Universidade Federal do Ceará

Campus Quixadá

Sistemas Operacionais

ESCALONAMENTO

PROF. SIDARTHA CARVALHO

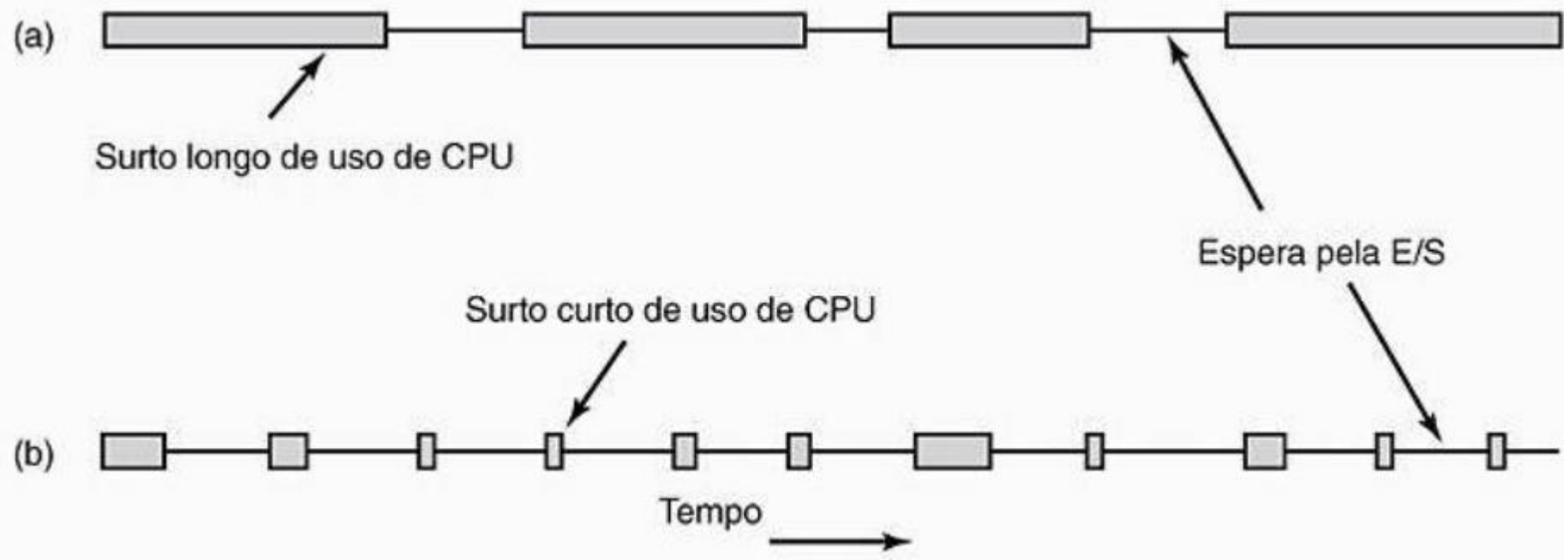
Escalonamento

Quando um computador é multiprogramado, muitas vezes ele possui variados processos que competem pela CPU ao mesmo tempo;

Essa situação ocorre sempre que dois ou mais processos estão simultaneamente no estado de pronto;

A parte do SO que faz a escolha de qual processo deve ser executado é chamado de **escalonador**, e o algoritmo que é usado para escolher o processo a ser executado é chamado de **algoritmo de escalonamento**;

Escalonamento



Escalonamento

Quando escalonar?

- Quando se cria um novo processo
- No término de um processo
- Quando um processo é bloqueado
- Quando um processo executa algum evento de Entrada e Saída (I/O)

Escalonamento

O algoritmo de escalonamento pode ser:

- **Não preemptivo:**
  - O processo executa até o fim, sem ser interrompido;
- **Preemptivo:**
  - O processo executa fatias de tempo (quantum) determinado pelo sistema operacional.

Para ambientes diferentes, são necessários diferentes algoritmos de escalonamento.

Categoriais de algoritmos de escalonamento

Três ambientes merecem distinção:

- **Lote**
  - Em sistemas em Lote, não há terminal com usuários esperando impacientemente por uma resposta rápida. Consequentemente, algoritmos com longo intervalo de tempo (ou que executam até terminar) para cada processo em geral são aceitáveis.
- **Interativo**
  - Em um ambiente com usuários interativos, a preempção é essencial para evitar que um processo se aposse da CPU e, com isso, negue serviço aos outros.
- **Tempo real**
  - Em sistemas com restrição de tempo real, a preempção é desnecessária algumas vezes, pois os processos sabem que não podem executar por longos períodos de tempo e em geral fazem seus trabalhos e bloqueiam rapidamente.

Categoriais de algoritmos de escalonamento

Um bom algoritmo de escalonamento é projetado de acordo com os critérios de utilização já mencionados

- Mas existem alguns critérios que são desejáveis para todos os casos
- Exemplo:
  - **Justiça:** Dar a cada processo uma porção justa da CPU
  - **Aplicação da política:** verificar se a política estabelecida é cumprida
  - **Equilíbrio:** manter ocupada todas as partes do sistema
    - CPU, E/S, etc...

Categoriais de algoritmos de escalonamento

Adicionalmente, alguns critérios se adequam aos seus ambientes de utilização

- **Sistemas em lote:**
  - **Vazão (Througput):**
    - Maximizar o número de tarefas/jobs por hora;
  - **Tempo de retorno:**
    - Minimizar o tempo entre a submissão e o término da execução do processo;
  - **Utilização da CPU:**
    - Manter a CPU ocupada todo o tempo;

Categoriais de algoritmos de escalonamento

Adicionalmente, alguns critérios se adequam aos seus ambientes de utilização

- **Sistemas interativos:**
  - **Tempo de resposta:**
    - Responder rapidamente às requisições
  - **Proporcionalidade:**
    - Satisfazer as perspectivas dos usuários

Categoriais de algoritmos de escalonamento

Adicionalmente, alguns critérios se adequam aos seus ambientes de utilização

- **Sistemas em tempo real:**
  - **Cumprimento dos prazos:**
    - Evitar a perda de dados;
  - **Previsibilidade:**
    - Evitar a degradação da qualidade em sistemas multimídias.

