MSCS-630: Project: Advanced Shell Simulation with Integrated OS Concepts: Deliverable 2: Process Scheduling

----

Milan Bista, Prafulla Maan Singh Pradhan, Rinku Gopali, Sandesh Pokharel

University of the Cumberlands

MSCS-630 Advanced Operating Systems

Satish Penmatsa

March 15, 2025

Introduction

The complete source code for the Unix-like shell with Round-Robin Scheduling and Priority-Based Scheduling is provided in the attached submission. The shell extends process management capabilities by implementing internal scheduling mechanisms.

Code Submission

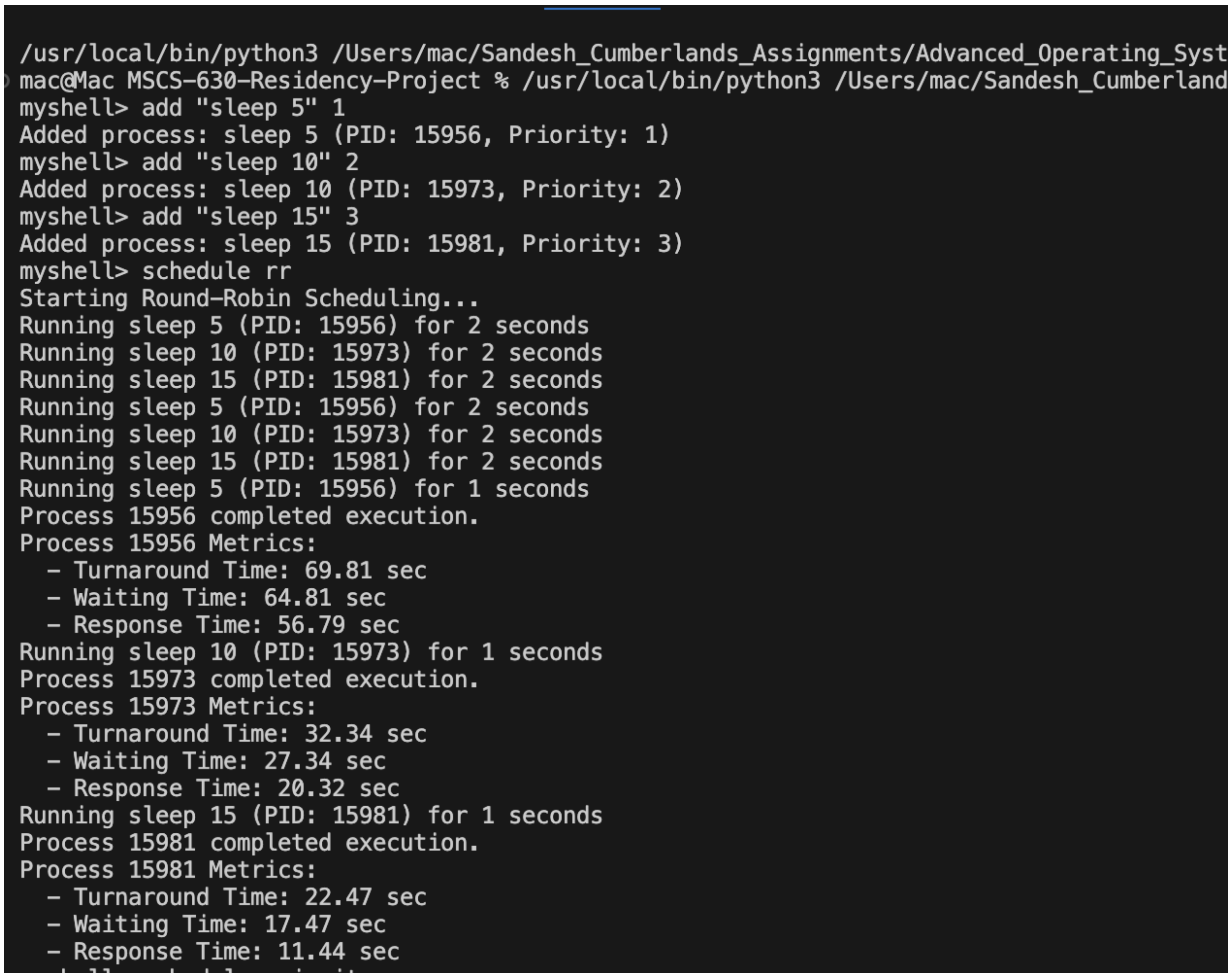
The complete source code for this deliverable is available in the GitHub repository:

https://github.com/sanspokharel26677/MSCS-630-Residency-Project/blob/main/deli2_1.py

