

Escalonamento

Decidindo qual processo vai executar

Escalonamento de processos

- Quando dois ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro

- A parte do sistema operacional responsável por essa decisão é chamada **escalador**, e o algoritmo usado para tal é chamado de **algoritmo de escalonamento**

Comportamento escalonamento-processo

Processos I/O-bound têm “prioridade” no escalonamento

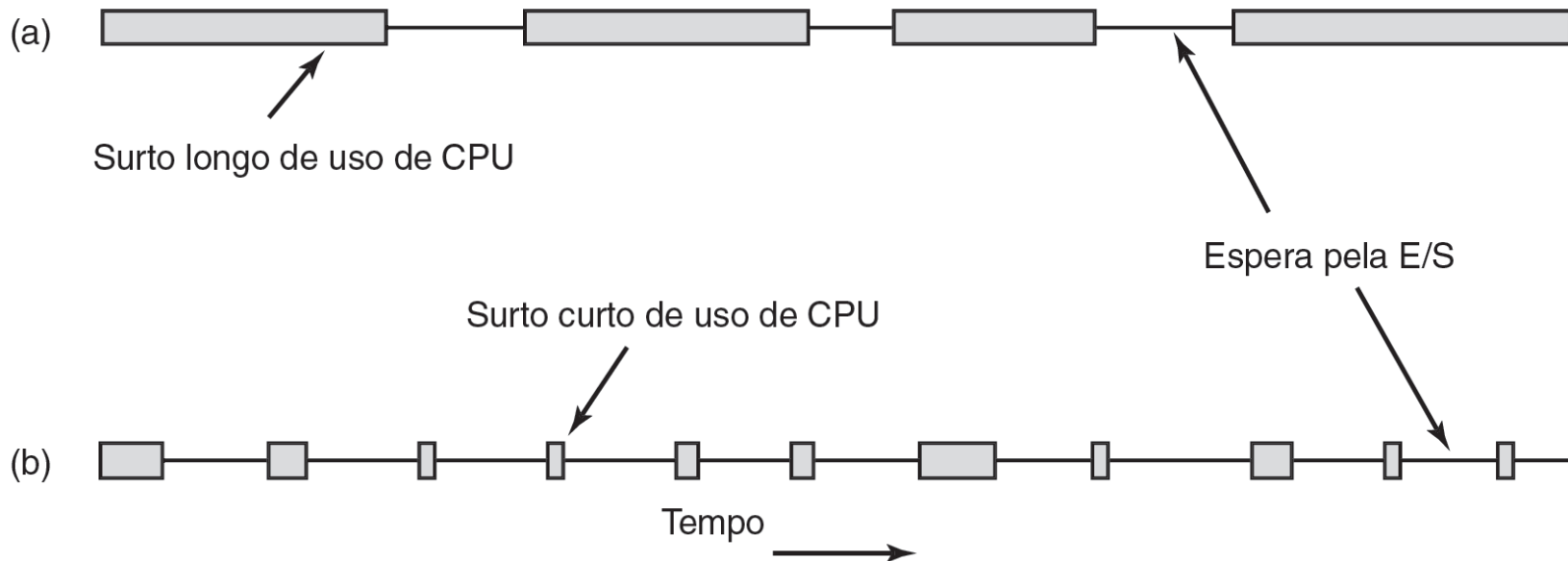
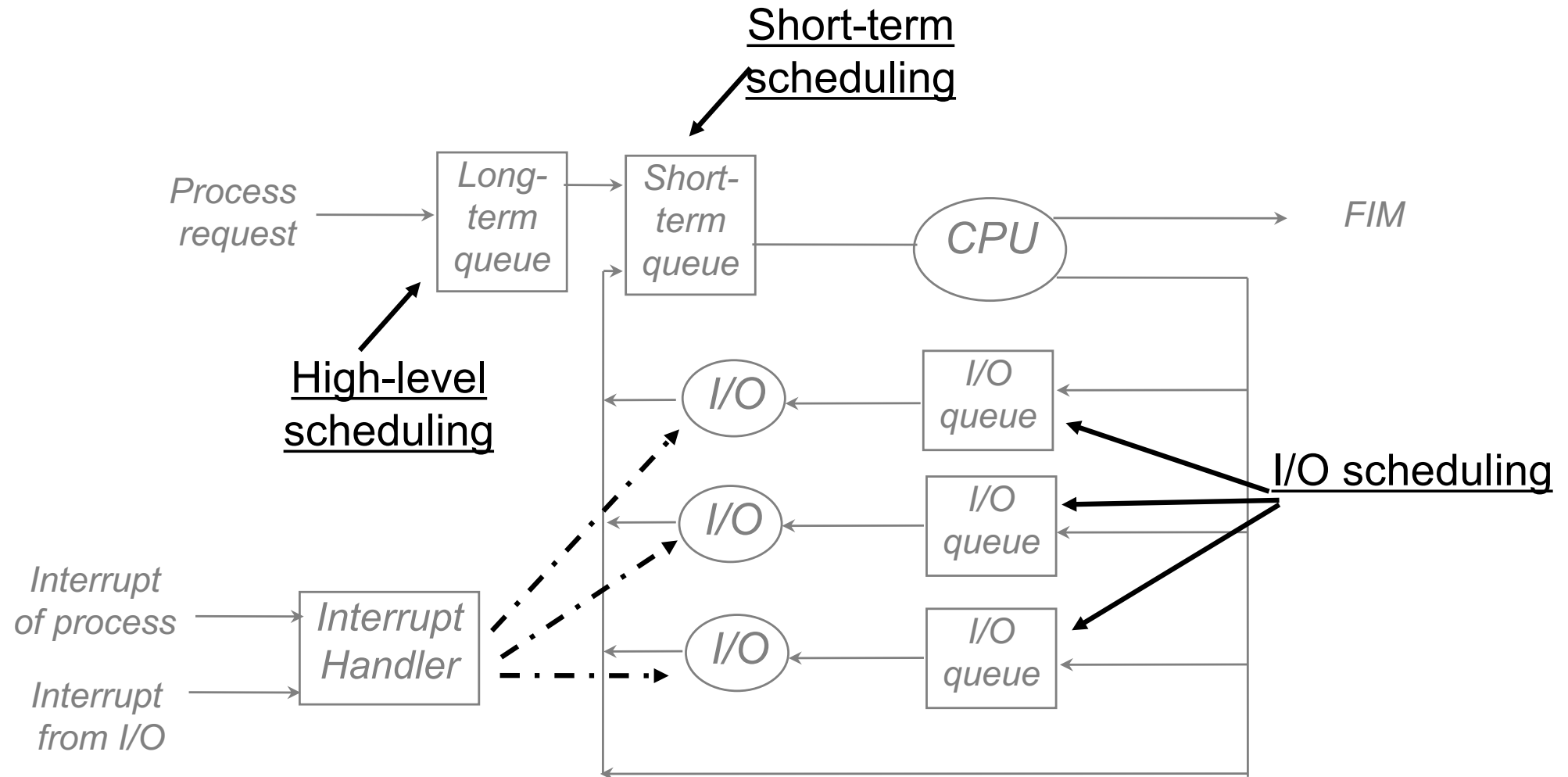


