

# Sistemas Operacionais

Cap.5 - Scheduling de CPU



Prof. MSc. Renzo P. Mesquita  
[renzo@inatel.br](mailto:renzo@inatel.br)

# Capítulo 5

## Scheduling de CPU

- 5.1. Conceitos Básicos;*
- 5.2. Critérios de Scheduling*
- 5.3. Algoritmos de Scheduling;*
- 5.4. Scheduling com Multiprocessadores;*
- 5.5. Avaliação de Algoritmos;*



# Objetivos

- Introduzir o Scheduling da CPU, que é a base dos Sistemas Operacionais multiprogramados;
- Descrever os principais Algoritmos de Scheduling da CPU;
- Discutir critérios de avaliação para a seleção de um Algoritmo de Scheduling da CPU para um sistema específico;



# 5.1. Conceitos Básicos

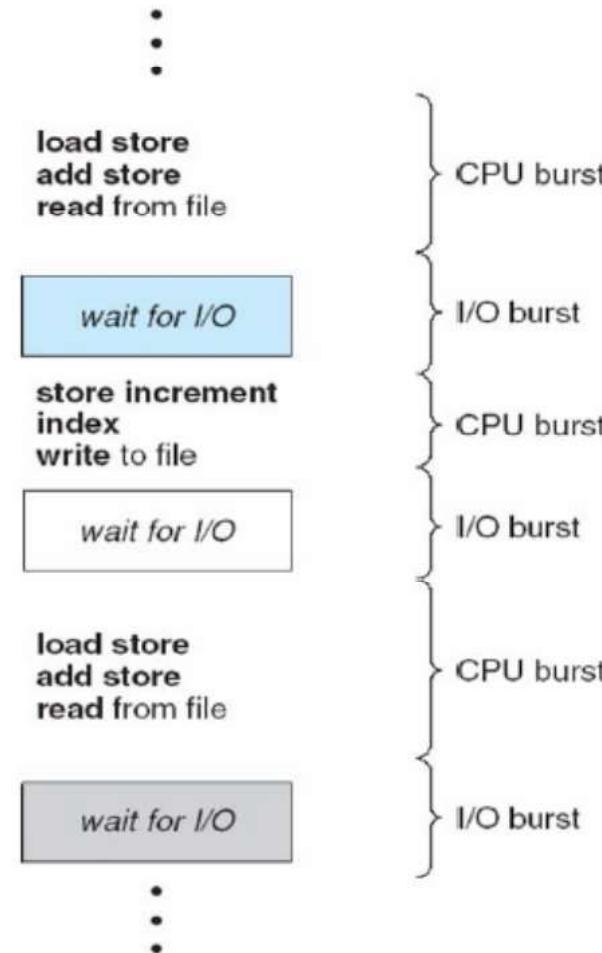
## 5.1.1. Ciclos de Picos de CPU

A execução de Processos é composta por um ciclo de execução de CPU e espera por operações de I/O. Os processos se alteram entre essas duas vertentes.

- A execução de um Processo começa com um pico de CPU, logo, é seguido por um pico de IO, que é seguido por outro pico de CPU e assim por diante;
- Vale salientar que a duração dos picos de CPU variam muito de um processo para outro e de um computador para outro;
- Um processo com longos picos de CPU é denominado CPU-Bound;
- Um processo com curtos picos de CPU é denominado IO-Bound;



## 5.1. Conceitos Básicos



Sequência alternada de picos de CPU e I/O.

# 5.1. Conceitos Básicos

## 5.1.2. Scheduler da CPU

*Sempre que a CPU fica ociosa, o SO tem de selecionar um dos processos da Fila de Prontos (Ready Queue) para ser executado. O processo de seleção é executado pelo Scheduler de Curto Prazo (Short-term Scheduler).*

- É bom ressaltar que a Fila de Prontos não é necessariamente uma Fila que utiliza unicamente da filosofia FIFO (First In First Out);
- Veremos que esta fila pode ser implementada utilizando diversos outros tipos de algoritmos, como por exemplo, uma fila de prioridades;



## 5.1. Conceitos Básicos

O Scheduler de Curto Prazo pode ser implementado utilizando de dois esquemas:

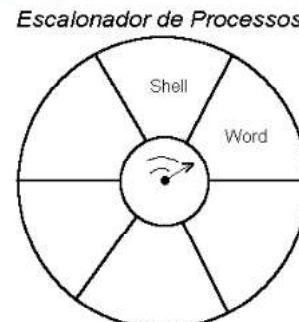
### *1) Esquema Não-preemptivo (Cooperativo)*

Quando o SO não interrompe (frusta) a execução do processo que usava a CPU. O processo interrompe sua execução só nos seguintes casos:

- Quando o Processo (em execução) passa de running para wait;
- Quando o Processo (em execução) termina;

### *2) Esquema Preemptivo*

Quando o SO interrompe (frusta) a execução do processo que usava a CPU. Exige modificações no Hardware do Sistema e no Kernel do SO. Sua implementação é mais complexa.