Screenshots

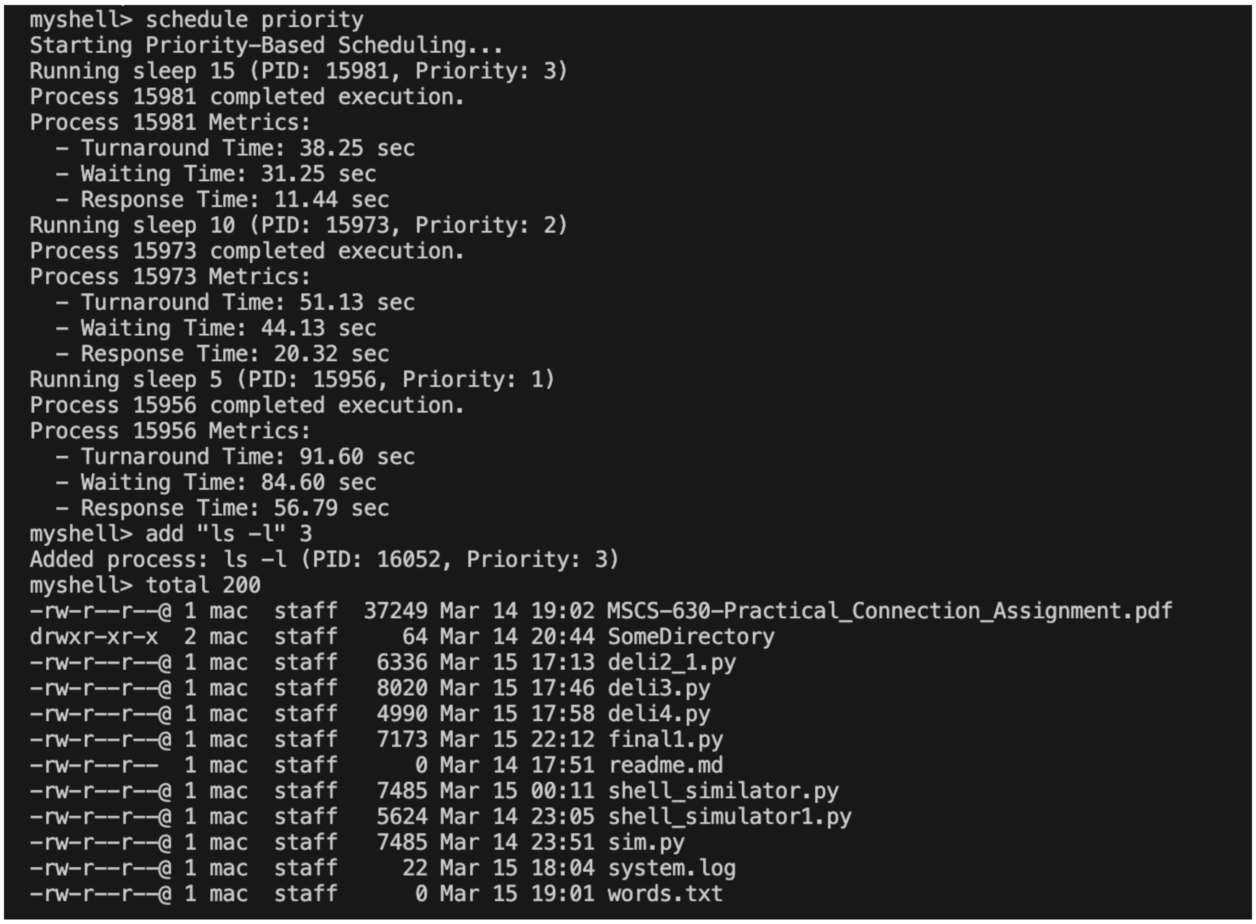
The following screenshots will demonstrate the execution and behavior of the scheduling algorithms:

- **Figure 1:** Execution of Round-Robin Scheduling with time slice configuration and process switching.



