CPU Scheduling

CPU scheduling

- CPU scheduling (lập lịch CPU) là một quá trình chọn lựa một tiến trình để cấp CPU, trong số rất nhiều tiến trình đang đợi (trạng thái Ready).
- Mục tiêu của lập lịch CPU nhằm đảo bảo hệ thống hoạt động hiệu quả, công bằng, tối ưu hóa việc sử dụng tài nguyên CPU.
- Các tiến trình đang đợi CPU sẽ được xếp vào hàng đợi Ready, được tổ chức theo kiểu hàng đợi FIFO, priority, tree,…
- Các thuật toán lập lịch CPU có thể thuộc nhóm Preemptive or Non preemptive.

Non Preemptive VS preemptive scheduling

- Lập lịch Non-Preemptive: một khi CPU đã được cấp phát cho một tiến trình, tiến trình sẽ được phép sử dụng CPU cho tới khi nó không cần nữa, gồm:
  - Xử lý xong các công việc: tiến trình sẽ chuyển sang trạng thái Terminated.
  - Đợi I/O: tiến trình sẽ chuyển sang trạng thái Waiting.
- Lập lịch Preemptive: dù CPU đã được cấp phát cho một tiến trình, nó có thể bị bộ lập lịch thu hồi bất kỳ lúc nào, dựa trên các tiêu chí như:
  - Có tiến trình vào hàng đợi Ready với nhu cầu dùng CPU ít hơn: SRTF
  - Có tiến trình vào hàng đợi Ready với độ ưu tiên cao hơn: Priority
  - Tiến trình đã sử dụng CPU hết thời gian quantum được cho phép: Round robin.
  - .....

Mục tiêu của các thuật toán lập lịch CPU

Đối với hệ thống:

- **CPU Utilization (mức sử dụng CPU):** tỉ lệ thời gian mà CPU được sử dụng để thực hiện các tác vụ tính toán. CPU Utilization cao: CPU được sử dụng nhiều; CPU utilization thấp: CPU được sử dụng ít.
- **Throughput (thông lượng):** số lượng công việc/tiến trình được xử lý trong một đơn vị thời gian.

Đối với tiến trình:

- **Turnaround Time(TAT):** còn gọi là thời gian quay vòng, tính từ lúc tiến trình được đưa vào hàng đợi cho đến khi hoàn thành.
- **Waiting Time(WT):** còn gọi là thời gian đợi, là thời gian mà một tiến trình phải chờ đợi trong hàng đợi để được cấp CPU hoặc tài nguyên.
- **Response Time(RT):** còn gọi là thời gian phản hồi, tính từ lúc tiến trình được đưa vào hàng đợi cho đến lần đầu tiên được cấp CPU.

Mục tiêu của các thuật toán lập lịch CPU

- Như vậy, mục tiêu của các thuật toán lập lịch CPU nhằm:
  - **Tối đa hóa:** CPU utilization, thoughput
  - **Tối thiểu hóa:** Turnaround Time(TAT), Waiting Time(WT), Response Time(RT).

Một số lưu ý

- **Arrival Time (AT):** là thời điểm tiến trình vào hàng đợi Ready.
- **Completion Time (CT):** là thời điểm tiến trình hoàn tất việc thực thi.
- **Burst Time (BT):** là khoảng thời gian tiến trình cần sử dụng CPU để thực thi.
- **Turn Around Time (TAT):** là thời gian chênh lệch giữa Completion Time và Arrival Time.

