Sistemas Operacionais

4º período

Professora: Michelle Hanne

PUC Minas

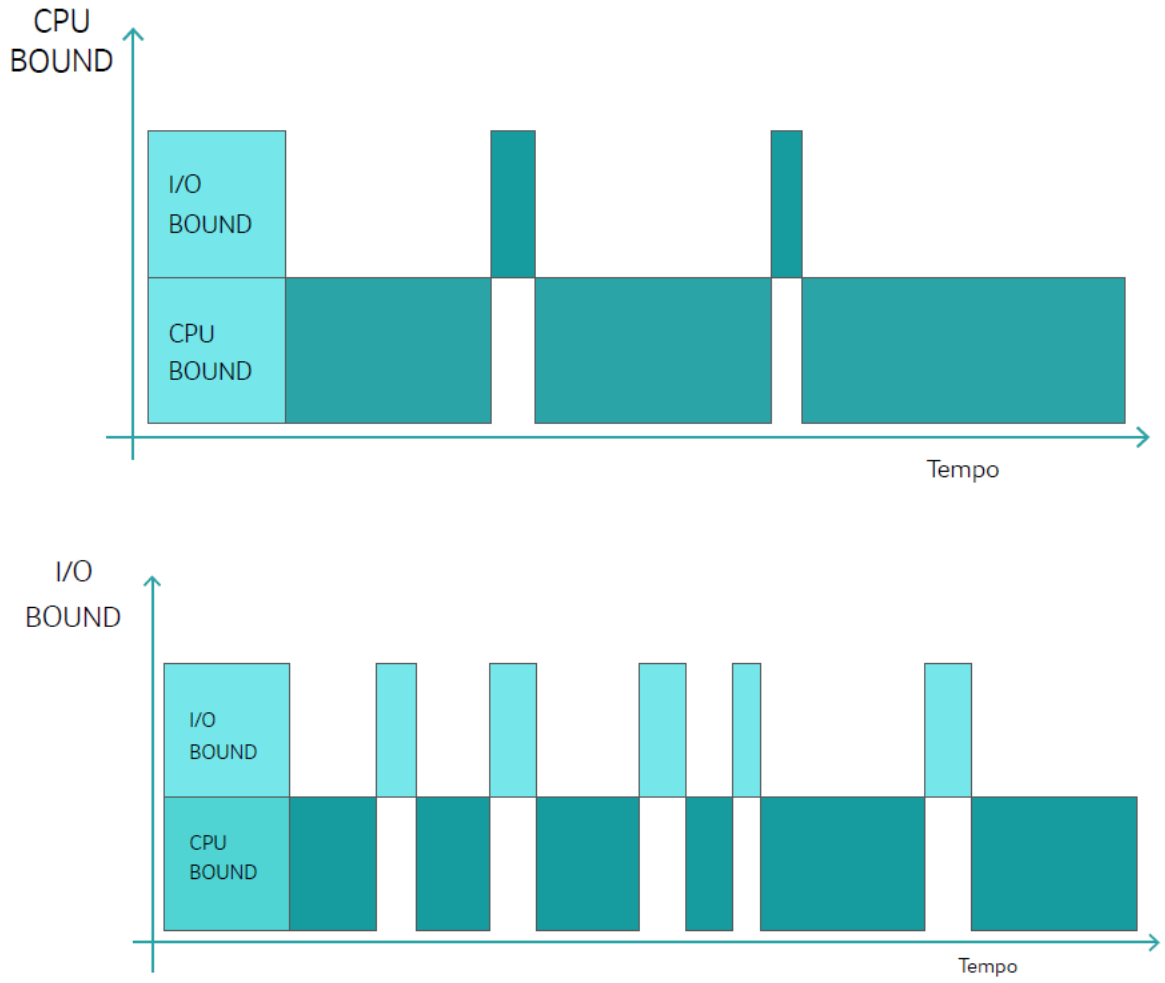
Tipos de Processos

Os processos podem ser classificados em dois tipos: CPU-BOUND e I/O-BOUND.

- **CPU-BOUND:** esse tipo de processo depende mais da CPU do que entradas e saídas. Como exemplo, *tem-se os processos focados em processamento de alto desempenho*.
- **I/O-BOUND:** esse tipo de processo realiza um número elevado de operações de entrada e saída, ficando um tempo maior no estado de ***Espera do que no estado de Execução ou Pronto***.

PUC Minas

Tipos de Processos

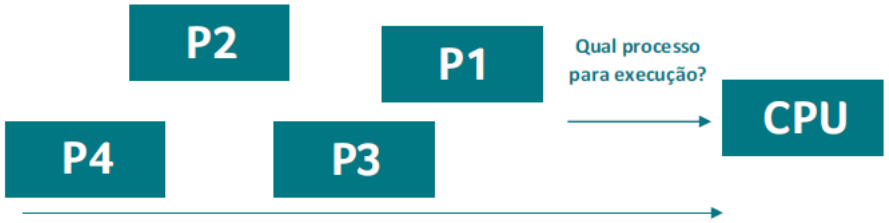


PUC Minas

Escalonamento de Processos

Os sistemas operacionais **multiprogramáveis** foram implementados a partir do momento que o conceito de compartilhamento **foi aplicado à CPU**. Vários processos podem ser executados concorrentemente na CPU.

O sistema operacional através de uma regra determina quem é o processo que terá a vez de ser processado pela CPU. Esse mecanismo é chamado de escalonamento (STUART, 2011).



PUC Minas

Escalonamento de Processos

Existem diversas técnicas e critérios para escalonamento. *Muitos dos problemas existentes no escalonamento de processos são válidos para thread.*

O escalonador de processos utiliza **diversos tipos de algoritmos para aplicação do escalonamento.**

- *Ex: FIFO (First in, first out) ou FCFS (First come, first served), SJF (Shortest Job First), SRT (Shortest Remaining Time), Algoritmo Loteria, Escalonamento Garantido, RR (Round-Robin), Múltiplas Filas e Algoritmo Fair-Share.*

PUC Minas

Critérios de Escalonamento

A principal função do algoritmo de escalonamento é definir qual processo tem a prioridade de execução na CPU.

Cada sistema operacional precisa de um algoritmo adequado ao seu tipo de processamento.