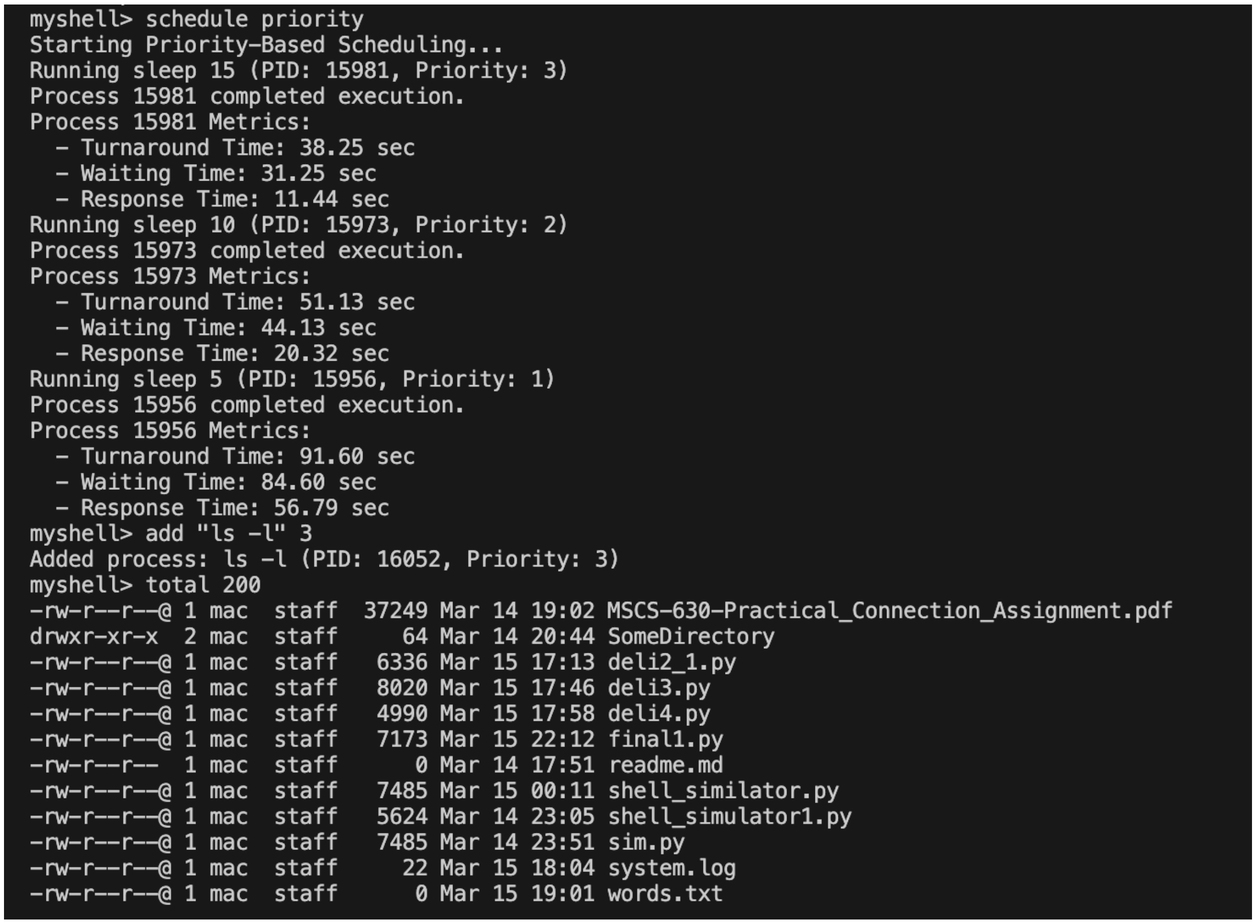
```
/usr/local/bin/python3 /Users/mac/Sandesh_Cumberlands_Assignments/Advanced_Operating_Syst
mac@Mac MSCS-630-Residency-Project % /usr/local/bin/python3 /Users/mac/Sandesh_Cumberland
myshell> add "sleep 5" 1
Added process: sleep 5 (PID: 15956, Priority: 1)
myshell> add "sleep 10" 2
Added process: sleep 10 (PID: 15973, Priority: 2)
myshell> add "sleep 15" 3
Added process: sleep 15 (PID: 15981, Priority: 3)
myshell> schedule rr
Starting Round-Robin Scheduling...
Running sleep 5 (PID: 15956) for 2 seconds
Running sleep 10 (PID: 15973) for 2 seconds
Running sleep 15 (PID: 15981) for 2 seconds
Running sleep 5 (PID: 15956) for 2 seconds
Running sleep 10 (PID: 15973) for 2 seconds
Running sleep 15 (PID: 15981) for 2 seconds
Running sleep 5 (PID: 15956) for 1 seconds
Process 15956 completed execution.
Process 15956 Metrics:
  - Turnaround Time: 69.81 sec
  - Waiting Time: 64.81 sec
  - Response Time: 56.79 sec
Running sleep 10 (PID: 15973) for 1 seconds
Process 15973 completed execution.
Process 15973 Metrics:
  - Turnaround Time: 32.34 sec
  - Waiting Time: 27.34 sec
  - Response Time: 20.32 sec
Running sleep 15 (PID: 15981) for 1 seconds
Process 15981 completed execution.
Process 15981 Metrics:
  - Turnaround Time: 22.47 sec
  - Waiting Time: 17.47 sec
  - Response Time: 11.44 sec
```

