Algoritmos de Escalonamento

- FCFS (First-Come, First-Served)
- SJF (Shortest Job First)
- Prioridade
- RR (Round Robin)
- Filas Multinível
- Filas Multinível com Realimentação

Escalonamento FCFS

- execução por ordem de chegada na fila de prontos
- caracteristicamente não preemptivo
  - ideal p/ sistemas batch
- implementação simples: fila
- funcionamento
  - processos que se tornam aptos são inseridos no final da fila
  - processo que está no início da fila é o próximo a executar
  - processo executa até que:
    - libere o processador (voluntariamente)
    - realize uma chamada de sistema (bloqueado)
    - termine sua execução

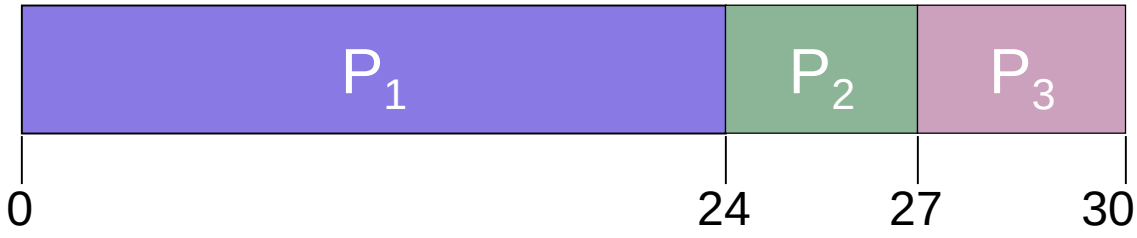
Ex. 1: Escalonamento FCFS (1)

Processo	Ciclo de CPU	chegada em t=0
P_1	24	
P_2	3	
P_3	3	

t de turnaround
P1 = 24
P2 = 27
P3 = 30
} t médio = 27

exemplo 1: ordem de chegada: P_1 P_2 P_3

Diagrama de Gantt



t de espera
P_1 = 0
P_2 = 24
P_3 = 27
} t médio = 17

Ex. 1: Escalonamento FCFS (2)

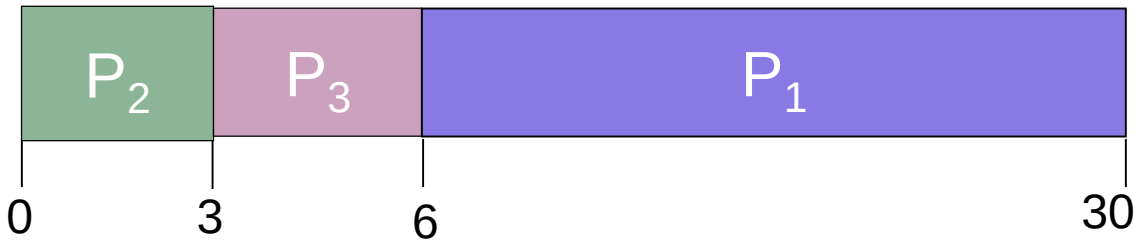
Processo	Ciclo de CPU
P_1	24
P_2	3
P_3	3

chegada em t=0

t de turnaround
P1 = 30
P2 = 3
P3 = 6
} t médio = 13

exemplo 2: ordem de chegada: P_2 P_3 P_1

Diagrama de Gantt



t de espera
P_2 = 0
P_3 = 3
P_1 = 6
} t médio = 3

Melhor !!!

