|  |  |
| --- | --- |
| **Name: Jal Bafana** | **Roll no: K005** |
| **Class: Btech. Cyber Security (Sem-4)** | **Batch: K1** |
| **Date of Experiment: 14.02.2025** | **Date of Submission: 14.02.2025** |

**Introduction to Priority Scheduling**

Priority scheduling is a CPU scheduling method where each process is given a priority level. The CPU is allocated to the process with the highest priority. If two processes share the same priority, the scheduling follows the First Come First Serve (FCFS) method.

**Types of Priority Scheduling**

1. **Pre-emptive Priority Scheduling**  
   In pre-emptive priority scheduling, if a process is running and a new process with a higher priority arrives, the current process is interrupted. The CPU is allocated to the higher-priority process. This approach is commonly used in real-time systems where immediate responses are necessary.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Priority** |
| P1 | 0 | 5 | 3 |
| P2 | 1 | 3 | 1 |
| P3 | 2 | 8 | 4 |
| P4 | 3 | 6 | 2 |

* + At time 0, P1 starts executing.
  + At time 1, P2 arrives with a higher priority and preempts P1.
  + After P2 finishes, the next highest-priority process is executed.

**Advantages:**

* + High-priority tasks get CPU time quickly.
  + Suitable for applications that require immediate processing.

**Disadvantages:**

* + Increased overhead due to frequent context switching.
  + Low-priority processes may experience starvation and never execute.

1. **Non-Pre-emptive Priority Scheduling**  
   In non-pre-emptive priority scheduling, once a process begins execution, it cannot be interrupted. The CPU is assigned to the highest-priority process available, and it runs until completion before the next process is scheduled.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Priority** |
| P1 | 0 | 4 | 2 |
| P2 | 1 | 6 | 1 |
| P3 | 2 | 3 | 3 |

* + At time 0, P1 begins execution.
  + P2 arrives at time 1 but waits until P1 finishes, as the scheduling is non-preemptive.
  + Once P1 completes, P2, which has the highest priority, starts.

**Advantages:**

* + Simple to implement.
  + Lower overhead due to fewer context switches.

**Disadvantages:**

* + A low-priority process might delay higher-priority processes that arrive later.
  + This can lead to inefficient CPU utilization.

**Comparison Table**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Pre-emptive Priority Scheduling** | **Non-Pre-emptive Priority Scheduling** |
| Interruptions | Yes, higher-priority processes can interrupt. | No, once a process starts, it runs to completion. |
| Response Time | Faster for high-priority processes. | May be delayed if a low-priority process is running. |
| Context Switching | High, due to frequent interruptions. | Low, as processes are not interrupted. |
| Starvation Issue | More severe. | Less severe, but still possible. |
| Use Case | Real-time systems, OS scheduling. | Batch processing, less time-critical systems. |

**Algorithms:**

**Pre-emptive Priority Scheduling Algorithm:**

1. **Input**: List of processes with their arrival times, burst times, and priorities.
2. **Sort**: Sort processes based on arrival time. If multiple processes arrive at the same time, sort them based on their priority.
3. **Process Execution**:
   * The process with the highest priority (smallest priority number) gets the CPU.
   * If a new process arrives with a higher priority than the running process, the running process is preempted, and the new process is allocated the CPU.
   * Continue until all processes are finished.
4. **Repeat** until all processes are executed.

**Non-Pre-emptive Priority Scheduling Algorithm:**

1. **Input**: List of processes with their arrival times, burst times, and priorities.
2. **Sort**: Sort processes based on arrival time. If multiple processes arrive at the same time, sort them based on priority.
3. **Process Execution**:
   * Select the process with the highest priority (smallest priority number) among all the processes that have arrived and are ready to execute.
   * The selected process executes until completion.
   * Once a process finishes, check for the next process with the highest priority and repeat the execution.
4. **Repeat** until all processes are executed.

# Pre-emptive Priority Scheduling

def preemptive\_priority\_scheduling(processes, n):

# Sorting the processes based on Arrival Time first and then by priority

processes.sort(key=lambda x: (x[1], x[3]))

completion\_times = [-1] \* n

waiting\_times = [0] \* n

turnaround\_times = [0] \* n

remaining\_burst\_times = [p[2] for p in processes]

time = 0

completed = 0

while completed < n:

# Find the process with the highest priority (smallest number)

highest\_priority = -1

idx = -1

for i in range(n):

if processes[i][1] <= time and remaining\_burst\_times[i] > 0:

if highest\_priority == -1 or processes[i][3] < processes[highest\_priority][3]:

highest\_priority = i

idx = i

# If no process is found, increment time

if idx == -1:

time += 1

continue

# Execute the process

remaining\_burst\_times[idx] -= 1

time += 1

# If the process finishes, update completion time

if remaining\_burst\_times[idx] == 0:

completion\_times[idx] = time

completed += 1

# Calculating Waiting Time and Turnaround Time

for i in range(n):

turnaround\_times[i] = completion\_times[i] - processes[i][1]

waiting\_times[i] = turnaround\_times[i] - processes[i][2]

return processes, completion\_times, waiting\_times, turnaround\_times

# Main function to get inputs and execute Pre-emptive Priority Scheduling

def main():