Finalizado: Introdução

Próxima: Algoritmos de Escalonamento

Seção 1

ALGORITMOS DE ESCALONAMENTO

Alguns algoritmos de escalonamento

- **Algoritmos não preemptivos (cooperativos)**
  - First-In First-Out (FIFO) ou First-Come First-Served (FCFS)
  - Shortest Job First (SJF)
- **Algoritmos preemptivos**
  - Round Robin (circular)
  - Baseado em prioridades
- Existem outros algoritmos de escalonamento
  - High Response Ratio Next (HRRN)
  - Shortest Remaining Time (SRT)
  - Escalonamento de frequência de CPU:
    - Ondemand e Interactive (Linux/Android)
  - etc...

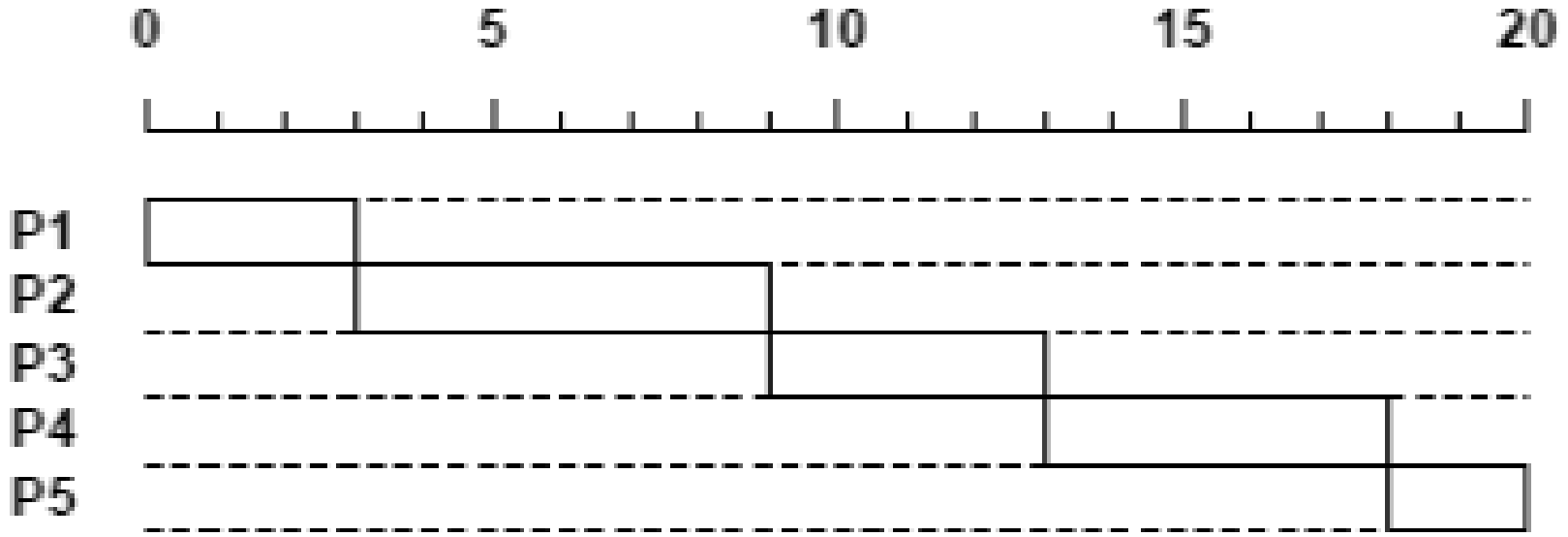
Primeiro a Chegar, Primeiro a ser Servido

First Come, First Served – FCFS ou First In, First Out (FIFO)

- Com esse algoritmo, a CPU é atribuída aos processos na ordem em que eles a requisitam.
  - Basicamente há uma fila única de processos prontos
- Quando o processo requisita a CPU ele é executado (dado que a CPU está ociosa).
  - Executa até que seja terminado.
  - Quando chega mais processos, estes são inseridos em uma fila.

FIFO - First In First Out

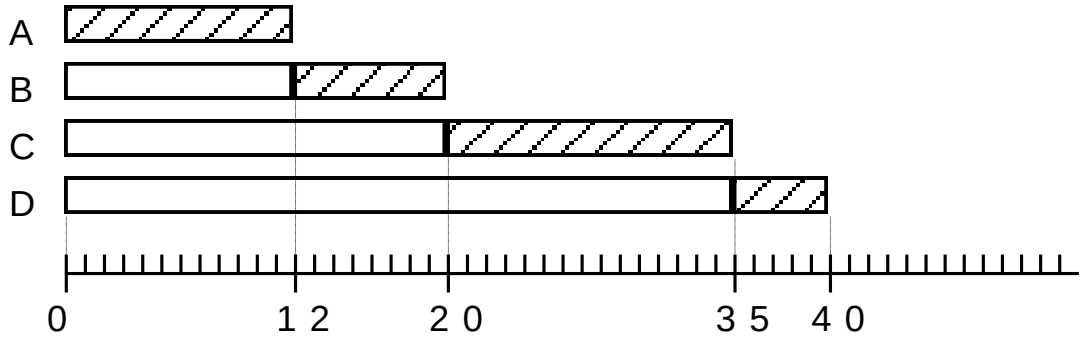
Processo	Tempo de chegada	Tempo de serviço
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



FIFO - First In First Out

- Desvantagem:
  - Prejudica processos *I/O bound*
- Tempo médio de espera na fila de execução:
  - Ordem A-B-C-D = (0 + 12 + 20 + 35) / 4 = 16.75 u.t.
  - Ordem D-A-B-C = (0 + 5 + 17 + 25) / 4 = 11.7 u.t.

