

## Aula 6

### Professores:

Felipe M. G. França  
Valmir C. Barbosa

### Conteúdo:

#### Escalonamento ou agendamento

- Introdução
- Algoritmos mais importantes

# Introdução

- ➡ Precisamos definir como escolhemos um processo para ser executado no processador, e por quanto tempo ele executa.
- ➡ O **escalonador** ou **agendador** é o responsável por comutar o uso do processador entre os processos do sistema.
- ➡ A administração executada pelo escalonador ao escolher um novo processo é chamada de **comutação de processo** ou de **contexto**.
- ➡ O algoritmo usado pelo escalonador para escolher os processos é chamado de algoritmo de **escalonamento** ou de **agendamento**.
- ➡ Um processo, em geral, alterna entre **rajadas de processamento** e de **espera por E/S**:
  - Em uma rajada de processamento, o processo está executando alguma computação no processador.
  - Em uma espera por E/S, o processo está esperando pelo término de uma operação de E/S iniciada anteriormente.

# Introdução

- ▶ Podemos classificar os processos em dois tipos, de acordo com seu comportamento:
  - **CPU-bound**: passam a maior parte do tempo computando.
  - **I/O-bound**: passam a maior parte do tempo esperando por E/S.



- ▶ O escalonamento eficiente dos processos I/O-bound é importante:
  - Com o aumento da velocidade de processamento, os processos CPU-bound tendem a ser processos I/O-bound.
  - Se o escalonador der prioridade aos processos IO-bound, o uso do processador e dos dispositivos de E/S será maximizado.

# Algoritmos de escalonamento

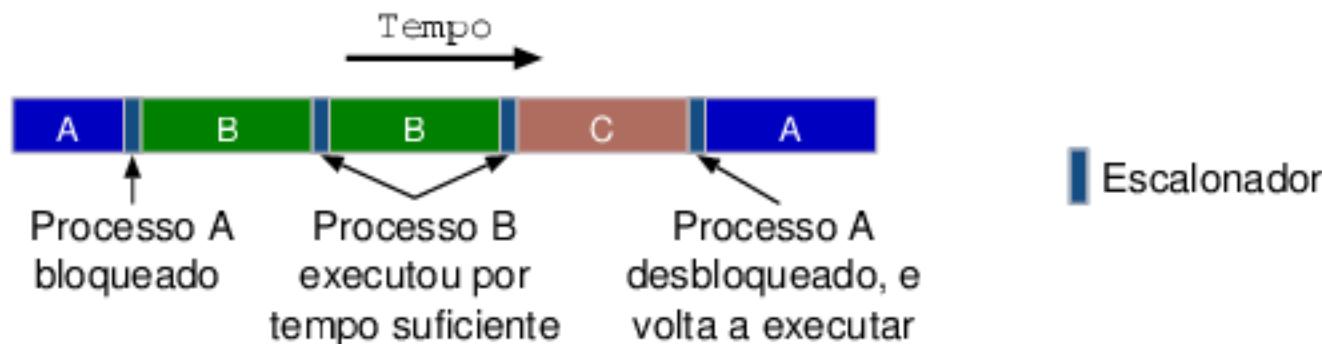
- ➡ Um bom algoritmo para o escalonamento dos processos deveria satisfazer os seguintes critérios:
  - **Imparcialidade:** cada processo deve ser capaz de executar no processador por um intervalo de tempo justo e adequado.
  - **Eficiência:** maximizar o uso do processador, de preferência mantendo-o ocupado por 100% do tempo (ou seja, não ocioso).
  - **Tempo de resposta:** minimizar o tempo de resposta para os processos dos usuários de um sistema interativo.
  - **Turnaround:** minimizar o tempo entre a submissão e a saída do trabalho de um usuário em um sistema em lote.
  - **Throughput:** maximizar o número de trabalhos processados por um sistema em lote em uma hora.
- ➡ Como alguns dos critérios acima são contraditórios, não existe um algoritmo que satisfaça todos os critérios.

# Algoritmos de escalonamento

- ➡ O escalonador deverá escolher um novo processo se um dos seguintes eventos ocorrerem ao processo em execução:
  - O processo terminou a sua execução.
  - O processo foi bloqueado, esperando por um evento externo.
- ➡ O escalonador pode também escolher um novo processo se um dos seguintes eventos ocorrerem:
  - Um novo processo é criado.
  - Uma interrupção é gerada por um dispositivo de E/S.
  - Uma interrupção é gerada pelo temporizador da placa-mãe.
- ➡ Quando um processo é criado, chamar o escalonador permite a reavaliação das prioridades dos processos.
- ➡ Se uma interrupção foi gerada por um dispositivo, então, em geral, um processo bloqueado pode finalmente voltar a executar.