Figura 2.31 Usos de surtos de CPU se alternam com períodos de espera por E/S. (a) Um processo orientado à CPU. (b) Um processo orientado à E/S.

Filas de Escalonamento

- High-level
 - Decide quantos programas são admitidos no sistema
 - Aloca memória e cria um processo
 - Controla a *long-term queue*
- Short-term
 - Decide qual processo deve ser executado
 - Controla a *short-term queue*
- I/O
 - Decide qual processo (com I/O) pendente deve ser tratado pelo dispositivo de I/O
 - Controla a *I/O queue*

Filas de Escalonamento



Categorias de Algoritmos de Escalonamento

- Em lote (batch)
 - Interativo
 - Tempo-real
-
- **batch**: *a quantity or consignment of goods produced at one time.*
Ex. Uma 'fornada' de pão 😊

Objetivos dos algoritmos de escalonamento

Todos os sistemas

Justiça — dar a cada processo uma porção justa da CPU

Aplicação da política — verificar se a política estabelecida é cumprida

Equilíbrio — manter ocupadas todas as partes do sistema

Sistemas em lote

Vazão (*throughput*) — maximizar o número de tarefas por hora

Tempo de retorno — minimizar o tempo entre a submissão e o término

Utilização de CPU — manter a CPU ocupada o tempo todo

Sistemas interativos

Tempo de resposta — responder rapidamente às requisições

Proporcionalidade — satisfazer às expectativas dos usuários

Sistemas de tempo real

Cumprimento dos prazos — evitar a perda de dados

Previsibilidade — evitar a degradação da qualidade em sistemas multimídia

Tabela 2.8 Alguns objetivos do algoritmo de escalonamento sob diferentes circunstâncias.

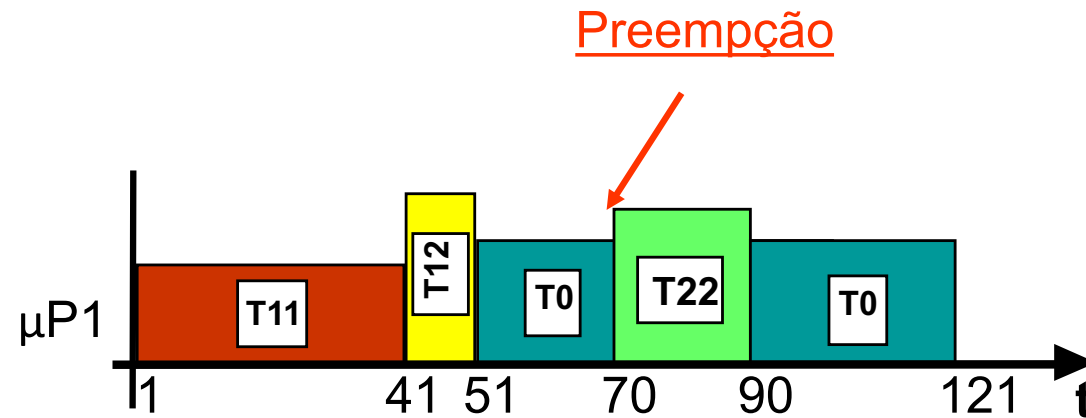
Tipos de Escalonamento

- Mecanismos de Escalonamento
 - Preemptivo x Não-preemptivo
- Políticas de Escalonamento
 - Round-Robin
 - FIFO (First-In First-Out)
 - Híbridos
 - Partições de Lote (Batch)
 - MFQ - Multiple Feedback Queue
 - SJF - Shortest Job First
 - SRJN - Shortest Remaining Job Next

