Sistemas Operacionais

Aula 10 - Escalonamento de CPU – Parte 01

Professor: Wellington Franco

Escalonamento de processos

- Na multiprogramação, vários processos querem executar:
  - Mas cada CPU só pode receber um processo por vez
  - **Problema:** Quem deve executar agora? P^1 ou P^2?
- O SO é o responsável por resolver este problema:
  - Escalonador e o(s) algoritmo(s) de escalonamento
- **O objetivo principal é sempre usar de forma eficiente a CPU!**

Objetivos do Escalonamento

- **Justiça:** Particionamento justo dos recursos de *hardware*
- **Aplicação da política:** Garantir que a política é atendida
- **Equilíbrio:** Todo o sistema deve manter-se ocupado
- **Vazão:** Maximizar o número de tarefas por hora
- **Tempo de Resposta:** Minimizar o tempo de término
- **Proporcionalidade:** Satisfazer as expectativas do usuário

Escalonador de processos

- Escolhe o melhor processo a ser executado:
  - Para isso, segue-se algum tipo de algorítimo pré-definido
  - Uma vez escolhido, troca-se um processo por outro
- Chaveamento entre processos é muito custoso:
  - Bloqueia e salva o estado do processo atual na CPU
  - Carrega e inicia o estado do processo escalonado

Escalonamento de processos

> *Sistemas computacionais da atualidade permitem que múltiplos programas sejam carregados em memória e executados de forma concorrente. Potencialmente, todos os processos são executados com as CPUs multiplexadas para eles. Ao trocar a CPU entre os processos, o SO faz o computador mais produtivo.*
>
> **SILBERSCHARTZ, P. B. GALVIN, G. GAGNE (2009)**

Conceitos básicos

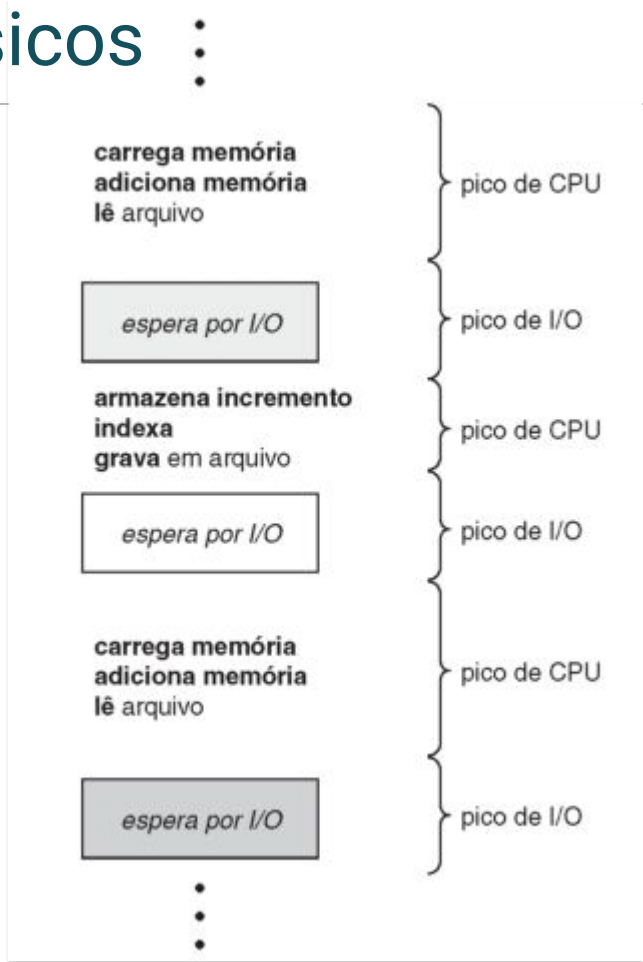
- Em um sistema com um único processador, apenas um processo pode ser executado por vez, qualquer outro deve esperar.