# 5.1. Conceitos Básicos

## 5.1.3. Despachante (Dispatcher)

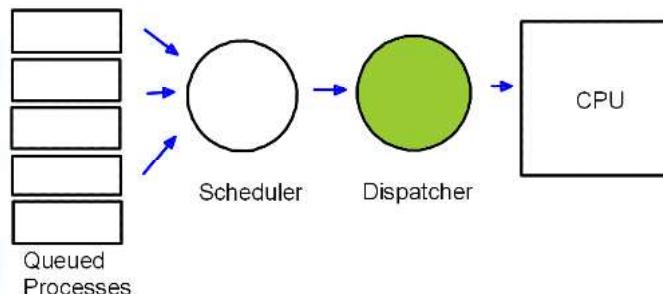
Outro componente envolvido no Scheduling da CPU é o Despachante.

*O Despachante é o módulo que passa o controle da CPU para o processo selecionado pelo Scheduler de Curto Prazo.*

O despachante deve ser implementado para ser extremamente rápido, uma vez que ele é chamado em cada Mudança de Contexto. Ele é um componente que tem as seguintes funções:

- Realizar a Mudança de Contexto de Processos na CPU;
- Mudança do sistema para a modalidade de usuário;

O tempo de que o Dispatcher precisa para interromper um Processo e iniciar a execução de outro é conhecido como Latência de Despacho.



## 5.2. Critérios de Scheduling

Diferentes Algoritmos de Scheduling da CPU têm propriedades distintas, e a escolha de um algoritmo específico pode favorecer uma classe de processos em vez de outra.

Muitos critérios têm sido sugeridos para a comparação de algoritmos de Scheduling da CPU. Os critérios são os seguintes:

### 1) Utilização da CPU

Indica quanto a CPU permanece ocupada;

### 2) Throughput

Número de processos que são completados por unidade de tempo;

### 3) Tempo de Turnaround

Intervalo de tempo entre a submissão e o término de um processo;

### 4) Tempo de Espera

Total de tempo que um processo fica na Ready Queue;

### 5) Tempo de Resposta

Intervalo de tempo que vai da submissão de um processo até quando este processo produz o seu primeiro resultado;

## 5.2. Critérios de Scheduling

O Algoritmo Perfeito seria aquele que maximiza-se a Utilização da CPU e o Throughput, e minimiza-se o Tempo de Turnaround, Tempo de Espera e Tempo de Resposta.

Dentre todos os critérios vistos, o mais utilizado e foco do nosso estudo é o *Tempo de Espera (Waiting Time)* .

*Turnaround*  
*Tempo de Espera*  
*Tempo de Resposta*



*Uso da CPU*  
*Throughput*

## 5.3. Algoritmos de Scheduling

*O Scheduling da CPU lida com a decisão de para qual dos processos da Fila de Pronto (Ready Queue) a CPU deve ser alocada.*

Existem diversos Algoritmos de Scheduling de CPU diferentes. Estudaremos os seguintes:

- 1) *FCFS (First-Come First Served);*
- 2) *SJF (Shortest Job First);*
- 3) *PS (Priority Scheduling);*
- 4) *RR (Round Robin Scheduling);*
- 5) *MQS (Multilevel Queue Scheduling);*
- 6) *MFS (Multilevel feedback-queue Scheduling);*

## 5.3. Algoritmos de Scheduling

### 5.3.1. FCFS (First Come First Served)

Neste esquema, o processo que solicita a CPU primeiro é o primeiro a usá-la.

- Implementado através de uma fila FIFO;
- É um Algoritmo Não-Preemptivo, ou seja, o Processo só libera a CPU quando precisa usar uma I/O ou quando termina;
- O Tempo médio de espera neste algoritmo muitas vezes não é mínimo e pode variar muito se os intervalos de pico de CPU dos Processos variarem muito;

Vamos ver um exemplo?

## 5.3. Algoritmos de Scheduling

### EXEMPLO 1

Considere o conjunto de Processos a seguir que chegam no momento 0, com o intervalo do pico de CPU dado em milissegundos. Se os processos chegarem na ordem P1,P2,P3, qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo FCFS?

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

Dica: Desenhe um Gráfico Gantt para facilitar o desenvolvimento do Exemplo;



## 5.3. Algoritmos de Scheduling

### EXEMPLO 2

*Agora considere o mesmo conjunto de Processos chegando na ordem  $P_2, P_3, P_1$ . Qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo FCFS?*

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

Dica: Desenhe um Gráfico Gantt para facilitar o desenvolvimento do Exemplo;



## 5.3. Algoritmos de Scheduling

### 5.3.2. SJF (Shortest Job First)

Neste esquema, quando a CPU está disponível, ela é atribuída ao Processo que tem o próximo pico de CPU mais curto.