Diz-se que um algoritmo/sistema operacional é **preemptivo** quando um processo entra na CPU e o mesmo pode ser retirado (da CPU) antes do término da sua execução

Escalonamento Preemptivo

- Permite a suspensão temporária de processos
- *Quantum* ou *time-slice*: período de tempo durante o qual um processo usa o processador a cada vez



Problema das trocas de processos

- Mudar de um processo para outro requer um certo tempo para a administração — salvar e carregar registradores e mapas de memória, atualizar tabelas e listas do SO, etc
- Isto se chama **troca de contexto**
- Suponha que esta troca dure 5 ms
- Suponha também que o *quantum* está ajustado em 20 ms
- Com esses parâmetros, após fazer 20 ms de trabalho útil, a CPU terá que gastar 5 ms com troca de contexto. Assim, 20% do tempo de CPU (5 ms a cada 25 ms) é gasto com o *overhead* administrativo...

Solução?

- Para melhorar a eficiência da CPU, poderíamos ajustar o *quantum* para 500 ms
 - Agora o tempo gasto com troca de contexto é menos do que 1% - “desprezível”...
- Considere o que aconteceria se dez usuários apertassem a tecla <ENTER> exatamente ao mesmo tempo, disparando cada um processo:
 - Dez processos serão colocados na lista de processo aptos a executar
 - Se a CPU estiver ociosa, o primeiro começará imediatamente, o segundo não começará cerca de ½ segundo depois, e assim por diante
 - O “azarado” do último processo somente começará a executar 5 segundos depois do usuário ter apertado <ENTER>, isto se todos os outros processos tiverem utilizado todo o seu *quantum*
 - Muitos usuários vão achar que o tempo de resposta de 5 segundos para um comando simples é “muita” coisa

“Moral da estória”

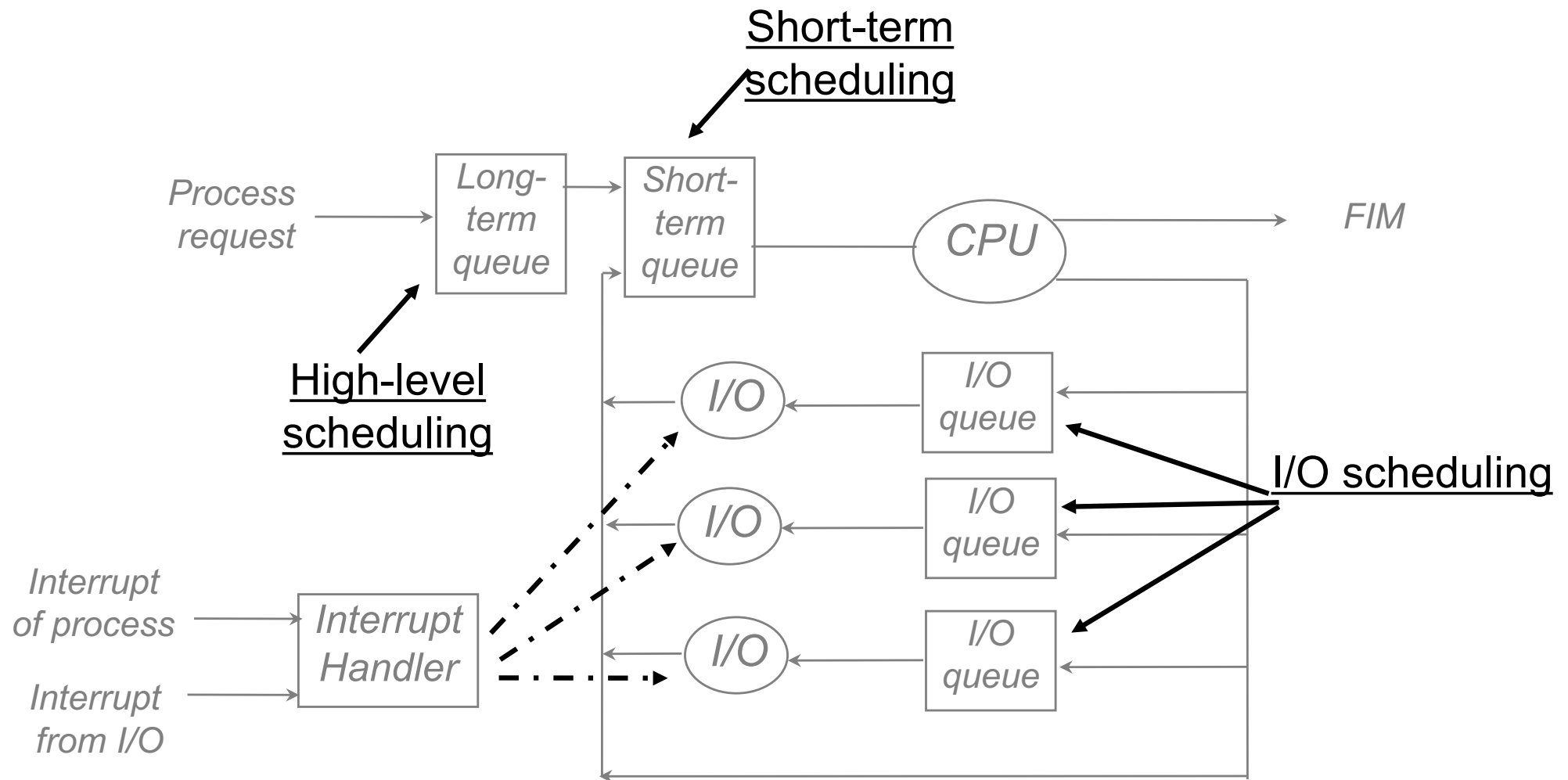
- Ajustar um *quantum* muito pequeno causa muitas trocas de contexto e diminui a eficiência da CPU, ...
- mas ajustá-lo para um valor muito alto causa um tempo de resposta inaceitável para pequenas tarefas interativas

