Sistemas Operacionais

Políticas de Escalonamento da CPU

Lesandro Ponciano

2025

Objetivos da Aula

- Analisar algoritmos clássicos de escalonamento
  - O primeiro a chegar é o primeiro a ser servido (FCFS)
  - O menor job primeiro (SJF)
  - Escalonamento com Prioridades
  - Round-Robin (RR)

Algoritmos de Escalonamento

- Devem decidir a qual dos processos que estão na fila de prontos a CPU deve ser alocada

- É uma decisão política
  - O primeiro a chegar é o primeiro a ser servido (FCFS)
  - O menor job primeiro (SJF)
  - Escalonamento com Prioridades
  - Round-Robin (RR)

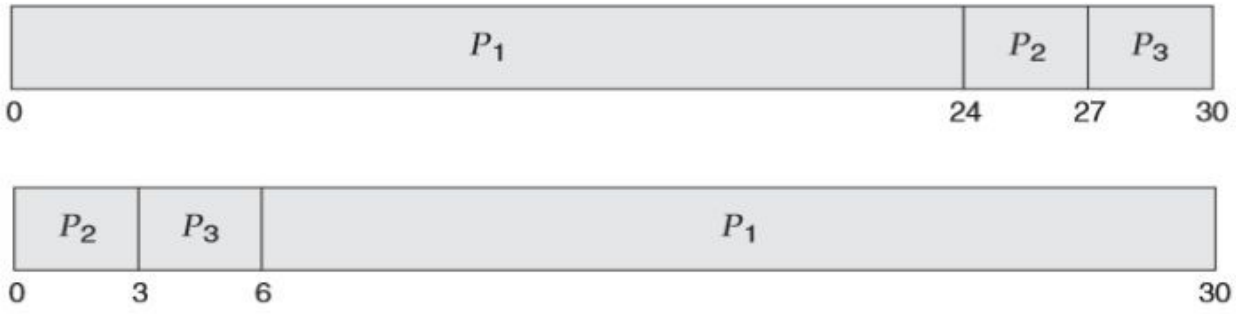
First-Come, First-Served (FCFS)

- O primeiro a chegar é o primeiro a ser servido
  - Processo que solicita a CPU primeiro a usa primeiro
  - Primeiro a entrar é o primeiro a sair (*first-in first-out*, FIFO)
  - Não usa preempção

- Pontos positivos
  - É fácil de implementar e de entender
- Ponto negativo
  - Tempo médio de espera não é mínimo e pode variar muito dependendo da ordem em que os processos chegam

First-Come, First-Served (FCFS)

Process	Burst Time
P_1	24
P_2	3
P_3	3

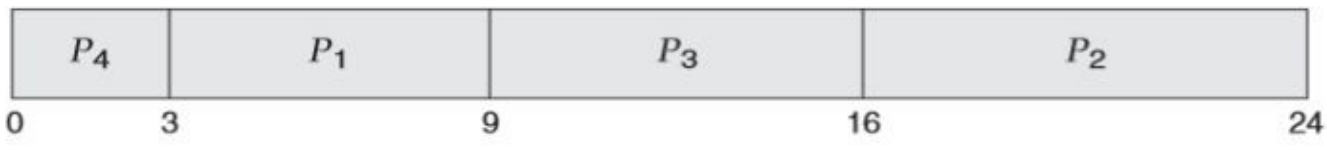


- Tempo médio de espera
  - Ordem 1 = (0+24+27)/3 = 17
  - Ordem 2 = (6+0+3)/3 = 3

Shortest-Job-First (SJF)

- O menor job (tarefa) primeiro
  - Associa a cada processo o intervalo do próximo pico de CPU
  - A CPU é atribuída ao processo que possui o pico mais curto
  - Empates são resolvidos usando FCFS

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



- Tempo médio de espera = (3+16+9+0)/4 = 7

Shortest-Job-First (SJF)

- Pontos positivos
  - É ótimo por fornecer o tempo médio de espera mínimo para um conjunto de processos
  - Reduz o tempo de espera de processos curtos e aumenta o tempo de espera de processos longos

