System Shutdown Simulation ==
[Application_Service] PID=89987 shutting down...
[Web_Server] PID=89988 shutting down...
[Database_Service] PID=89989 shutting down...
[Application_Service] PID=89997 shutdown completed.

[Web_Server] PID=89998 shutdown completed.

[Database_Service] PID=89999 shutdown completed.

[SYSTEM] All services have been shut down successfully.
ashishyaduvanshi@ashishyaduvanshi-MacBook-Air: ~
```