- Mas como calcular o próximo pico de CPU de cada Processo?
- Ex: Através de uma estimativa:  $T_{n+1} = a t_n + (1 - a) T_n$ , onde:
  - $T_{n+1}$ : previsão do CPU burst no tempo  $n+1$ ;
  - $t_n$ : valor real do CPU burst no tempo  $n$ ;
  - $T_n$ : previsão do CPU burst no tempo  $n$ , baseado no histórico do passado;
  - $a$ : uma constante;
- O SJF é comprovadamente ótimo, pelo fato de fornecer o menor tempo médio de espera para um determinado número de Processos;
- O algoritmo SJF pode ser tanto Não-Preemptivo quanto Preemptivo;

Vamos ver um exemplo?

## 5.3. Algoritmos de Scheduling

### EXEMPLO 3

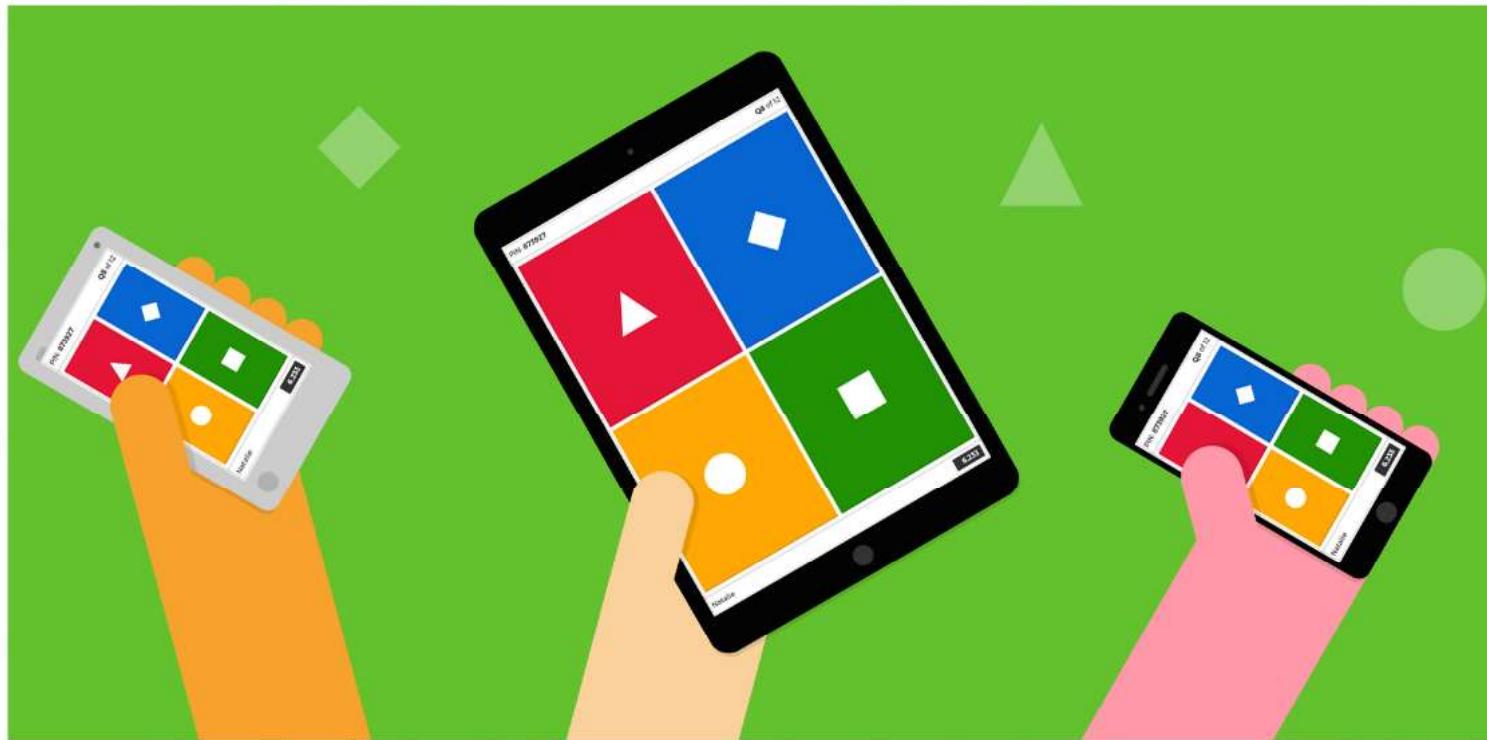
Considere o conjunto de Processos a seguir que chegam no momento 0, com o próximo pico de CPU dado em milissegundos. Qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo SJF Não-Preemptivo?