- Pontos negativo
  - Como saber o tamanho do próximo pico de CPU?
  - Faz mais sentido de ser usado como escalonador de longo prazo do que escalonador de curto prazo
    - Usuário informa o tempo de uso do CPU ao criar o job
  - Pode gerar inanição (*starvation*)

SRTF: SJF com Preempção

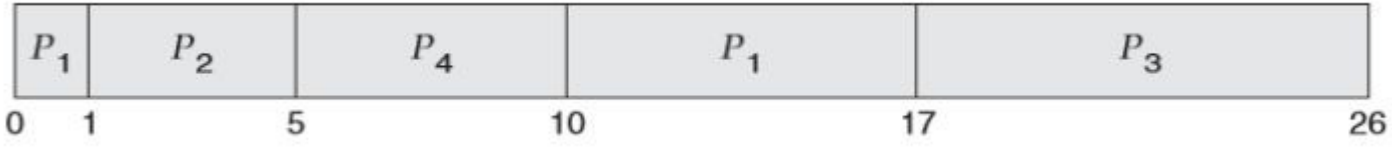
- O SJF com preempção também é chamado *Shortest Remaining Time First (SRTF)*

- Quando um novo processo chega com um tempo de execução menor que o tempo restante do processo que está na CPU, a execução do processo que está na CPU é interrompida e a CPU é alocada ao novo processo

Exemplo de SJF com Preempção

- Processo com tempo restante mais curto primeiro

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



- Tempo médio de espera
  - Com preempção: [(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4=6,5
  - Sem preempção:[(0-0)+(8-1)+(12-3)+(17-2)]/4=31/4=7,75

Escalonamento com Prioridades

- A CPU é alocada ao processo com prioridade mais alta
  - Uma prioridade é associada a cada processo
  - Empates são resolvidos usando FCFS
  - Dependendo do sistema
    - 0 pode indicar uma prioridade alta ou baixa
    - Usaremos valores baixos para indicar alta prioridade

Escalonamento com Prioridades

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Escalonamento com Prioridades

- Pode ou não ter preempção
  - Quando há preempção um processo pode deixar a CPU para dar lugar a outro processo de maior prioridade

- Problema
  - Inanição: um processo de baixa prioridade nunca ter acesso à CPU
  - Tal problema pode ser minimizado usando envelhecimento, aumenta gradualmente a prioridade de processos mais velhos

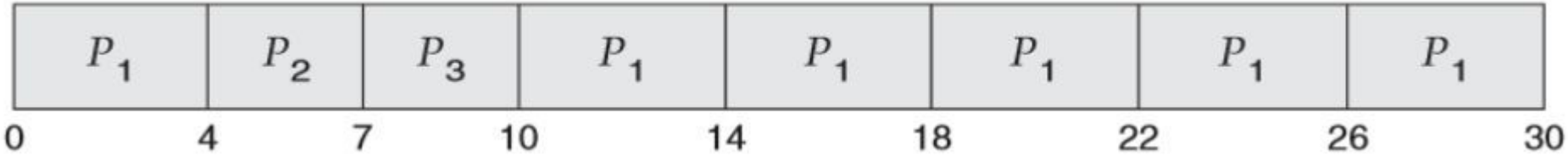
Escalonamento Round-Robin (RR)

- Cada processo tem direito a um *quantum* de tempo na CPU
  - Quantum costuma ter duração entre 10 e 100 milissegundos
  - Usa FCFS em uma fila circular
- Uma de duas coisas pode acontecer
  - O processo pode liberar a CPU voluntariamente se tiver um pico de CPU menor que a duração do quantum
  - O SO removerá o processo da CPU quando o quantum terminar
    - Troca de contexto para outro processo na fila