# Algoritmos de escalonamento

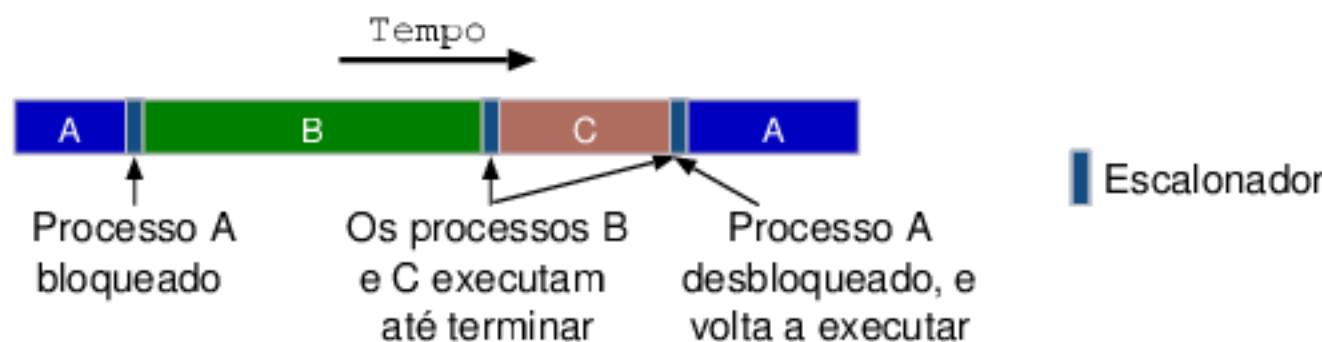
- ➡ Se o hardware possui um temporizador que gera periodicamente interrupções:
  - Sempre que uma interrupção ocorre, é verificado se o processo já executou por muito tempo no processador.
  - Se o processo já executou por muito tempo, então um novo processo é escolhido, pelo escalonador, para executar.
  - Neste caso, o algoritmo do escalonador é **preemptivo**.



O algoritmo do escalonador é chamado a cada interrupção gerada pelo temporizador, e poderá, dependendo do algoritmo, e do processo em execução, manter este processo, ou escolher um novo processo.

# Algoritmos de escalonamento

- ➡ Se não existe um temporizador no hardware, ou se o algoritmo para o escalonamento não usa este temporizador:
  - Um processo pode executar no processador por um intervalo arbitário de tempo.
  - O escalonador somente será chamado quando o processo executando no processador terminar, ou for bloqueado.
  - Neste caso, o algoritmo do escalonador é **não-preemptivo**.



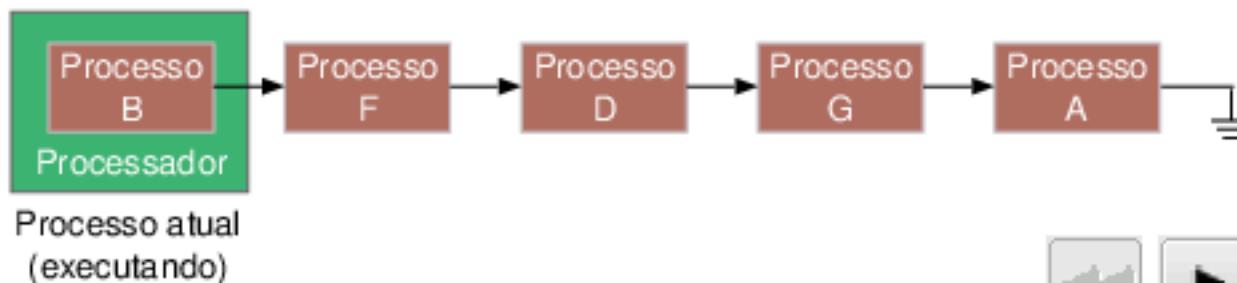
O algoritmo do escalonador é chamado somente quando o processo que está executando no processador ou é bloqueado pela espera de algum evento externo, ou termina de executar.

# Algoritmos de escalonamento

- Vamos estudar os seguintes algoritmos para o escalonador, cujos tipos são indicados entre parênteses:
- Escalonamento round robin (preemptivo).
  - Escalonamento por prioridade (preemptivo).
  - Trabalho mais curto primeiro (não-preemptivo).
  - Escalonamento por sorteio (preemptivo).
  - Escalonamento de dois níveis (preemptivo).

# Escalonamento round robin

- ➡ Cada processo pode executar no processador durante um certo intervalo de tempo, chamado de **quantum**.
- ➡ O escalonador será chamado se:
  - O processo já executou no processador durante o intervalo definido pelo quantum.
  - O processo foi bloqueado esperando pela ocorrência de algum evento externo.
- ➡ O escalonador deverá manter uma lista de processos prontos:
  - O primeiro processo da lista é o que está executando.
  - Um processo suspenso irá para o final da lista.

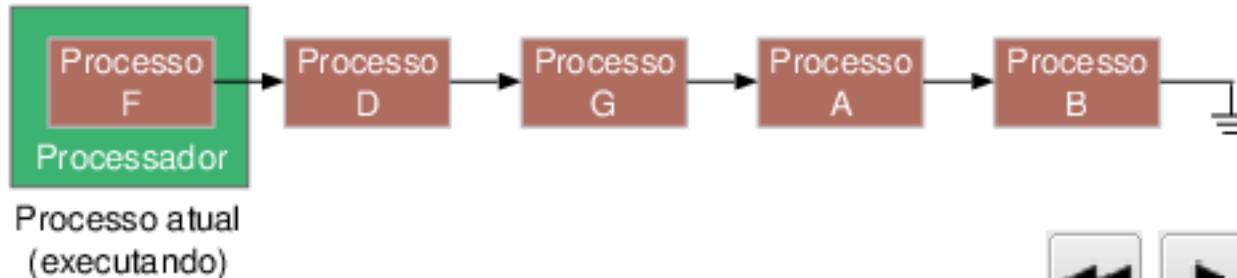


O processo B está agora executando no processador. Os processos F (próximo a executar), D, G, e A estão no estado pronto.