Processo	Tempo
A	12
B	8
C	15
D	5



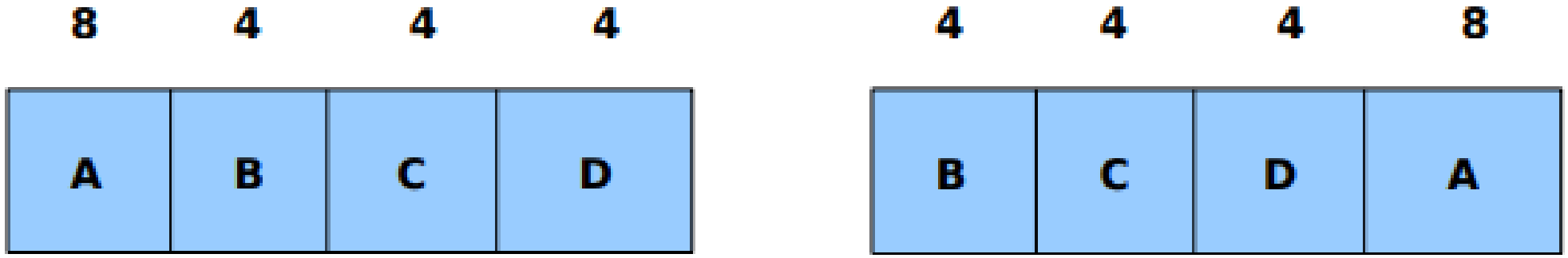
FIFO - First In First Out

Qual a vantagem e a desvantagem do uso deste algoritmo?

- **Vantagem:**
  - É de fácil entendimento e de fácil implementação.
- **Desvantagem:**
  - Quando tem-se programas orientados a CPU e programas orientados a E/S
  - Um programa orientado a CPU executa 1 segundo, e um programa orientado a E/S que executa pouco tempo de CPU, mas necessita de executar mil leituras no disco antes de terminar

Shortest Job First (SJF)

Quando várias tarefas igualmente importantes estiverem postados na fila de entrada à espera de serem iniciados, o escalonador escolhe a tarefa mais curta primeiro.



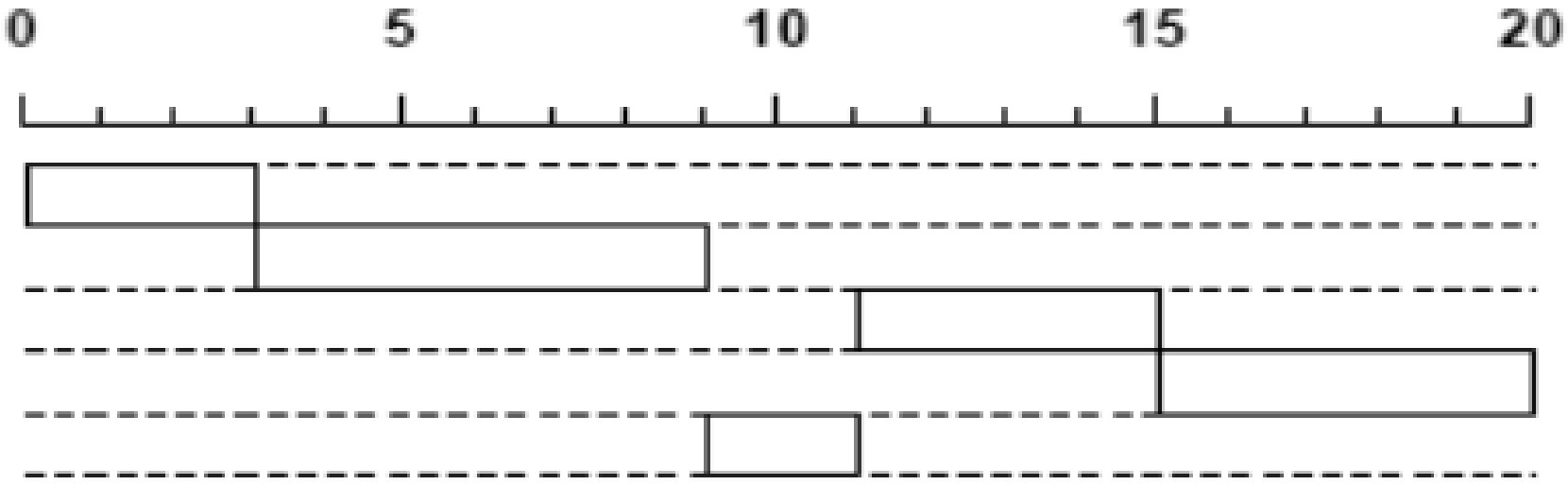
Shortest Job First (SJF)

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Shortest Job First (SJF)

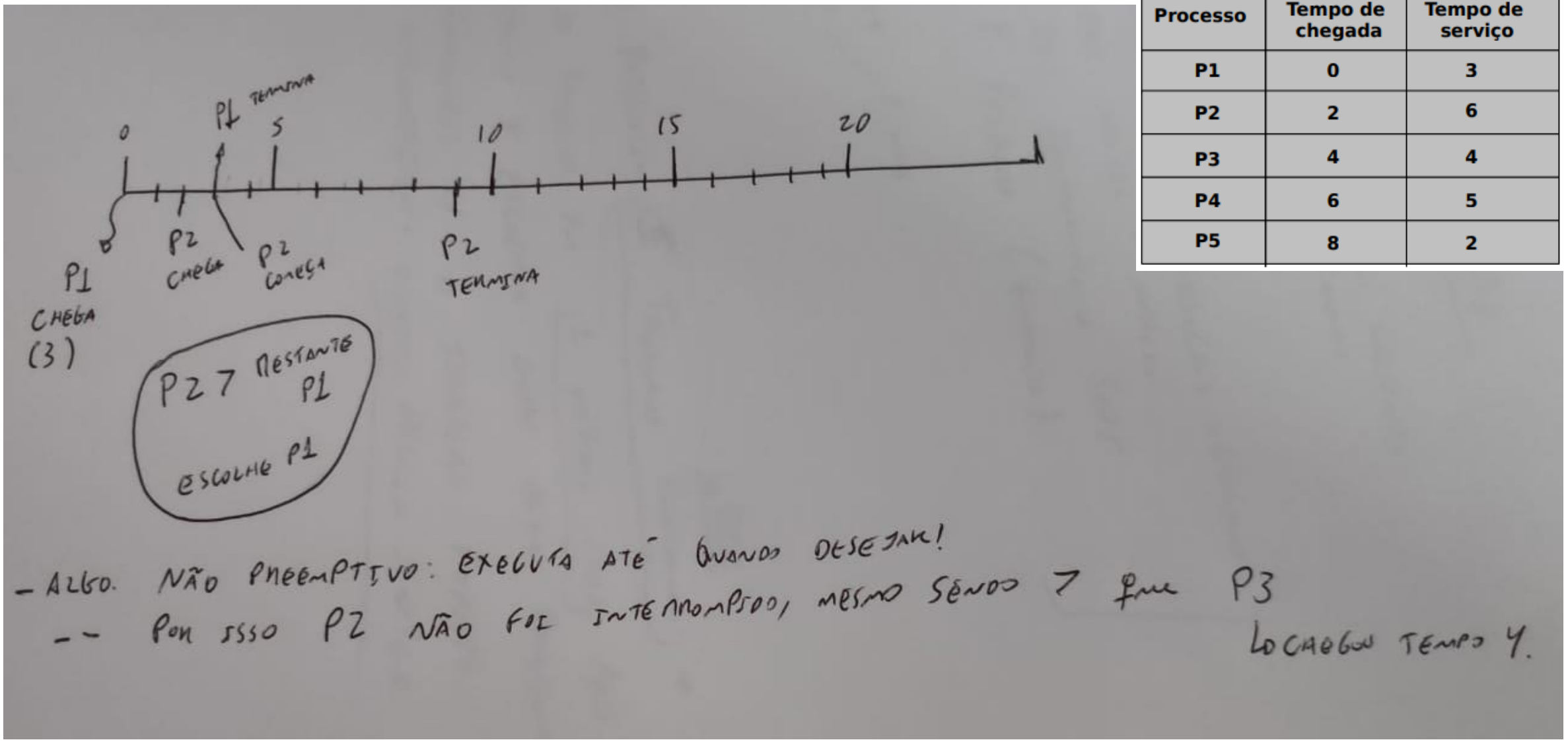
Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Pausem o vídeo e tentem entender o algoritmo!
No próximo slide eu explico.