- Um sistema em jogo, por exemplo, precisa de um algoritmo que tenha como característica um tempo de resposta alto.
- Já um sistema de impressão precisa que um algoritmo controle bem as filas de impressão.

Alguns critérios são determinantes para escolha de um algoritmo de escalonamento, tais como:

- Utilização da CPU: em quase todos os sistemas, a CPU deve estar ocupada na sua carga máxima, 100%.

Porém, critérios podem ser adotados para garantir uma melhor eficiência da CPU. Um sistema com 40% de utilização do processador pode ser considerado um sistema de baixa utilização, mas um valor de 80% pode ser considerado um nível ideal de utilização (STUART, 2011).

PUC Minas

Critérios de Escalonamento

A utilização da CPU é dada pela fórmula:

- **(a soma total dos tempos de processamento de cada processo)/ (tempo total).**

Um sistema com três processos (A, B, C) com tempos de CPU 10, 20, 30 sendo processados sequencialmente, tem como valor de utilização: (10+20+30) / (10+20+30) = 1 = 100%, ou seja, nesse sistema a CPU terá 100% de utilização.

- Para explicitar a utilização da CPU, um servidor com um núcleo de processamento utiliza 10 minutos de processamento de um tempo total de 20 minutos.
  - Nesse caso, 10/20 = 0,5, ou seja, a CPU tem 50% de utilização. Em outro exemplo, os mesmos tempos de processamento, porém um processador com 4 núcleos, então, teria 0,5/4 = 0,125, ou seja, 12,5% ocupada.

PUC Minas

Throughput

É a quantidade de processos que é processado em um determinado intervalo de tempo. **A relação é direta, quanto maior o *throughput* melhor será o desempenho do sistema.**

- Um sistema que contém 3 processos e leva 20 segundos para processá-los possui um **throughput de (3/20) = 0,15**, mas um outro sistema que processe esses mesmos 3 processos num intervalo de 40 segundos, (3/40), terá o **throughput com valor igual a 0,075**. Sendo assim, o primeiro sistema seria mais desejável a um melhor desempenho.

PUC Minas

Tempo de Retorno (TurnAround)

É tempo total medido desde a admissão do processo na CPU até o seu término. O tempo de retorno leva em consideração o tempo perdido na alocação de memória, fila de espera de processos de pronto, processamento na CPU e nas trocas de contextos.

- *Exemplo:* Um sistema que possui um tempo de processamento na CPU de 10s, tempo de chegada de 0s, leva exatos 10 segundos para ser executado por completo, porém um segundo processo que chegue logo em seguida com tempo de chegada de 0s e um tempo de processamento de 20s, terá o tempo de retorno em 30 segundos (10+0+20) = 30.

PUC Minas

Tempo de Resposta

É o tempo de resposta é o tempo medido entre a admissão e a primeira resposta do sistema.

- É muito utilizado em sistemas interativos. Um dispositivo de entrada ou saída pode limitar esse tempo de resposta.

PUC Minas

Escalonador de Processos

Multiprogramação: muitos processos ou threads aptos a executar

Problema: Não há processadores/núcleos livres para todos

Objetivo: manter os processadores/núcleos ocupados e finalizar as tarefas o quanto antes

Escalonador (scheduler): **decide quem será o próximo processo da fila de prontos a executar**

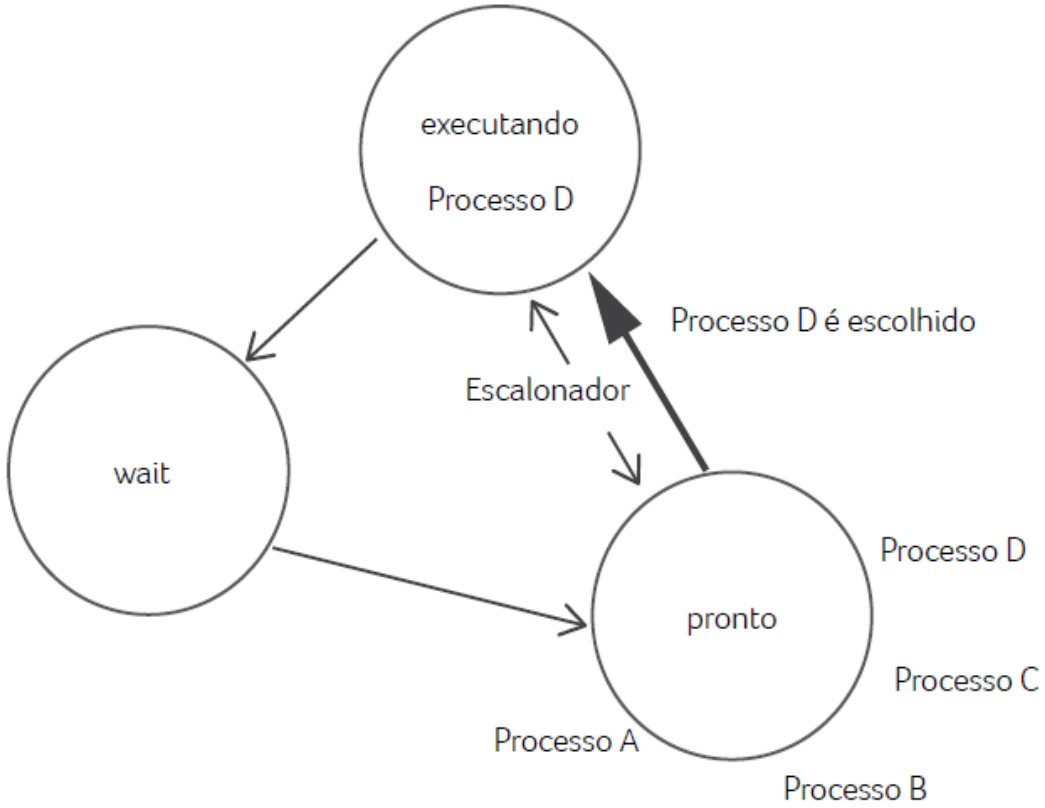
Algoritmo de escalonamento: regra seguida pelo escalonador para determinar a escolha

PUC Minas

Escalonador de Processos

O escalonador de processos faz uso de algoritmo para possibilitar que processos concorrentes sejam executados na CPU, tanto os processos CPU-BOUND quanto os processos I/O-BOUND (TANENBAUM, 2016).