# Escalonamento round robin

- ➡ Cada processo pode executar no processador durante um certo intervalo de tempo, chamado de **quantum**.
- ➡ O escalonador será chamado se:
  - O processo já executou no processador durante o intervalo definido pelo quantum.
  - O processo foi bloqueado esperando pela ocorrência de algum evento externo.
- ➡ O escalonador deverá manter uma lista de processos prontos:
  - O primeiro processo da lista é o que está executando.
  - Um processo suspenso irá para o final da lista.

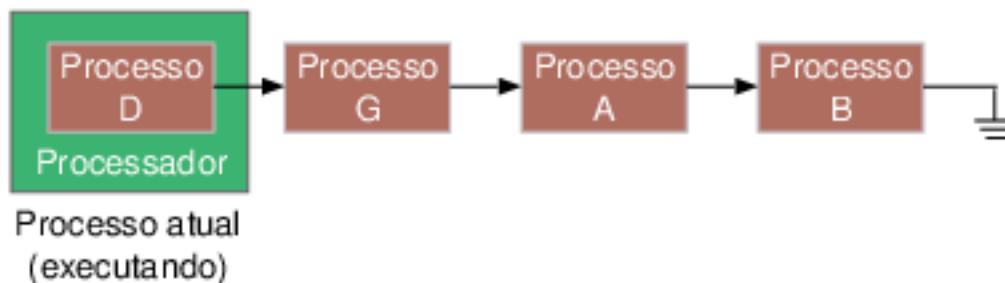


O processo B executou por todo o seu quantum e foi suspenso, e o processo F, o próximo da lista, foi o escolhido para executar.



# Escalonamento round robin

- ➡ Cada processo pode executar no processador durante um certo intervalo de tempo, chamado de **quantum**.
- ➡ O escalonador será chamado se:
  - O processo já executou no processador durante o intervalo definido pelo quantum.
  - O processo foi bloqueado esperando pela ocorrência de algum evento externo.
- ➡ O escalonador deverá manter uma lista de processos prontos:
  - O primeiro processo da lista é o que está executando.
  - Um processo suspenso irá para o final da lista.

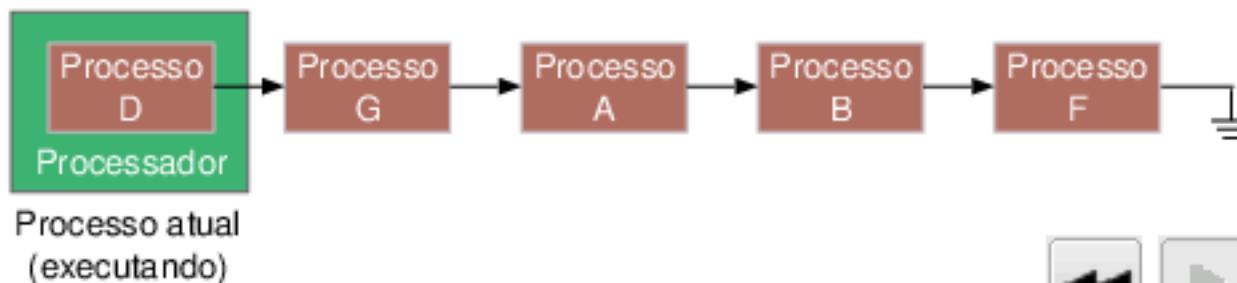


O processo F bloqueou devido a uma operação de E/S. Então, o escalonador é acionado, e escolhe o processo D para executar.



# Escalonamento round robin

- ➡ Cada processo pode executar no processador durante um certo intervalo de tempo, chamado de **quantum**.
- ➡ O escalonador será chamado se:
  - O processo já executou no processador durante o intervalo definido pelo quantum.
  - O processo foi bloqueado esperando pela ocorrência de algum evento externo.
- ➡ O escalonador deverá manter uma lista de processos prontos:
  - O primeiro processo da lista é o que está executando.
  - Um processo suspenso irá para o final da lista.



O processo F voltou ao estado pronto porque a operação de E/S terminou. Então, o processo será colocado no final da lista.



## Escalonamento round robin

- ➡ A grande questão do algoritmo é a escolha de um tamanho adequado para o quantum de um processo:
  - Se usarmos um quantum pequeno, a porcentagem de uso do processador para comutar os processos será significativa.
  - Se usarmos um quantum muito grande, teremos um péssimo tempo de resposta para os usuários interativos.
  - Um quantum de 100ms em geral é o mais adequado.
- ➡ O problema do algoritmo é o de que os processos são igualmente importantes:
  - Todos os processos possuem o mesmo quantum.
  - Alguns processos devem ser mais importantes do que outros, como os processos que gerenciam os dispositivos.

# Escalonamento por prioridade

- ➡ A cada processo do sistema é atribuída uma **prioridade**, e o escalonador escolhe o processo com a maior prioridade.
- ➡ Agora podemos, usando as prioridades, classificar os processos de acordo com o seu grau de importância.
- ➡ Como garantir o uso do processador por todos os processos?
- ➡ O escalonador pode reduzir, a cada interrupção, a prioridade do processo que está executando no processador:
  - A comutação de processo ocorre quando a prioridade deste processo não for mais a maior, ou quando este for bloqueado.



