Sistemas operacionais

Aula 3 - Gerência de processador

INTRODUÇÃO

Um ponto decisivo, no desenvolvimento de um Sistema Operacional, é como alocar de forma eficiente o processador (CPU) para os vários processos prontos para serem executados.

A parte do SO responsável por realizar a seleção do processo a ser executado é o escalonador, que distribui o uso da CPU entre os vários processos prontos.

Para determinar a ordem com que os processos serão executados, o escalonador utiliza critérios definidos por um determinado algoritmo ou política de escalonamento de processos.

Esta atividade denomina-se Gerência de Processador, objeto de nosso estudo nesta aula.

OBJETIVOS

Descrever o funcionamento do escalonador;

Distinguir políticas preemptivas de não preemptivas;

Identificar as políticas de escalonamento.

GERÊNCIA DE PROCESSADOR

Fonte da Imagem: Sergey Nivens / Shutterstock

É a parte de um SO em que são definidos a ordem e os critérios para executar os processos em uma máquina.

O procedimento de determinar qual processo será executado, por quanto tempo, em que ordem, dentre os diversos processos que estejam na fila de *prontos* é chamado escalonamento.

ESCALONAMENTO PREEMPTIVO E NÃO PREEMPTIVO

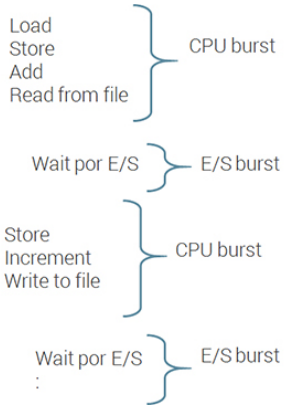
Quantum
É o tempo máximo que um processo pode executar a cada vez que passa ao estado de executando. Após esse tempo ele é interrompido e colocado no fim da fila de prontos.

A maioria dos sistemas operacionais, de uso geral atuais, é preemptiva. Sistemas mais antigos, como o Windows 3.*, PalmOS 3 e MacOS 8 e 9 operavam de forma cooperativa.

TIPOS DE CICLO

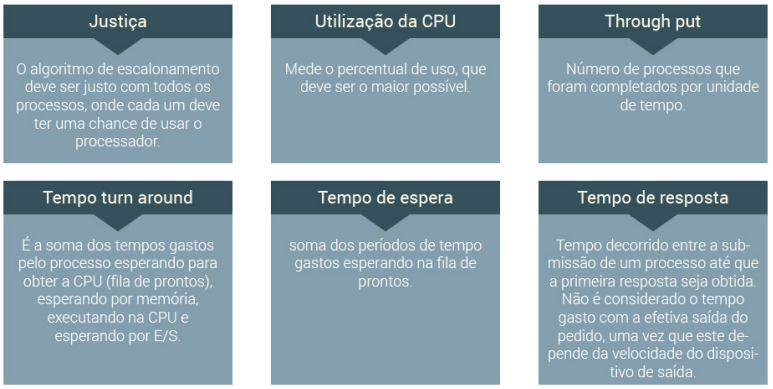
A execução de um processo é uma sucessão de ciclos de execução de instruções (CPU burst) e espera pelo término da operação de entrada e saída (E/S burst).

A execução começa com um CPU burst, seguido de um E/S burst e novamente um CPU burst:



CRITÉRIOS DE ESCALONAMENTO

Alguns critérios devem ser observados na avaliação dos algoritmos de escalonamento:



POLÍTICAS DE ESCALONAMENTO NÃO PREEMPTIVAS

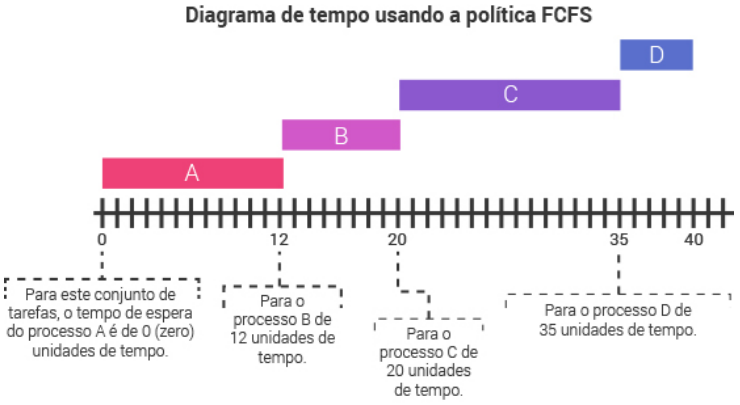
Você já ouviu falar no Escalonamento First Come First Served (FCFS)?