Escalonador de Processos



Diversos algoritmos são implementados de acordo com o sistema em uso. O escalonador tem que levar uma série de condições em questão, como:

- **Justiça:** todos os processos devem ser processados em igualdade de condições, sem adiamento indefinido.
- **Produtividade (100%):** a maximização das tarefas por tempo tem que ser a maior possível, de preferência em 100%.
- **Previsibilidade:** sempre usar o mesmo tempo e os recursos computacionais na execução da mesma tarefa.
- **Tempo de resposta:** minimizar ao máximo em sistemas interativos.
- **Sobrecarga (overhead):** minimizar o desperdício de recursos.
- **Balanceamento:** balancear os recursos; devem ser balanceados no uso de recursos.
- **Cumprimento dos prazos:** sistema tem que atender no tempo correto as requisições.
- **Proporcionalidade:** Satisfazer as perspectivas dos usuários.

PUC Minas

Algoritmos de Escalonamento

- Sempre que um **processo for bloqueado ou esperar um dispositivo de entrada e saída,** o escalonador de processos deve usar os algoritmos para definir qual processo deverá ter prioridade de execução.
- Os algoritmos de escalonamento podem ser divididos em três **categorias: lote, interativo, e de tempo real.**

Categoria	*Característica*
Sistema em *Batch*	*Throughput (Taxa de Transferência)* Tempo de retorno Utilização
Sistemas interativos	Tempo de resposta Proporcionalidade
Sistema de tempo real	Cumprimento dos prazos Previsibilidade

PUC Minas

Escalonamento Não Preemptivo

- Os sistemas operacionais das primeiras gerações eram predominantemente por processamento em *batch*, ou seja, não preemptivo. **Cada processo que ganhava a CPU, tinha o direito de ser totalmente processado para depois liberar a CPU para o próximo processo.**
- O sistema de processamento **não preemptivo (*batch*)** também pode ser exemplificado no processo de impressão. Cada processo de impressão entra em uma fila de impressão (*pool* de impressão), e somente um por vez é atendido e totalmente processado pelo sistema (TANENBAUM, 2016).

- O processo retém a CPU.
- Sem decisão ativa do escalonador
- Não exige temporizador de hardware

PUC Minas

Escalonamento Não Preemptivo

Existem alguns algoritmos que satisfazem um sistema em lote:

- primeiro a chegar, primeiro a sair (*First IN First OUT* - FIFO),
- o menor trabalho primeiro (*Short Job First* - SJF)
- próximo de menor tempo (*Short Remaing Time Next* - SRTN)
- FCFS (First come, first served)

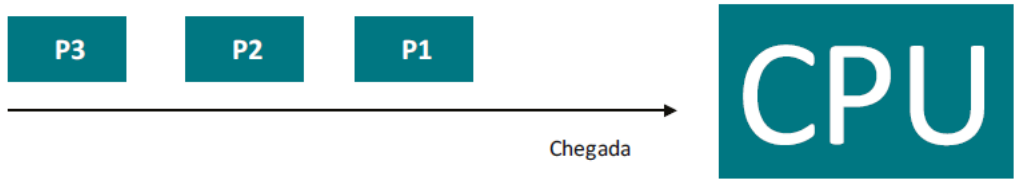
Um exemplo clássico de um sistema em batch é um backup de dados, pois é executado em lote sem a necessidade de interação com o usuário.

PUC Minas

Escalonamento FIFO

Esse algoritmo satisfaz a condição de **quem chega primeiro é atendido.** Quando o processo está com a permissão de ser executado pela CPU, ele é totalmente processado até o seu término, e só então, libera a CPU para o próximo processo na fila.

- Exemplos: Fila de Impressão. Outro exemplo são os circuitos eletrônicos de buffer e controle de fluxo que usam o algoritmo FIFO para ler e escrever os dados.



PUC Minas

Escalonamento FIFO

Processo	Tempo de Chegada	Tempo de Processamento	Tempo de Execução
A	0	24	24-0 = 24
B	2	3	(24+3) -2 = 25
C	1	4	(24+3+4)-1 = 30

Nesse cenário, com a ordem de chegada (A, B, C), o tempo médio de espera total seria de 15 segundos: (0 + 22 + 23) / 3 = 15

Cada processo é executado por vez na CPU, mas seus tempos de espera são diferentes. O tempo de espera de A é 0s, ou seja, imediatamente é processado; o tempo de espera de B é de 22 segundos, ou seja, leva os 24 segundos de processamento de A menos os 2 segundos de chegada. O tempo de espera do processo C é de 23 segundos, pois leva os 24 segundos de A menos 1 segundo de C.

O tempo médio de processamento seria (24 + 25 + 30) /3 = 26,33 segundos.

PUC Minas

Escalonamento SJF (*Short Job First*)

O tempo de execução do processo na CPU é determinante, pois é previamente conhecido pelo sistema operacional.

Esse tempo de processamento determina a ordem de chegada dos processos na CPU.

Rotinas diárias previamente conhecidas pelo tempo gasto em sua execução podem ser organizadas usando conceito do algoritmo: trabalho mais curto primeiro.

- Esse algoritmo pode ser aplicado em sistemas que são orientados por tarefas de entrada e saída como servidores de rede e compiladores.

PUC Minas

Escalonamento SJF (*Short Job First*)