Turn Around Time = Completion Time – Arrival Time

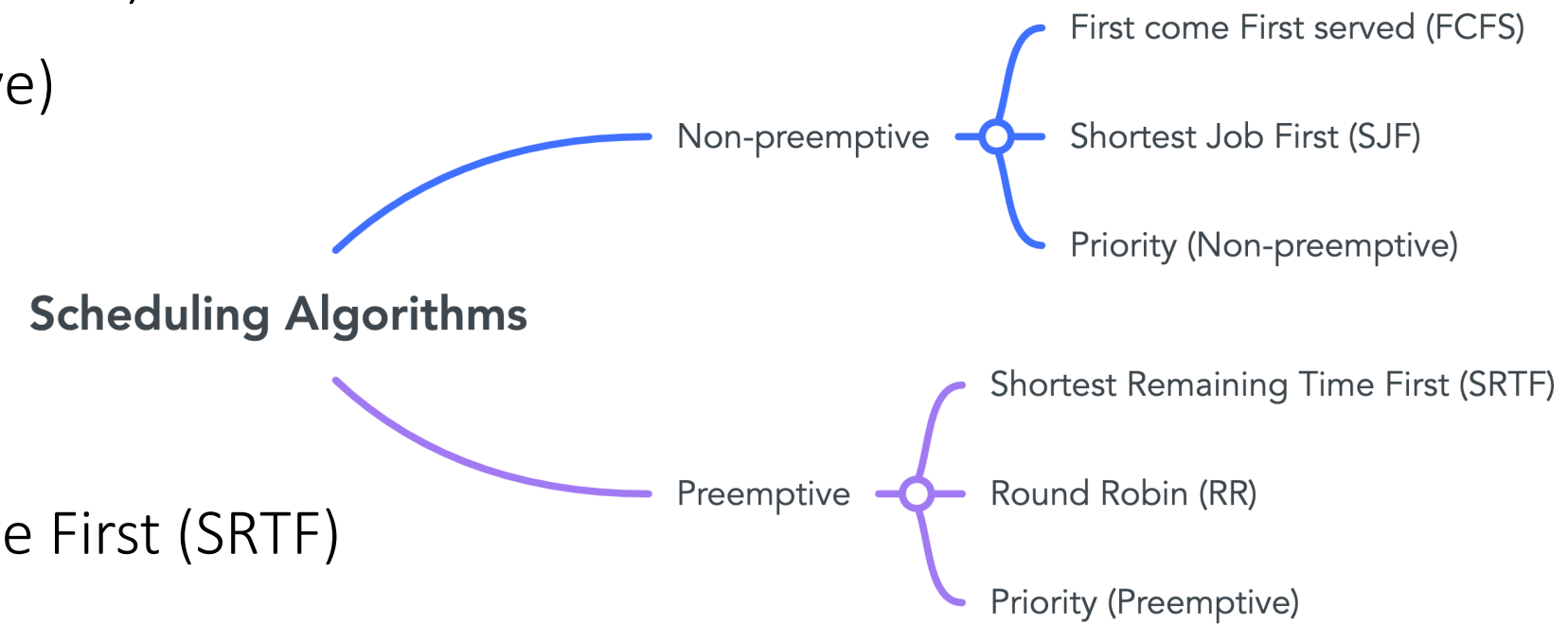
- **Waiting Time (WT):** là thời gian chênh lệch giữa Turn Around Time và Burst Time

Waiting Time = Turn Around Time – Burst Time

Các thuật toán lập lịch CPU

Non-preemptive Algorithm:

- First come First served (FCFS)
- Priority (Non-preemptive)
- Shortest Job First (SJF)



Preemptive Algorithm:

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

Một số loại thuật toán lập lịch khác:

- Multilevel queue scheduling
- Multilevel feedback queue scheduling
- Real time scheduling

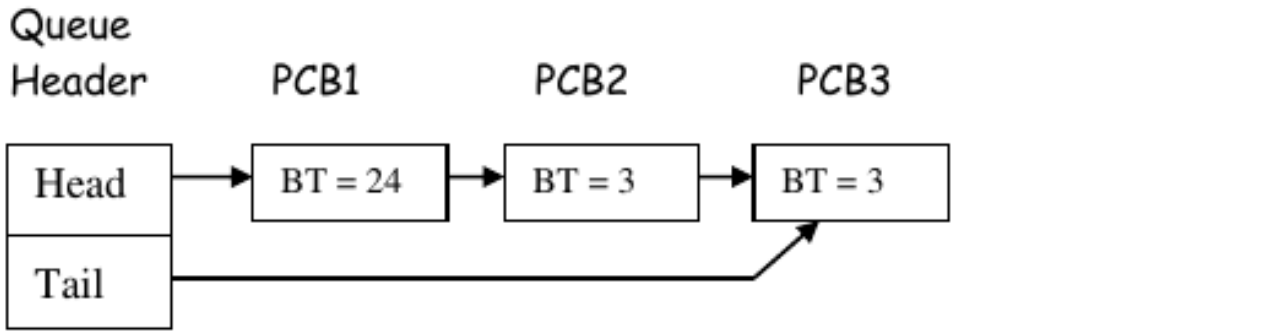
First Come First Served Scheduling (FCFS)

- **Ý tưởng:** tiến trình nào yêu cầu CPU trước thì sẽ được cấp phát CPU trước.
- **Hoạt động:**
  - Các yêu cầu CPU của tiến trình được giữ trong hàng đợi ready theo cơ chế FIFO.
  - Khi một process vào hàng đợi, PCB của nó được liên kết đến cuối hàng đợi.
  - Khi CPU rảnh, hệ điều hành sẽ cấp CPU cho tiến trình nằm ở đầu hàng đợi.

First Come First Served Scheduling (FCFS) – Ex1

- Cho tập các tiến trình như hình dưới, giả sử tất cả các tiến trình đều đến vào thời điểm 0, theo thứ tự P1 →P2→P3 với nhu cầu sử dụng CPU được ghi nhận trong cột CPU Burst Time. Sử dụng giải thuật FCFS để tính: TAT của từng tiến trình, WT trung bình, TAT trung bình.

Process	CPU Burst Time
P1	24
P2	3
P3	3



The Gantt chart is:

First Come First Served Scheduling (FCFS) – Ex1

- TAT = CT – AT and WT = TAT – BT.
- Average TAT= (24 + 27 + 30)/ 3 = 81/3=27ms.
- Average WT = (0 + 24 + 27)/ 3 = 17ms.