Quantum grande:
Diminui número de mudanças de contexto e *overhead*
do S.O., mas...
Ruim para processos interativos

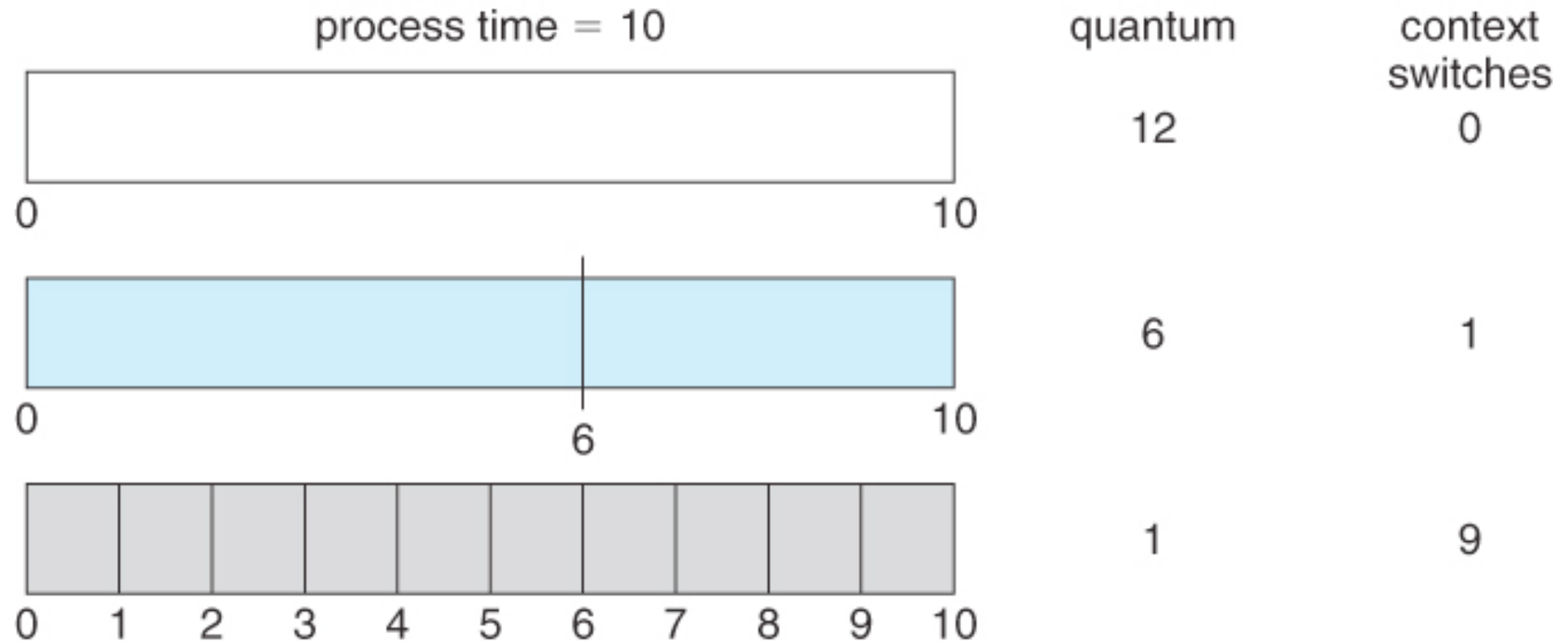
Escalonamento de Processos

Algoritmos (cont.)

Relembrando: Filas de Escalonamento



Relembrando: *Quantum* e Troca de Contexto



Categorias de Algoritmos de Escalonamento (Agendamento)

- Em lote (batch)
- Interativo
- Tempo-real
- Híbrido

Earliest Deadline First (EDF)

- Preemptivo
- Considera o momento em que a resposta deve ser entregue
- Processo se torna mais prioritário quanto mais próximo do deadline
- Algoritmo complexo