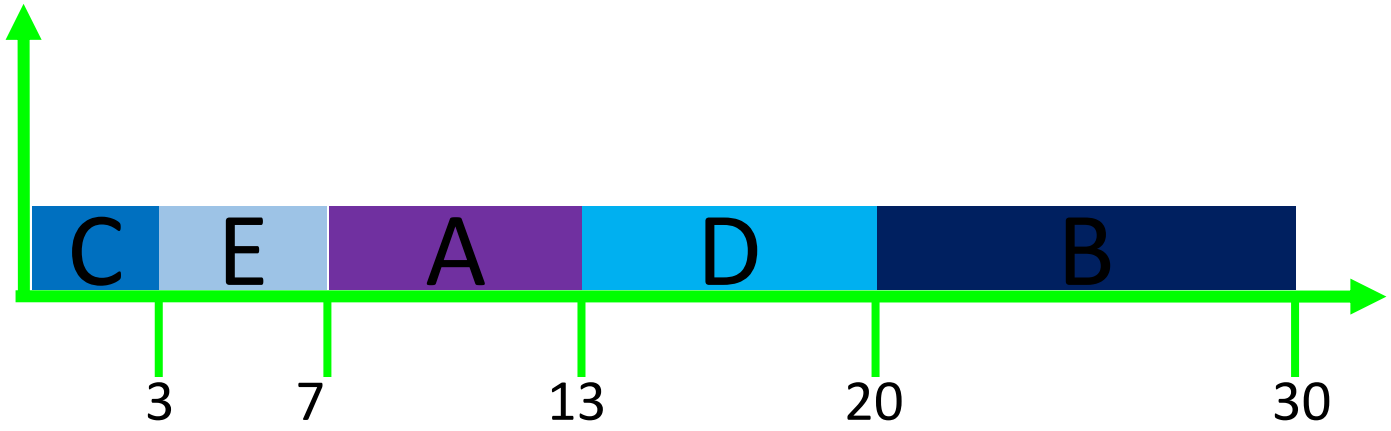
Processo Ordenados	Tempo de Chegada	Tempo Previsto de Processamento	Tempo de Execução
B	0	3	3 - 0 = 3
C	1	4	(3+4) -1 = 6
A	0	24	(3 + 4 + 24)-0 = 31

Então, o tempo médio de processamento seria (3 + 6 + 31) /3 = 13,33 segundos.

Nesse cenário, o tempo médio de espera seria (0 + 2 + 7/ 3) = 3 segundos

Escalonamento SJF (*Short Job First*)

Processo	Tempo previsto
A	6
B	10
C	3
D	7
E	4



Retorno médio = (3+7+13+20+30)/5

= 73/5 = 14,6

PUC Minas

Escalonamento Preemptivo

O processo pode ser temporariamente suspenso. Decisão ativa do escalonador.

O SO atribui um valor numérico a cada processo. “Nem todos os processos são igualmente importantes”

O processo com a prioridade mais importante executa primeiro. Atenção para os critérios diferentes de cada SO.

PUC Minas

Escalonamento Preemptivo

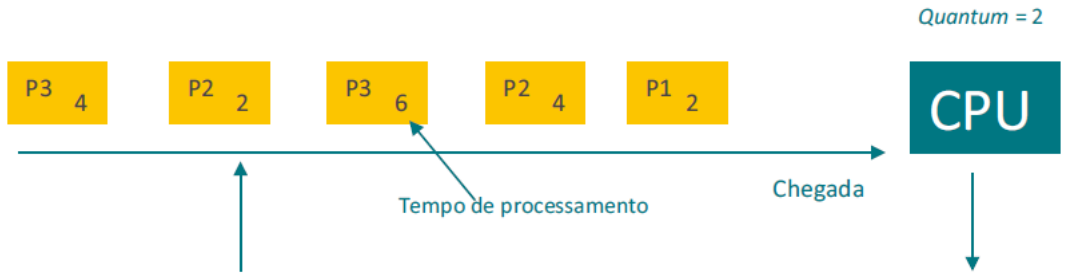
Os sistemas interativos, como jogos, por exemplo, possuem diversos tipos de algoritmos que satisfazem o seu objetivo.

Os algoritmos de interatividade são: *Round-Robin, por prioridade, múltiplas filas, garantido, sorteio*. Esses algoritmos trabalham com alternância de processos na CPU.

PUC Minas

Algoritmo Circular (*Round-Robin*)

- No algoritmo de **alternância circular**, a cada processo é atribuído um tempo chamado de **quantum**.
- Esse **quantum** é o tempo que esse processo **tem de CPU a cada ciclo de processamento.** Um processo que tem **10 segundos e um quantum de 2 segundos**, irá ser processado em 5 vezes, alterando com outros processos na fila.
- Após o seu processamento, ele é colocado no final da fila.



PUC Minas

Algoritmo Circular (*Round-Robin*)

- **Nesse algoritmo o importante é o tamanho do quantum**. A cada preempção, o sistema tem que registrar o tempo gasto para a administração dessa troca de contexto (salvar, carregar registradores, mapas de memória, listas de tabelas, etc.).
- *A ideia é minimizar o tempo gasto pela troca de contexto. Um sistema que tem um quantum de 4ms e um tempo de troca de contexto de 1ms, tem 20% do tempo gasto pela CPU só para fazer a troca de processos, mas um sistema com um quantum de 99ms, tem, nesse mesmo cenário, 1% de perda na troca de contexto.*

O algoritmo Round-Robin é utilizado em sistemas destinados à otimização de áudio e vídeo na internet.

PUC Minas

Algoritmo Circular (*Round-Robin*)

Agora, **suponha que 10 usuários** apertem a **tecla Enter** ao mesmo tempo, quanto tempo o último usuário levaria para receber uma resposta? O conceito é estabelecer um **balanceamento entre o tamanho do quantum e tempo de troca de contexto**, pois quantum pequeno gera muitas trocas e reduz a eficiência da CPU, e quantum longo, gera demora nas requisições curtas em sistemas interativos.

A duração da fatia de tempo depende do sistema operacional e do processador. Como cada fatia de tempo é pequena (aproximadamente 20 milissegundos), vários segmentos parecem estar sendo executados ao mesmo tempo. No Linux Kernel 2.6, a fatia é de 20 ms; no Windows, por sua vez, é de aproximadamente 15 ms:(STUART, 2011).

Algoritmo Circular (*Round-Robin*)

Processo	Tempo de Chegada	Tempo de Processamento	Tempo de Espera	Tempo de Execução considerando a preempção
A	0	6	0	(13 - 0 - 6) = 7
B	2	3	2	(9 - 2 - 3) = 4.
C	1	4	3	(11 - 1 - 4) = 6.

quantum (tempo de processo na CPU) = 2 ut (unidade de tempo)

O tempo médio de espera será (0 + 2 + 3) /3 = 1,66.

O tempo médio de processamento será (7 + 4 + 6) /3 = 5,66.