- Multiprogramação é a ideia de haver sempre algum processo em execução para que a CPU seja otimizada

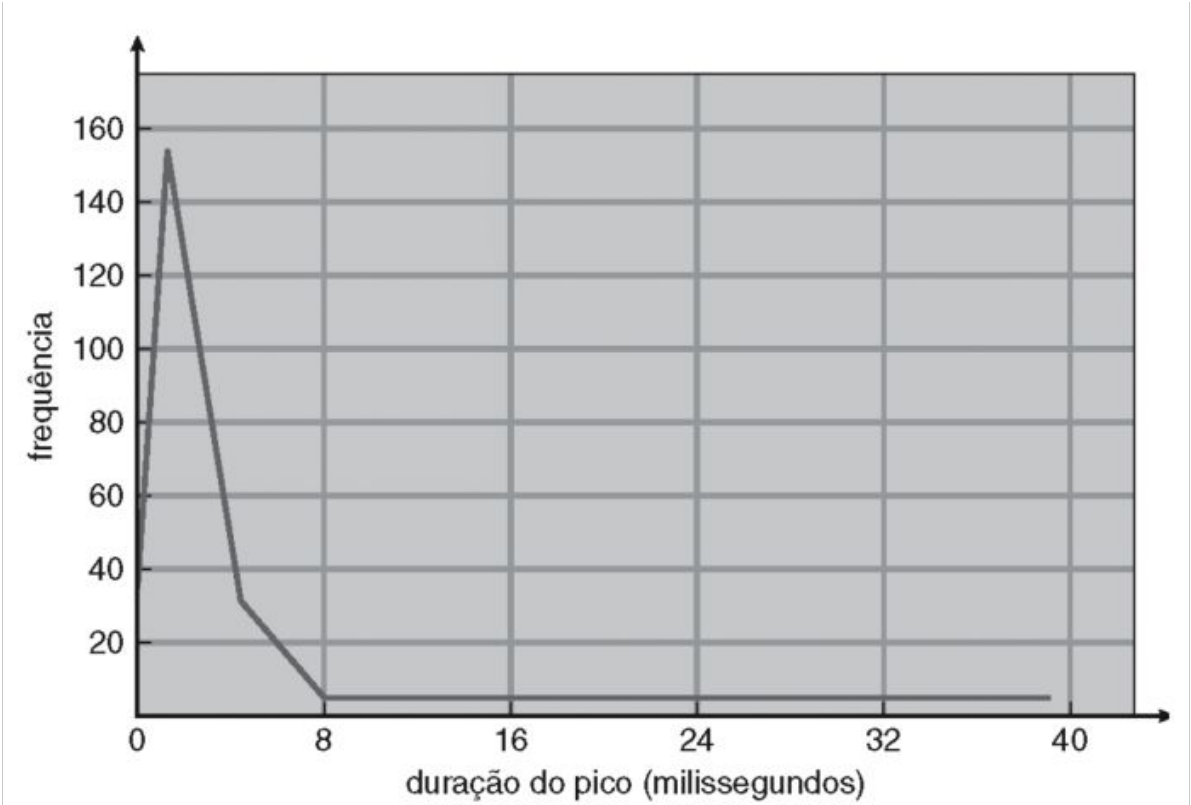
- Sempre que um processo tiver de esperar por algum recurso (ou I/O), outro processo pode assumir o uso da CPU
  - Escalonamento (ou Scheduling) é uma função básica do SO; é essencial

Conceitos básicos

- A execução de processos é composta por um ciclo de execução da CPU e espera por operações de I/O.
- Os processos se alternam entre esses dois estados
- O último pico de CPU termina quando o sistema solicita o encerramento da execução



Histograma com conceitos básicos



Scheduler da CPU

- Sempre é selecionado um processo da fila de pronto
- O processo de seleção é executado pelo **scheduler de curto prazo**
- O scheduler seleciona um processo entre os processos na memória para execução e aloca a CPU
- A fila de prontos ocorre como FIFO (primeiro a entrar, primeiro a sair) ?