Escalonamento FCFS: características

☺ fácil implementação

☹ execução de processo depende do t de execução dos processos que estão na frente

☹ prejudica processos I/O-bound

☹ tempo médio de resposta é alto

☹ efeito comboio

- todos os outros processos menores aguardam pela execução do processo maior

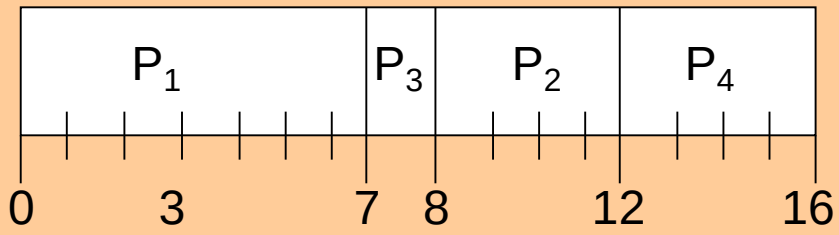
Escalonamento SJF

- “menor” processo primeiro (processo c/ menor ciclo de CPU)
- associa a cada processo o tempo de execução na CPU (normalmente derivado de execuções anteriores)
- ideia: aumentar o *throughput* → tarefas menores primeiro
- desempate: FCFS (em geral)
- 2 abordagens:
  - **não-preemptiva**: processo ativo não pode ser preemptado até completar seu ciclo
  - **preemptiva**: novo processo c/ ciclo de CPU menor que o tempo restante do processo em execução pode preemptá-lo (SRTF - *Shortest-Remaining-Time-First* → *preempção por tempo*)

SJF não-preemptivo

Processo	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (**não-preemptivo**)



Tempo médio de espera

(0 + 6 + 3 + 7)/4 = 4

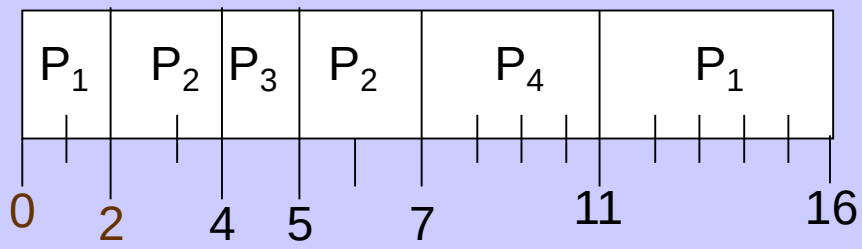
Tempo de turnaround

(7 + 10 + 4 + 11)/4 = 8

SJF preemptivo

Processo	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (**preemptivo**)



Tempo médio de espera

(9 + 1 + 0 +2)/4 = 3

Tempo de turnaround

(16 + 5 +1 + 6)/4 = 7

Escalonamento SJF: características

☺ fornece t médio de espera mínimo na fila de prontos

☺ favorece processos I/O-bound

☹ dificuldade: determinar o tempo do próximo ciclo de CPU de cada processo

- aplicações
  - escalonador de longo prazo (*long term scheduler*)
  - uso: comparação de algoritmos de escalonamento da CPU
  - aproximações do SJF:
    - projeção do valor do próximo ciclo de CPU c/ base no passado
    - obtenção da média exponencial dos tempos de duração dos ciclos de CPU medidos anteriormente

Escalonamento por prioridades

- cada processo possui uma prioridade (valor inteiro)
- CPU é alocada ao processo c/ maior prioridade (em geral, menor valor = maior prioridade)
- desempate: FCFS, SJF, RR, ...
- 2 abordagens:
  - **não-preemptivo:** processo libera espontaneamente o processador
  - **preemptivo:** processo em execução é interrompido caso chegue à fila de prontos um processo de maior prioridade

Ex.: Prioridade não preemptiva

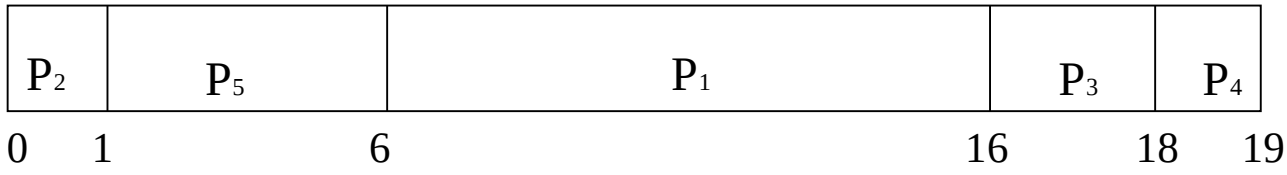
Processo	ciclo CPU	Prioridade
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

t de turnaround

P1 = 16
P2 = 1
P3 = 18
P4 = 19
P5 = 6

t médio = 12

- **Prioridade não-preemptiva**



- tempo médio de espera = (6+0+16+18+1)/5 = 8,2

Como definir a prioridade de um processo?