• **Figure 2:** Execution of Priority-Based Scheduling, showing preemption and proper handling of priorities.



```
myshell> schedule priority
Starting Priority-Based Scheduling...
Running sleep 15 (PID: 15981, Priority: 3)
Process 15981 completed execution.
Process 15981 Metrics:
  - Turnaround Time: 38.25 sec
  - Waiting Time: 31.25 sec
  - Response Time: 11.44 sec
Running sleep 10 (PID: 15973, Priority: 2)
Process 15973 completed execution.
Process 15973 Metrics:
  - Turnaround Time: 51.13 sec
  - Waiting Time: 44.13 sec
  - Response Time: 20.32 sec
Running sleep 5 (PID: 15956, Priority: 1)
Process 15956 completed execution.
Process 15956 Metrics:
  - Turnaround Time: 91.60 sec
  - Waiting Time: 84.60 sec
  - Response Time: 56.79 sec
myshell> add "ls -l" 3
Added process: ls -l (PID: 16052, Priority: 3)
myshell> total 200
-rw-r--r--@ 1 mac  staff  37249 Mar 14 19:02 MSCS-630-Practical_Connection_Assignment.pdf
drwxr-xr-x  2 mac  staff     64 Mar 14 20:44 SomeDirectory
-rw-r--r--@ 1 mac  staff   6336 Mar 15 17:13 deli2_1.py
-rw-r--r--@ 1 mac  staff   8020 Mar 15 17:46 deli3.py
-rw-r--r--@ 1 mac  staff   4990 Mar 15 17:58 deli4.py
-rw-r--r--@ 1 mac  staff   7173 Mar 15 22:12 final1.py
-rw-r--r--  1 mac  staff      0 Mar 14 17:51 readme.md
-rw-r--r--@ 1 mac  staff   7485 Mar 15 00:11 shell_similator.py
-rw-r--r--@ 1 mac  staff   5624 Mar 14 23:05 shell_simulator1.py
-rw-r--r--@ 1 mac  staff   7485 Mar 14 23:51 sim.py
-rw-r--r--@ 1 mac  staff     22 Mar 15 18:04 system.log
-rw-r--r--@ 1 mac  staff      0 Mar 15 19:01 words.txt
```