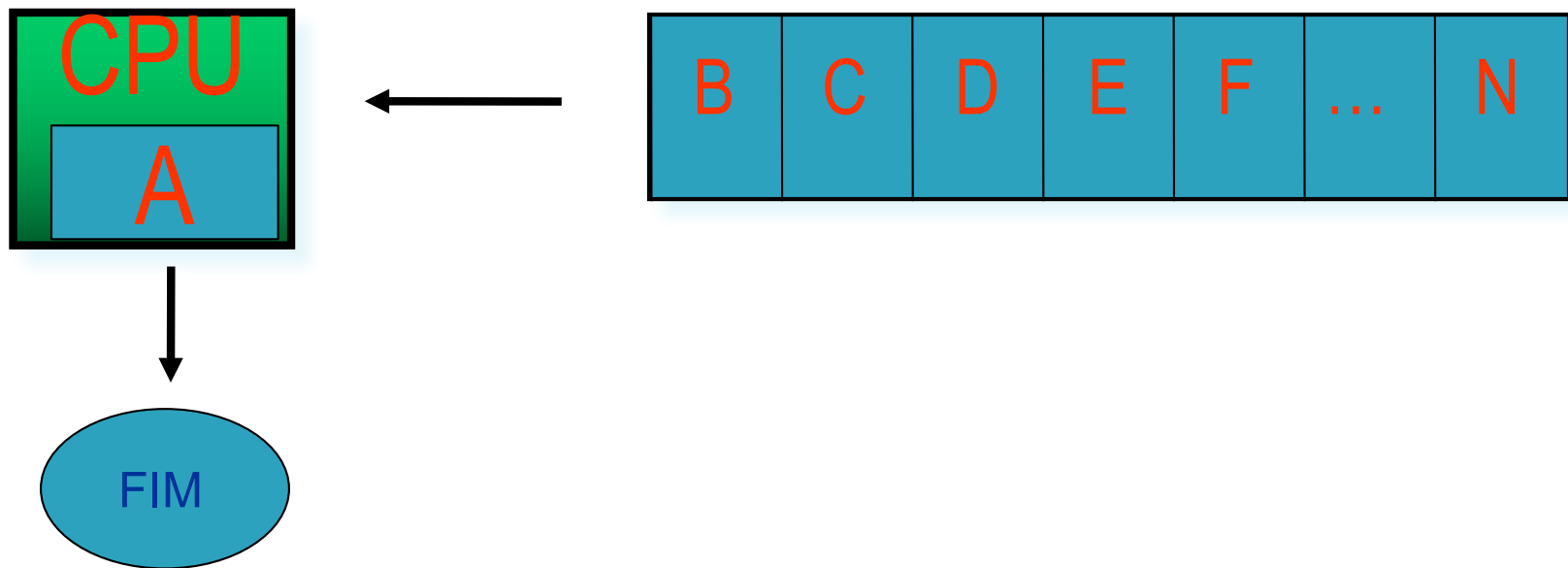
Priorização dinâmica

Escalonamento em Sistemas em Lote

- First-come first-served (ou FIFO)
- Shortest Job First (*job mais curto primeiro*) – SJF
- Shortest Remaining Time/Job First/Next – SRTF/SRJN

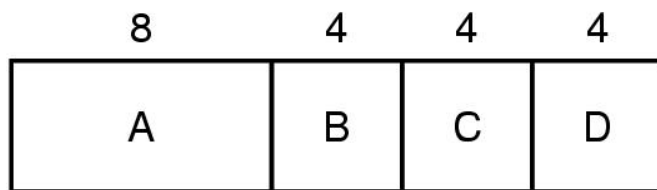
First-In First-Out (FIFO)

- Uso de uma lista de processos sem prioridade
- Escalonamento não-preemptivo
- Simples e justo

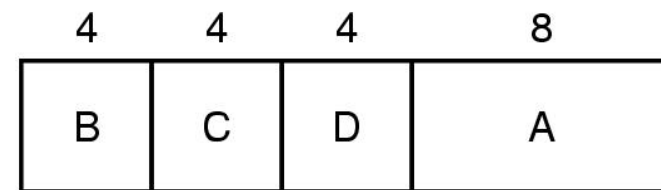


Escalonamentos baseados no tempo de execução

- Shortest Job First (não-preemptivo)
- Shortest Remaining Job Next (preemptivo)
- Melhora o tempo de resposta
- Não é justo: pode causar estagnação (*starvation*)
 - Pode ser resolvida alterando a prioridade dinamicamente



(a)



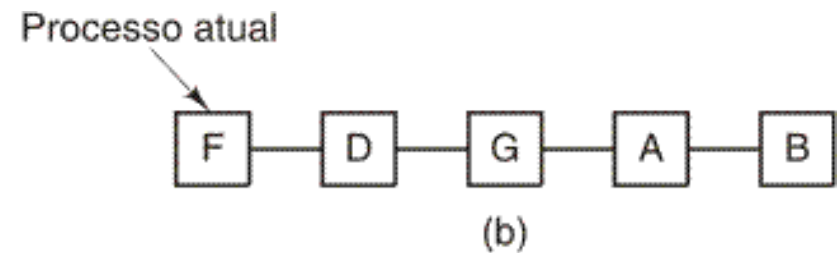
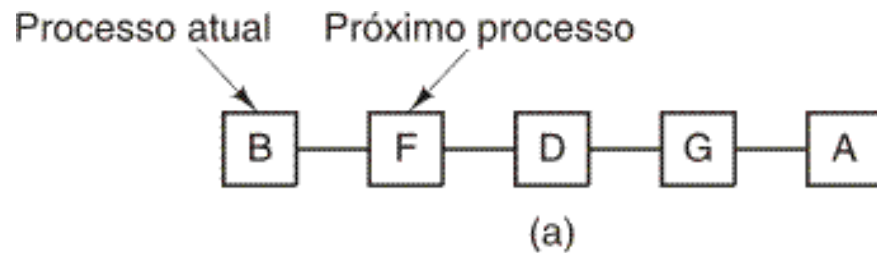
(b)

Exemplo de escalonamento *job mais curto primeiro* (**Shortest Job First – SJF**)