Shortest Job First (SJF)

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2



Shortest Job First (SJF)

Convém observar que a tarefa mais curta primeiro é adequada somente para situações em que todas as tarefas estejam disponíveis simultaneamente e que não seja necessário ordem na execução.

Finalizado: Algoritmos de Escalonamento Não-preemptivos

Próxima: Algoritmos de escalonamento Preemptivos

Shortest Job First (SJF) – Versão Preemptiva

Com este algoritmo, o escalonador sempre escolhe o processo cujo o tempo de execução restante seja o menor

- Neste caso, há interrupção do processo quando um processo com menor tempo de execução chega

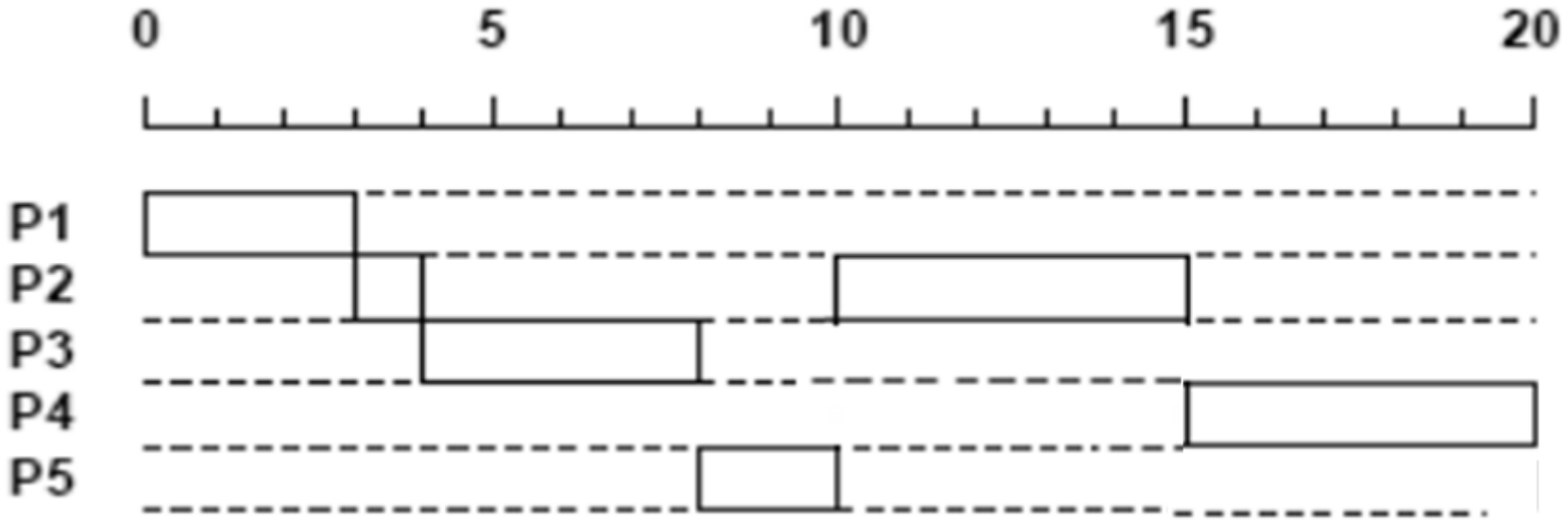
Novamente, o tempo de execução deverá ser previamente conhecido.

Quando chega um novo job, seu tempo total é comparado ao tempo restante do processo em curso.

- Se para terminar, o novo job precisar de menos tempo que o processo atual, então esse será suspenso e o novo job será iniciado