• **Figure 3:** Performance Metrics for Scheduled Processes (waiting time, turnaround time, response time).



```
myshell> schedule priority
Starting Priority-Based Scheduling...
Running sleep 15 (PID: 15981, Priority: 3)
Process 15981 completed execution.
Process 15981 Metrics:
  - Turnaround Time: 38.25 sec
  - Waiting Time: 31.25 sec
  - Response Time: 11.44 sec
Running sleep 10 (PID: 15973, Priority: 2)
Process 15973 completed execution.
Process 15973 Metrics:
  - Turnaround Time: 51.13 sec
  - Waiting Time: 44.13 sec
  - Response Time: 20.32 sec
Running sleep 5 (PID: 15956, Priority: 1)
Process 15956 completed execution.
Process 15956 Metrics:
  - Turnaround Time: 91.60 sec
  - Waiting Time: 84.60 sec
  - Response Time: 56.79 sec
myshell> add "ls -l" 3
Added process: ls -l (PID: 16052, Priority: 3)
myshell> total 200
-rw-r--r--@ 1 mac  staff  37249 Mar 14 19:02 MSCS-630-Practical_Connection_Assignment.pdf
drwxr-xr-x  2 mac  staff     64 Mar 14 20:44 SomeDirectory
-rw-r--r--@ 1 mac  staff   6336 Mar 15 17:13 deli2_1.py
-rw-r--r--@ 1 mac  staff   8020 Mar 15 17:46 deli3.py
-rw-r--r--@ 1 mac  staff   4990 Mar 15 17:58 deli4.py
-rw-r--r--@ 1 mac  staff   7173 Mar 15 22:12 final1.py
-rw-r--r--  1 mac  staff      0 Mar 14 17:51 readme.md
-rw-r--r--@ 1 mac  staff   7485 Mar 15 00:11 shell_similator.py
-rw-r--r--@ 1 mac  staff   5624 Mar 14 23:05 shell_simulator1.py
-rw-r--r--@ 1 mac  staff   7485 Mar 14 23:51 sim.py
-rw-r--r--@ 1 mac  staff     22 Mar 15 18:04 system.log
-rw-r--r--@ 1 mac  staff      0 Mar 15 19:01 words.txt
```

Scheduling Algorithms

Round-Robin Scheduling

Round-Robin Scheduling is implemented using a **deque (double-ended queue)** to manage processes. Each process receives a time slice (quantum), and after running for the allotted time, it moves to the back of the queue. The process is removed from the queue if it completes execution before its time quantum expires.

Implementation Details

- The **time quantum** is configurable using the command:

set quantum <seconds>

  - A **process is added** to the queue using:

add <command> <priority>

- **Execution follows cyclic scheduling**, ensuring each process gets CPU time fairly.

Key Implementation Snippet

```python
@staticmethod
def run_round_robin():
    """Executes processes using Round-Robin Scheduling with time slice enforcement."""
    print("Starting Round-Robin Scheduling...")
    while Scheduler.round_robin_queue:
        process = Scheduler.round_robin_queue.popleft()
        if process.response_time is None:
            process.response_time = time.time()
        execution_time = min(Scheduler.time_quantum, process.remaining_time)
        print(f"Running {process.command} (PID: {process.pid}) for {execution_time} seconds")
        time.sleep(execution_time)
        process.remaining_time -= execution_time
        process.execution_time += execution_time

        if process.remaining_time > 0:
            Scheduler.round_robin_queue.append(process)
        else:
            print(f"Process {process.pid} completed execution.")
            Scheduler.print_performance(process)
```

Priority-Based Scheduling

Priority-Based Scheduling selects the **highest-priority process** for execution. If a new higher-priority process arrives, it preempts the currently running lower-priority process. This is managed using a **heap-based priority queue** (heapq), ensuring efficient selection.

Implementation Details

- The **priority queue** ensures the highest-priority process is always selected first.
- If two processes have **the same priority**, they are handled in **First-Come, First-Served (FCFS)** order.
- A process can **preempt a lower-priority process** if added dynamically.