Process	Burst Time(BT)	Arrival Time(AT)	Completion Time(CT)	Turn Around Time(TAT)	Waiting Time(WT)
P1	24	0	24	24	0
P2	3	0	27	27	24
P3	3	0	30	30	27

First Come First Served Scheduling (FCFS) – Ex2

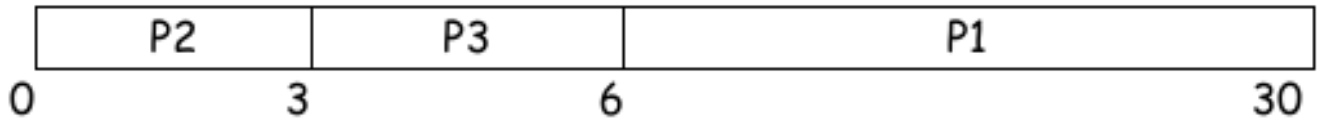
- **Ví dụ 2:** với dữ liệu tương tự ví dụ 1 nhưng các tiến trình đến theo thứ tự P2 → P3 → P1. Hãy vẽ sơ đồ Gantt và tính TAT trung bình, WT trung bình với giải thuật FCFS

Process	CPU Burst Time
P2	3
P3	3
P1	24

First Come First Served Scheduling (FCFS) – Ex2

- Average TAT= (30+3+6)/ 3 = 13ms.
- Average WT = (6+0+3)/ 3 = 3ms.

Solution: The Gantt chart is



Process	Burst Time(BT)	Arrival Time(AT)	Completion Time(CT)	Turn Around Time(TAT)	Waiting Time(WT)
P2	3	0	3	3	0
P3	3	0	6	6	3
P1	24	0	30	30	6

First Come First Served Scheduling (FCFS) – Ex3

- Cho tập hợp các tiến trình như hình dưới (dữ liệu tương tự VD1), thời gian vào hàng đợi của từng tiến trình được liệt kê tại cột Arrival Time. Hãy tính TAT trung bình và WT trung bình với giải thuật FCFS.

Process	CPU Burst Time	Arrival Time
P1	24	0.0
P2	3	0.4
P3	3	1.0

First Come First Served Scheduling (FCFS) – Ex3

- Average TAT= (24 + 26.6 + 29)/ 3 = 26.53ms.
- Average WT = (0 + 23.6 + 26)/ 3 = 16.53ms.

Process	Burst Time(BT)	Arrival Time(AT)	Completion Time(CT)	Turn Around Time(TAT)	Waiting Time(WT)
P1	24	0.0	24	24	0
P2	3	0.4	27	26.6	23.6
P3	3	1.0	30	29	26

Một số nhận xét về First Come First Served Scheduling (FCFS)

- FCFS thuộc nhóm lập lịch non-preemptive.
- Thứ tự đến của các tiến trình đóng một vai trò quan trọng trong việc tính thời gian đợi trung bình.
- Thời gian đợi của tiến trình bị ảnh hưởng bởi thời gian đợi của những tiến trình trước đó.
- Nếu tất cả các tiến trình đến cùng lúc, thời gian hoàn tất (CT) = thời gian quay vòng (TAT).
- Hiệu suất lập lịch kém vì thời gian đợi trung bình cao.
- *Convoy Effect*: tất cả các tiến trình phải đợi một tiến trình có nhu cầu sử dụng CPU lớn trao trả CPU mới có thể hoạt động.

Shortest job first scheduling (SJF)

- **Ý tưởng:** cấp phát CPU cho tiến trình có nhu cầu sử dụng CPU ít nhất (điều kiện là tiến trình đó đã vào hàng đợi Ready).
- **Hoạt động:**
  - Các tiến trình được giữ trong hàng đợi Ready theo thứ tự tăng dần nhu cầu sử dụng CPU (burst time).
  - Khi CPU rảnh, hệ điều hành sẽ cấp CPU cho tiến trình ở đầu hàng đợi (có nhu cầu sử dụng CPU ít nhất).
  - Nếu có hai tiến trình có nhu cầu sử dụng CPU giống nhau, giải thuật FCFS sẽ được áp dụng đối với hai tiến trình đó.

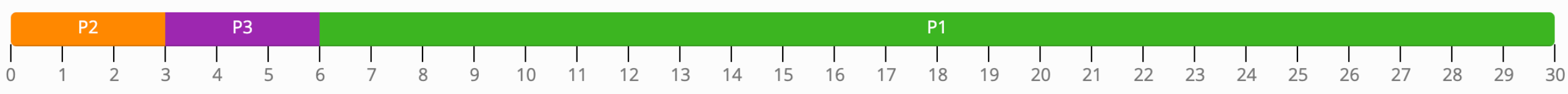
Shortest job first scheduling (SJF) – Ex1

- Cho tập tiến trình như bảng dưới, giả sử tất cả các tiến trình đều đến vào thời điểm 0. Hãy tính TAT trung bình và WT trung bình khi sử dụng giải thuật SJF.

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	24			
P2	0	3			
P3	0	3			
Average					

Shortest job first scheduling (SJF) – Ex1

- Cho tập tiến trình như bảng dưới, giả sử tất cả các tiến trình đều đến vào thời điểm 0. Hãy tính TAT trung bình và WT trung bình khi sử dụng giải thuật SJF.