<u>Process</u>	<u>Burst Time</u>
$P_1$	6
$P_2$	8
$P_3$	7
$P_4$	3

Dica: Desenhe um Gráfico Gantt para facilitar o desenvolvimento do Exemplo;



# HORA DO **Kahoot!**



**ACESSE:** <https://kahoot.it>

## 5.3. Algoritmos de Scheduling

### EXEMPLO 4

Considere o conjunto de Processos a seguir que chegam em momentos distintos, com o próximo pico de CPU dado em milissegundos. Qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo SJF Preemptivo?

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	8
$P_2$	1	4
$P_3$	2	9
$P_4$	3	5

Dica: Desenhe um Gráfico Gantt para facilitar o desenvolvimento do Exemplo;

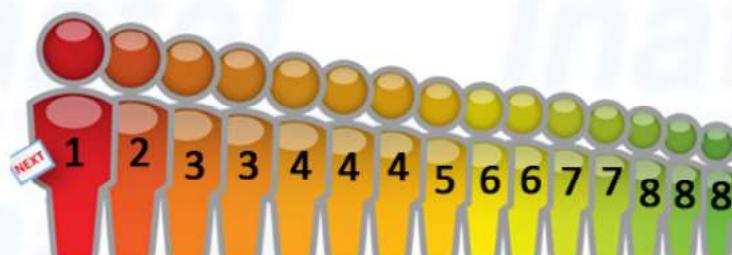


# 5.3. Algoritmos de Scheduling

## 5.3.3. PS (*Priority Scheduling*)

Neste esquema, uma prioridade é associada a cada processo, e a CPU é alocada ao processo de prioridade mais alta.

- Processos de prioridades iguais são agendados na ordem FCFS;
- SJF é portanto um caso particular de priority scheduling;
- A prioridade é usualmente dada por um valor numérico;
- Convenção que iremos adotar: valor numérico mais baixo significa maior prioridade;
- O algoritmo PS pode ser tanto Não-Preemptivo quanto Preemptivo;



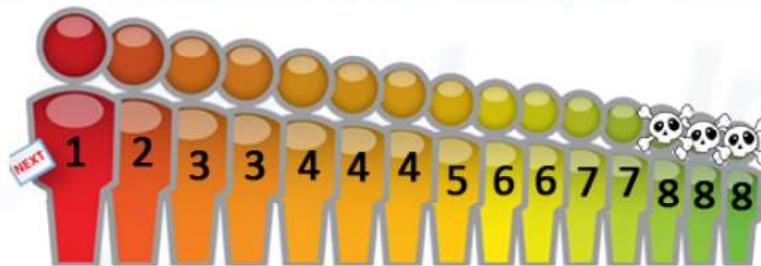
## 5.3. Algoritmos de Scheduling

### 5.3.3. PS (Priority Scheduling)

#### 5.3.3.1 Starvation (ou Inanição)

Acontece quando um processo está esperando pelo uso da CPU mas nunca é selecionado para usá-la;

- Starvation (ou inanição) é o maior problema do PS;
- Como resolver este problema? Utilizando uma técnica chamada Aging (ou Envelhecimento);
- o Aging é uma técnica que aumenta gradualmente a prioridade dos processos que esperam no sistema por muito tempo;



## 5.3. Algoritmos de Scheduling

### EXEMPLO 5

Considere o conjunto de Processos a seguir que chegam no momento 0, com o intervalo do pico de CPU dado em milissegundos. Se os processos chegarem na ordem P1,P2,P3,P4 e P5 qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo PS Não-Preemptivo?

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
$P_1$	10	3
$P_2$	1	1
$P_3$	2	4
$P_4$	1	5
$P_5$	5	2

Dica: Desenhe um Gráfico Gantt para facilitar o desenvolvimento do Exemplo;

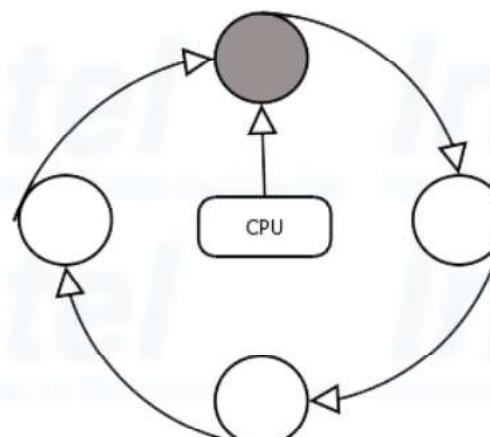


## 5.3. Algoritmos de Scheduling

### 5.3.4. RR (Round Robin)

Este algoritmo foi projetado especialmente para sistemas de tempo compartilhado. Ele é semelhante ao FCFS, mas a preempção é adicionada para permitir que o sistema se alterne entre os processos.