Algoritmos de escalonamento

> *O escalonamento de CPU lida com o problema de decidir quais os processos na fila de prontos deverá ser alocado a CPU em um determinado momento. Diversos critérios foram sugeridos para comparar os algoritmos de escalonamento: nível de utilização de CPU, vazão, tempo de espera, tempo de resposta...*
>
> **SILBERSCHARTZ, P. B. GALVIN, G. GAGNE (2009)**

Tipo de algoritmos de escalonamento

- Um processo não pode monopolizar a CPU:
  - Prejudica a segurança e impossibilita a multiprogramação
- O SO é o responsável por resolver este problema:
  - Periodicamente, é gerada uma interrupção de *software*
  - O controle é devolvido para o Sistema Operacional
  - Sistema Operacional decide se o processo continua ou não
- Escalonamento pode ser: **preemptivo** ou **não-preemptivo**

Tipo de algoritmos de escalonamento

Escalonamento Não-Preemptivo

Mantém o processo em execução até o mesmo entrar em estado bloqueado ou liberar a CPU por conta própria

Escalonamento Preemptivo

Processo é executado por um tempo máximo fixado

Scheduling com preempção

- As decisões do escalonador de CPU podem ocorrer quando um processo:
  - Muda do estado de execução para o estado de espera
  - Muda do estado de executando para pronto
  - Muda do estado de espera para pronto
  - Quando um processo é finalizado

Scheduling com preempção

- As decisões do escalonador de CPU podem ocorrer quando um processo:
  - Muda do estado de execução para o estado de espera
  - Muda do estado de executando para pronto
  - Muda do estado de espera para pronto
  - Quando um processo é finalizado
  - As condições 1 e 4 é dito **não-preemptivo**. As condições 2 e 3 é dito **preemptivo**

Critérios de scheduling

- É utilizado para a comparação de algoritmos de scheduling da CPU
  - **Utilização da CPU**: manter a CPU o mais ocupada possível. Pode variar de 0 a 100 por cento.
  - **Vazão:** quando a CPU está ocupada executando processos, trabalho está sendo realizado (quantidades de processo/unidade de tempo)
  - **Tempo de turnaround:** quantidade necessária de tempo para executar um processo

Critérios de scheduling

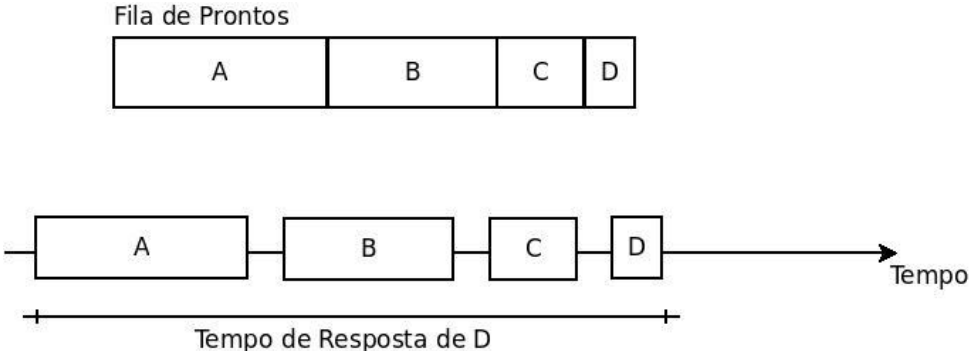
- É utilizado para a comparação de algoritmos de scheduling da CPU
  - **Tempo de espera:** quantidade de tempo que um processo aguardou na fila de prontos
  - **Tempo de resposta:** quantidade de tempo entre a requisição de execução de um programa e a produção da primeira resposta. (sistemas interativos)

Critérios de otimização dos scheduling

- É desejável a maximização dos critérios:
  - Utilização máxima de CPU
  - Vazão máxima
  - Tempo de turnaround mínimo
  - Tempo de espera mínimo
  - Tempo de resposta mínimo

Algoritmos de escalonamento

- Escalonamento **First-In-First-Out (FIFO)/Primeiro a Entrar, Primeiro a Ser Atendido (FCFS):**
  - Processos prontos alocados em uma fila
  - Processos são alocados à CPU na ordem de chegada
  - Jobs grandes podem atrasar (e muito) jobs pequenos



Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**
  - O processo que solicita a CPU primeiro é o primeiro a usá-la.
  - A implementação da política FCFS é facilmente gerenciada com uma fila FIFO.
  - Quando um processo entra na fila de prontos, seu PCB é inserido no final da fila. Quando a CPU estiver livre, ela é alocada para o processo na cabeça da fila.
  - O processo em execução é então removido da fila

Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**
    - **Desvantagem**: tempo médio de espera é geralmente bem longo

Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**

Processo	Duração do Pico
P_1	24
P_2	3
P_3	3

  - Supondo que chegaram na ordem P1, P2, P3

Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**

  - Tempo de espera médio = (0 + 24 + 27)/3 = 17

Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**
  - Supondo que a ordem foi: P2, P3, P1

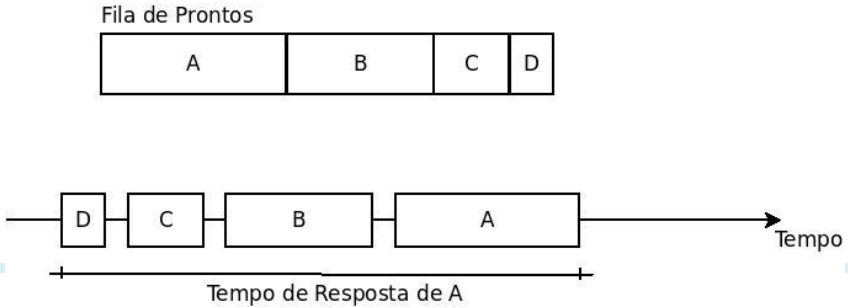
  - Tempo de espera médio = (0 + 3 + 6)/3 = 3

Algoritmos de scheduling

- **Primeiro a Entrar, Primeiro a Ser Atendido (FCFS)**
  - **“Efeito comboio”:** todos os outros processos esperam que o grande processo saia da CPU
  - Não usa preempção

Algoritmos de escalonamento

- Escalonamento **Shortest job first**:
  - Cada processo possui associado uma estimativa do seu tempo total de processamento
  - Processo com tempo mais curto é alocado à CPU
  - Situação adequada apenas em situações onde todos os processos estão disponíveis e a estimativa também



Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**
  - Associa a cada processo a duração do seu próximo pico de CPU
  - Usa essas durações para escalonar o processo com a menor duração de pico de CPU
  - Se tiver dois processos com a mesma duração, o primeiro a entrar na fila será o executado

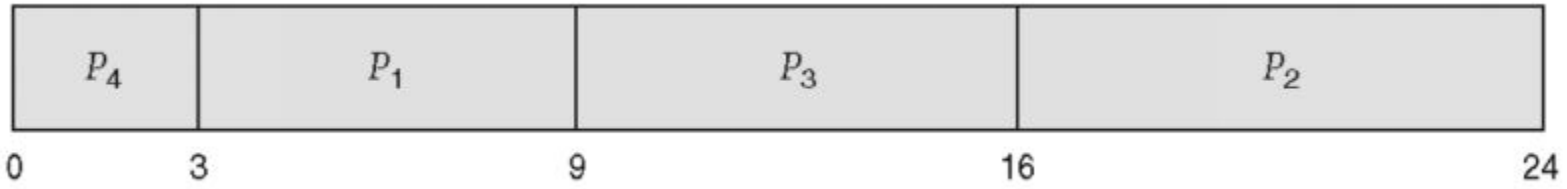
Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**

Processo	Duração do Pico
P_1	6
P_2	8
P_3	7
P_4	3

Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**



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

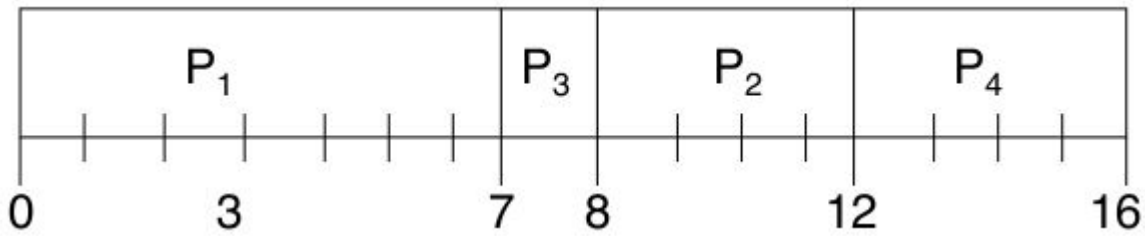
Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**
  - SJF não-preemptivo