Exemplo

- Quantum de 4 milissegundos

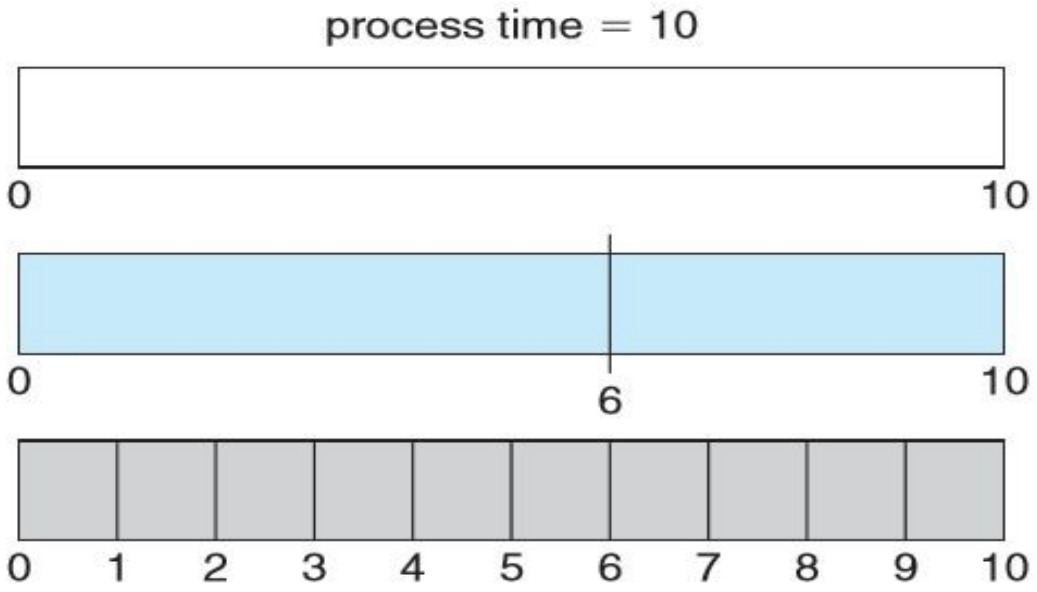
Process	Burst Time
P_1	24
P_2	3
P_3	3



- Tempo médio de espera
  - [(10-4)+4+7]/3=5,66

Escalonamento Round-Robin (RR)

- Duração do quantum e número de trocas de contexto



quantum	context switches
12	0
6	1
1	9

Escalonamento de Filas Multiníveis

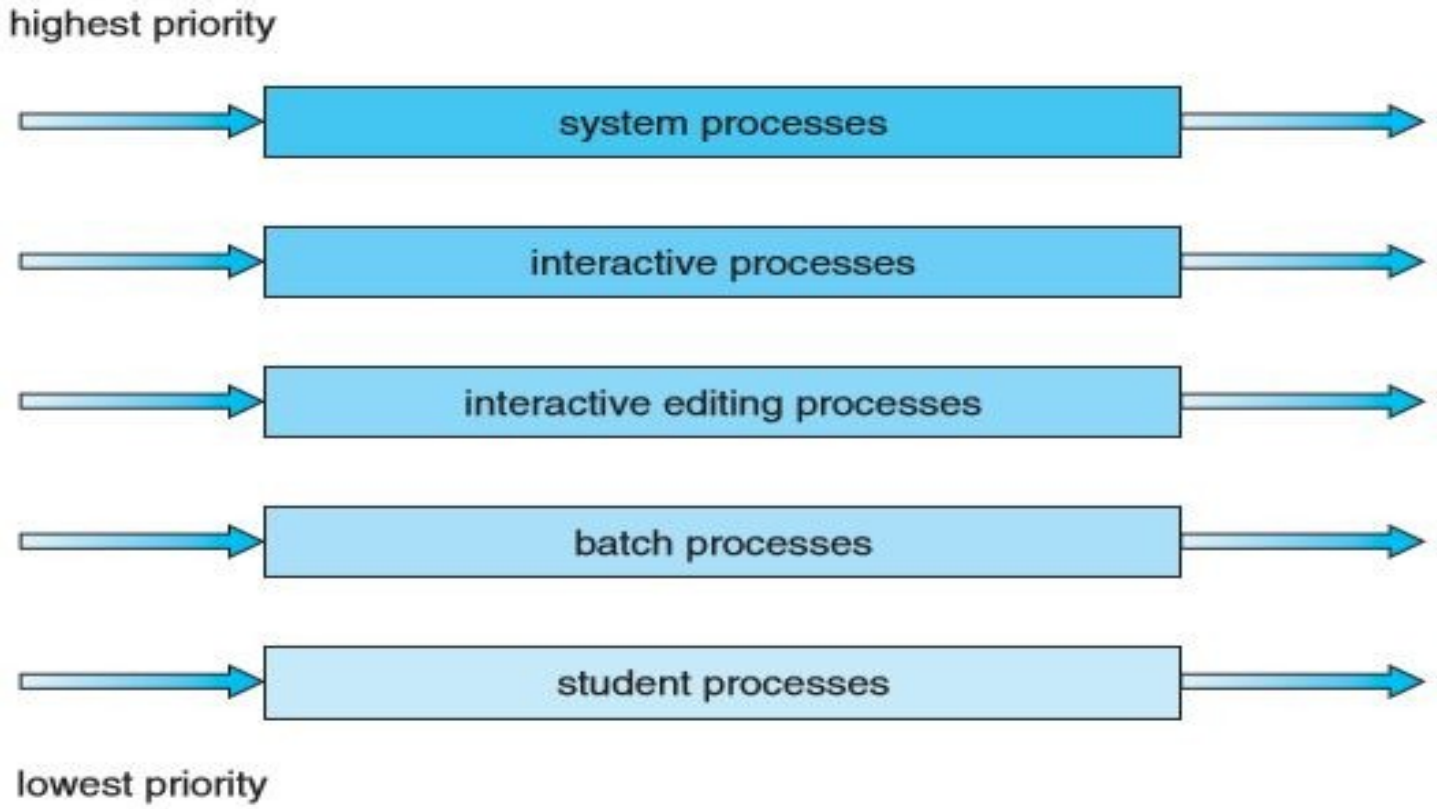
- Divide-se os processos por categorias
  - Prioridade, uso de memória, tipo do processo
- Divide-se a fila de prontos em diversas filas separadas
- Processos são atribuídos às filas de acordo com sua categoria
- Cada fila tem seu próprio algoritmo de escalonamento, como FCFS, SJF, prioridades, RR