# Take input from user

print("Jal Bafana - K005")

n = int(input("Enter the number of processes: "))

processes = []

for i in range(n):

name = input(f"Enter name of process {i + 1}: ")

arrival\_time = int(input(f"Enter arrival time of process {name}: "))

burst\_time = int(input(f"Enter burst time of process {name}: "))

priority = int(input(f"Enter priority of process {name} (lower number means higher priority): "))

processes.append([name, arrival\_time, burst\_time, priority])

# Run Pre-emptive Priority Scheduling

print("\nPre-emptive Priority Scheduling Results:")

processes\_copy = [process[:] for process in processes] # Copy for Pre-emptive scheduling

processed\_data = preemptive\_priority\_scheduling(processes\_copy, n)

processes, completion\_times, waiting\_times, turnaround\_times = processed\_data

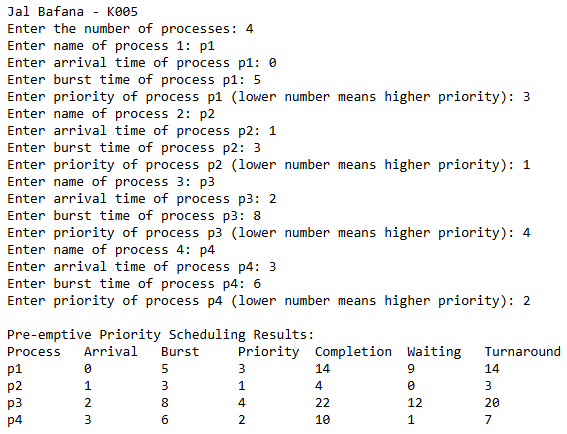
print(f"{'Process':<10}{'Arrival':<10}{'Burst':<10}{'Priority':<10}{'Completion':<12}{'Waiting':<10}{'Turnaround':<10}")

for i in range(n):

print(f"{processes[i][0]:<10}{processes[i][1]:<10}{processes[i][2]:<10}{processes[i][3]:<10}{completion\_times[i]:<12}{waiting\_times[i]:<10}{turnaround\_times[i]:<10}")

if \_\_name\_\_ == "\_\_main\_\_":

main()



# Non-Pre-emptive Priority Scheduling

def non\_preemptive\_priority\_scheduling(processes, n):

processes.sort(key=lambda x: (x[1], x[3])) # Sort by Arrival Time and then Priority

completion\_times = [-1] \* n

waiting\_times = [0] \* n

turnaround\_times = [0] \* n

remaining\_burst\_times = [p[2] for p in processes]

time = 0

completed = 0

while completed < n:

idx = -1

for i in range(n):

if processes[i][1] <= time and remaining\_burst\_times[i] > 0:

if idx == -1 or processes[i][3] < processes[idx][3]:

idx = i

if idx == -1: # No process ready to execute, increase time

time += 1

continue

remaining\_burst\_times[idx] -= 1

time += 1

if remaining\_burst\_times[idx] == 0:

completion\_times[idx] = time

completed += 1

for i in range(n):

turnaround\_times[i] = completion\_times[i] - processes[i][1]

waiting\_times[i] = turnaround\_times[i] - processes[i][2]

return processes, completion\_times, waiting\_times, turnaround\_times

# Main function to get inputs and execute Non-Pre-emptive Priority Scheduling

def main():

print("Jal Bafana - K005")

# Take input from user

n = int(input("Enter the number of processes: "))

processes = []

for i in range(n):

name = input(f"Enter name of process {i + 1}: ")

arrival\_time = int(input(f"Enter arrival time of process {name}: "))

burst\_time = int(input(f"Enter burst time of process {name}: "))

priority = int(input(f"Enter priority of process {name} (lower number means higher priority): "))

processes.append([name, arrival\_time, burst\_time, priority])

# Run Non-Pre-emptive Priority Scheduling

print("\nNon-Pre-emptive Priority Scheduling Results:")

processes\_copy = [process[:] for process in processes] # Copy for Non-Pre-emptive scheduling

processed\_data = non\_preemptive\_priority\_scheduling(processes\_copy, n)

processes, completion\_times, waiting\_times, turnaround\_times = processed\_data

print(f"{'Process':<10}{'Arrival':<10}{'Burst':<10}{'Priority':<10}{'Completion':<12}{'Waiting':<10}{'Turnaround':<10}")

for i in range(n):

print(f"{processes[i][0]:<10}{processes[i][1]:<10}{processes[i][2]:<10}{processes[i][3]:<10}{completion\_times[i]:<12}{waiting\_times[i]:<10}{turnaround\_times[i]:<10}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