- Trabalha com a idéia de QUANTUM de Tempo (Tempo máximo que um processo pode usar a CPU a cada vez que ele é alocado nela);
- Geralmente, um quantum de tempo tem duração de 10ms a 100ms;
- A Fila de Prontos é tratada como uma Fila Circular;



## 5.3. Algoritmos de Scheduling

### EXEMPLO 6

Considere o conjunto de Processos a seguir que chegam no momento 0, com o intervalo do pico de CPU dado em milissegundos. Se usarmos um Quantum de tempo de 4ms e os processos chegarem na ordem  $P_1, P_2, P_3$ , qual é o Tempo Médio de Espera destes processos utilizando o Algoritmo RR?

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

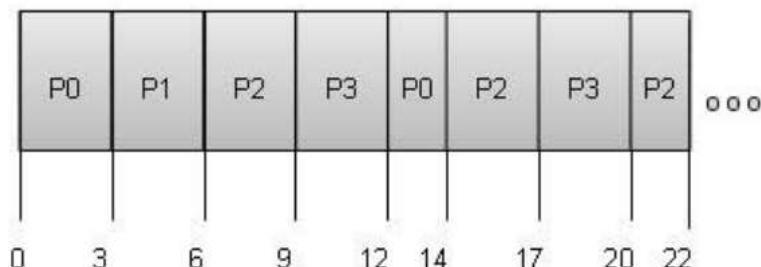


## 5.3. Algoritmos de Scheduling

### 5.3.4. RR (Round Robin)

O Desempenho do RR depende muito do tamanho do Quantum de tempo.

- Se o Quantum Time é pequeno (menor que o CPU-burst dos processos), há alta taxa de Context-Switching (logo, causando muito overhead);
- Se o Quantum Time tende ao infinito, o RR tende a ser um FCFS;



*Exemplo:*

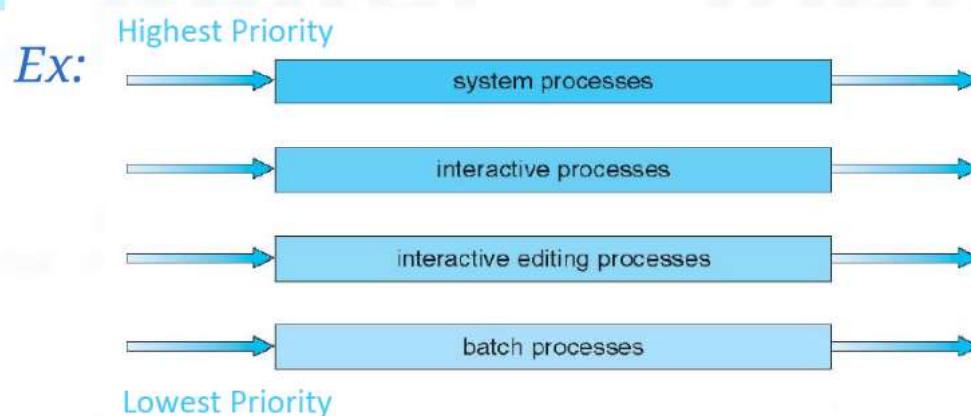
Quantum Pequeno = Muitos Context-Switching;

## 5.3. Algoritmos de Scheduling

### 5.3.5. MQS (Multilevel Queue Scheduling)

Algoritmo criado para situações em que os processos são facilmente classificados em diferentes grupos. Por exemplo, uma divisão comum é feita entre processos de *Foreground* (Interativos) e *Background* (Batch).

- Divide a Ready Queue em várias filas separadas;
- Cada processo é PERMANENTEMENTE designado para somente uma das filas, de acordo com suas características;
- Cada Fila pode ter seu próprio Algoritmo de Scheduling;
- Cada fila tem prioridade absoluta sobre filas de menor prioridade;
- Existe um Scheduler principal entre as filas, de prioridade fixa e com Preempção;