Processo atual  
(executando)



Processo F	7
---------------	---

Processo D	9
---------------	---

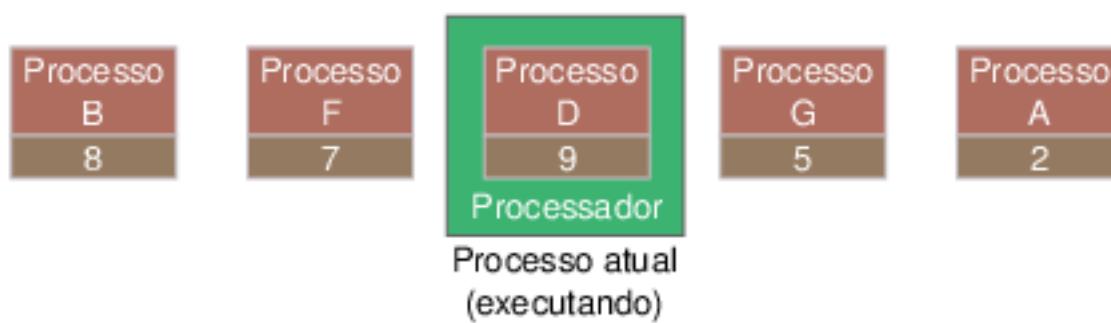
Processo G	5
---------------	---

Processo A	2
---------------	---

O processo B está atualmente executando no processador. Mais quatro processos estão no estado pronto: F, D, G e A. As prioridades são dadas abaixo dos processos.

# Escalonamento por prioridade

- A cada processo do sistema é atribuída uma **prioridade**, e o escalonador escolhe o processo com a maior prioridade.
- Agora podemos, usando as prioridades, classificar os processos de acordo com o seu grau de importância.
- Como garantir o uso do processador por todos os processos?
- O escalonador pode reduzir, a cada interrupção, a prioridade do processo que está executando no processador:
  - A comutação de processo ocorre quando a prioridade deste processo não for mais a maior, ou quando este for bloqueado.

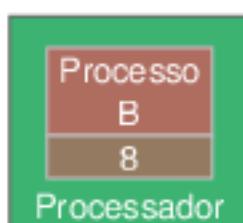


Depois de duas interrupções do temporizador, o processo B terá prioridade 8, e como o processo D possui prioridade 9, este será escolhido para executar no processador.



# Escalonamento por prioridade

- A cada processo do sistema é atribuída uma **prioridade**, e o escalonador escolhe o processo com a maior prioridade.
- Agora podemos, usando as prioridades, classificar os processos de acordo com o seu grau de importância.
- Como garantir o uso do processador por todos os processos?
- O escalonador pode reduzir, a cada interrupção, a prioridade do processo que está executando no processador:
  - A comutação de processo ocorre quando a prioridade deste processo não for mais a maior, ou quando este for bloqueado.



Processo atual  
(executando)



Processo F
7

Processo G
5

Processo A
2

Suponha que o processo D bloqueou por executar uma operação de E/S. Então o processo B é o escolhido para executar no processador.

# Escalonamento por prioridade

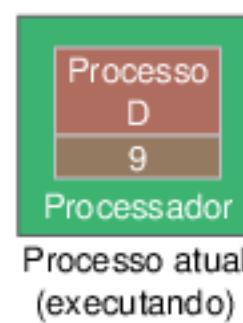
- A cada processo do sistema é atribuída uma **prioridade**, e o escalonador escolhe o processo com a maior prioridade.
- Agora podemos, usando as prioridades, classificar os processos de acordo com o seu grau de importância.
- Como garantir o uso do processador por todos os processos?
- O escalonador pode reduzir, a cada interrupção, a prioridade do processo que está executando no processador:
  - A comutação de processo ocorre quando a prioridade deste processo não for mais a maior, ou quando este for bloqueado.

Processo B
8

Processo F
7

Processo G
5

Processo A
2



O processo B continua a sua execução. Suponha que o processo D foi desbloqueado. Então, o processo B será comutado com o D, pois a prioridade de D ainda é 9.

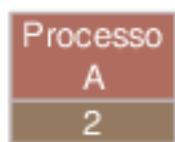
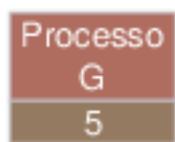
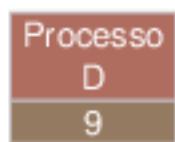
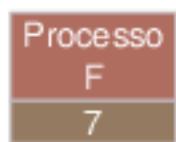


# Escalonamento por prioridade

- Uma outra possibilidade é a do escalonador associar a cada um dos processos um quantum:
- O processo executa somente durante o seu quantum.
  - Quando o processo usar o seu quantum, o próximo processo com a maior prioridade é escolhido para executar.



Processo atual  
(executando)



Um sistema com cinco processos no estado pronto. O processo B está atualmente executando no processador. As prioridades são dadas abaixo dos processos.



- A prioridade de um processo pode ser atribuída estaticamente, ou seja, no momento da criação deste processo.