Então, o tempo de espera de cada processo será:

- **Processo A:** tempo de chegada = 0, pois, ao chegar à CPU, o processo começa.
- **Processo B:** tempo de chegada **= 2**, pois ele chega no tempo 2.
- **Processo C:** tempo de chegada **= 1.** Há dois outros processos sendo processados pela ordem de chegada (A, B); nesse caso, será descontado o tempo de processamento dos processos**: 4 - 1 = 3.**

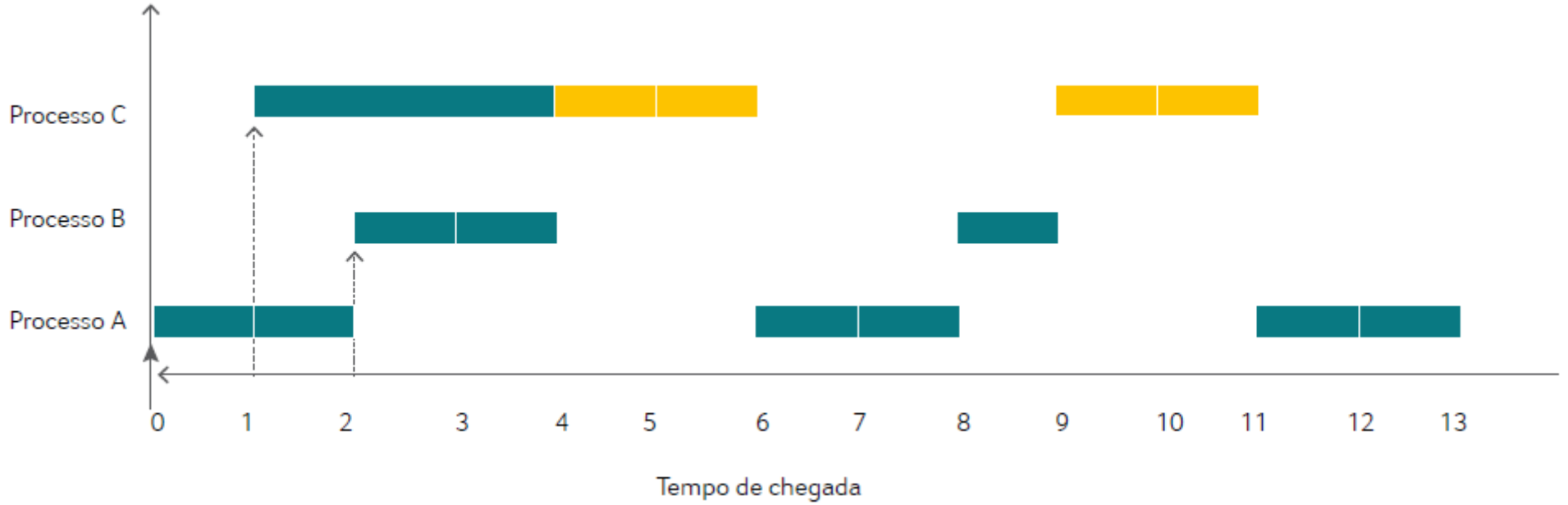
O tempo de processamento de cada processo tem que levar em consideração a preempção, ou seja, a cada 2 ut, o processo sofre uma preempção. Nesse caso, o tempo de processamento de cada processo seria:

- **Processo A: tempo de processamento total - tempo de chegada - tempo de CPU = (13 - 0 - 6) = 7.**
- **Processo B: (9 - 2 - 3) = 4.**
- **Processo C: (11 - 1 - 4) = 6.**

PUC Minas

Algoritmo Circular (*Round-Robin*)

Processo	Tempo de Chegada	Tempo de Processamento	Tempo de Espera	Tempo de Execução considerando a preempção
A	0	6	0	(13 - 0 - 6) = 7
B	2	3	2	(9 - 2 - 3) = 4.
C	1	4	3	(11 - 1 - 4) = 6.



PUC Minas

Algoritmo por Prioridade

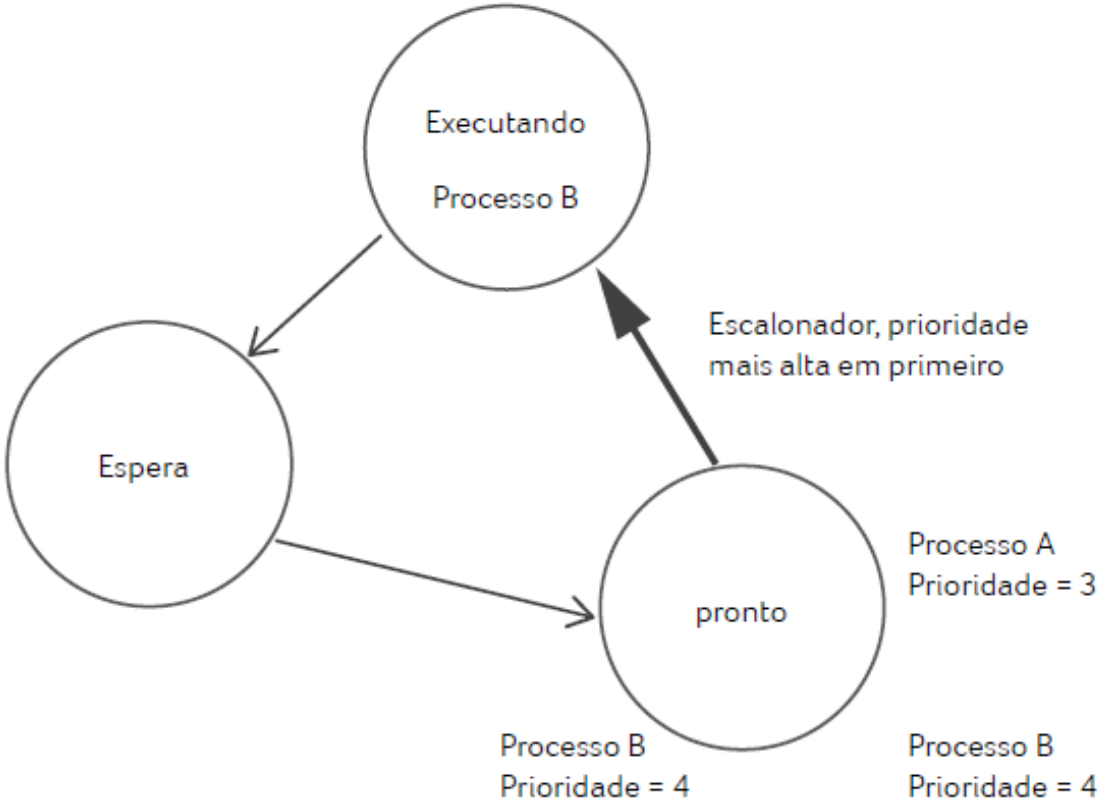
No algoritmo por prioridade, a cada processo é atribuída uma prioridade, estaticamente ou dinamicamente, e a política de prioridade (a mais alta primeiro ou a mais baixa primeiro) determina quem é executado na CPU.