Escalonamento de Filas Multiníveis

- Há um escalonador com prioridades e com preempção que decidirá sobre o acesso dos processos das filas à CPU

- Cada fila tem uma prioridade

Escalonamento de Filas Multiníveis



Usando Retroalimentação

- Escalonamento de Filas Multiníveis não permite que processos mudem de fila

- Incluindo-se retroalimentação, possibilita-se que um processo mude de fila ao longo da execução
  - Permite-se, por exemplo, que ele saia de uma fila de processos intensivos em E/S e mude para uma fila de processos intensivos em CPU

Processadores Múltiplos

- Normalmente cada processador mantém sua fila de processos
- Afinidade com o processador
  - Ocorre quando o processo possui características associadas com um dado processador, ex: dados em *caching*
  - Deve-se escalonar o processo para o processador com o qual ele tem afinidade
- Balanceamento de carga
  - Manter a carga uniforme entre os diversos processadores

Atividade de Fixação

- Mostre o uso da CPU ao longo do tempo e calcule o tempo médio de espera dos processos abaixo quando os seguintes algoritmos são usados
  - FCFS
  - SJF (com e sem preempção)
  - Prioridades (com e sem preempção)
  - RR (quantum de 25)

Processo	Chegada	Pico de CPU	Prioridade
P1	0	75	3
P2	20	26	2
P3	30	49	1
P4	40	100	0

Referências

TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson Prentice Hall, 2009. xvi, 653 p. ISBN 9788576052371

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais: princípios básicos. Rio de Janeiro, RJ: LTC, 2013. xvi, 432 p. ISBN 9788521622055

PONCIANO, L; Brasileiro, Francisco . Assessing Green Strategies in Peer-to-Peer Opportunistic Grids. Journal of Grid Computing, v. 11, p. 129-148, 2013.

Ponciano, Lesandro; Brito, Andrey ; Sampaio, Lívia ; Brasileiro, Francisco. Energy Efficient Computing through Productivity-Aware Frequency Scaling. In: 2012 International Conference on Cloud and Green Computing (CGC), 2012, Xiangtan. p. 191-198.

Sistemas Operacionais

Prof. Dr. Lesandro Ponciano

https://orcid.org/0000-0002-5724-0094