Escalonamento em Sistemas Interativos

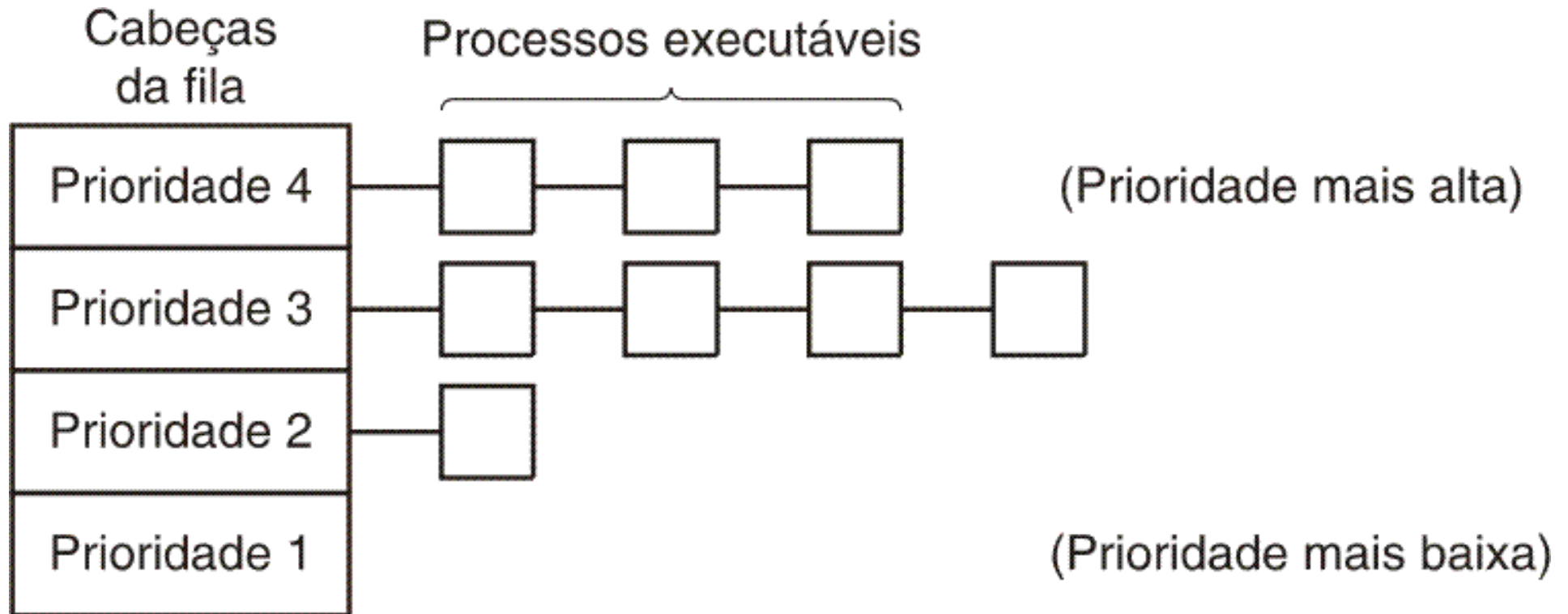
- Round-robin
- Priority
- Multiple queues
- Shortest process next
- Guaranteed scheduling
- Lottery scheduling
- Fair-share scheduling
- Chaveamento circular
- Escalonamento por prioridades
- Filas múltiplas
- Processo mais curto é o próximo
- Escalonamento garantido
- Escalonamento por loteria
- Escalonamento por fração justa

Escalonamento em Sistemas Interativos (1)



- Escalonamento por chaveamento/alternância circular (*round-robin*)
 - a) lista de processos executáveis
 - b) lista de processos executáveis depois que B usou todo o seu quantum

Escalonamento em Sistemas Interativos (2)