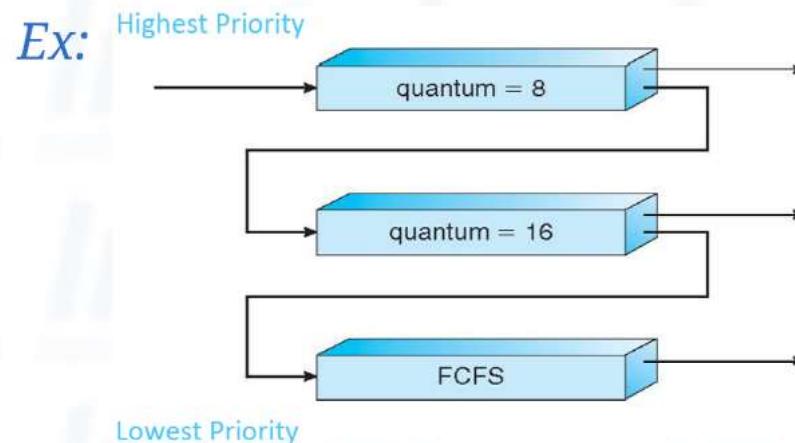


## 5.3. Algoritmos de Scheduling

### 5.3.6. MFS (Multilevel feedback-queue Scheduling)

Funcionamento parecido com o MQS, porém, permite a alternância de processos entre filas.

- Separa os Processos de acordo com seus CPU-Bursts (Picos de CPU);
- Processos CPU-Bound vão se deslocando para as filas de mais baixa prioridade e Processos IO-Bound e interativos vão se deslocando para filas de mais alta prioridade;
- Utiliza da técnica de Aging (Envelhecimento) para evitar o Starvation (Move o processo entre filas);
- É o esquema mais geral e flexível, porém, é o mais complexo em termos de implementação;



## 5.4. Scheduling com Múltiplos Processadores

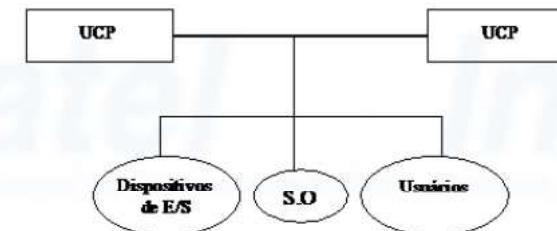
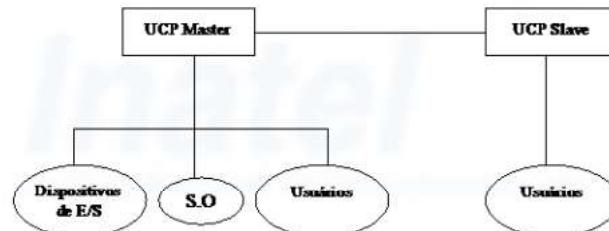
Como já vimos anteriormente, duas abordagens podem ser usadas para multiprocessamento:

1. *Multiprocessamento Assimétrico (Master-Slave);*
2. *Multiprocessamento Simétrico;*

No Multiprocessamento Assimétrico, existe um processador designado como master, que é o scheduler dos demais processadores. Só ele acessa a Ready Queue;

No Multiprocessamento Simétrico, pode-se ter duas situações:

- Cada Processador examina uma Ready Queue Comum;
- Cada Processador pode ter sua própria Ready Queue;



## 5.4. Scheduling com Múltiplos Processadores

### 5.4.1. Afinidade com o Processador (Process Affinity)

Ocorre quando um Processo busca ser executado sempre em um mesmo Processador para tentar sempre que possível, aproveitar dados deixado por ele na cache em outros momentos;  
A afinidade pode ser implementada de forma leve e forte.

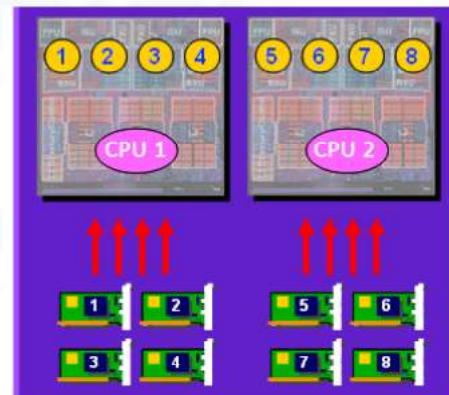
#### A) Leve

Quando o SO tem uma política de tentar manter um processo sendo executado no mesmo processador, mas sem garantir que ele fará isso.

#### B) Forte

Quando o SO obriga um processo ser executado em um único processador (Ex: Alguns sistemas Linux);

Ex:



## 5.4. Scheduling com Múltiplos Processadores

### 5.4.2. Balanceamento de Carga (*load balancing*)

No caso de cada processador ter sua própria Fila de Prontos, o SO deve implementar políticas para **distribuir uniformemente** a carga de processamento entre todos os processadores.