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	24	30	30	6
P2	0	3	3	3	0
P3	0	3	6	6	3
Average				39 / 3 = 13	9 / 3 = 3

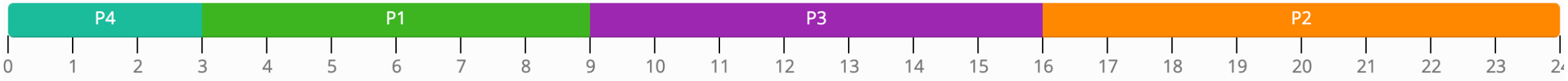
Shortest job first scheduling (SJF) – Ex2

- Cho tập tiến trình như bảng dưới, giả sử tất cả các tiến trình đều đến vào thời điểm 0. Hãy tính TAT trung bình và WT trung bình khi sử dụng giải thuật SJF.

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**6**			
P2	**0**	**8**			
P3	**0**	**7**			
P4	**0**	**3**			
Average					

Shortest job first scheduling (SJF) – Ex2

- *Average waiting time (SJF): (3 + 16 + 9 + 0) / 4 = 7 unit.*
- *Average waiting time (FCFS): 10.25 unit.*



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	6	9	9	3
P2	0	8	24	24	16
P3	0	7	16	16	9
P4	0	3	3	3	0
Average				52 / 4 = 13	28 / 4 = 7

Một số nhận xét về Shortest job first scheduling (SJF)

- SJF có thể thuộc một trong hai loại:
  - Non-Preemptive: giải thuật SJF, CPU không bị thu hồi khi tiến trình vẫn đang cần.
  - Preemptive: giải thuật SRTF, CPU có thể bị thu hồi tại bất kỳ thời điểm nào, tùy thuộc tiêu chí (độ ưu tiên, thời gian dùng CPU,…)
- Nếu một tiến trình mới vào Ready queue và có nhu cầu dùng CPU ít hơn tiến trình đang thực thi thì SJF loại:
  - Non-Preemptive: cho phép tiến trình đang thực thi vẫn tiếp tục sử dụng CPU theo yêu cầu của nó.
  - Preemptive: ngưng tiến trình đang thực thi và giao CPU cho tiến trình mới vào.
- Lập lịch SJF loại preemptive được gọi là Shortest-Remaining-Time-First (SRTF).

Shortest job first scheduling (SJF) – Ex3

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: Preemptive SJF (SRTF) & Non-preemptive SJF

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**8**			
P2	**1**	**4**			
P3	**2**	**9**			
P4	**3**	**5**			
			Average	**?**	**?**

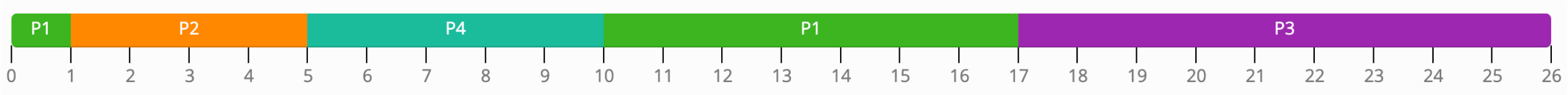
Shortest job first scheduling (SJF) – Ex3

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: Non-preemptive SJF

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	8	8	8	0
P2	1	4	12	11	7
P3	2	9	26	24	15
P4	3	5	17	14	9
			Average	57 / 4 = 14.25	31 / 4 = 7.75

Shortest job first scheduling (SJF) – Ex3

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: Preemptive SJF (SRTF)



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	8	17	17	9
P2	1	4	5	4	0
P3	2	9	26	24	15
P4	3	5	10	7	2
Average				52 / 4 = 13	26 / 4 = 6.5

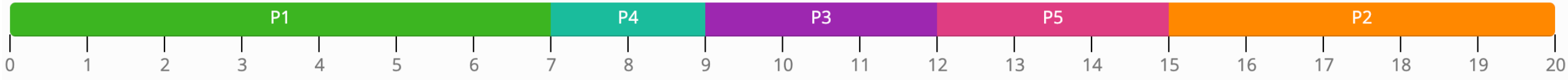
Shortest job first scheduling (SJF) – Ex4

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: Preemptive SJF (SRTF) & Non-preemptive SJF

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**7**			
P2	**1**	**5**			
P3	**2**	**3**			
P4	**6**	**2**			
P5	**12**	**3**			
Average					

Shortest job first scheduling (SJF) – Ex4

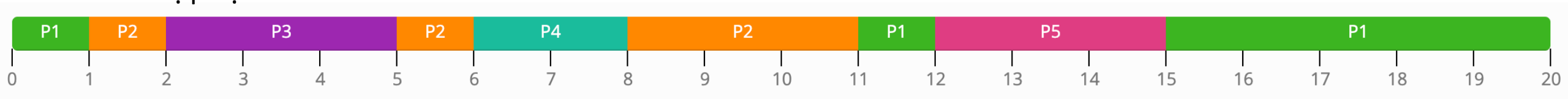
- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: SJF



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	7	7	7	0
P2	1	5	20	19	14
P3	2	3	12	10	7
P4	6	2	9	3	1
P5	12	3	15	3	0
Average				42 / 5 = 8.4	22 / 5 = 4.4