Um algoritmo de escalonamento com quatro classes de **prioridade**

Escalonamento Híbrido

Multiple Feedback Queue

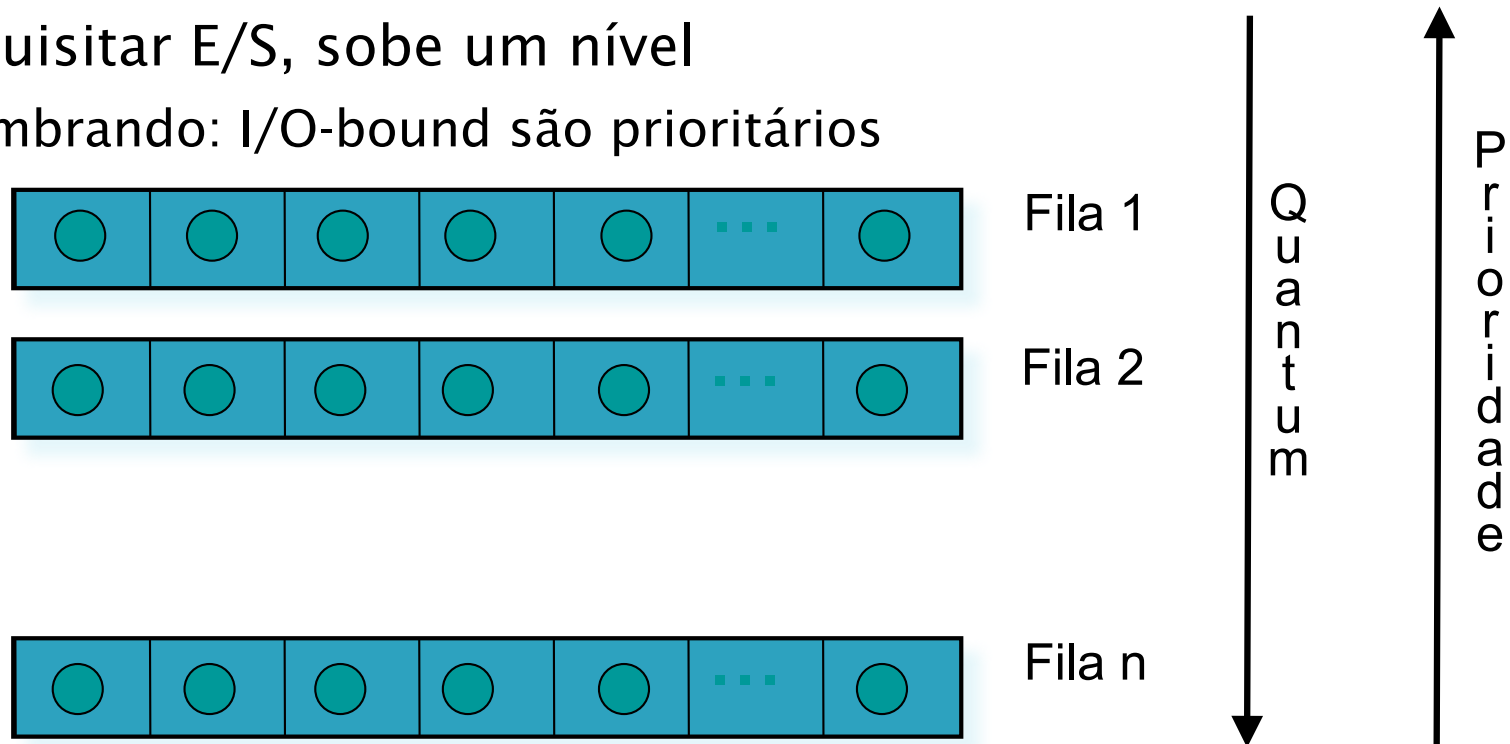
- Como saber *a priori* se o processo é CPU-bound ou I/O-bound?
- MFQ usa abordagem de prioridades dinâmicas
- Adaptação baseada no comportamento de cada processo
- Usado no VAX / VMS...

Segue...

Escalonamentos Híbridos

Multiple Feedback Queue

- Novos processos entram na primeira fila (prioridade mais alta)
- Se acabar o quantum, desce um nível
- Se requisitar E/S, sobe um nível
 - Lembrando: I/O-bound são prioritários



Fair-share Scheduling

Uso da CPU distribuído igualmente entre usuários ou grupos de usuários (e não entre processos)

Exemplo 1

- **Usuários** A, B, C, D, 1 processo cada:
 - $100\% \text{ CPU} / 4 = 25\%$ para cada usuário
 - Se B iniciar um segundo processo:
 - $25\% \text{ B} / 2 = 12,5\%$ para cada processo de B
 - Continua 25% para A, C, D
 - Se novo usuário E:
 - $100\% \text{ CPU} / 5 = 20\%$ para A, B, C, D, E

Fair-share Scheduling

Uso da CPU distribuído igualmente entre usuários ou grupos de usuários (e não entre processos)

Exemplo 2

- **Grupos** 1, 2, 3:
 - 100% CPU / 3 grupos = 33.3% por grupo
 - Grupo 1: (33.3% / 3 usuários) = 11.1% por usuário
 - Grupo 2: (33.3% / 2 usuários) = 16.7% por usuário
 - Grupo 3: (33.3% / 4 usuários) = 8.3% por usuário

Algoritmos de Escalonamento

Em lote

- ✓ First-come first-served (ou FIFO)
- ✓ Shortest Job First (job mais curto primeiro) – SJF
- ✓ Shortest Remaining Job Next – **SRJN**

Em sistemas de tempo real

- ✓ Earliest Deadline First (EDF)