# Escalonamento por prioridade

→ Uma outra possibilidade é a do escalonador associar a cada um dos processos um quantum:

- O processo executa somente durante o seu quantum.
- Quando o processo usar o seu quantum, o próximo processo com a maior prioridade é escolhido para executar.

Processo B
10
Processador

Processo atual  
(executando)

Processo F
7

Processo D
9

Processo G
5

Processo A
2

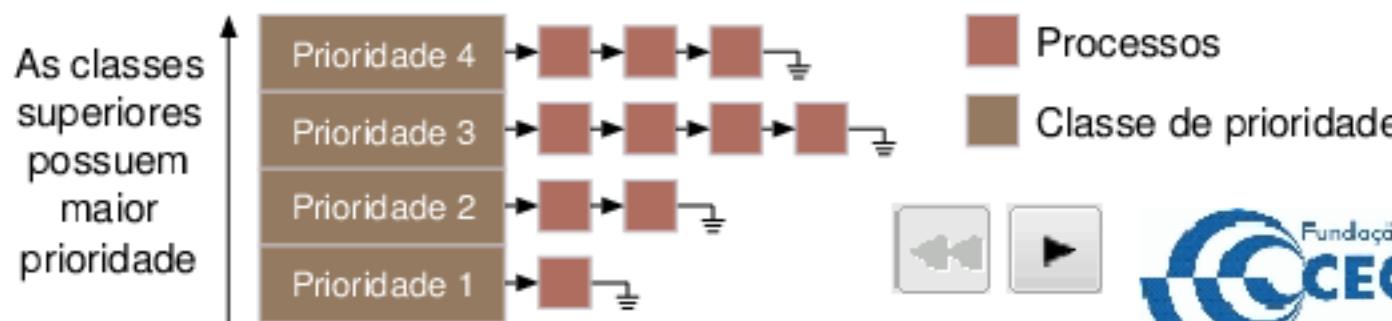
O algoritmo escolhe um novo processo a cada nova interrupção do temporizador, de acordo com a prioridade. Depois de executar todos os processos, o processo com a maior prioridade é novamente executado.



→ A prioridade de um processo pode ser atribuída estaticamente, ou seja, no momento da criação deste processo.

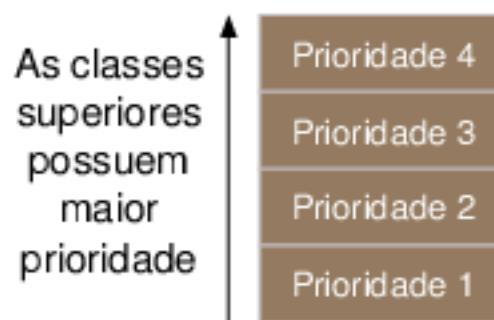
# Escalonamento por prioridade

- ➡ As prioridades também podem ser atribuídas dinamicamente:
  - Pode-se reduzir ou aumentar a prioridade de um processo.
  - Pode-se melhorar o uso do processador e dos dispositivos:
    - Associar a um processo a prioridade  $1/f$ , onde  $f$  é a fração usada do quantum antes de o processo ser bloqueado.
    - Dá maior prioridade aos processos IO-bound: as operações de E/S são executadas em paralelo com as computações.
- ➡ Podemos organizar os processos em **classes de prioridades**:
  - O algoritmo round robin é usado para escolher os processos da mesma classe.
  - Os processos de uma classe são executados após todos os processos das classes mais prioritárias terem sido executados.



# Escalonamento por prioridade

- ➡ As prioridades também podem ser atribuídas dinamicamente:
  - Pode-se reduzir ou aumentar a prioridade de um processo.
  - Pode-se melhorar o uso do processador e dos dispositivos:
    - Associar a um processo a prioridade  $1/f$ , onde  $f$  é a fração usada do quantum antes de o processo ser bloqueado.
    - Dá maior prioridade aos processos IO-bound: as operações de E/S são executadas em paralelo com as computações.
- ➡ Podemos organizar os processos em **classes de prioridades**:
  - O algoritmo round robin é usado para escolher os processos da mesma classe.
  - Os processos de uma classe são executados após todos os processos das classes mais prioritárias terem sido executados.



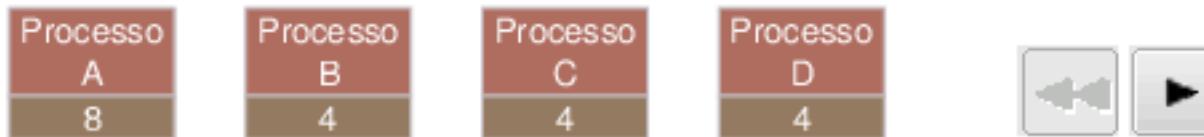
Processos

Classe de prioridade



## Trabalho mais curto primeiro

- Desenvolvido para os sistemas em lote, em que sabemos, a priori, o tempo de execução dos trabalhos executados pelos processos.
- O escalonador sempre escolhe o processo da lista de processos prontos que possui o menor tempo de execução:



Quatro trabalhos submetidos a um sistema de lote. Os trabalhos foram submetidos na ordem dada acima. Os tempos de execução são dados abaixo dos processos que executam estes trabalhos.

- O tempo médio de retorno (da submissão à geração da saída) dos trabalhos é minimizado por este algoritmo.