Shortest job first scheduling (SJF) – Ex4

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: SRJF



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	7	20	20	13
P2	1	5	11	10	5
P3	2	3	5	3	0
P4	6	2	8	2	0
P5	12	3	15	3	0
Average				38 / 5 = 7.6	18 / 5 = 3.6

Shortest job first scheduling (SJF) – Ex5

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: Preemptive SJF (SRTF) & Non-preemptive SJF

Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**4**			
P2	**1**	**1**			
P3	**2**	**2**			
P4	**2**	**1**			
			Average		

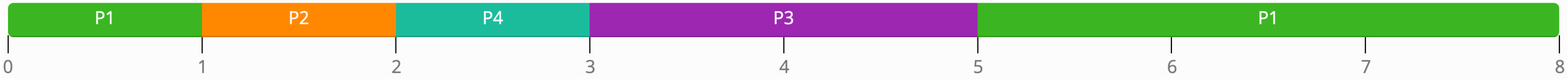
Shortest job first scheduling (SJF) – Ex5

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: SJF

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	4	4	4	0
P2	1	1	5	4	3
P3	2	2	8	6	4
P4	2	1	6	4	3
Average				18 / 4 = 4.5	10 / 4 = 2.5

Shortest job first scheduling (SJF) – Ex5

- Cho bảng dữ liệu dưới đây, hãy tính TAT trung bình và WT trung bình với các thuật toán lập lịch: SRTF



Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	4	8	8	4
P2	1	1	2	1	0
P3	2	2	5	3	1
P4	2	1	3	1	0
			Average	13/4 = 3.25	5/4=1.25

Priority Scheduling (Non Preemptive)

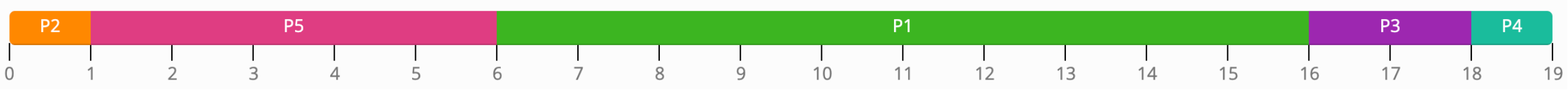
- **Ý tưởng:** mỗi tiến trình có thêm tham số độ ưu tiên, CPU được cấp cho tiến trình có độ ưu tiên cao nhất trong hàng đợi Ready.
- **Hoạt động:**
  - Các tiến trình trong hàng đợi Ready được sắp thứ tự độ ưu tiên từ cao → thấp, bất kể cùng hay khác thời điểm đến.
  - CPU được cấp cho tiến trình có độ ưu tiên cao nhất trong hàng đợi.
  - Nếu có các tiến trình cùng độ ưu tiên, sử dụng giải thuật FCFS đối với các tiến trình đó.

Priority Scheduling (Non Preemptive) – Ex1

- Biết các tiến trình dưới đây đến hàng đợi Ready vào thời điểm 0, theo thứ tự P1 → P2 → P3 → P4 → P5. Tính TAT trung bình và WT trung bình với giải thuật Priority (non preemptive)

Job	Priority	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**3**	**10**			
P2	**1**	**1**			
P3	**4**	**2**			
P4	**5**	**1**			
P5	**2**	**5**			
Average					

Priority Scheduling (Non Preemptive) – Ex1



Job	Priority	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	3	10	16	16	6
P2	1	1	1	1	0
P3	4	2	18	18	16
P4	5	1	19	19	18
P5	2	5	6	6	1
Average				60 / 5 = 12	41 / 5 = 8.2

Priority Scheduling (Non Preemptive) – Ex2

- Cho dữ liệu tiến trình như bảng dưới đây. Hãy tính WT trung bình và TAT trung bình với giải thuật lập lịch Priority (non preemptive)?

Job	Arrival Time	Burst Time	Priority	Turnaround Time	Waiting Time
P1	0	2	4		
P2	0	3	1		
P3	2	3	2		
P4	4	4	3		
			Average	?	?

Priority Scheduling (Non Preemptive) – Ex2

- Cho dữ liệu tiến trình như bảng dưới đây. Hãy tính WT trung bình và TAT trung bình với giải thuật lập lịch Priority (non preemptive)?

Job	Arrival Time	Burst Time	Priority	Turnaround Time	Waiting Time
P1	0	2	4	12	10
P2	0	3	1	3	0
P3	2	3	2	4	1
P4	4	4	3	6	2
Average				25 / 4 = 6.25	13 / 4 = 3.25

Một số nhận xét về Priority Scheduling (Non Preemptive)

- Giải thuật lập lịch Priority có thể thuộc một trong hai nhóm:
  - Preemptive: tiến trình có thể bị thu hồi CPU nếu có một tiến trình khác vào hàng đợi Ready với độ ưu tiên cao hơn.
  - Non-preemptive: tiến trình không bị thu hồi CPU suốt quá trình hoạt động.
- Một vấn đề nghiêm trọng với giải thuật Priority là indefinite blocking (chặn vô thời hạn) hay starvation ("chết đói"). Là hiện tượng tiến trình chờ cấp CPU nhưng liên tục có các tiến trình với độ ưu tiên cao hơn chen ngang → giải pháp là aging.
- Aging là kỹ thuật tăng độ ưu tiên của một tiến trình khi nó chờ đợi quá lâu trong hàng đợi Ready.