Existem duas abordagens para o balanceamento de carga:

#### A) Migração por Expulsão

Uma **tarefa** fica **verificando constantemente a carga** de cada processador, e quando encontra um desequilíbrio, tenta equilibrar a carga entre processadores;

#### B) Migração por Absorção

Quando um **processador ocioso extraí** uma tarefa que está esperando em um processador ocupado;

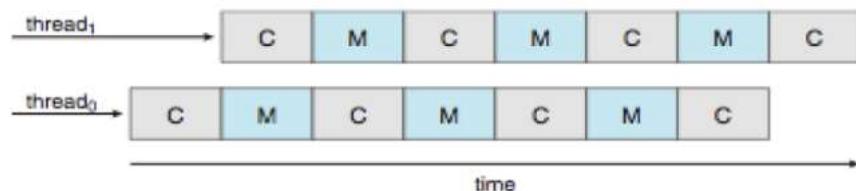
Obs: Ambas as técnicas podem ser usadas de forma conjunta em um Sistema Operacional;

## 5.4. Scheduling com Múltiplos Processadores

*Processadores com múltiplos núcleos dentro de um mesmo chip físico são denominados Processadores Multicore. Estes Processadores são geralmente mais rápidos e consomem menos energia.*

Quando um Processador acessa a memória, ele gasta um tempo significativo esperando até os dados ficarem disponíveis. Essa situação é conhecida como Queda de Memória (Memory Stall). O Processador pode gastar até 50% do tempo esperando que os dados fiquem disponíveis. Como resolver este problema?

- Atribuir 2 Threads de Hardware a cada núcleo; Assim, se um Thread for interrompido enquanto espera a memória, o núcleo poderá passar para outro Thread (Hyper-Threading);
- Portanto, em um sistema dual-core e dual-thread, existem quatro processadores lógicos na perspectiva do SO;



*M - Memory Stall;  
C - Computação;*

## 5.5. Avaliação de Algoritmos

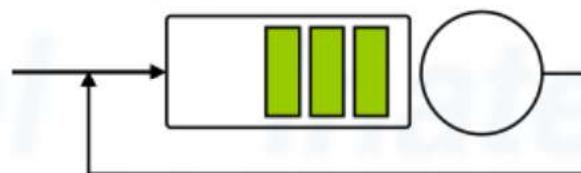
Como avaliar cada um dos Algoritmos que vimos anteriormente? Podemos utilizar de diferentes modelagens para testá-los.

### *1. Modelagem Determinística*

Determinar o comportamento e desempenho dos algoritmos segundo uma **carga de processos pré-definida**; É o modelo que temos usado em sala de aula;

### *2. Modelos de Enfileiramento (Queueing Model)*

Determinam-se ou estimam-se as **distribuições de probabilidade** dos CPU-bursts e dos tempos de chegada dos processos na Ready Queue; **Utiliza-se da teoria de filas**.



## 5.5. Avaliação de Algoritmos

### 3. Simulações

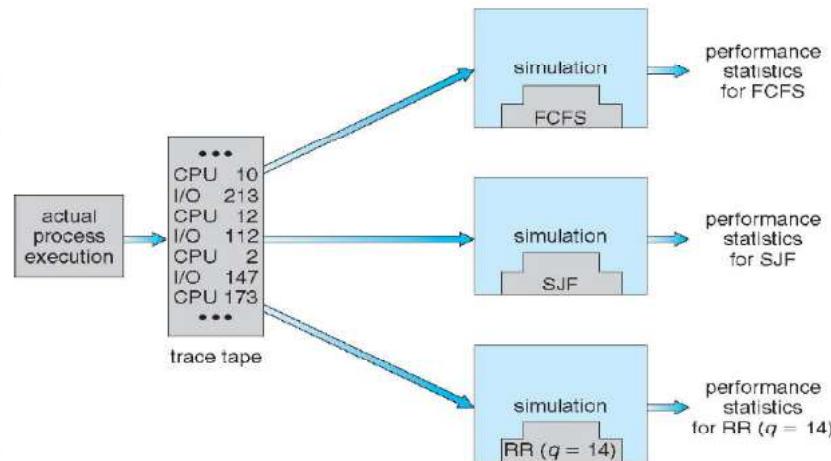
Envolve a programação (simulação) de um sistema computacional. Processos, CPU-bursts e tempo de chegada dos processos são gerados aleatoriamente pelo **simulador utilizando do Modelo de Enfileiramento**.

### 4. Implementação

**Codificação real** do Algoritmo de Scheduling **no SO** para a verificação de seu comportamento. É o método mais eficiente.

*Ex:*

Simulações.



# 5.5. Avaliação de Algoritmos

## TRABALHO PRÁTICO 2

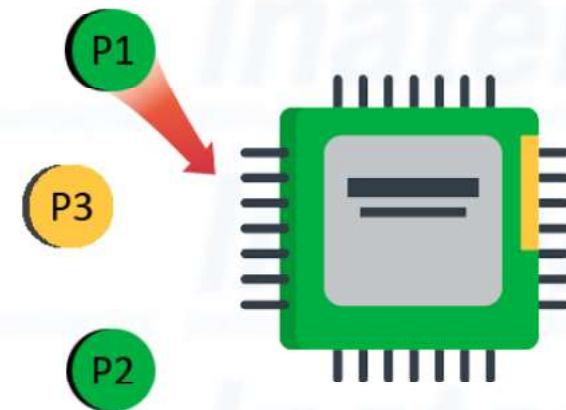
Agora que você já conhece os conceitos fundamentais de escalonamento de CPU, crie um pequeno simulador em Java ou na linguagem da sua escolha capaz de SIMULAR A FILA DE PRONTOS (READY QUEUE) de um Sistema Operacional.