Trata-se do algoritmo de escalonamento de implementação mais simples. Com este algoritmo de escalonamento, o primeiro processo que solicita a CPU é o primeiro a ser alocado. Dessa forma, os processos que estão prontos para serem executados pela CPU são organizados em uma fila, que funciona baseada na política FIFO (glossário).

Vejamos um exemplo. Considere o seguinte conjunto de processos:

Processor	Duração de Execução
A	12
B	8
C	15
D	5

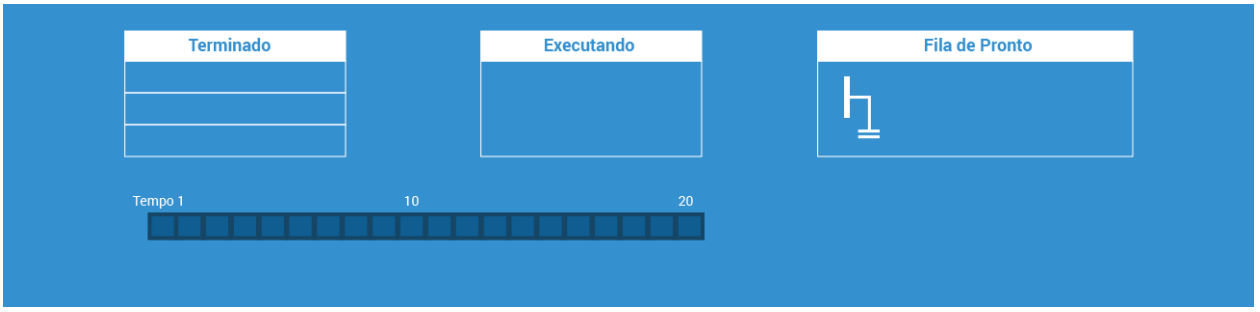
Supondo que a ordem de chegada dos processos seja: A – B – C – D utilizando FCFS, a ordem de execução dos processos é mostrada a seguir:



O tempo médio de espera na fila de prontos é de (**0+12+20+35**)/4, que equivale a **16,75 unidades de tempo**.

Nesta política de escalonamento, o tempo médio de espera é, com frequência, um tanto longo. Outro ponto é que processos importantes podem ser obrigados a esperar devido à execução de outros processos menos importantes dado que o escalonamento FCFS não considera qualquer mecanismo de distinção entre processos.

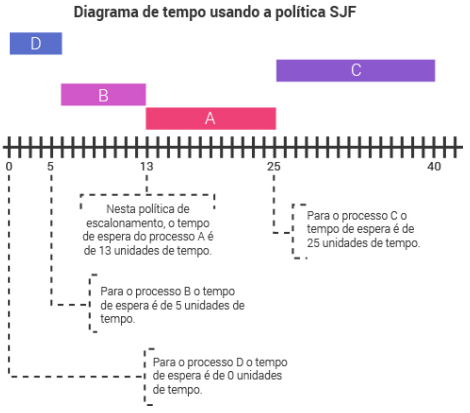
Como vimos, a fila de execução funciona baseada na política FIFO. Veja, a seguir, como essa política funciona:



ESCALONAMENTO SHORTEST JOB FIRST (SJF)

Nesta política, o processo que tem o menor ciclo de processamento (tempo de execução) será selecionado para usar o processador.