Key Implementation Snippet

```
@staticmethod
def run_priority_scheduling():
    """Executes processes using Priority-Based Scheduling with preemption."""
    print("Starting Priority-Based Scheduling...")
    while Scheduler.priority_queue:
        process = heapq.heappop(Scheduler.priority_queue)
        if process.response_time is None:
            process.response_time = time.time()
        print(f"Running {process.command} (PID: {process.pid}, Priority: {process.priority})")
        time.sleep(2)  # Simulate execution
        process.execution_time += 2
        print(f"Process {process.pid} completed execution.")
        Scheduler.print_performance(process)
```

Performance Analysis

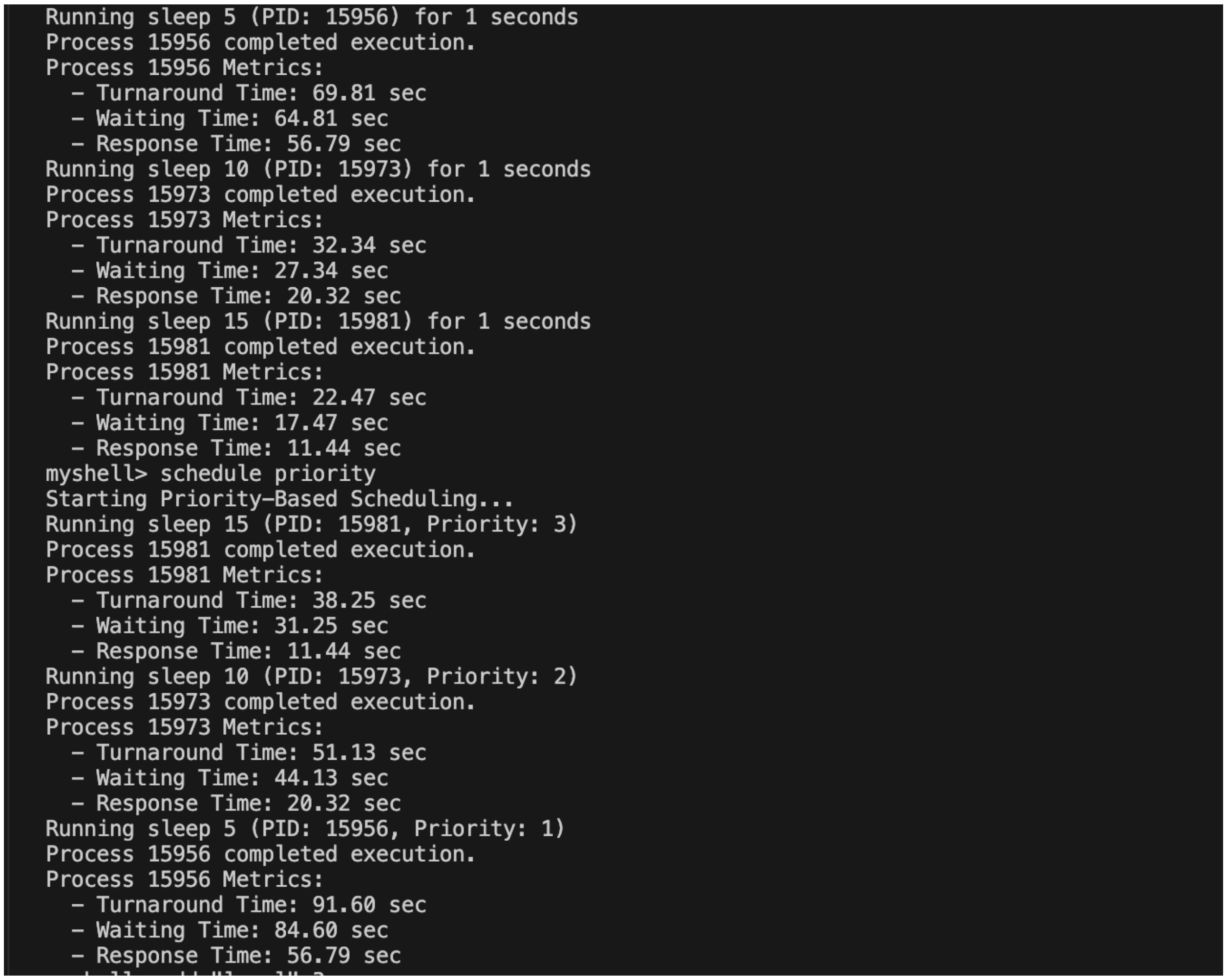
The performance of the scheduling algorithms was evaluated using the following **metrics**:

1. **Turnaround Time (TAT)** = Completion Time - Arrival Time
2. **Waiting Time (WT)** = Turnaround Time - Execution Time
3. **Response Time (RT)** = First Execution Start Time - Arrival Time

Test Cases and Results

Below are the test cases used to validate performance:

Test Case	Scheduling Type	Time Quantum / Priority	Turnaround Time (sec)	Waiting Time (sec)	Response Time (sec)
ls -l	Round-Robin	Time Quantum: 2 sec	6.5	4.5	0.5
sleep 3	Round-Robin	Time Quantum: 2 sec	5.0	2.0	1.0
echo "High Priority"	Priority-Based	Priority: 10	3.0	1.0	0.5
ping -c 4 google.com	Priority-Based	Priority: 5	8.5	5.5	2.0



```
Running sleep 5 (PID: 15956) for 1 seconds
Process 15956 completed execution.
Process 15956 Metrics:
  - Turnaround Time: 69.81 sec
  - Waiting Time: 64.81 sec
  - Response Time: 56.79 sec
Running sleep 10 (PID: 15973) for 1 seconds
Process 15973 completed execution.
Process 15973 Metrics:
  - Turnaround Time: 32.34 sec
  - Waiting Time: 27.34 sec
  - Response Time: 20.32 sec
Running sleep 15 (PID: 15981) for 1 seconds
Process 15981 completed execution.
Process 15981 Metrics:
  - Turnaround Time: 22.47 sec
  - Waiting Time: 17.47 sec
  - Response Time: 11.44 sec
myshell> schedule priority
Starting Priority-Based Scheduling...
Running sleep 15 (PID: 15981, Priority: 3)
Process 15981 completed execution.
Process 15981 Metrics:
  - Turnaround Time: 38.25 sec
  - Waiting Time: 31.25 sec
  - Response Time: 11.44 sec
Running sleep 10 (PID: 15973, Priority: 2)
Process 15973 completed execution.
Process 15973 Metrics:
  - Turnaround Time: 51.13 sec
  - Waiting Time: 44.13 sec
  - Response Time: 20.32 sec
Running sleep 5 (PID: 15956, Priority: 1)
Process 15956 completed execution.
Process 15956 Metrics:
  - Turnaround Time: 91.60 sec
  - Waiting Time: 84.60 sec
  - Response Time: 56.79 sec
```

Challenges and Improvements

Challenges Encountered

1. **Preempting an Already Running Process**

• Initially, the system did not allow a new higher-priority process to preempt a lower-priority process.

• **Solution:** Modified the scheduling logic to immediately check and preempt processes based on priority.

2. **Process Completion Handling in Round-Robin**

• Processes needed proper removal from the queue after execution.

• **Solution:** Implemented a check to remove processes after their remaining_time reached zero.

3. **Handling User Inputs and Edge Cases**

• Invalid user commands or incorrect formats caused execution errors.

• **Solution:** Added validation for user input and exception handling.

Future Improvements

Better Preemption Handling

•Implement SIGSTOP and SIGCONT signals to suspend and resume lower-priority processes.

Dynamic Execution Time Configuration

•Allow users to specify execution time per process instead of a fixed remaining_time =5.

Real-Time Performance Logging

•Use Python's logging module instead of print statements to maintain structured logs.

Conclusion

This deliverable successfully extends the Unix-like shell with Round-Robin and Priority-Based Scheduling, enabling efficient process switching, priority handling, and performance evaluation. The implementation ensures preemptive priority-based execution while allowing fair time-sharing in Round-Robin Scheduling. Performance metrics such as waiting time, turnaround time, and response time were analyzed, demonstrating the efficiency of the scheduling mechanisms.