## Trabalho mais curto primeiro

- Desenvolvido para os sistemas em lote, em que sabemos, a priori, o tempo de execução dos trabalhos executados pelos processos.
- O escalonador sempre escolhe o processo da lista de processos prontos que possui o menor tempo de execução:

Processo A	Processo B	Processo C	Processo D
8	4	4	4



Executando os trabalhos na ordem de submissão, teremos os seguintes tempos de retorno para os processos: 8 para o A, 12 para o B, 16 para o C, e 20 para o D. O tempo médio será de 14.

- O tempo médio de retorno (da submissão à geração da saída) dos trabalhos é minimizado por este algoritmo.

## Trabalho mais curto primeiro

- Desenvolvido para os sistemas em lote, em que sabemos, a priori, o tempo de execução dos trabalhos executados pelos processos.
- O escalonador sempre escolhe o processo da lista de processos prontos que possui o menor tempo de execução:

Processo A	Processo B	Processo C	Processo D
8	4	4	4



Executando os trabalhos de acordo com o menor tempo de execução, teremos os seguintes tempos de retorno para os processos: 4 para o B, 8 para o C, 12 para o D, e 20 para o A. O tempo médio será de 11.

- O tempo médio de retorno (da submissão à geração da saída) dos trabalhos é minimizado por este algoritmo.

## Trabalho mais curto primeiro

→ O tempo é minimizado somente quando temos todos os trabalhos:

Processo A	Processo B
2	4

Suponha os trabalhos, executados pelos processos A e B, foram submetidos no tempo 0. Abaixo dos processos estão os tempos de execução dos trabalhos.



→ O algoritmo também pode ser aplicado aos processos interativos que continuamente esperam por um comando, e o executam:

- Como não sabemos os tempos de execução dos comandos, podemos usar o seguinte algoritmo de **envelhecimento**:
- Suponha que o tempo estimado para o comando atual foi  $T_0$ , e que o tempo real de execução foi  $T_1$ .
- Então, o tempo estimado para o próximo comando será de  $\alpha T_0 + (1-\alpha) T_1$ , onde  $\alpha$  é, em geral, 1/2.

## Trabalho mais curto primeiro

→ O tempo é minimizado somente quando temos todos os trabalhos:

Processo A
2

Processo B
4

Processo C
1

Processo D
1

Processo E
1

Se mais três trabalhos forem submetidos no tempo 3, teremos os seguintes tempos de retorno: 2 para o A, 6 para o B, 4 para o C, 5 para o D, e 6 para o E. A média será 4,6.



→ O algoritmo também pode ser aplicado aos processos interativos que continuamente esperam por um comando, e o executam:

- Como não sabemos os tempos de execução dos comandos, podemos usar o seguinte algoritmo de **envelhecimento**:
- Suponha que o tempo estimado para o comando atual foi  $T_0$ , e que o tempo real de execução foi  $T_1$ .
- Então, o tempo estimado para o próximo comando será de  $\alpha T_0 + (1-\alpha) T_1$ , onde  $\alpha$  é, em geral, 1/2.

# Trabalho mais curto primeiro

→ O tempo é minimizado somente quando temos todos os trabalhos:

Processo A	Processo B	Processo C	Processo D	Processo E
2	4	1	1	1

Se os trabalhos C, D e E chegaram no tempo 3, e se executarmos na ordem B, C, D, E e A, os tempos de retorno serão: 9 para o A, 4 para o B, 2 para o C, 3 para o D, e 4 para o E, com uma média de 4,4.

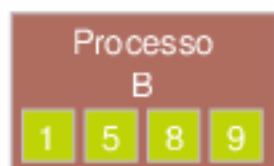


→ O algoritmo também pode ser aplicado aos processos interativos que continuamente esperam por um comando, e o executam:

- Como não sabemos os tempos de execução dos comandos, podemos usar o seguinte algoritmo de **envelhecimento**:
- Suponha que o tempo estimado para o comando atual foi  $T_0$ , e que o tempo real de execução foi  $T_1$ .
- Então, o tempo estimado para o próximo comando será de  $\alpha T_0 + (1-\alpha) T_1$ , onde  $\alpha$  é, em geral, 1/2.

## Escalonamento por sorteio

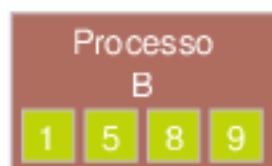
- A idéia é associar a cada um dos recursos do sistema, como o tempo do processador, bilhetes de loteria de processos.
- Quando aplicada ao algoritmo de escalonamento:
  - Existirá um conjunto de bilhetes associado ao processador.
  - Cada processo possuirá um subconjunto deste conjunto.
  - O escalonador escolherá um dos bilhetes aleatoriamente, e o processo que possuir o bilhete será o escolhido para executar.
  - A probabilidade de um processo ser escolhido será proporcional à fração do número total de bilhetes que ele possui.



Neste exemplo, existem 10 bilhetes associados ao processador, sendo que estes foram distribuídos como dado acima. A probabilidade de os processos serem escolhidos é a seguinte: A - 30%, B - 40%, C - 20%, e D - 10%.