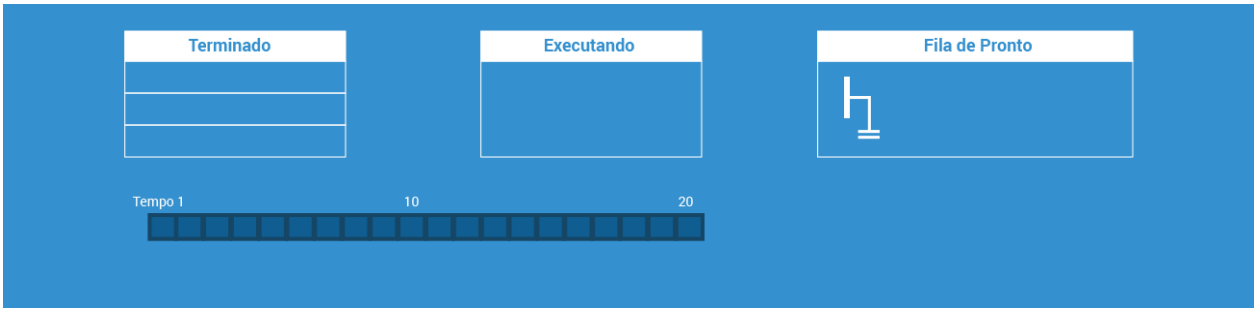
Considerando o mesmo conjunto de tarefas apresentadas, teríamos o diagrama de tempo conforme mostrado a seguir:



O tempo médio de espera na fila de prontos é de (**13+5+25+0**)/4, que equivale a 10,75 unidades de tempo. Em média, nessa política de escalonamento, os processos tiveram que esperar menos para serem executados pelo processador.

Esta política é considerada ótima, mas impossível de implementar na prática já que não há modo de saber o tempo de duração do processo.

Veja, abaixo, um exemplo animado:



, Existe uma variante preemptiva deste algoritmo onde o escalonamento é realizado não com base no tempo total de execução do processo, mas no tempo previsto para a duração de seu próximo CPU *burst*.

POLÍTICAS DE ESCALONAMENTO PREEMPTIVAS

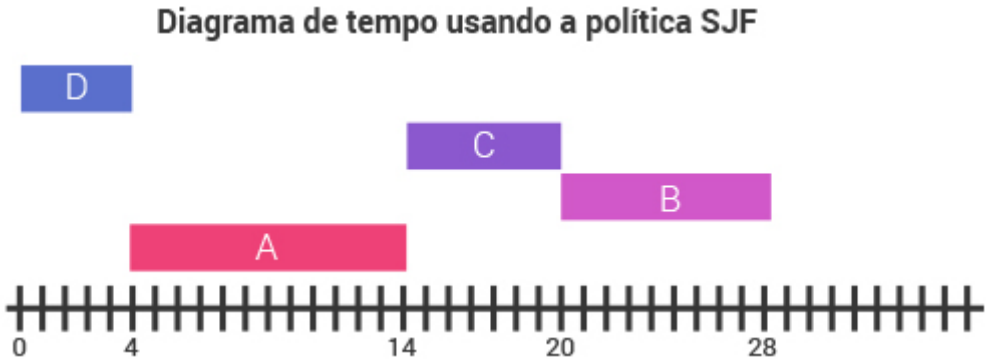
Escalonamento por prioridade

A cada processo é associada uma prioridade e a CPU é alocada para o processo com a mais alta prioridade. Processos com prioridades iguais são escalados segundo a política FCFS. Para facilitar a compreensão, veja o exemplo a seguir.

Considere o seguinte conjunto de processos:

Processor	Prioridade	Duração de Execução
A	2	10
B	4	8
C	3	6
D	1	4

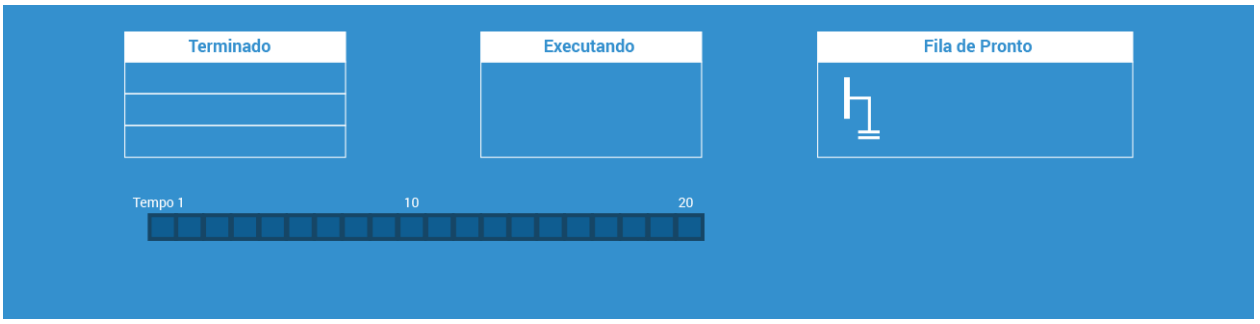
Dessa forma, baseado na política de escalonamento por prioridades (quanto menor o número, maior a prioridade), a ordem de execução dos processos é mostrado a seguir:



A prioridade de um processo pode ser de dois tipos:

Prioridade estática: não muda durante a vida do processo;
Prioridade dinâmica: prioridade ajustada de acordo com o tipo de processamento e/ou carga do sistema.