Round Robin

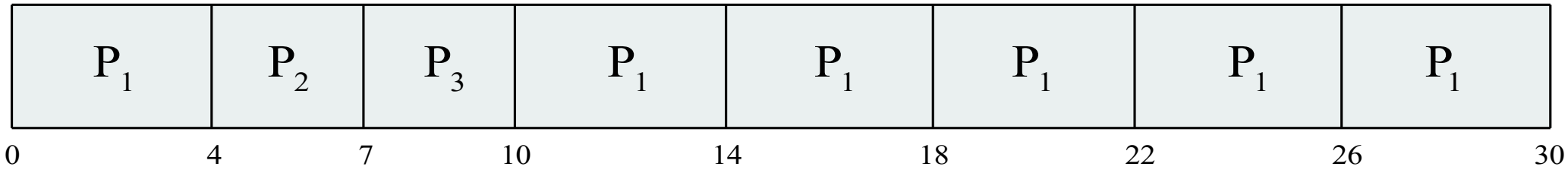
- Mỗi tiến trình được sử dụng CPU trong một đơn vị thời gian nhỏ (được gọi là quantium q), giá trị q thường là 10 → 100 milliseconds. Khi hết quantium, tiến trình sẽ bị thu hồi CPU và vào cuối hàng đợi Ready để đợi lần cấp CPU kế tiếp.
- Bộ lập lịch sẽ ngắt mỗi chu kỳ quantum để giao CPU cho tiến trình kế tiếp trong hàng đợi Ready.
- Đánh giá về hiệu suất
  - q lớn $\Rightarrow$ Round Robin trở thành FIFO
  - q nhỏ $\Rightarrow$ chi phí chuyển ngữ cảnh rất cao → giảm hiệu năng hệ thống.

Round Robin

Process	Burst Time
P_1	24
P_2	3
P_3	3

quantum = 4

- Biểu đồ Gantt:



- Thông thường Round Robin có TAT trung bình > SJF nhưng response time tốt hơn.
- q cần phải lớn hơn so với thời gian chuyển ngữ cảnh (context switch):
  - q thường từ 10 milliseconds ⇒ 100 milliseconds,
  - Context switch < 10 microseconds

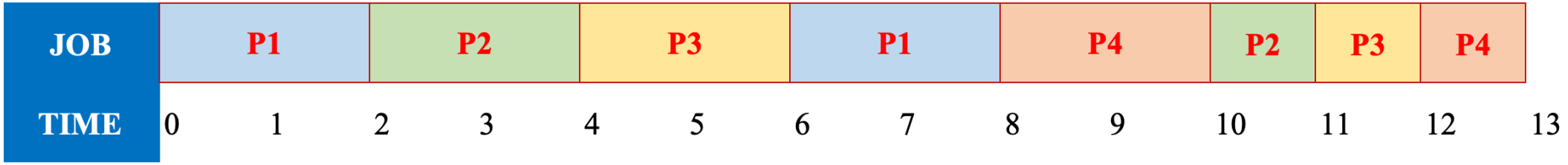
Round Robin – Ex1

- Cho thông tin tiến trình như bảng bên dưới, biết quantium = 2. Hãy xác định TAT trung bình và WT trung bình nếu sử dụng giải thuật Round Robin?

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**4**			
P2	**1**	**3**			
P3	**2**	**3**			
P4	**4**	**3**			
Average					

Round Robin – Ex1

- Cho thông tin tiến trình như bảng bên dưới, biết quantium = 2. Hãy xác định TAT trung bình và WT trung bình nếu sử dụng giải thuật Round Robin?



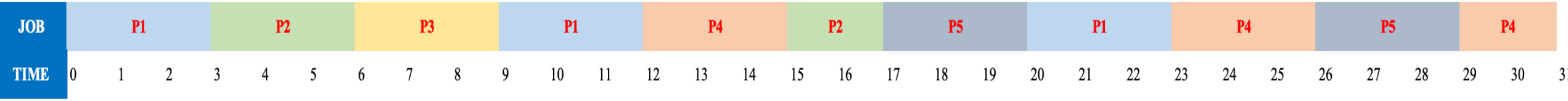
Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	4	8	8	4
P2	1	3	11	10	7
P3	2	3	12	10	7
P4	4	3	13	9	6
Average				37 / 4 = 9.25	24 / 4 = 6

Round Robin – Ex2

- Cho tập các tiến trình như bảng dưới đây, biết quantum = 3. Hãy tính TAT trung bình và WT trung bình khi sử dụng giải thuật Round Robin?