Quando dois processos concorrentes chegam a CPU, o de maior prioridade ganha a CPU, mas a cada interrupção do relógio, decrementa o valor da prioridade. Quando a prioridade do processo fica menor que a prioridade do próximo processo, **há uma alternância de processo, e o próximo passa ter a vez na CPU.**

PUC Minas

Algoritmo por Prioridade

Entre dois processos, um envia em *background* e o outro exibe informações no vídeo, o segundo processo tem prioridade sobre o primeiro, como gravar tem prioridade sobre impressão. O sistema por prioridade é utilizado pelo Windows para determinar qual processo será processado ou tem a prioridade de execução.



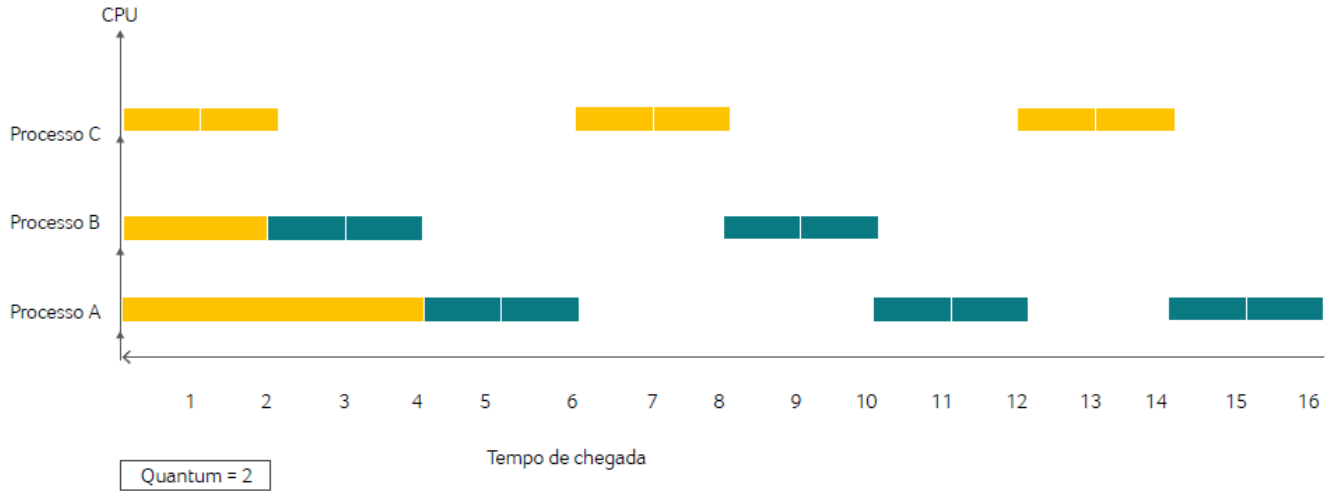
PUC Minas

Algoritmo por Prioridade

Processo	Tempo de Chegada	Tempo de Processamento	Prioridade	Tempo de Espera (Quantum=2)	Tempo de Execução considerando a preempção
A	0	6	2	4	(16 - 0 - 6) = 10
B	0	4	3	2	(10 -0- 4) = 6
C	0	6	4	0	(14 - 0- 6) = 8

A permissão de execução na CPU é determinada pelo valor mais alto de prioridade. A preempção será realizada a cada duas unidades de tempo na CPU, ou seja, *quantum* igual a 2.

O tempo médio de espera será (4 + 2 + 0) /3 = 2
O tempo médio de processamento será (10 + 6 + 8) /3 = 8



PUC Minas

Algoritmo MúltiplasFilas

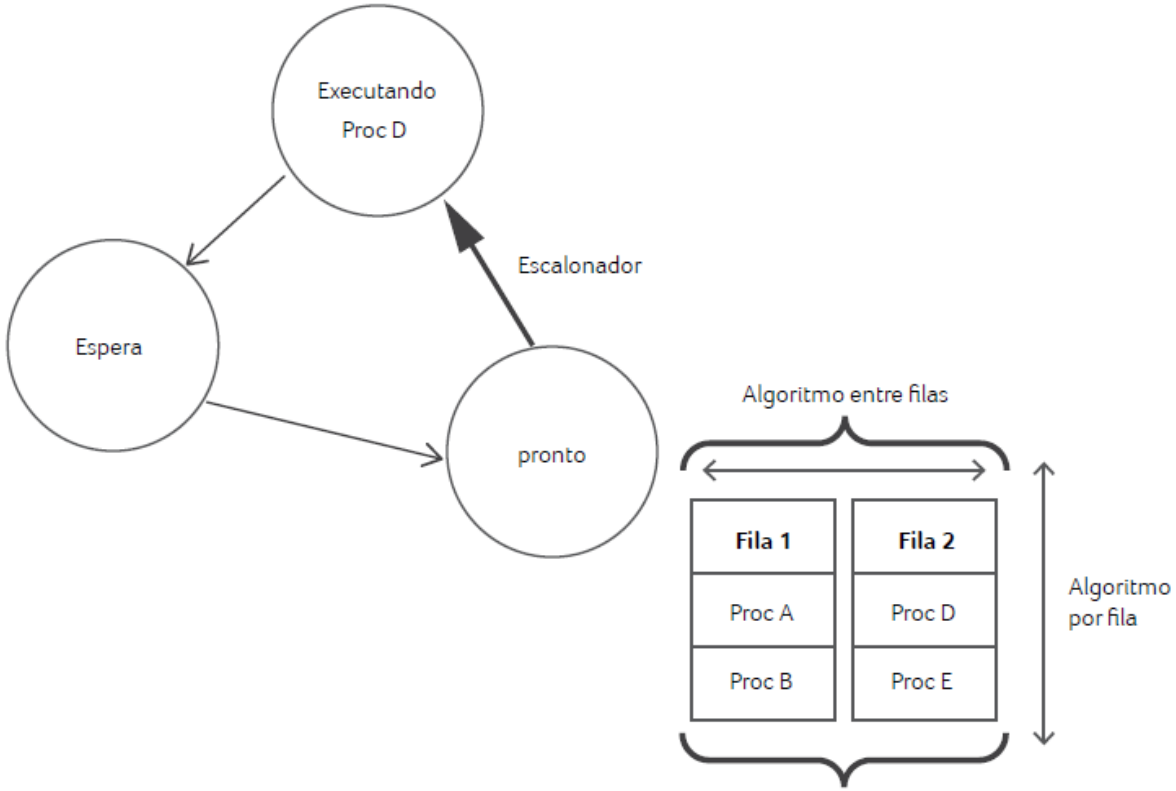
Como os sistemas possuem características diferentes de processamento, torna-se difícil que um único mecanismo de escalonamento seja adequado a todos os tipos de processo.