- **Prioridade estática**
  - processo é criado com uma prioridade, a qual é mantida durante todo seu tempo de vida
- **Prioridade dinâmica**
  - prioridade é ajustada de acordo com o seu comportamento (estado de execução do processo, situação do sistema, ...)
    - Ex.: ajustar a prioridade em função da fatia de tempo realmente utilizada pelo processo

      q = 100 ms

      P1 usou 2 ms → nova prioridade = 1/0.02 = 50

      P2 usou 50 ms → nova prioridade = 1/0.5 = 2

Escalonamento por prioridades: características

☺ pode-se personalizar os processos através de prioridades:

- priorizar processos I/O-bound
- priorizar processos mais importantes (critério ???)

☹ problema → postergação indefinida ("*starvation*")

- processos c/ prioridade baixa podem não executar "nunca"

solução → envelhecimento ("***aging***")

- aumento gradual de prioridade

☹ processo com prioridade estática pode ficar mal classificado e ser penalizado/favorecido em relação aos demais

- Ex.: processos que durante sua execução trocam de padrão de comportamento (CPU-bound a I/O-bound e vice-versa)

solução: múltiplas filas com realimentação

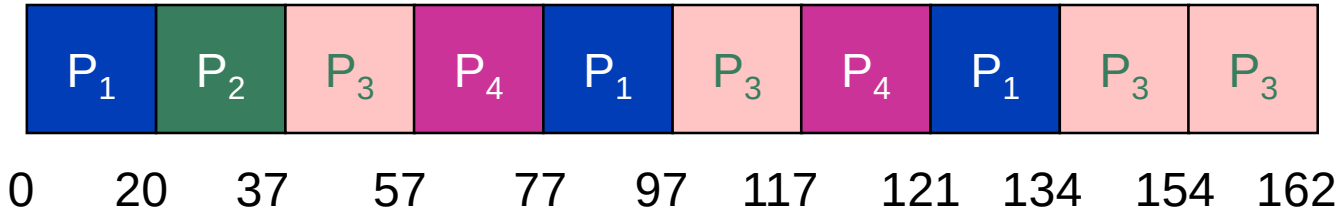
Escalonamento Round Robin

- projetado especialmente p/ sistemas time-sharing
- cada processo recebe uma fatia de tempo da CPU – *quantum* (em geral entre 20 e 500 ms)
  - exige timer (interrupção de tempo)
- fatia de tempo termina → processo é preemptado e movido p/ fim da fila de prontos (ordem FIFO)
- caracteristicamente preemptivo
- fila de prontos é tratada como fila circular
- *scheduler*: acionado a cada término da fatia

Ex.: RR

Processo	Tempo de execução
P_1	53
P_2	17
P_3	68
P_4	24

Diagrama de Gantt (*quantum* = 20 u.t.)



obs.: tipicamente, tem-se tempo de *turnaround* médio maior que em SJF, porém melhor *resposta*

Escalonamento Round Robin: características (1)

- tipicamente, apresenta maior t de *turnaround* que SJF, mas melhor t de resposta
- processo executa menos que fatia se terminar ou pedir E/S
- com ***n*** processos na fila de prontos e fatia=***q***
  - cada processo obtém ***1/n*** do tempo da CPU em fatias de *no máximo* ***q*** u.t. cada
  - nenhum processo espera por mais de ***(n-1) * q*** u.t. para ser atendido

☺ melhor distribuição de t da CPU – algoritmo justo

☺ implementação simples

Escalonamento Round Robin: características (2)