Veja um exemplo animado:



, Este escalonamento pode ser tanto preemptivo quanto não preemptivo.
O maior problema deste escalonamento é o *Starvation* (processo de baixa prioridade ser indefinidamente postergado).
Um método para prevenir "*starvation*" é o "*aging*". Consiste em ir aumentando gradualmente a prioridade de um processo, à medida que ele fica esperando.

Escalonamento Round Robin ou circular

Especialmente útil para sistemas de tempo compartilhado, permite a implementação da multiprogramação em sistemas monoprocessados.

Cada processo ganha um tempo limite para sua execução. Após esse tempo ele é interrompido e colocado no fim da fila de *prontos*. Este tempo é chamado de fatia de tempo, "*time-slice*" ou *quantum*. Geralmente se situa entre 100 e 300 ms.

Assim, um processo executa durante um quantum específico. Se o *quantum* for suficiente para este processo finalizar, outro processo do início da fila é selecionado para executar. Se durante sua execução o processo bloquear (antes do término do *quantum*), outro processo da fila de *prontos* também é selecionado.

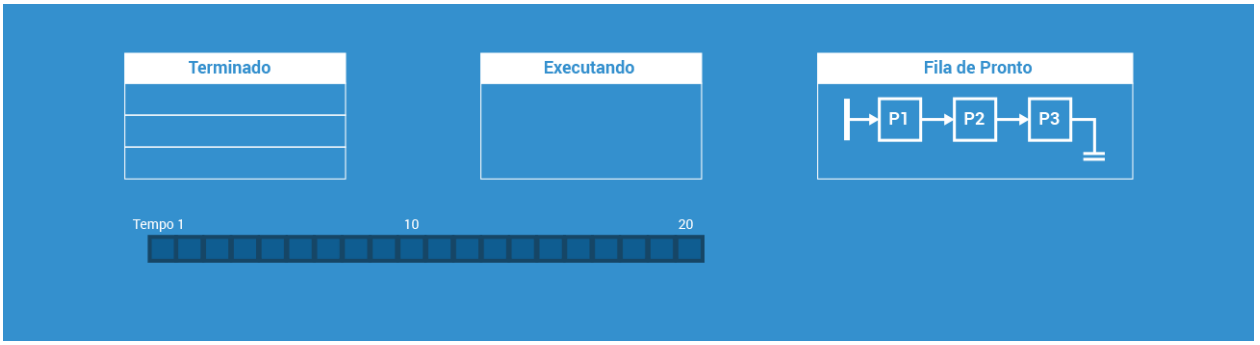
Se terminar a fatia de tempo do processo e ele estiver em execução, o mesmo é retirado do processador, que é disponível para outro processo. Veja o próximo exemplo:

Processor	Duração de Execução
A	12
B	8
C	15
D	5

É o mesmo conjunto de tarefas utilizado no exemplo de FCFS. Para escalonar este conjunto de tarefas utilizando o algoritmo de escalonamento Round Robin, com *quantum* de 4 unidades de tempo, teríamos o diagrama de tempo representado a seguir:

O tempo médio de espera é geralmente longo. O desempenho do algoritmo depende bastante da escolha do *quantum*. A troca de contexto (alternar entre um processo e outro) requer certa quantidade de tempo para ser realizada. Sendo assim, se o *quantum* definido for muito pequeno, ocasiona uma grande quantidade de trocas de processos e a eficiência da CPU é reduzida; de forma análoga, a definição do *quantum* muito grande pode tornar a política Round Robin numa FCFS comum.