O algoritmo por múltiplas filas subdivide os processos em grupos de duas ou mais filas. Depois, prioriza cada fila por meio de um algoritmo (qualquer algoritmo). Cada fila, individualmente, tem seu próprio escalonador e um critério de prioridade.

PUC Minas

Algoritmo MúltiplasFilas

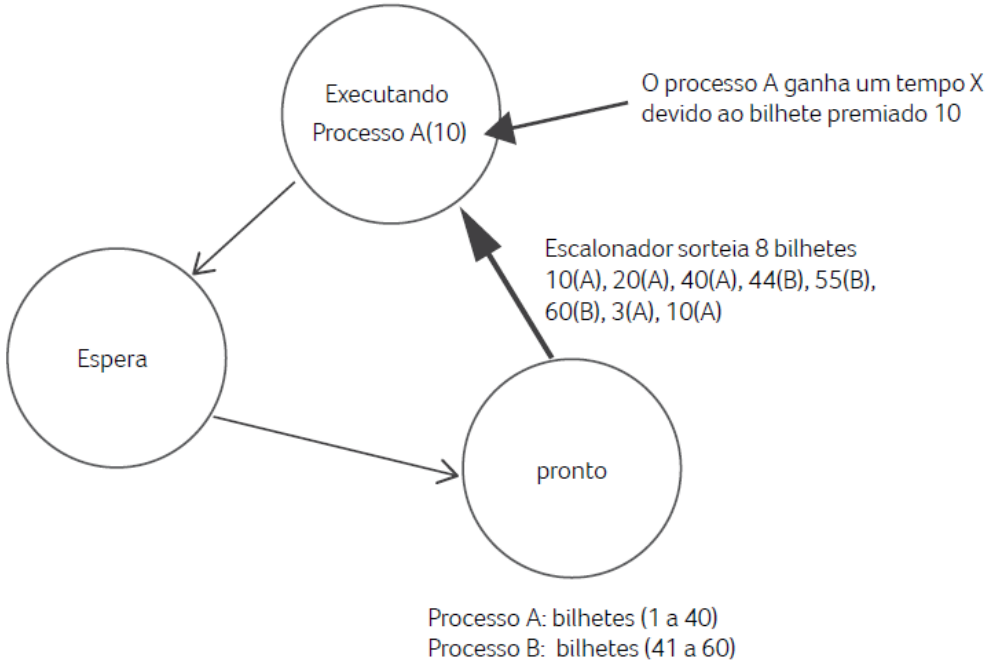
Exemplo: Esse algoritmo é utilizado no agendamento de diferentes tipos de pacote – como pacotes de dados em tempo real e sem tempo real, em nós de sensores com restrições de recursos em redes de sensores sem fio a fim de reduzir o consumo de energia dos sensores e os atrasos na transmissão dos dados de ponta a ponta.



PUC Minas

Algoritmo Sorteio (Loteria)

O algoritmo é semelhante a um sorteio de loteria. Porém, os prêmios são recursos do sistema. Esses recursos podem ser, **por exemplo, o tempo de execução na CPU atribuído a determinado processo.** Na hora da decisão, um sorteio aleatório é realizado e o processo ganhador tem um tempo x para executar todos os seus recursos.



PUC Minas

Algoritmo Sorteio (Loteria)

Nesse algoritmo, processos podem trocar bilhetes entre si, dando mais prioridade na execução das tarefas. Exemplos:

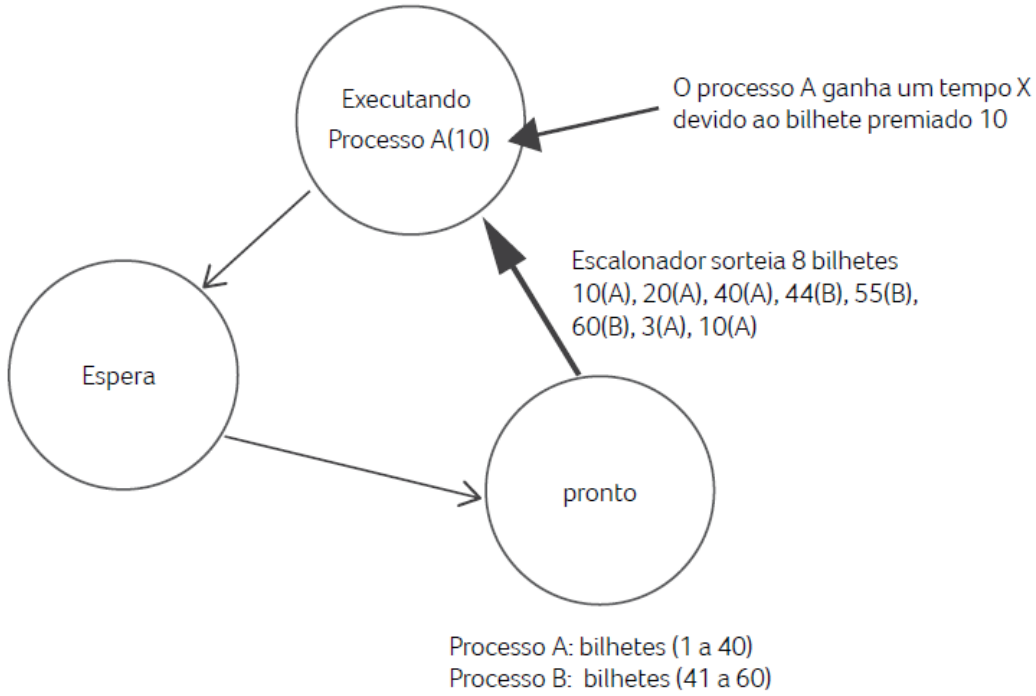
Servidores consolidados com virtualização, os acessos a dispositivos de rede física de máquinas virtuais convidadas precisam ser coordenados. Os dispositivos de rede virtualizados são necessário para atender às cargas de trabalho executadas simultaneamente a partir de várias VMs, visto que é importante alocar largura de banda de E/S de rede de forma justa e estável entre várias máquinas virtuais.

