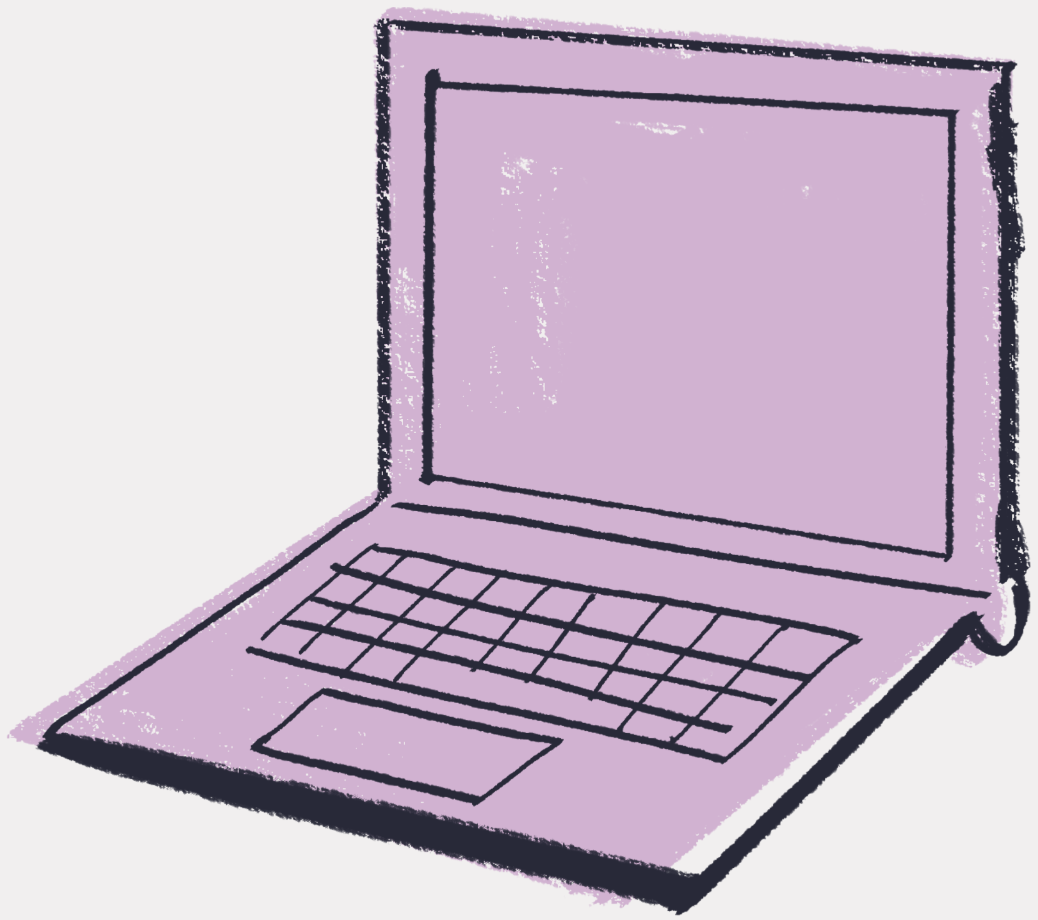
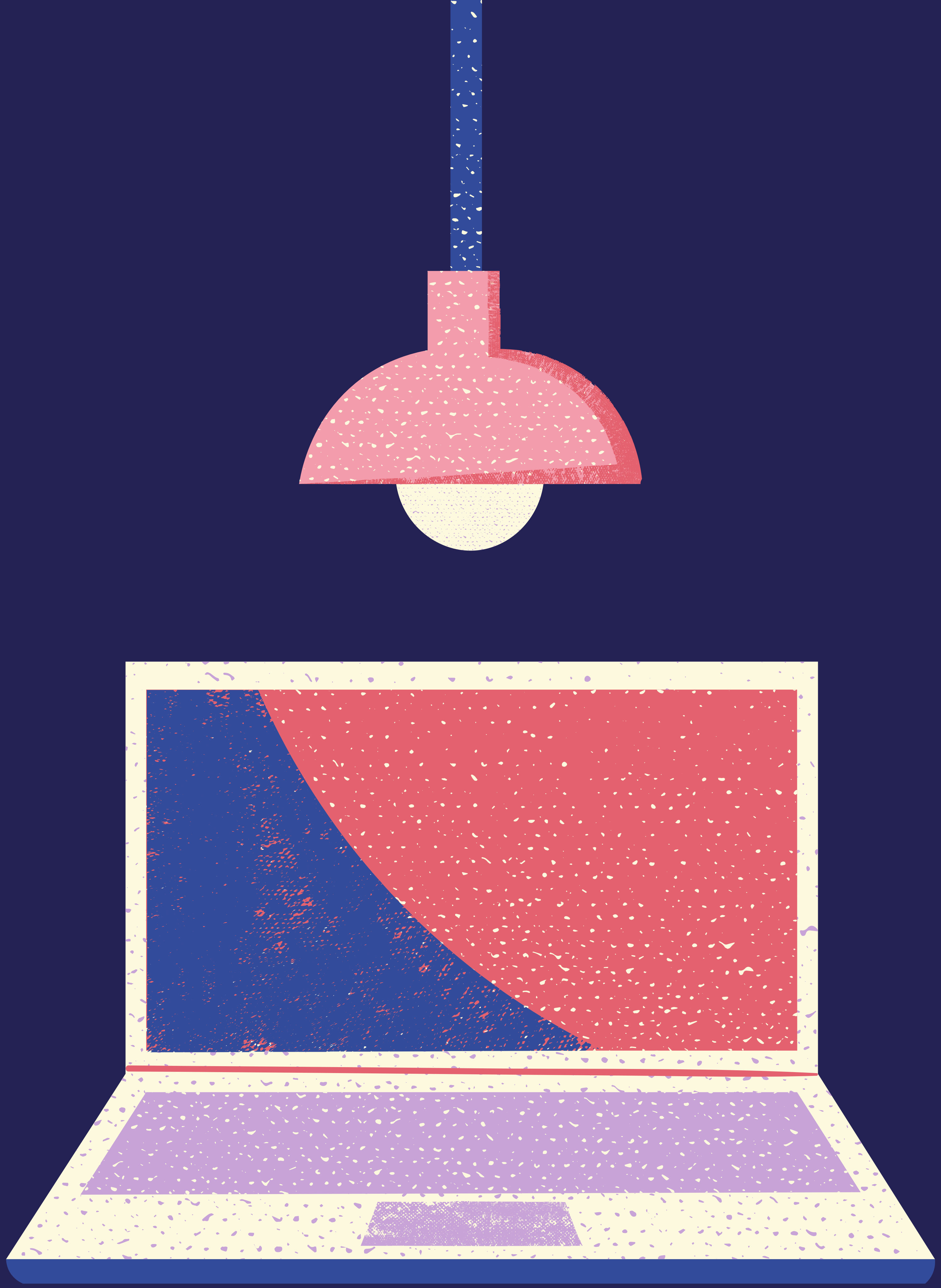
Multi-Level Feedback Queue CPU Scheduling