Shortest Job First (SJF) – Versão Preemptiva

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

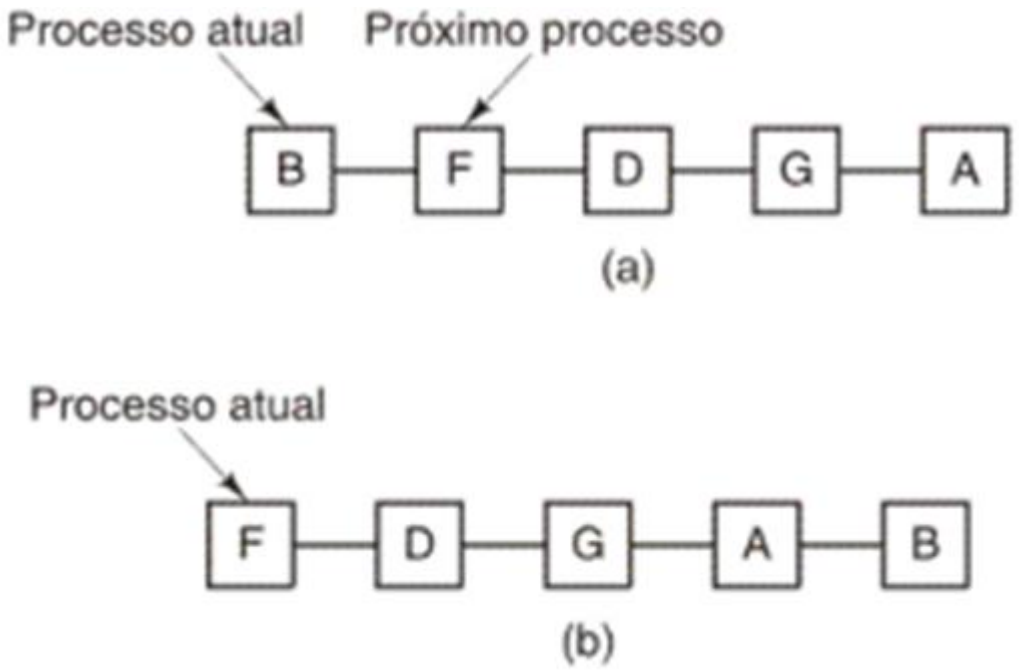


Escalonamento por alternância circular

Round Robin

- Um dos algoritmos mais antigos;
- A cada processo é atribuído um intervalo de tempo, **o quantum**, no qual ele é permitido executar;
- Se no final do quantum o processo não terminou, a CPU sofre uma preempção e outro processo entra para executar;
- Quando um processo termina seu quantum, ele é colocado no final da fila.

Round Robin


Processo atual
Próximo processo
B
F
D
G
A
(a)
Processo atual
F
D
G
A
B
(b)

Round Robin

O que interessa em um escalonador circular é o tamanho do **quantum**.

A alternância de um processo para outro requer uma certa quantidade de tempo (troca de contexto).

Vamos supor que para cada processo, a troca de contexto dure 1 ms. Suponha também que o **quantum** é de 4 ms.

- Com este exemplo, após 4ms de trabalho, a CPU gastará 1ms para alternar o processo
- Assim, 20% da CPU será gasta com a troca de contexto dos processos gerando um overhead muito alto.

Round Robin

Para melhorar a eficiência, vamos supor um **quantum** de 100 ms, agora o tempo gasto com o overhead é de 1% do tempo da CPU

No entanto, considere o que pode acontecer em um sistema de tempo compartilhado se houver 50 solicitações de mudança dentro de um curto intervalo de tempo e com grande variação nas necessidades de CPU

1. O primeiro processo iniciará imediatamente (se a CPU estiver ociosa)
2. O segundo esperará 100ms
3. O terceiro esperará mais 100ms
4. ...
5. O último pode ter que esperar 5s antes de ter uma oportunidade

Round Robin

Essa situação é especialmente ruim se alguma das solicitações próximas ao fim da fila exigir apenas alguns milissegundos de tempo da CPU

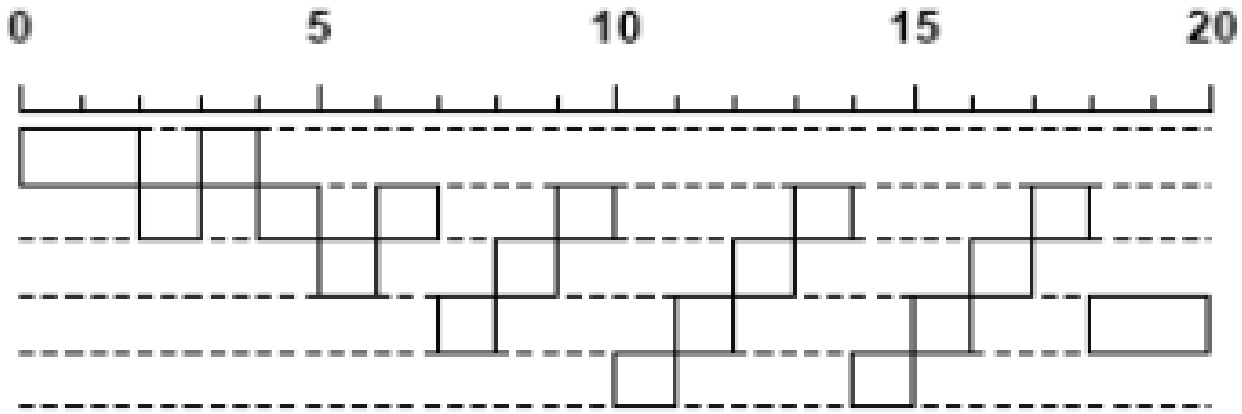
- Com um quantum curto, os usuários teriam obtido o melhor serviço

A conclusão pode ser formulada assim:

- Adotar um quantum muito curto causa muito chaveamento de processos e reduz a eficiência da CPU
- Mas, um quantum muito longo pode gerar uma resposta pobre às requisições interativas curtas
- Um quanto em torno de 20 ms a 50 ms é o razoável