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	**0**	**9**			
P2	**0**	**5**			
P3	**3**	**3**			
P4	**4**	**8**			
P5	**7**	**6**			
			Average		

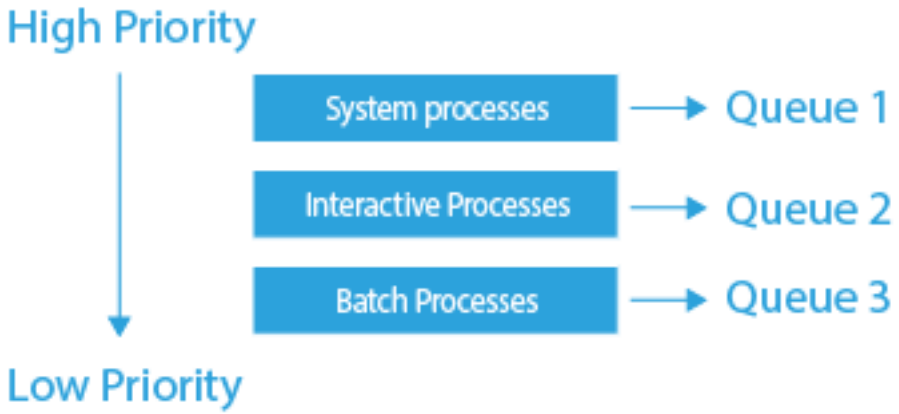
Round Robin – Ex2



Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	9	23	23	14
P2	0	5	17	17	12
P3	3	3	9	6	3
P4	4	8	31	27	19
P5	7	6	29	22	16
Average				95 / 5 = 19	64 / 5 = 12.8

Multilevel Queue Scheduling

- Các tiến trình được chia thành nhiều hàng đợi dựa trên mức độ ưu tiên, mỗi hàng đợi có một mức độ ưu tiên khác nhau. Tiến trình có độ ưu tiên cao được đặt trong hàng đợi có độ ưu tiên cao và ngược lại.
- Mức độ ưu tiên của tiến trình được xác định dựa trên loại, đặc điểm, mức độ quan trọng... Ví dụ:
  - System process
  - Interactive processes
  - Interactive editing processes
  - Batch processes
  - Student processes



Ưu điểm của Multilevel Queue Scheduling

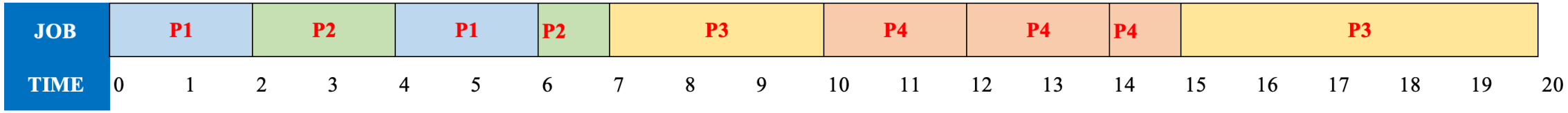
- Sử dụng tài nguyên hiệu quả bằng cách nhóm các tiến trình có nhu cầu tài nguyên tương tự nhau vào các hàng đợi riêng biệt.
- Cải thiện thời gian phản hồi bằng cách đưa các tiến trình cần thời gian phản hồi nhanh vào các hàng đợi có độ ưu tiên cao.
- Thông lượng tốt hơn vì hệ thống có thể thực thi các tiến trình hiệu quả hơn, thực thi nhiều tiến trình đồng thời ở nhiều hàng đợi.
- Linh hoạt hơn bằng cách thay đổi độ ưu tiên của hàng đợi, điều này làm cho hệ thống thích ứng với sự thay đổi nhu cầu tài nguyên.
- Công bằng: ngăn cản hiện tượng có tiến trình giữ CPU quá lâu làm ảnh hưởng đến các tiến trình khác.

Nhược điểm của Multilevel Queue Scheduling

- Phức tạp vì cần duy trì và quản lý nhiều hàng đợi với mức độ ưu tiên khác nhau.
- Tăng chi phí liên quan đến thuật toán lập lịch (trên nhiều hàng đợi).
- Tính linh hoạt: dù lập lịch hàng đợi đa cấp linh hoạt hơn, nhưng có thể không phù hợp với tất cả các loại ứng dụng hoặc nhu cầu công việc.
- Starvation: vẫn có thể xảy ra nếu hàng đợi độ ưu tiên cao có quá nhiều tiến trình so với hàng đợi có độ ưu tiên thấp.

Ví dụ về Multilevel Queue Scheduling

Processes	Arrival Time	CPU burst time	Queue No.
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1