Tanmayee Inaganti AP22110011499

Raya Srujana Reddy AP22110011502

Darsi Tejaswi AP22110011503



Outline of the Presentation

- Introduction to CPU Scheduling
- Problem Statement
- Methodology: Algorithm
- Results
- Conclusion
- References

Problem Statement 

- Efficient CPU scheduling is essential for maximizing system performance in multitasking operating systems. Traditional algorithms such as FCFS, SJF, and Round Robin often face limitations in balancing fairness, efficiency, and responsiveness.
- The Multilevel Feedback Queue (MLFQ) Scheduling Algorithm is designed to overcome these challenges by dynamically adjusting process priorities based on execution behavior.

Objective:

Design and implement a CPU scheduling algorithm using the Multilevel Feedback Queue (MLFQ) approach to optimize process scheduling by reducing waiting and turnaround times while ensuring fairness and preventing starvation in multitasking systems.

Methodology : Algorithm

1.Initialization:

Create a process list with attributes: Process ID, Arrival Time (AT), Burst Time (BT), and initialize CT, TAT, and WT to 0.

Create three empty queues: q0, q1, and q2 for multi-level scheduling.

Set simulation current time c to the earliest arrival time (a[0]).

2.Start Scheduling: Repeat until all processes are completed (check() returns true):

- **Add New Arrivals to Queue 1:** Move all processes whose AT ≤ c from the process list to q0.
- **Serve Queue 1 (Time Quantum = t0):** Dequeue a process from q0.

If BT > t0:

Reduce BT by t0, increment c by t0, and move the process to q1.

Otherwise: Mark the process as complete, set CT = c + BT, and increment c.

- **Serve Queue 2 (Time Quantum = t1):**

If q0 is empty, dequeue a process from q1.

If BT > t1:

Reduce BT by t1, increment c by t1, and move the process to q2.

Otherwise: Mark the process as complete, set CT = c + BT, and increment c.

Methodology : Algorithm

- **Serve Queue 3 (No Time Quantum):** If q0 and q1 are empty, dequeue a process from q2. Mark the process as complete, set CT = c + BT, and increment c.
- **Handle Idle Time:** If all queues are empty and no new processes have arrived, increment c.

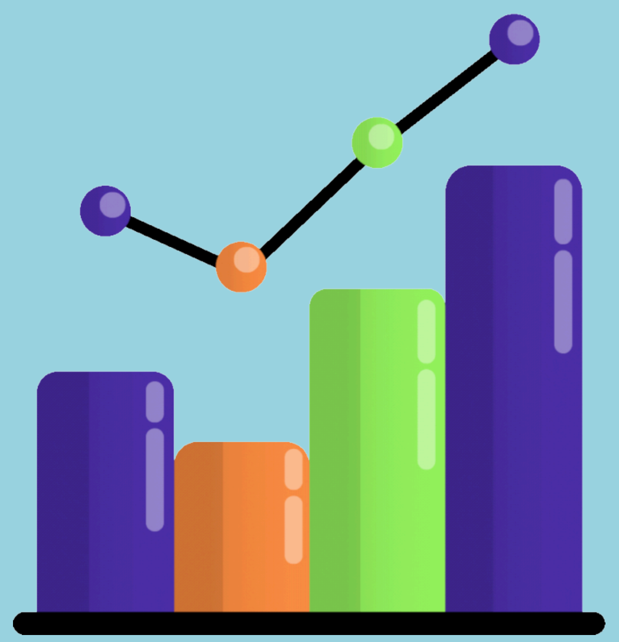
3. **Record Execution:**
   Track process execution in a Gantt Chart.

4. **Calculate Metrics:**
   For each process:
   TAT = CT - AT.
   WT = TAT - BT

5. **Display Results:**
   Print process details (AT, BT, CT, TAT, WT).

6. **Compute and display:**
   Average TAT.
   Average WT.
   Display the Gantt Chart.

Results



Multilevel Feedback Queue Scheduling outperforms traditional algorithms:

- **Turnaround Time:** Significantly lower than FCFS and Round Robin due to dynamic adjustments.
- **Waiting Time:** Comparable to FCFS and Round Robin, MLFQ is lower.

Execution Highlights:

- Processes dynamically moved between queues based on remaining burst time.
- Gantt Chart demonstrates smooth execution transitions across queues.

Key Observation:

- MLFQ combines the strengths of FCFS and Round Robin, ensuring efficiency.