## Escalonamento por sorteio

- A idéia é associar a cada um dos recursos do sistema, como o tempo do processador, bilhetes de loteria de processos.
- Quando aplicada ao algoritmo de escalonamento:
  - Existirá um conjunto de bilhetes associado ao processador.
  - Cada processo possuirá um subconjunto deste conjunto.
  - O escalonador escolherá um dos bilhetes aleatoriamente, e o processo que possuir o bilhete será o escolhido para executar.
  - A probabilidade de um processo ser escolhido será proporcional à fração do número total de bilhetes que ele possui.



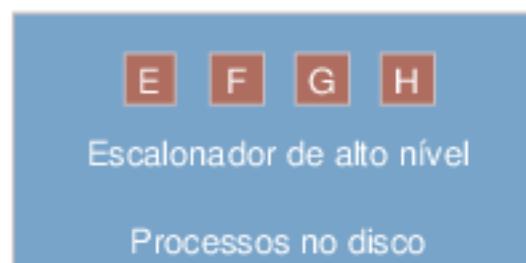
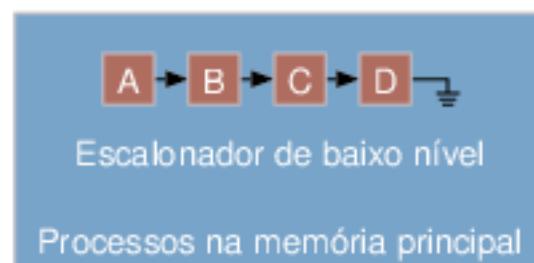
Supoha que o escalonador escolheu o bilhete 9. Entao, o processo B será executado. Se nos próximos sorteios forem escolhidos os números 2, 8, 6, 3, e 4, os seguintes processos executarão, em ordem: A, B, D, C, A.

## Escalonamento por sorteio

- ➡ Se um processo possui uma fração  $f$  dos bilhetes, então o processo usará uma fração  $f$  do tempo total do processador.
- ➡ Ao contrário do escalonamento por prioridades, em que não é claro o significado da prioridade, aqui o significado da fração  $f$  é claro.
- ➡ Este algoritmo possui as seguintes propriedades:
  - Um processo sempre terá uma probabilidade de ser escolhido proporcional ao número de bilhetes que possui.
  - Dois ou mais processos que executam cooperativamente uma tarefa podem trocar os seus bilhetes entre si:
    - Um processo cliente pode passar os seus bilhetes a um processo servidor, que os retornará após executar o serviço.
  - Pode ser usado para resolver problemas que seriam difíceis de serem resolvidos por outros algoritmos:
    - Os processos de um servidor de vídeo, que devem dividir o processador de acordo com a taxa de quadros usada.

# Escalonamento de dois níveis

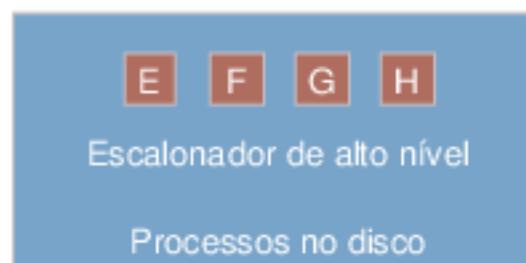
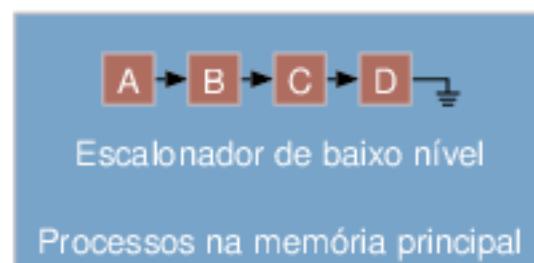
- ➔ Usado se a memória não pode armazenar todos os processos, ou seja, alguns dos processos devem ser armazenados no disco.
- ➔ Como o tempo de comutação do disco é grande, devemos então fazer um escalonamento em dois níveis, com dois escalonadores:
  - O **escalonador de baixo nível** escolhe o próximo processo a ser executado, dentre os processos que estão na memória.
  - O **escalonador de alto nível** escolhe um dos processos do disco para ser carregado na memória, no lugar de um outro processo.



No exemplo acima, temos oito processos: A, B, C, D, E, F, G, e H. Os processos A, B, C e D estão na memória, e os processos E, F, G e H estão no disco.

# Escalonamento de dois níveis

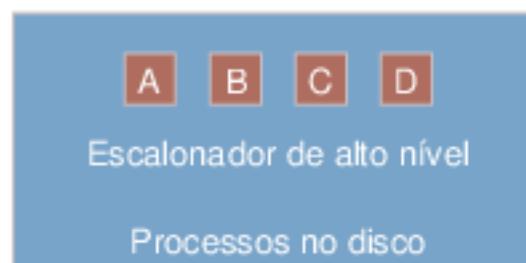
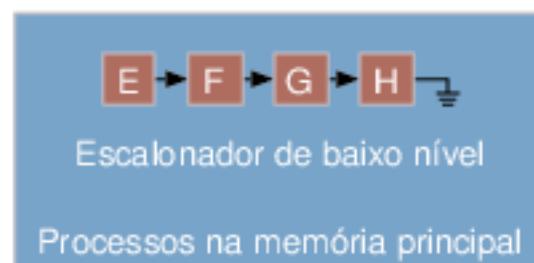
- ➔ Usado se a memória não pode armazenar todos os processos, ou seja, alguns dos processos devem ser armazenados no disco.
- ➔ Como o tempo de comutação do disco é grande, devemos então fazer um escalonamento em dois níveis, com dois escalonadores:
  - O **escalonador de baixo nível** escolhe o próximo processo a ser executado, dentre os processos que estão na memória.
  - O **escalonador de alto nível** escolhe um dos processos do disco para ser carregado na memória, no lugar de um outro processo.