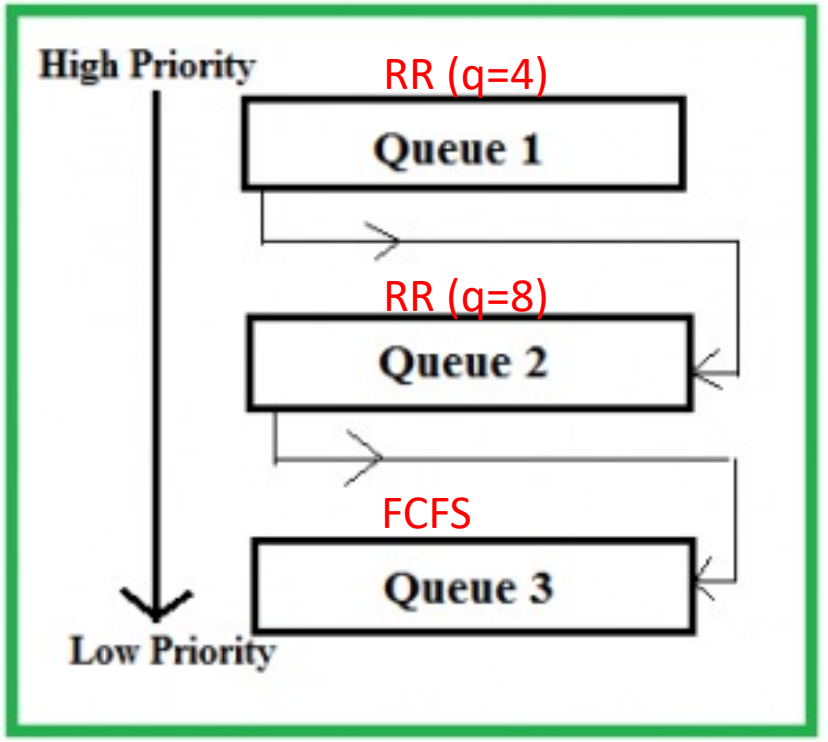
- Queue 1 có độ ưu tiên cao hơn Queue 2 => Round Robin được sử dụng ở Queue 1 (quantum=2), FCFS được sử dụng ở Queue 2.
- Cả hai queue đều được xử lý từ đầu.
  - Tiến trình ở Queues 1 (P1, P2) thực thi trước (vì queue có độ ưu tiên cao hơn) với giải thuật Round Robin và sẽ kết thúc sau 7 đơn vị thời gian.
  - Vì không còn tiến trình trong Q1, các tiến trình trong Q2 bắt đầu thực thi (P3). Nhưng khi P3 đang thực thi thì P4 vào Q1. P4 nằm ở queue có độ ưu tiên cao hơn nên sẽ được ưu tiên thực thi, P3 sẽ bị tạm dừng. Sau khi P4 kết thúc, P3 tiếp tục thực thi.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling

- Lập lịch Multilevel Feedback Queue Scheduling (MLFQ) tương tự lập lịch Multilevel Queue(MLQ) Scheduling nhưng các tiến trình không cố định tại một queue mà có thể di chuyển giữa các queue. Điều này làm cho việc lập lịch trở nên linh hoạt hơn.
- Ưu điểm của MLQF Scheduling:
  - Nhiều hàng đợi.
  - Độ ưu tiên được điều chỉnh linh hoạt.
  - Phân chia thời gian: mỗi hàng đợi được phân chia một khoảng thời gian nhằm xác định thời gian mà CPU được dành riêng cho mỗi hàng đợi trước khi chuyển sang hàng đợi có độ ưu tiên thấp hơn.
  - Cơ chế phản hồi: điều chỉnh độ ưu tiên của tiến trình dựa trên nhu cầu của tiến trình đó theo thời gian.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling

- Khi một tiến trình bắt đầu, OS có thể đặt nó vào bất kỳ hàng đợi nào trong số 3 hàng đợi (hình minh họa) dựa trên độ ưu tiên của nó. Giả sử tiến trình này có độ ưu tiên cao, do đó nó sẽ được đặt vào Queue 1.
- Ở Q1 (Round Robin, q=4), tiến trình thực thi 4 đơn vị thời gian. Nếu nó hoàn tất việc thực thi trong khoảng thời gian quantium thì độ ưu tiên của nó không bị ảnh hưởng. Do đó, lần kế tiếp vào hàng đợi, nó vẫn sẽ được xếp vào Q1.
- Nếu nó không hoàn tất trong khoảng thời gian q thì độ ưu tiên của nó sẽ giảm và được chuyển sang Queue 2 (Round Robin, q=8).
- Điều tương tự sẽ xảy ra ở tất cả các hàng đợi. Nếu một tiến trình không thể hoàn tất trong một khoảng thời gian cho phép, nó sẽ phải di chuyển xuống hàng đợi có độ ưu tiên thấp hơn ở lần kế tiếp.
- Ở hàng đợi cuối cùng, các tiến trình được lập lịch bằng thuật toán FCFS. Điều này nhằm đảm bảo tiến trình được cung cấp đủ CPU mà nó cần.
- Một tiến trình ở hàng đợi thấp chỉ có thể thực thi một khi các hàng đợi có độ ưu tiên cao hơn không còn tiến trình nào.
- Một tiến trình đang thực thi ở hàng đợi thấp có thể bị ngắt bất kỳ lúc nào bởi một tiến trình vào hàng đợi có độ ưu tiên cao hơn.



Nguồn tham khảo

- http://www.codequiz.in/cpu-scheduling/
- https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
- https://www.studytonight.com/operating-system/cpu-scheduling
- https://ess.cs.tu-dortmund.de/Software/AnimOS/CPU-Scheduling/
- https://www.geeksforgeeks.org/multilevel-queue-mlq-cpu-scheduling/
- https://www.geeksforgeeks.org/multilevel-feedback-queue-scheduling-mlfq-cpu-scheduling/
- https://www.geeksforgeeks.org/difference-between-multilevel-queue-mlq-and-multi-level-feedback-queue-mlfq-cpu-scheduling-algorithms/