Round Robin

Round Robin (RR)
q = 1

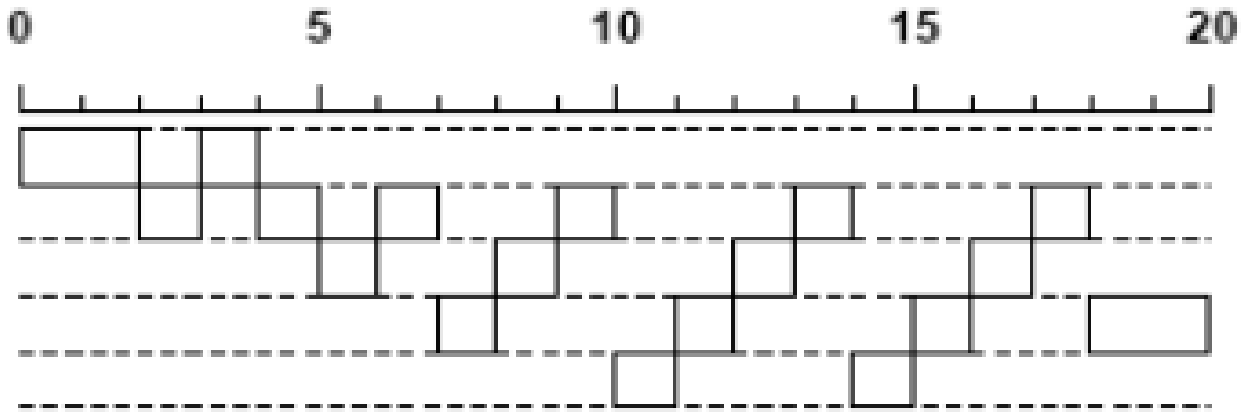


Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

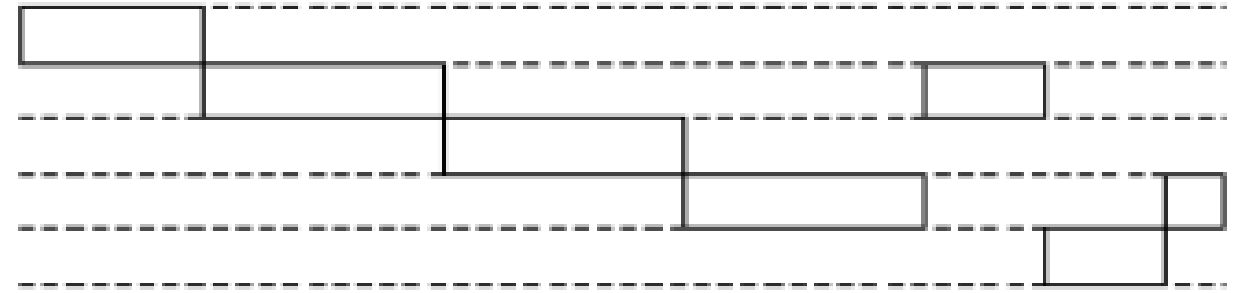
Round Robin

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Round Robin (RR)
q = 1



Round Robin (RR)
q = 4



Round Robin

Por ser preemptivo, um processo perde o processador quando:

- Libera explicitamente o processador (yield)
- Realize uma chamada de sistema (bloqueado)
- Termina sua execução
- Quando sua fatia de tempo é esgotada

Se *quantum* $\rightarrow \infty$ (tende ao infinito), obtém-se o comportamento de um escalonador FIFO.

Escalonamento por Prioridade

O escalonamento circular pressupõe que todos os processos são igualmente importantes. Para que haja diferenças externas de execução, foi criado o escalonamento por prioridade.

A ideia básica é simples:

- A cada processo é atribuído uma prioridade, e o processo com maior prioridade é executado primeiro

Mesmo em um PC comum com um único proprietário, pode haver múltiplos processos com prioridades diferentes.

- Exemplo, um processo como um **serviço** de correio eletrônico deve ter menor prioridade que um processo que atualiza um vídeo na tela

Escalonamento por Prioridade

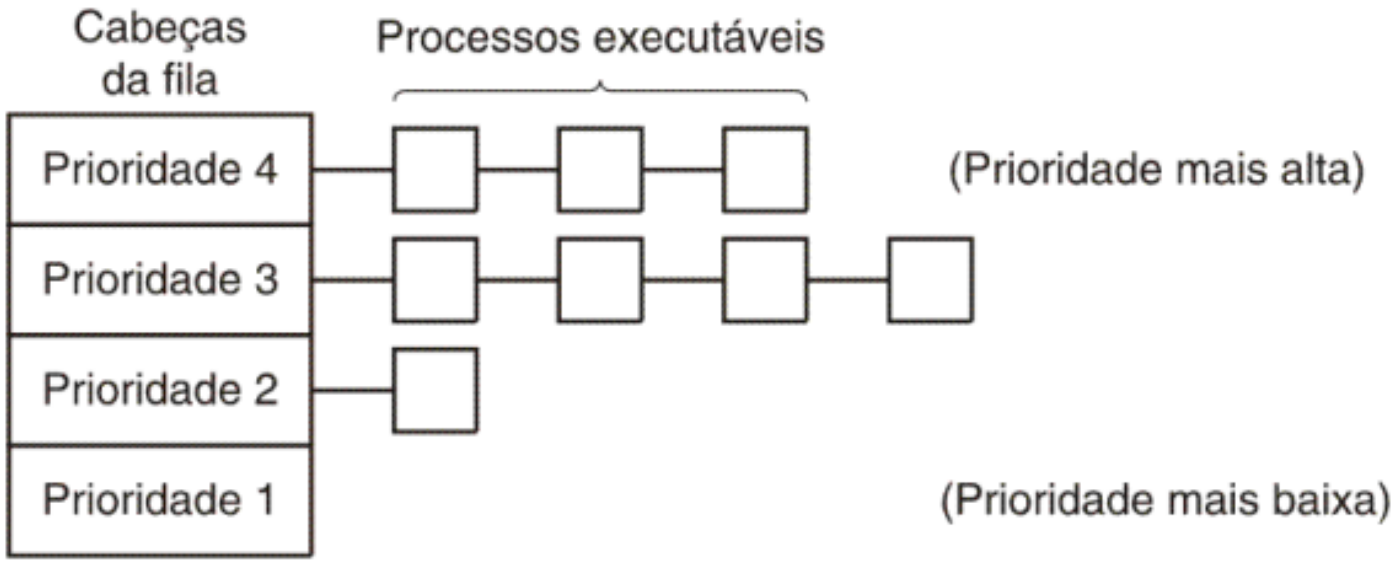
Para evitar que processos de alta prioridade executem infinitamente, o escalonador pode reduzir a prioridade do processo a cada execução do mesmo.

As prioridades são atribuídas pelo S.O.

- No Linux, o comando **nice** altera a prioridade de um processo de acordo com que o usuário desejar.