☹ não equilibra o compartilhamento entre diferentes tipos de processos → fila circular trata processos igualmente

- processos I/O-bound são prejudicados: esperam da mesma forma que os CPU-bound, mas muito provavelmente não usam toda a sua fatia

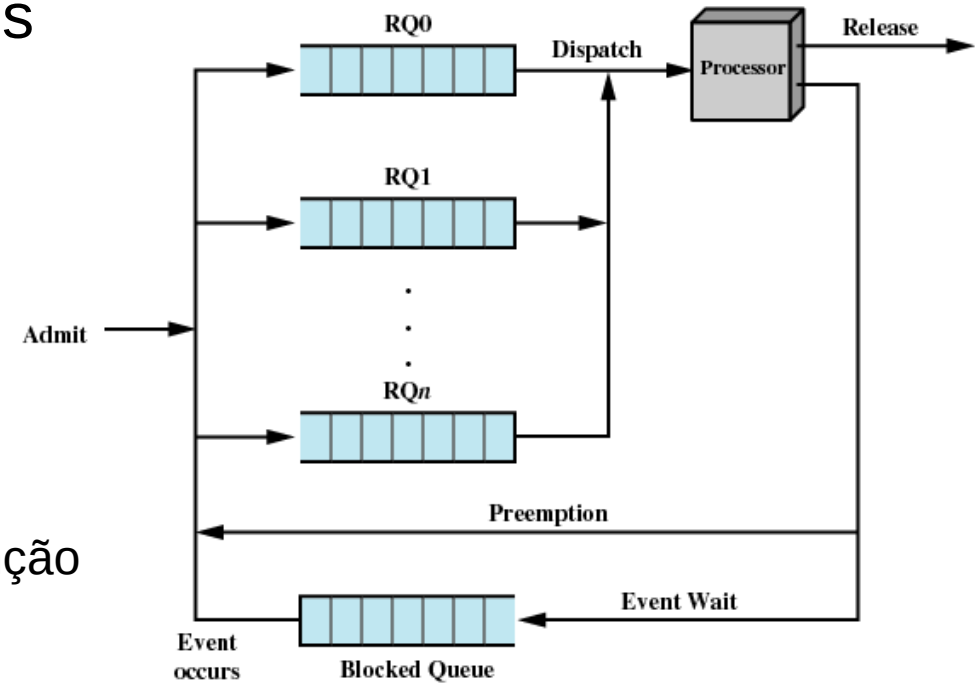
☹ tamanho da fatia de tempo ?? → define desempenho

- fatia grande
  - tende a FCFS
  - aumenta o tempo de resposta dos processos no final da fila de prontos
- fatia pequena
  - tempo de troca de contexto torna-se significativo

☹ tempo da troca de contexto afeta desempenho

Filas multinível

- p/ situações em que processos podem ser classificados em diferentes grupos
- diversas filas de prontos
  - cada processo é associado exclusivamente a uma fila
  - cada fila possui:
    - alg. de escalonamento – em função das características do processo
    - prioridade
  - SO escalona processos de uma fila se todas as outras de maior prioridade estiverem vazias