MLFQ

```
Enter no.of processes:
8
enter Arrival times for 8processes  in ascending order:
0 2 3 6 7 8 10 11
enter Burst times for 8processes :
40 30 25 17 60 10 2 14

Enter time quantum for queue 1
15
Enter time quantum for queue 2
30
Move Process p1 from queue-1 to queue-2
Queue 1: EMPTY
Queue 2: P0
Queue 3: EMPTY

Move Process p2 from queue-1 to queue-2
Queue 1: P2 P3 P4 P5 P6 P7
Queue 2: P0 P1
Queue 3: EMPTY

Move Process p3 from queue-1 to queue-2
Queue 1: P3 P4 P5 P6 P7
Queue 2: P0 P1 P2
Queue 3: EMPTY

Move Process p4 from queue-1 to queue-2
Queue 1: P4 P5 P6 P7
Queue 2: P0 P1 P2 P3
Queue 3: EMPTY

Move Process p5 from queue-1 to queue-2
Queue 1: P5 P6 P7
Queue 2: P0 P1 P2 P3 P4
Queue 3: EMPTY

Process P6 completed its execution
Queue 1: P6 P7
Queue 2: P0 P1 P2 P3 P4
Queue 3: EMPTY

Process P7 completed its execution
Queue 1: P7
Queue 2: P0 P1 P2 P3 P4
```

```
Gantt Chart:
--------------------------------------------------------------------------------------------------
|  P1  |  P2  |  P3  |  P4  |  P5  |  P6  |  P7  |  P8  |  P1  |  P2  |  P3  |  P4  |  P5  |  P5  |
--------------------------------------------------------------------------------------------------
0     15     30     45     60     75     85     87     101     126     141     151     153     183     198

P-No    AT      BT      CT      TAT     WT
P0:     0       40      126     126     86
P1:     2       30      141     139     109
P2:     3       25      151     148     123
P3:     6       17      153     147     130
P4:     7       60      198     191     131
P5:     8       10      85      77      67
P6:     10      2       87      77      75
P7:     11      14      101     90      76

Average Turnaround time: 124.375
Average waiting time: 99.625

--------------------------------
Process exited after 79.71 seconds with return value 0
Press any key to continue . . .
```

FCFS

Input

Algorithm

First Come First Serve, FCFS

Arrival Times

0 2 3 6 7 8 10 11

Burst Times

40 30 25 17 60 10 2 14

Solve

Output

FCFS

Gantt Chart

A	B	C	D	E	F	G	H

0 40 70 95 112 172 182 184 198

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
A	0	40	40	40	0
B	2	30	70	68	38
C	3	25	95	92	67
D	6	17	112	106	89
E	7	60	172	165	105
F	8	10	182	174	164
G	10	2	184	174	172
H	11	14	198	187	173
			Average	1006 / 8 = 125.75	808 / 8 = 101

Round Robin

Input

Algorithm

Round-Robin, RR

Arrival Times

0 2 3 6 7 8 10 11

Burst Times

40 30 25 17 60 10 2 14

Time Quantum

30

Solve

Output

RR

Gantt Chart

A	B	C	D	E	F	G	H	A	E

0 30 60 85 102 132 142 144 158 168 198

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
A	0	40	168	168	128
B	2	30	60	58	28
C	3	25	85	82	57
D	6	17	102	96	79
E	7	60	198	191	131
F	8	10	142	134	124
G	10	2	144	134	132
H	11	14	158	147	133
			Average	1010 / 8 = 126.25	812 / 8 = 101.5

Conclusion

- **Multilevel Feedback Queue (MLFQ) Scheduling Algorithm** is a powerful approach for CPU scheduling in modern operating systems.
- **Advantages over Traditional Algorithms:**
  - **Fairness:** Dynamic priority adjustments prevent starvation and ensure process equality.
  - **Efficiency:** Achieves lower average waiting and turnaround times compared to FCFS and Round Robin.
  - **Scalability:** Handles diverse workloads effectively, making it suitable for multitasking systems.
- **Significance in OS Development:**
  - MLFQ ensures optimal CPU resource allocation, contributing to smoother OS performance.
  - The algorithm demonstrates practical utility in balancing user and system-level tasks.
- **Final Thought:**
  - MLFQ represents a robust scheduling paradigm for current and next-generation operating systems, blending fairness, adaptability, and efficiency.

References

Geeks for Geeks : Multilevel Feedback Queue Scheduling Algorithm. Accessed for theoretical insights and implementation techniques.

Naukri 360: Multilevel Feedback Queue in Operating Systems. Referenced for real-world scheduling examples and practical use cases

YouTube Tutorials:

1. "Multilevel Feedback Queue Scheduling Algorithm (Gantt Chart)": YouTube Link
2. "MLFQ with Example in Operating System": YouTube Link
3. "Multilevel Feedback Queue with Explanation": YouTube Link