Escalonamento por Prioridade

Muitas vezes é conveniente agrupar processos em classes de prioridade e usar o escalonamento de prioridade entre as classes



Finalizado: Algoritmos de escalonamento

Próxima: Medidas de desempenho

Seção 3

MEDIDAS DE DESEMPENHO DOS ALGORITMOS DE ESCALONAMENTO

Medidas de Desempenho dos Algoritmos de Escalonamento

Considere os seguintes itens:

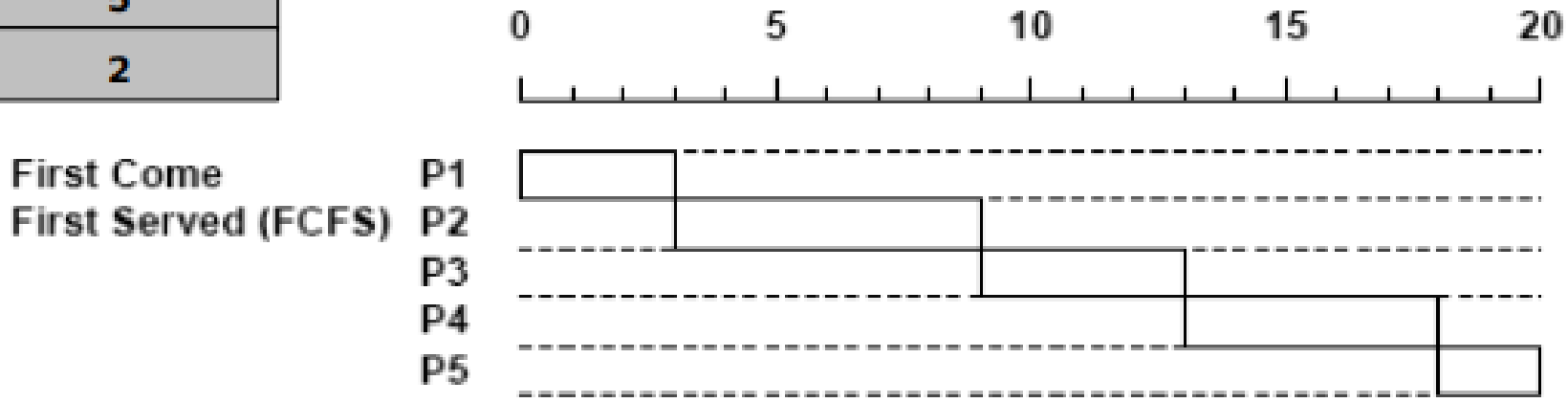
- **Tempo de chegada:**
  - Tempo em que o processo entra no sistema;
- **Tempo de serviço (T_s):**
  - Tempo total de processamento necessário para o processo;
- **Turn-around Time (T_q):**
  - Intervalo entre a entrada do processo no sistema e o seu término (inclusive as esperas);
- **T_q / T_s :**
  - Tempo de desempenho do processo;

Medidas de Desempenho dos Algoritmos de Escalonamento – FIFO/FCFS

Calcular o desempenho dos algoritmos de escalonamento utilizando a tabela abaixo e o algoritmo FIFO:

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2



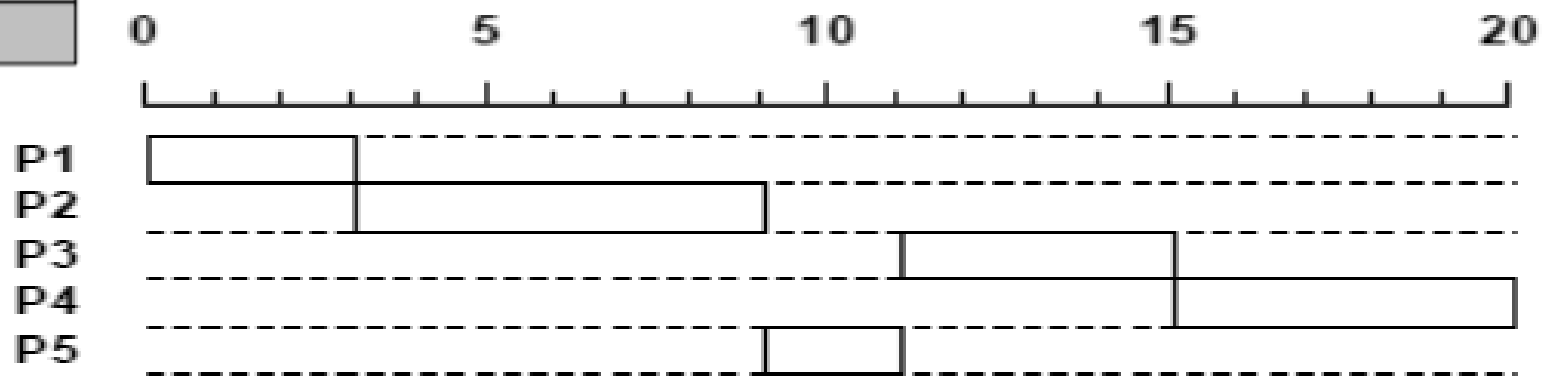
	Process	1	2	3	4	5	
	Arrival Time	0	2	4	6	8	Mean
	Service Time (Ts)	3	6	4	5	2	
FCFS	Finish Time	3	9	13	18	20	
	Turnaround Time (Tq)	3	7	9	12	12	8.60
	Tq / Ts	1.00	1.17	2.25	2.40	6.00	2.56

Medidas de Desempenho dos Algoritmos de Escalonamento – Tarefa mais curta 1º

Calcular o desempenho dos algoritmos de escalonamento utilizando a tabela abaixo e o algoritmo SJF (Shortest Job First - tarefa mais curta primeiro):

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Processo	Tempo de chegada	Tempo de serviço
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2



	Process	1	2	3	4	5	
	Arrival Time	0	2	4	6	8	Mean
	Service Time (Ts)	3	6	4	5	2	
SPN	Finish Time	3	9	15	20	11	
	Turnaround Time (Tq)	3	7	11	14	3	7.60
	Tq / Ts	1.00	1.17	2.75	2.80	1.50	1.84

Finalizado: Medidas de desempenho

Próxima: Escalonamento no Linux

Seção 4

ESCALONAMENTO UTILIZADO PELO SISTEMA OPERACIONAL LINUX

Escalonamento utilizado pelo sistema operacional Linux

O problema básico de escalonamento em sistemas operacionais é como satisfazer simultaneamente objetivos conflitantes:

- Tempo de resposta rápido,
- Boa vazão (num. processos por tempo) para processos background,
- Evitar espera indefinida,
- Conciliar processos de alta prioridade com de baixa prioridade,
- Etc...
- Ou seja, determinar a política de escalonamento

Escalonamento utilizado pelo sistema operacional Linux

Os sistemas operacionais Linux implementam o escalonamento da seguinte forma:

Os processos são divididos em três grandes classes:

1. **processos interativos**
2. **processos batch**
3. **processos tempo real**

Em cada classe, os processos podem ser ainda subdivididos em I/O-bound ou CPU-bound

- De acordo com a proporção de tempo que ficam esperando por operações de entrada e saída ou utilizando o processador.