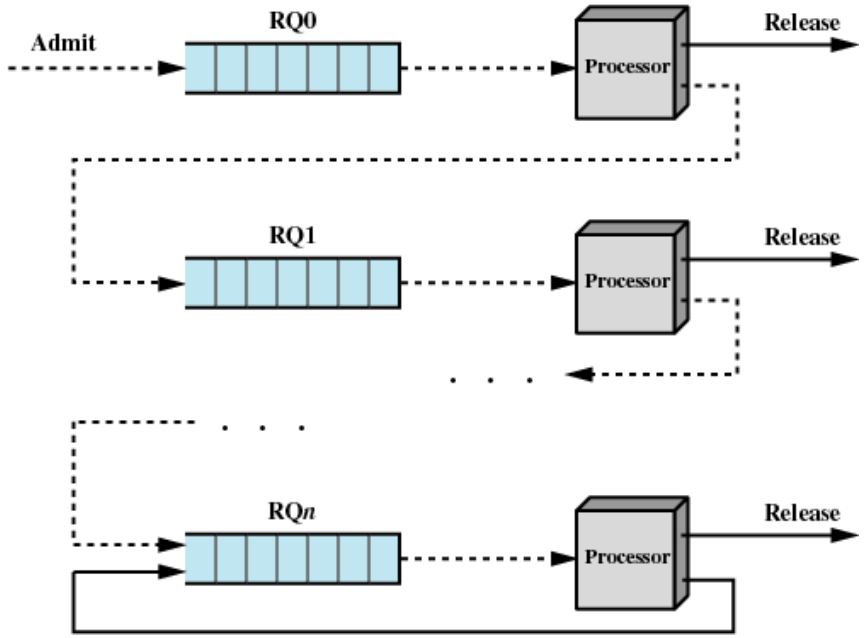


Ex.: Filas multinível

- fila de prontos é particionada:
  - fila foreground (processos interativos)
  - fila background (processos batch)
- cada fila possui um algoritmo:
  - foreground - RR
  - background - FCFS
- escalonamento deve ser feito entre as filas:
  - prioridade fixa: atende as filas em ordem
    - ex.: todos os foreground depois todos os background
  - t da CPU é dividido entre as filas
    - ex.: 80% foreground em RR e 20% background em FCFS

Filas multinível e realimentação

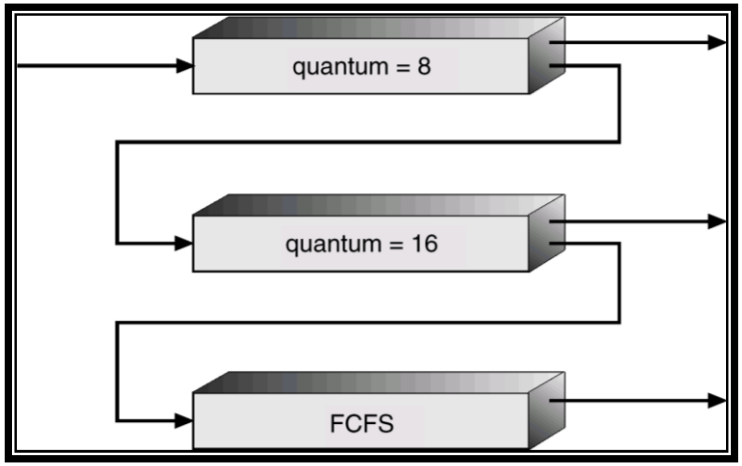
- processos em filas de baixa prioridade tendem a postergação indefinida
  - **<u>solução</u>**: *Filas multinível com realimentação*



Filas multinível com realimentação (feedback)

- processos movem-se de uma fila para outra
- base: prioridade dinâmica
  - em função do t de uso da CPU → prioridade aumenta/diminui
- *aging* → evita postergação indefinida

- Ex.: três filas: Q0, Q1 e Q2
  - novo processo entra em Q0
  - processo ganha CPU e recebe 8 ms
  - se processo não termina em 8 ms, é movido p/ Q1
  - em Q1, processo recebe 16 ms
  - se processo ainda não termina, é movido p/ Q2



Exercício

Cinco processos, de A até E, ficam disponíveis para rodar em lote simultaneamente. O tempo estimado de execução de cada processo é de 10, 6, 2, 4 e 8 minutos, respectivamente. Suas prioridades são 3, 5, 2, 1 e 4, respectivamente, sendo 5 a maior prioridade. Para cada um dos seguintes algoritmos de escalonamento, faça o diagrama de Gantt e determine o tempo médio de turnaround para cada algoritmo, desconsiderando o overhead de troca de contexto:

a) Round-robin (q = 3)

b) Prioridade não preemptiva

c) FCFS (ordem de execução: A, B, C, D, E)

d) SJF não preemptivo