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

Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**
  - SJF não-preemptivo



    - Tempo de espera médio = (0 + 6 + 3 + 7)/4 = 4

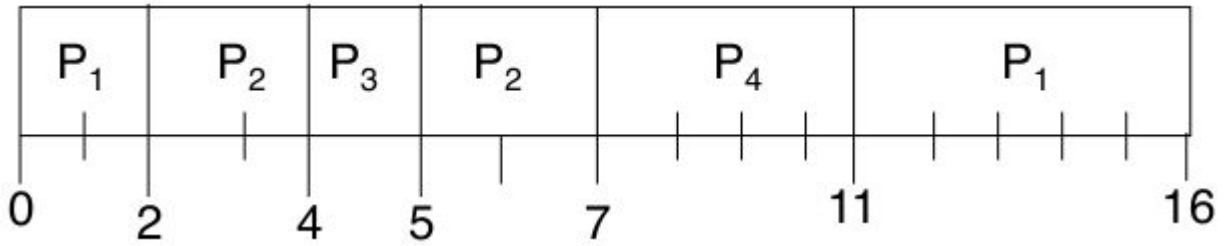
Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**
  - SJF preemptivo

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

Algoritmos de scheduling

- **Scheduling Menor-Job-Primeiro (SJF)**
  - SJF preemptivo



    - Tempo de espera médio = (9 + 1 + 0 +2)/4 = 3

Algoritmos de scheduling

- **Scheduling por prioridade**
  - Cada processo recebe um nível de prioridade
  - A CPU é alocada para o processo com a maior prioridade (menor valor número)

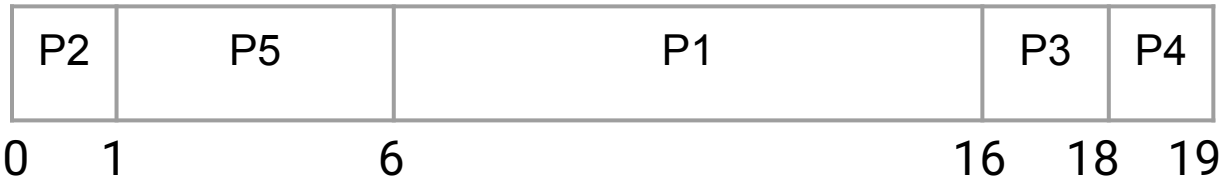
Algoritmos de scheduling

- **Scheduling por prioridade**

Processo	Duração do Pico	Prioridade
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Algoritmos de scheduling

- **Scheduling por prioridade**



Tempo de espera médio = 8,2

Starvation

- Também conhecido como adiamento indefinido:
  - Um processo nunca recebe a UCP para processamento
- Ocorre devido a falhas nas políticas de escalonamento:
  - Em políticas baseadas em prioridades, processos de baixa prioridade podem nunca receber a CPU;
  - Nesse caso, o **envelhecimento** evita o starvation.
- Sistema Operacional deve mirar a justiça entre os processos.

Algoritmos de scheduling

- **Scheduling por prioridade**
  - Com prempção e não-preempção
  - Problema Starvation (bloqueio indefinido): processos de baixa prioridade podem nunca serem executados
    - Solução: **Envelhecimento** - a medida que o tempo passa a prioridade dos processos aumenta

Algoritmos de scheduling

- **Scheduling Round-Robin**
  - Cada processo recebe uma pequena quantidade de tempo de CPU (quantum), usualmente entre 10 e 100 milissegundos.

  - Depois que esse tempo esgota, o processo é interrompido e inserido no fim da fila de prontos.