Escalonamento utilizado pelo sistema operacional Linux

O escalonador do Linux não distingue processos interativos de processos batch, diferenciando-os apenas dos processos de tempo real

Como todos os outros escalonadores UNIX, o escalonador Linux privilegia os processos **I/O-bound** em relação aos **CPU-bound**

- De forma a oferecer um melhor tempo de resposta às aplicações interativas
- Reduzir tempo de espera

Escalonamento utilizado pelo sistema operacional Linux

O escalonador do Linux é baseado em **time-sharing**

- Ou seja, o tempo do processador é dividido em fatias de tempo (quantum) as quais são alocadas aos processos.
- Se durante a execução de um processo o quantum é esgotado, um novo processo é selecionado para execução, provocando então uma troca de contexto.

Esse procedimento é completamente transparente ao processo e baseia-se em interrupções de tempo.

- Esse comportamento confere ao Linux um escalonamento do tipo **preemptivo**.

Escalonamento utilizado pelo sistema operacional Linux

Outra característica do escalonador Linux é a existência de prioridades dinâmicas.

- O escalonador do Linux monitora o comportamento de um processo e **ajusta dinamicamente sua prioridade**, visando equalizar o uso do processador entre os processos.

Processos que recentemente ocuparam o processador durante um período de tempo considerado “longo” têm sua **prioridade reduzida**.

- De forma análoga, aqueles que estão há muito tempo sem executar recebem um aumento na sua prioridade, sendo então beneficiados em novas operações de escalonamento.

Escalonamento utilizado pelo sistema operacional Linux

O sistema Linux trabalha com dois tipos de prioridades: estática e dinâmica.

- As prioridades estáticas são utilizadas exclusivamente por processos de tempo real
  - Neste caso, a prioridade do processo tempo real é **definida pelo usuário e não é modificada pelo escalonador**.
- O esquema de prioridades dinâmicas é aplicado aos processos interativos e batch.
  - Aqui, a prioridade é calculada, considerando-se a prioridade base do processo e a quantidade de tempo restante em seu quantum.

Escalonamento utilizado pelo sistema operacional Linux

O escalonador do Linux executa os processos de prioridade dinâmica apenas quando não há processos de tempo real.

- Em outros termos, os processos de prioridade estática recebem uma prioridade maior que os processos de prioridade dinâmica.

Para selecionar um processo para execução, o escalonador do Linux prevê três políticas diferentes :

- **SCHED_FIFO**
- **SCHED_RR**
- **SCHED_OTHER**

Escalonamento utilizado pelo sistema operacional Linux

SCHED_FIFO:

- Essa política é valida apenas para os **processos de tempo real.**
- Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade.
- Nessa política, quando um processo é alocado ao processador, ele executa até que uma de três situações ocorra:
  1. Um processo de tempo real de prioridade superior torna-se apto a executar.
  2. O processo libera espontaneamente o processador para processos de prioridade igual à sua.
  3. O processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização.

Escalonamento utilizado pelo sistema operacional Linux

SCHED_RR [ROUND ROBIN]:

- Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade.
- Quando um processo é alocado ao processador, ele executa até que uma de quatro situações ocorra:
  1. Seu período de execução (quantum) tenha se esgotado, nesse caso o processo é inserido no final de sua fila de prioridade.
  2. Um processo de prioridade superior torna-se apto a executar.
  3. O processo libera espontaneamente o processador para processos de prioridade igual a sua.
  4. O processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização.

Escalonamento utilizado pelo sistema operacional Linux

SCHED_OTHER:

- Corresponde a um esquema de filas multinível de prioridades dinâmicas com timesharing. Os processo **interativos e batch** recaem nessa categoria.
- O escalonador do Linux é executado a partir de duas formas diferentes:
  - A primeira é a **forma direta** através de uma chamada explícita à rotina que implementa o escalonador.
    - Essa é a maneira utilizada pelo núcleo do Linux quando, por exemplo, detecta que um processo deverá ser bloqueado em decorrência de uma operação de entrada e saída ou de sincronização.

Escalonamento utilizado pelo sistema operacional Linux

SCHED_OTHER:

- A segunda forma, denominada de lazy, também é consequência do procedimento de escalonamento, ocorrendo tipicamente em uma das duas situações:
  1. Rotina de tratamento de interrupção de tempo que atualiza os temporizadores e realiza a contabilização de tempo por processo.
     - Essa rotina, ao detectar que um processo esgotou seu quantum de execução aciona o escalonador para que seja efetuada uma troca de processo
  2. Um processo de mais alta prioridade é desbloqueado pela ocorrência do evento que esperava.
     - A parte do código que efetua o desbloqueio, isto é, trata os eventos de sincronização e de entrada e saída, consulta a prioridade do processo atualmente em execução e compara-a com a do processo que será desbloqueado.

Para saber mais...

Sobre os escalonadores do Linux, procurem por Completely Fair Scheduler (CFS) :

https://trepo.tuni.fi/bitstream/handle/10024/96864/GRADU-1428493916.pdf

Também é possível consultar a documentação oficial do Kernel Linux.

Dúvidas?

Sistemas operacionais: Parte 4 - Escalonamento de processos

Definição. Sistemas em lote, interativos e de tempo real. Métricas de avaliação dos algoritmos escalonadores. Algoritmos não preemptivos (cooperativos). First-In First-Out (FIFO) First-Come First-Served (FCFS). Shortest Job First (SJF). Algoritmos preemptivos. Round Robin (circular). Baseado em prioridades. High Response Ratio Next (HRRN). Shortest Remaining Time (SRT). Escalonamento de frequência de CPU: Ondemand e Interactive (Linux/Android). Medidas de desempenho. Escalonamento de processos no Linux.