O Escalonador de baixo nível escolhe somente dentre os processos que estão na memória, no caso, A, B, C, ou D. Este escalonador pode usar qualquer algoritmo que vimos, como, por exemplo, o algoritmo de escalonamento round robin.

# Escalonamento de dois níveis

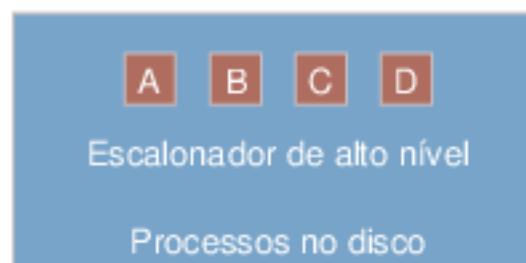
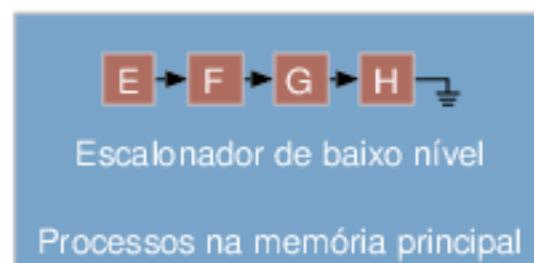
- ➔ Usado se a memória não pode armazenar todos os processos, ou seja, alguns dos processos devem ser armazenados no disco.
- ➔ Como o tempo de comutação do disco é grande, devemos então fazer um escalonamento em dois níveis, com dois escalonadores:
  - O **escalonador de baixo nível** escolhe o próximo processo a ser executado, dentre os processos que estão na memória.
  - O **escalonador de alto nível** escolhe um dos processos do disco para ser carregado na memória, no lugar de um outro processo.



O Escalonador de alto nível escolhe, de tempos em tempos, alguns processos do disco e os troca com outros processos que estão na memória. No exemplo, todos os processos do disco são trocados com todos os processos da memória.

# Escalonamento de dois níveis

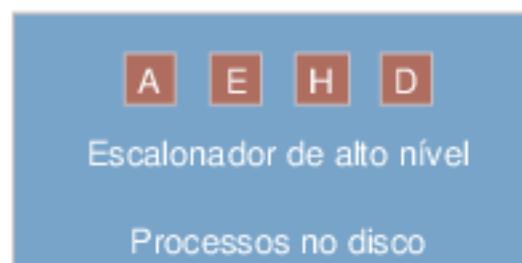
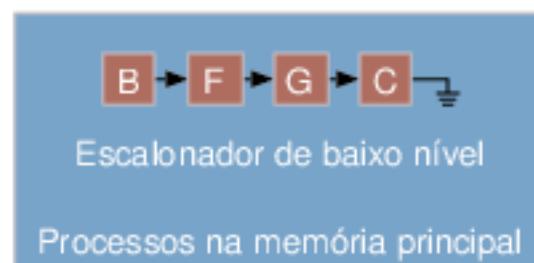
- ➔ Usado se a memória não pode armazenar todos os processos, ou seja, alguns dos processos devem ser armazenados no disco.
- ➔ Como o tempo de comutação do disco é grande, devemos então fazer um escalonamento em dois níveis, com dois escalonadores:
  - O **escalonador de baixo nível** escolhe o próximo processo a ser executado, dentre os processos que estão na memória.
  - O **escalonador de alto nível** escolhe um dos processos do disco para ser carregado na memória, no lugar de um outro processo.



O escalonador de baixo nível agora escolhe, usando o algoritmo de escalonamento round robin, dentre os processos E, F, G, ou H.

# Escalonamento de dois níveis

- ➔ Usado se a memória não pode armazenar todos os processos, ou seja, alguns dos processos devem ser armazenados no disco.
- ➔ Como o tempo de comutação do disco é grande, devemos então fazer um escalonamento em dois níveis, com dois escalonadores:
  - O **escalonador de baixo nível** escolhe o próximo processo a ser executado, dentre os processos que estão na memória.
  - O **escalonador de alto nível** escolhe um dos processos do disco para ser carregado na memória, no lugar de um outro processo.



O escalonador de alto nível agora escolhe trocar o processo B pelo E, e o processo C pelo H. Depois disso, o escalonador de baixo nível escolherá entre os processos F, G, B e C.

## Escalonamento de dois níveis

- ➡ O escalonador de nível mais baixo pode usar qualquer um dos algoritmos que já estudamos.
- ➡ O escalonador de nível mais alto, que pode também usar qualquer um destes algoritmos, deve levar em conta o seguinte:
  - O tempo decorrido desde que um processo foi armazenado no disco ou na memória.
  - O tempo de processamento usado por um processo.
  - O tamanho do processo.
  - A prioridade do processo.
- ➡ Podemos usar algoritmos de escalonamento diferentes para cada um dos escalonadores.