CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Chandubhai S Patel Institute of Technology

Faculty of Technology and Engineering

Chandubhai S. Patel Institute of Technology

Department of Computer Science & Engineering

Date: 22/02/25

Practical 6

Academic Year	:	2023-24	Semester	:	4th
Course code	:	CSE208	Course name	:	Operating System

Perform Linux Commands for the following

Practical 6: Implementing CPU Scheduling Algorithms

You are a software developer working on a simulation project for an operating systems course. Your task is to demonstrate and compare the performance of two CPU scheduling algorithms:

- Shortest Remaining Time First (SRTF)
- Round Robin (RR)

Project Requirements:
Input: Simulate scheduling for 6 processes, each with specific arrival times and burst times.
Output: For each algorithm:

- A Gantt chart visualizing the execution sequence of processes.
- A table showing the following metrics for each process:
- Completion Time (CT): When the process finishes execution.
- Turnaround Time (TAT): Total time taken by the process from arrival to completion.
- Waiting Time (WT): Time the process spends waiting in the ready queue.
- Response Time (RT): Time from arrival to the first response by the CPU.

```
23cs070@67da6ba712a7dc36e53df5c5:~$ pip install tabulate
23cs070@67da6ba712a7dc36e53df5c5:~$ nano cpu_scheduling.py
```

Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~

File Edit View Terminal Tabs Help

GNU nano 6.2 cpu_scheduling.py *

```
import numpy as np
from tabulate import tabulate

# Step 1: Define a Process Class
class Process:
    def __init__(self, pid, arrival, burst):
        self.pid = pid
        self.arrival = arrival
        self.burst = burst
        self.remaining = burst
        self.completion = 0
        self.turnaround = 0
        self.waiting = 0
        self.response = -1

# Step 2: Implement Shortest Remaining Time First (SRTF) Algorithm
def srtf_scheduling(processes):
    n = len(processes)
    time = 0
    completed = 0
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~

File Edit View Terminal Tabs Help

GNU nano 6.2 cpu_scheduling.py *

```
    completed = 0
    gantt = []
    while completed < n:
        available = [p for p in processes if p.arrival <= time and p.remaining >
        if not available:
            time += 1
            continue
        available.sort(key=lambda p: (p.remaining, p.arrival))
        current = available[0]
        if current.response == -1:
            current.response = time - current.arrival
        gantt.append(current.pid)
        current.remaining -= 1
        time += 1
        if current.remaining == 0:
            current.completion = time
            current.turnaround = current.completion - current.arrival
            current.waiting = current.turnaround - current.burst
            completed += 1
    return gantt
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~