Algoritmos de scheduling

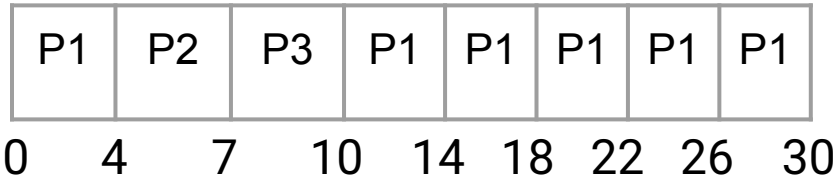
- **Scheduling Round-Robin**

Processo	Duração do Pico
P1	24
P2	3
P3	3

Quantum = 3

Algoritmos de scheduling

- **Scheduling Round-Robin**



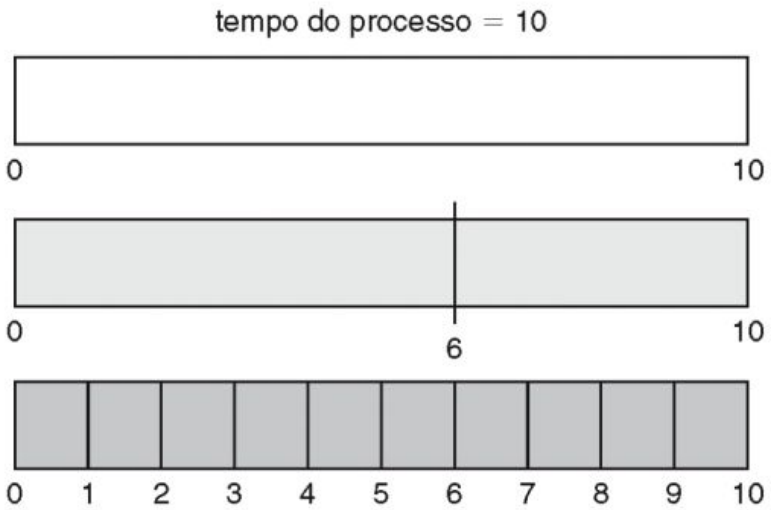
Tempo de espera médio = (10-4 + 4 + 7) = 5,66

Algoritmos de scheduling

- **Scheduling Round-Robin**
  - Desempenho
    - Se q grande -> FCFS
    - Se q pequeno -> Compartilhamento do processador -> overhead
      - q precisa ser grande em relação ao tempo de troca de contexto, caso contrário o overhead será muito grande.

Algoritmos de scheduling

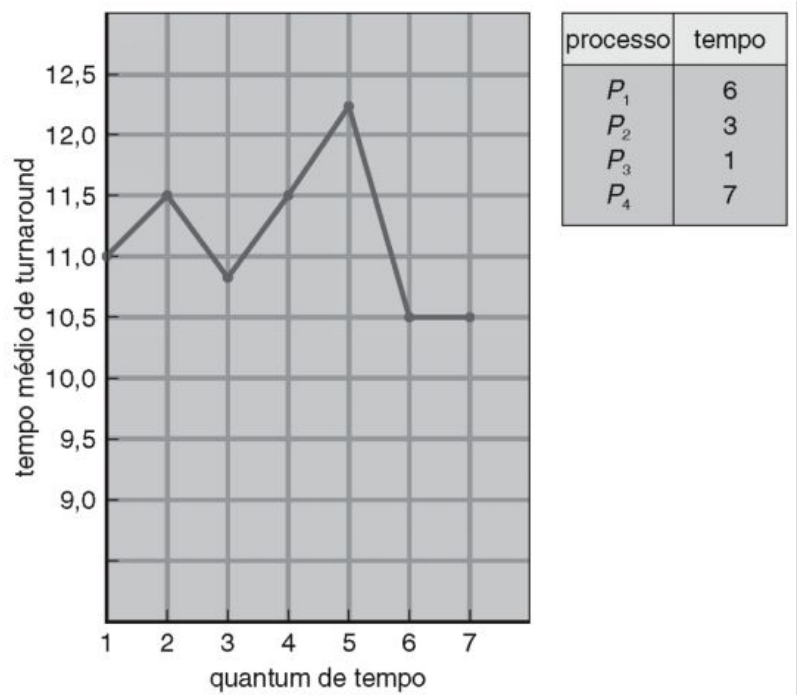
- **Scheduling Round-Robin**



quantum	mudanças de contexto
12	0
6	1
1	9

Algoritmos de scheduling

- **Scheduling Round-Robin**



processo	tempo
P_1	6
P_2	3
P_3	1
P_4	7

Dúvidas??

E-mail: wellington@crateus.ufc.br