Ajuste de largura de banda e atender ao agendamento proporcional dos requisitos de entrega de dados de rede. Servidores de vídeo, etc.

PUC Minas

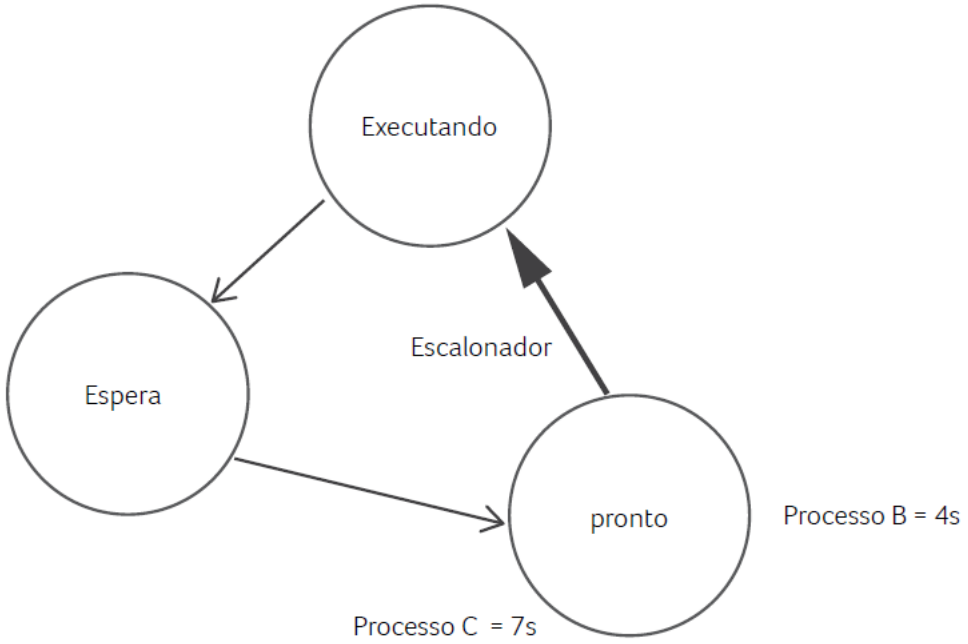
Algoritmo Sorteio (Loteria)

- Imagine dois processos (A e B): o processo A com 40 bilhetes (1 a 40), e o processo B com 20 bilhetes (41 a 60). O escalonador escolhe um número aleatório de 1 a 60. Então, se o escalonador sortear um número de 1 a 40, o processo A é executado; entre 41 a 60, o processo B é sorteado.
- Imagine que o escalonador tenha sorteado oito bilhetes nesse formato (10, 20, 40, 44, 55, 60, 3, 10). Nesse cenário, os processamentos respectivamente seriam de **A, A, A, B, B, B, A, A, ou seja, o processo A seria executado cinco vezes, e o processo B, três vezes.**



Algoritmo SJRF (*Short Job Remain First*)

- Nesse algoritmo, o tempo restante para processamento determina a ordem de execução dos processos na CPU
- **Quando um processo chega a CPU, o sistema operacional verifica se o tempo total desse processo é menor do que o tempo que falta para processar o processo (job) em execução.** Caso seja menor, **o job em execução é suspenso e um novo é inicializado na CPU.**



PUC Minas

Algoritmo SJRF (*Short Job Remain First*)

Esse sistema faz uma preempção (alternância) na escolha dos processos, porém todo o processo é executado.

Jobs curtos tem melhor desempenho e o tempo de processo tem que ser previamente conhecido (TANENBAUM, 2016).

Esse algoritmo é uma versão preemptiva do algoritmo SJF onde trabalha com sistema que são orientados a tarefas de entrada e saída como servidores de rede.

PUC Minas

Algoritmo de Escalonamento em Tempo Real

Um sistema de tempo real não precisa ser ultrarrápido, **mas precisa ter previsibilidade** (ou seja, seu tempo de resposta deve ser conhecido no melhor e pior caso de operação). Tempos de acesso a disco e sincronizações excessivas são essenciais para minimizar esperas e latência.

Existem dois tipos de sistema de tempo real**: Hoftreal time e Hardreal time**. O Softreal time pode ser demonstrado no som ou vídeo online, quando há atrasos é sentido pelo usuário na qualidade do som ou imagem, e o Hardreal time, em que aperda de prazos pelo sistema pode perturbar o objeto controlado, com graves consequências humanas ou ao ambiente, como sistemas de aviação e sistemas hospitalares (UTI), por exemplo.

- São exemplos de sistema de tempo real QNX, RT-Linux e VxWorks.

PUC Minas

Algoritmo de Escalonamento em Tempo Real

Algoritmos de sistemas de tempo real podem ser **estáticos ou dinâmicos**. O algoritmo estático toma a decisão de escalonamento **antes do início da execução das tarefas**, o **dinâmico em tempo de execução.**

Um sistema de detecção de gás pode usar um algoritmo de tempo real para evitar incidentes, ou um sistema de monitoração de conversas em tempo real para contagem de fonemas em tempo real e aplicativo para monitoramento da taxa de fala.

Referências

TANENBAUM, Andrew S.; BOS, Herbert. Sistemas operacionais modernos. 4. ed. São Paulo: Pearson Education do Brasil, c2016.

MACHADO, Francis Berenger; MAIA, Luiz Paulo. Arquitetura de sistemas operacionais. 5. ed. Rio de Janeiro: LTC, 2013. 263 p. ISBN: 9788521622109.

STUART, Brian L. **Princípios de sistema operacionais**: projetos e aplicações. São Paulo: Cenage Learning, 2011. 637 p. ISBN: 9788522107339.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais. 9. ed. Rio de Janeiro, RJ: LTC, c2015.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. Sistemas operacionais. 4. ed. Porto Alegre: Bookman, 2010.