File Edit View Terminal Tabs Help

GNU nano 6.2 cpu_scheduling.py *

```
# Step 3: Implement Round Robin (RR) Scheduling
def round_robin_scheduling(processes, quantum):
    time = 0
    gantt = []
    processes.sort(key=lambda p: p.arrival)
    for p in processes:
        p.remaining = p.burst

    i = 0
    while any(p.remaining > 0 for p in processes):
        if i >= len(processes):
            time += 1
            i = 0
            continue
        p = processes[i]
        if p.remaining > 0 and p.arrival <= time:
            if p.response == -1:
                p.response = time - p.arrival
            execute_time = min(p.remaining, quantum)
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~

File Edit View Terminal Tabs Help

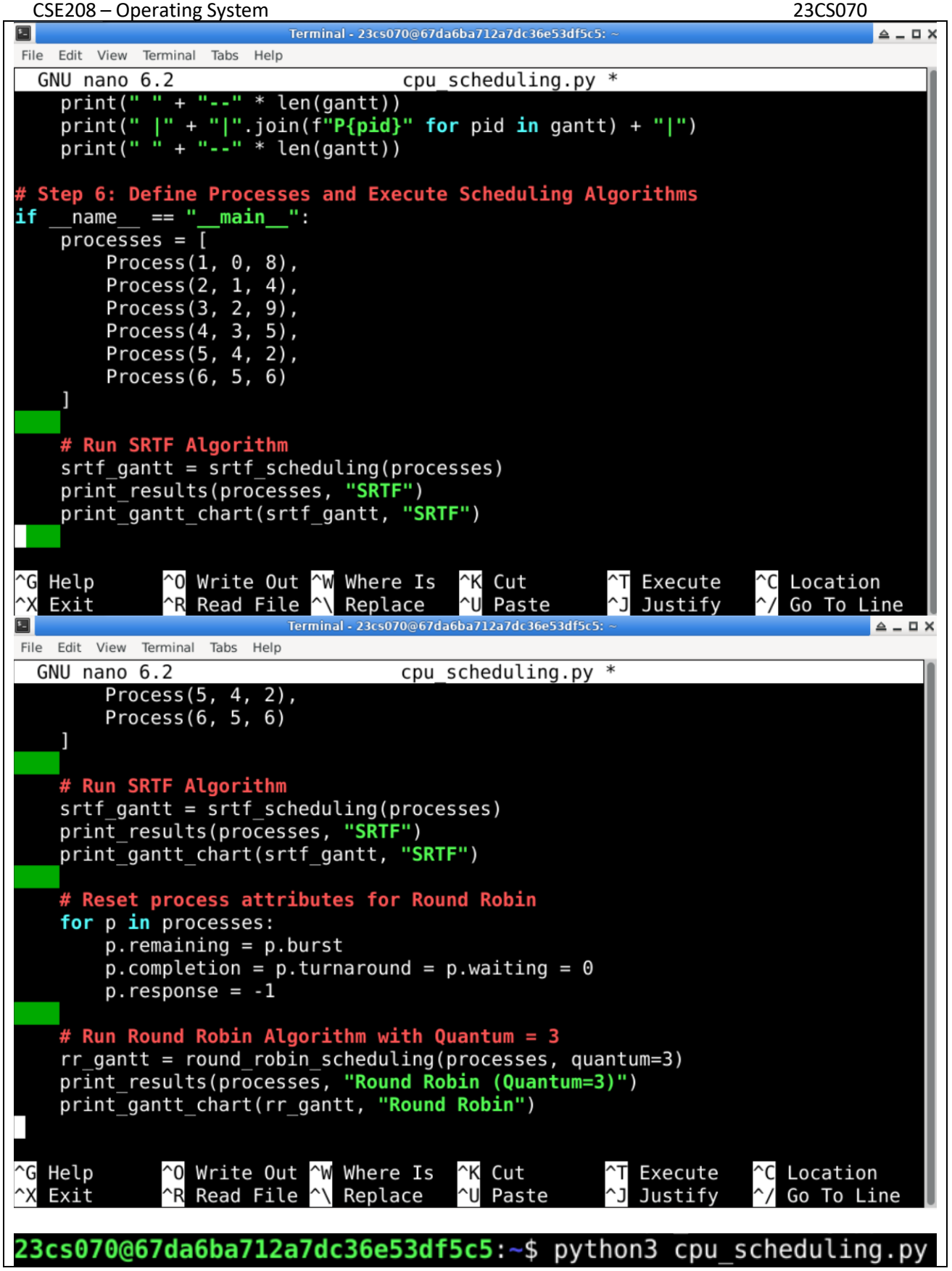
GNU nano 6.2 cpu_scheduling.py *

```
            gantt.extend([p.pid] * execute_time)
            time += execute_time
            p.remaining -= execute_time
            if p.remaining == 0:
                p.completion = time
                p.turnaround = p.completion - p.arrival
                p.waiting = p.turnaround - p.burst
        i += 1
    return gantt

# Step 4: Print Results in Table Format
def print_results(processes, algo):
    headers = ["PID", "Arrival", "Burst", "Completion", "TAT", "Waiting", "Resp>
    data = [[p.pid, p.arrival, p.burst, p.completion, p.turnaround, p.waiting, >
    print(f"\n{algo} Scheduling Results:")
    print(tabulate(data, headers, tablefmt="grid"))

# Step 5: Print Gantt Chart
def print_gantt_chart(gantt, title):
    print(f"\n{title} Gantt Chart:")
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line


Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~
File Edit View Terminal Tabs Help
GNU nano 6.2 cpu_scheduling.py *
print(" " + "--" * len(gantt))
print(" |" + "|".join(f"P{pid}" for pid in gantt) + "|")
print(" " + "--" * len(gantt))
# Step 6: Define Processes and Execute Scheduling Algorithms
if __name__ == "__main__":
processes = [
Process(1, 0, 8),
Process(2, 1, 4),
Process(3, 2, 9),
Process(4, 3, 5),
Process(5, 4, 2),
Process(6, 5, 6)
]
# Run SRTF Algorithm
srtf_gantt = srtf_scheduling(processes)
print_results(processes, "SRTF")
print_gantt_chart(srtf_gantt, "SRTF")
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
Terminal - 23cs070@67da6ba712a7dc36e53df5c5: ~
File Edit View Terminal Tabs Help
GNU nano 6.2 cpu_scheduling.py *
Process(5, 4, 2),
Process(6, 5, 6)
]
# Run SRTF Algorithm
srtf_gantt = srtf_scheduling(processes)
print_results(processes, "SRTF")
print_gantt_chart(srtf_gantt, "SRTF")
# Reset process attributes for Round Robin
for p in processes:
p.remaining = p.burst
p.completion = p.turnaround = p.waiting = 0
p.response = -1
# Run Round Robin Algorithm with Quantum = 3
rr_gantt = round_robin_scheduling(processes, quantum=3)
print_results(processes, "Round Robin (Quantum=3)")
print_gantt_chart(rr_gantt, "Round Robin")
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
23cs070@67da6ba712a7dc36e53df5c5:~$ python3 cpu_scheduling.py

```
23cs070@67da6ba712a7dc36e53df5c5:~$ python3 cpu_scheduling.py

SRTF Scheduling Results:
+-------+-----------+---------+--------------+-------+-----------+------------+
|   PID |   Arrival |   Burst |   Completion |   TAT |   Waiting |   Response |
+=======+===========+=========+==============+=======+===========+============+
|     1 |         0 |       8 |           25 |    25 |        17 |          0 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     2 |         1 |       4 |            5 |     4 |         0 |          0 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     3 |         2 |       9 |           34 |    32 |        23 |         23 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     4 |         3 |       5 |           12 |     9 |         4 |          4 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     5 |         4 |       2 |            7 |     3 |         1 |          1 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     6 |         5 |       6 |           18 |    13 |         7 |          7 |
+-------+-----------+---------+--------------+-------+-----------+------------+

SRTF Gantt Chart:
 ------------------------------------------------------------------------
 |P1|P2|P2|P2|P2|P5|P5|P4|P4|P4|P4|P4|P6|P6|P6|P6|P6|P6|P1|P1|P1|P1|P1|P1|P1|P3|
P3|P3|P3|P3|P3|P3|P3|P3|
 ------------------------------------------------------------------------
Round Robin (Quantum=3) Scheduling Results:
+-------+-----------+---------+--------------+-------+-----------+------------+
|   PID |   Arrival |   Burst |   Completion |   TAT |   Waiting |   Response |
+=======+===========+=========+==============+=======+===========+============+
|     1 |         0 |       8 |           33 |    33 |        25 |          0 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     2 |         1 |       4 |           22 |    21 |        17 |          2 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     3 |         2 |       9 |           36 |    34 |        25 |          4 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     4 |         3 |       5 |           27 |    24 |        19 |          6 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     5 |         4 |       2 |           14 |    10 |         8 |          8 |
+-------+-----------+---------+--------------+-------+-----------+------------+
|     6 |         5 |       6 |           30 |    25 |        19 |          9 |
+-------+-----------+---------+--------------+-------+-----------+------------+

Round Robin Gantt Chart:
 ------------------------------------------------------------------------
 |P1|P1|P1|P2|P2|P2|P3|P3|P3|P4|P4|P4|P5|P5|P6|P6|P6|P1|P1|P1|P2|P3|P3|P3|P4|P4|
P6|P6|P6|P1|P1|P3|P3|P3|
 ------------------------------------------------------------------------
```