O Simulador deve possibilitar no mínimo:

- *Escalonar processos com um Algoritmo PREEMPTIVO;*
- *Escalonar processos com um Algoritmo NÃO-PREEMPTIVO;*

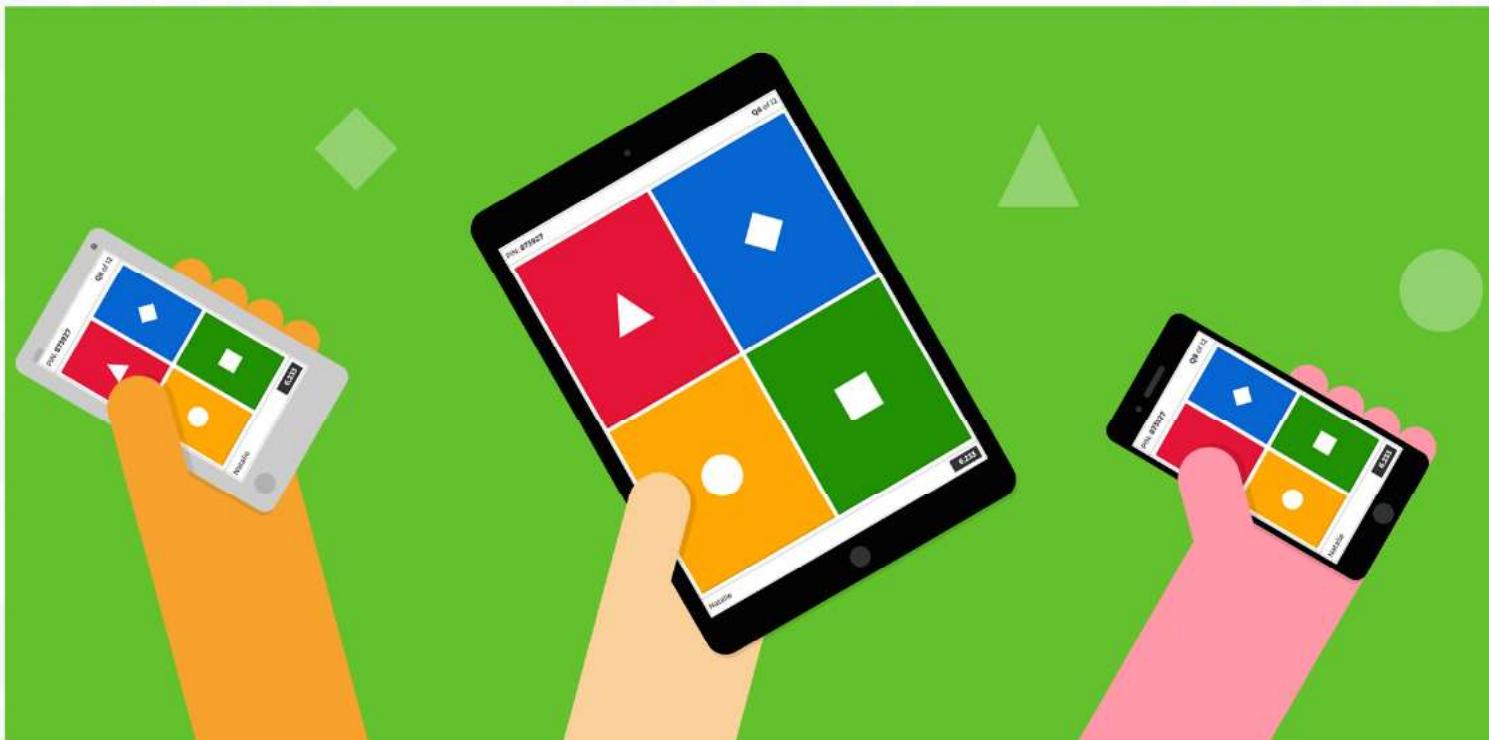
### IMPORTANTE:

- Use de seus conhecimentos e criatividade :)
- O Trabalho pode ser individual ou em dupla;
- Fique atento à data de apresentação proposta pelo professor.



HORA DO

# Kahoot!



ACESSE:

<https://kahoot.it>

**FIM  
DO  
CAPÍTULO 5**



**EXERCÍCIOS**