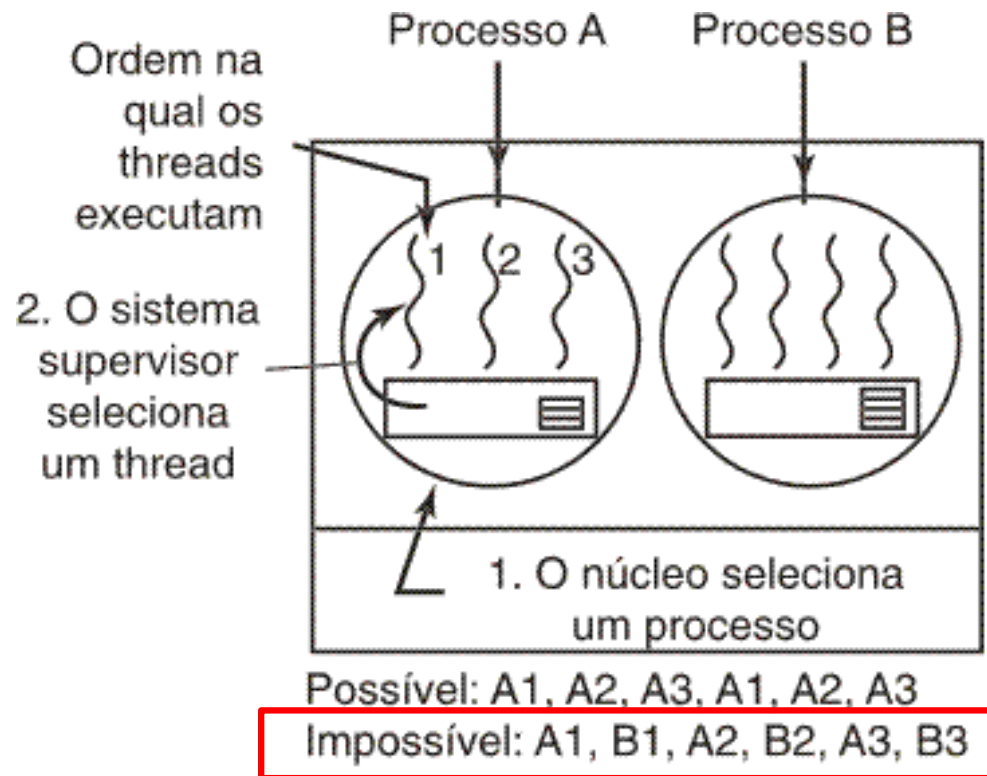
Em sistemas interativos

- ✓ Round-robin
- ✓ Priority
- ✓ Multiple queues (**híbrido**)
 - **Shortest process next**
 - Guaranteed scheduling
 - Lottery scheduling
- ✓ Fair-share scheduling

Algoritmos de Escalonamento

Round Robin	<ul style="list-style-type: none">• Quantum constante• Sem prioridade
Com Prioridade	<ul style="list-style-type: none">• Cada processo possui uma prioridade e o de maior prioridade executa primeiro• Para evitar que os processos de maior prioridade tomem conta do processador, a prioridade é decrementada
Menor Job Primeiro	<ul style="list-style-type: none">• Difícil estimar o tempo
Filas múltiplas	<ul style="list-style-type: none">• Criação de classes de Prioridades alocadas em diferentes filas
Garantido	<ul style="list-style-type: none">• Estimar (prometer) a um processo o tempo de sua execução e cumprir• Necessário conhecimento dos processos executando e a serem executados
Loteria	<ul style="list-style-type: none">• Distribuição de bilhetes que dão acesso à CPU• Lembra o escalonamento com prioridade, mas bilhetes são trocados entre processos

Escalonamento de Threads (1)

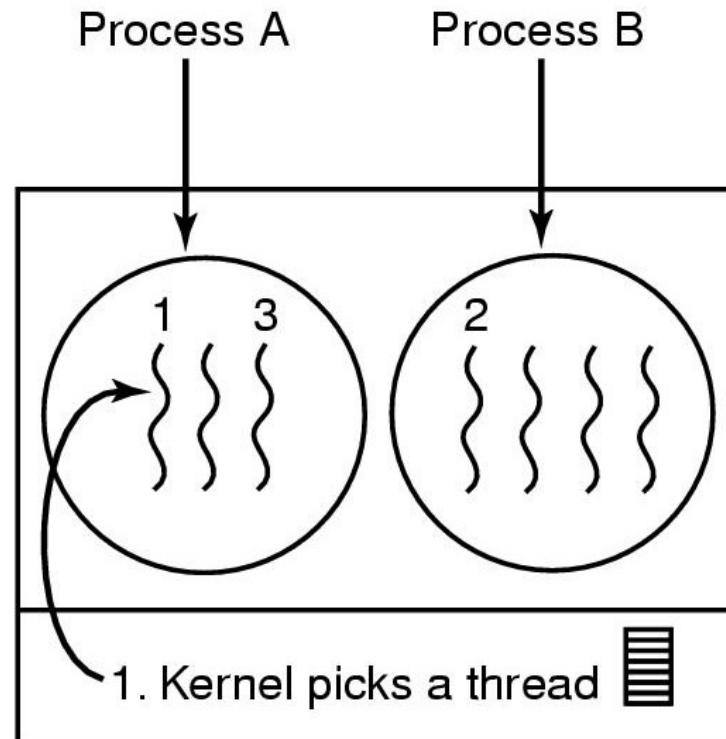


(a)

Possível escalonamento de **threads de usuário**

- processo com quantum de 50-mseg
- threads executam 5 mseg por surto de CPU

Escalonamento de Threads (2)



Possible: A1, A2, A3, A1, A2, A3

Also possible: A1, B1, A2, B2, A3, B3

Possível escalonamento de **threads de núcleo**

- processo com quantum de 50-mseg
- threads executam 5 mseg por surto de CPU

Conclusões

Como funcionam dois ou mais programas ao mesmo tempo?

- Conceitos
 - Processos x *Threads* (processos leves)
- Interrupção
 - Cooperação hardware-software
- Escalonamento
 - Tipos de processos
 - CPU-bound x I/O-bound
 - Lote (batch) x interativo
 - Filas de escalonamento
 - Long-term (admissão)
 - Short-term
 - I/O
- Escalonamento (cont.)
 - Objetivos
 - Justiça
 - Eficiência
 - Tempo de Resposta
 - Conceitos
 - Preempção
 - *Quantum (time-slice)*
 - Troca de contexto
 - Algoritmos
 - Propósito x Complexidade x Eficiência

✓ **Gerência
de Processos**