Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Title: CPU Scheduling Priority Scheduling Algorithm

Submitted To

Atia Rahman Orthi
Lecturer
Department Of Computer Science & Engineering

Submitted By

Md. Atikul Islam Atik
Reg No:21201063
Sec:B1

Submission Date: 7.3.2025

Problem:

Process	Priority	Arrive Time	Burst Time
1	10	0	5
2	20	1	4
3	30	2	2
4	40	4	1

Output:

Process	Priority	Arrival	Burst	CT	TAT	WT	RT
1	10	0	5	12	12	7	0
2	20	1	4	8	7	3	0
3	30	2	2	4	2	0	0
4	40	4	1	5	1	0	0

Source Code:

```python
# time quanta = 1

def priority_preemptive(process_list, time_quanta):
    time = 0
    gantt_chart = []
    completed = {}
    backup = {}
    first_response = {}

    # Sort processes by arrival time initially
    process_list.sort(key=lambda x: x[3])

    for process in process_list:
        pid = process[1]
        backup[pid] = {"arrival_time": process[3], "burst_time": process[2], "priority": process[0]}
        first_response[pid] = -1

    remaining_processes = process_list[:]

    while remaining_processes:

        available_processes = []
        for p in remaining_processes:
            if p[3] <= time:
                available_processes.append(p)

        if not available_processes:
            gantt_chart.append("Idle")
            time += 1
            continue

        # Select the highest-priority process (higher number = higher priority)
        process = max(available_processes, key=lambda x: x[0])

        pid = process[1]

        # Record first response time if it's the first execution
        if first_response[pid] == -1:
            first_response[pid] = time - process[3]

        execution_time = min(time_quanta, process[2])

        gantt_chart.append(pid)
        time += execution_time
        process[2] -= execution_time

        # If process is completed
        if process[2] == 0:
            completion_time = time
            arrival_time = backup[pid]["arrival_time"]
            burst_time = backup[pid]["burst_time"]
            turnaround_time = completion_time - arrival_time
            waiting_time = turnaround_time - burst_time
            response_time = first_response[pid]

            completed[pid] = [arrival_time, burst_time, backup[pid]["priority"],
                              completion_time, turnaround_time, waiting_time, response_time]

            remaining_processes.remove(process)

    return {"gantt_chart": gantt_chart, "completed": completed}

def print_table(completed):
    print("\nProcess\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time\tResponse Time")
    for key, values in completed.items():
        print(f"{key}\t{values[0]}\t\t{values[1]}\t\t{values[2]}\t\t{values[3]}\t\t{values[4]}\t\t{values[5]}\t\t{values[6]}")

def collect_input():
    process_list = []
    n = int(input("Enter the number of processes: "))
    time_quanta = int(input("Enter the time quanta: "))  # Collecting time quanta

    for i in range(n):
        priority = int(input(f'Enter the priority for process {i+1} (Higher value -> Higher priority): '))
        pid = f"P{i+1}"
        arrival_time = int(input(f'Enter the arrival time for process {i+1}: '))
        burst_time = int(input(f'Enter the burst time for process {i+1}: '))
        process_list.append([priority, pid, burst_time, arrival_time])

    return process_list, time_quanta

if __name__ == "__main__":
    inputs = collect_input()
    result = priority_preemptive(inputs[0], inputs[1])

    print("\nGantt Chart:", result["gantt_chart"])
    print_table(result["completed"])
```

[Live Link of Code]

Output :

```
 atik  os/lab4   main !?   v3.13.2   09:54 PM
 python -u "/home/atik/Codes/python/os/lab4/priority_premptive.py"
Enter the number of processes: 4
Enter the time quanta: 1
Enter the priority for process 1 (Higher value -> Higher priority): 10
Enter the arrival time for process 1: 0
Enter the burst time for process 1: 5
Enter the priority for process 2 (Higher value -> Higher priority): 20
Enter the arrival time for process 2: 1
Enter the burst time for process 2: 4
Enter the priority for process 3 (Higher value -> Higher priority): 30
Enter the arrival time for process 3: 2
Enter the burst time for process 3: 2
Enter the priority for process 4 (Higher value -> Higher priority): 40
Enter the arrival time for process 4: 4
Enter the burst time for process 4: 1

Gantt Chart: ['P1', 'P2', 'P3', 'P3', 'P4', 'P2', 'P2', 'P2', 'P1', 'P1', 'P1', 'P1']

Process Arrival Time    Burst Time      Priority        Completion Time Turnaround Time Waiting Time    Response Time
P3      2               2               30              4               2               0               0
P4      4               1               40              5               1               0               0
P2      1               4               20              8               7               3               0
P1      0               5               10              12              12              7               0

 atik  os/lab4   main !?   v3.13.2   09:55 PM
```

Algorithm:

Input Process Details: Each process should have an ID, priority, arrival time, and burst time.

Initialize Variables:

- Set currentTime = 0.
- Set remainingTime = burstTime for all processes.

Find the Next Process to Execute:

- Identify the highest-priority process that has already arrived and still has remaining execution time.
- If multiple processes have the same priority, use arrival time as a tiebreaker (earlier arrival gets preference).

Execute the Process:

- Run the selected process for 1 time unit.
- Decrease its remainingTime by 1.

Check for Process Completion:

- If a process finishes execution (remainingTime == 0), record:
  - **Completion Time (CT)** = currentTime + 1.
  - **Turnaround Time (TAT)** = CT - Arrival Time.
  - **Waiting Time (WT)** = TAT - Burst Time.

Repeat Steps 3-5 until all processes are completed.

Display Results:

- Print **CT, TAT, WT** for each process.
- Optionally, calculate and display average TAT and WT for overall evaluation.

Conclusion:

- Priority scheduling is one of the most common scheduling algorithms used by the operating system to schedule processes based on their priority. Each process is assigned a priority.
  - **Waiting Time (WT):** The time a process spends waiting in the queue before it starts executing.
  - **Response Time (RT):** The time from process arrival to its first execution.

The output clearly shows the execution order in the Gantt chart and provides detailed scheduling results for each process. The results demonstrate that the Round Robin algorithm ensures fair CPU allocation by giving each process a fixed time slice, reducing starvation. However, selecting an appropriate time quantum is crucial to balancing responsiveness and efficiency.