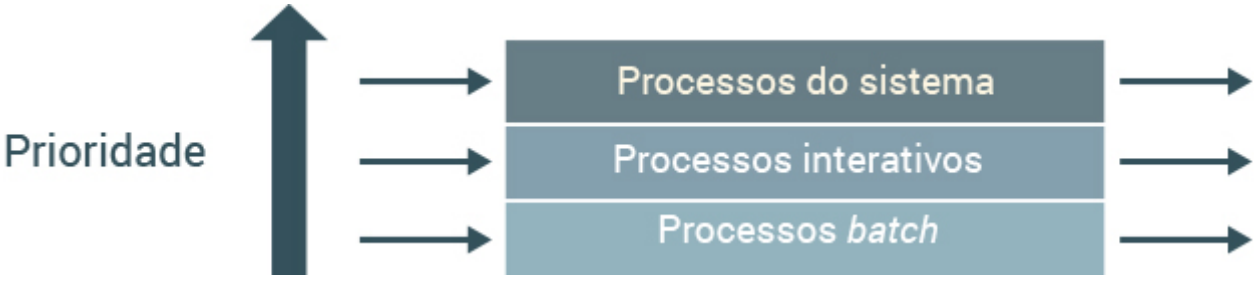
Veja um exemplo animado:



Escalonamento por múltiplas filas

Implementa diversas filas de processos no estado de *pronto*, uma para cada prioridade. Os processos devem ser classificados previamente em função do tipo de escalonamento para poderem ser corretamente alocados.

Cada fila separada tem seu próprio algoritmo de escalonamento. Ex.: fila de *forground* (interativos) usa escalonamento *Round Robin*, enquanto os processos da fila de *background* (*batch*) utilizam algoritmo FCFS.

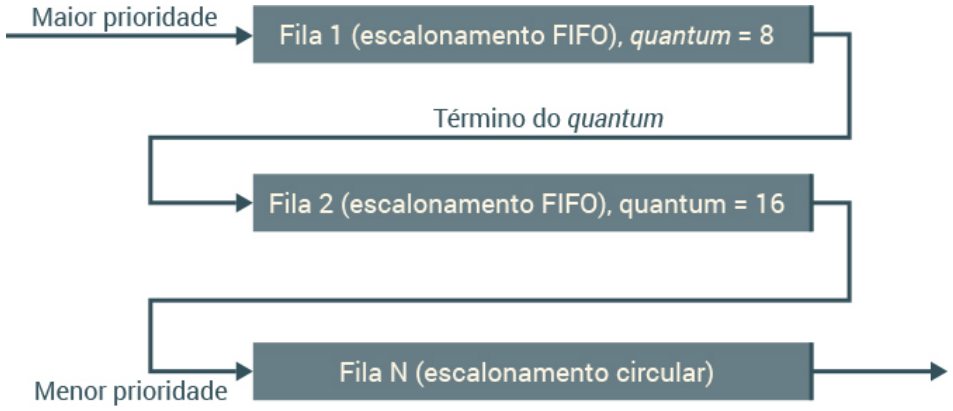


, Note que a prioridade é da fila não do processo.

Escalonamento por múltiplas filas com realimentação

No caso de processos que alterem seu comportamento, no decorrer do tempo, o esquema anterior é falho, porque não permite a movimentação do processo entre as filas. Este escalonamento permite a troca entre as filas.

O sistema tenta identificar dinamicamente o comportamento de cada processo, ajustando suas prioridades de execução e mecanismos de escalonamento. Este esquema é dito mecanismo adaptativo. Veja o exemplo:

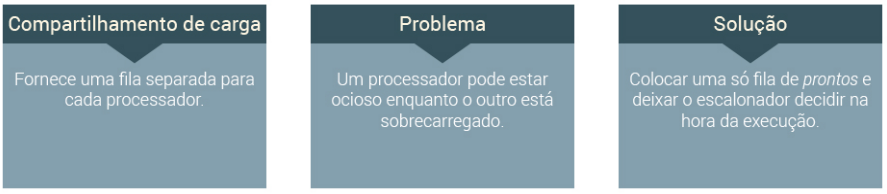


Quando o processo é criado, é posto na fila 1 (de mais alta prioridade). Se ele esgota seu *quantum*, é posto uma fila abaixo e recebe um novo *quantum* (maior). Se também esgota esse *quantum*, vai descendo sucessivamente de fila, até atingir a de menor prioridade, onde o escalonamento é circular.

Neste esquema, os processos "I/O bound" ficam por mais tempo nas filas de alta prioridade, enquanto os "CPU bound" gastam seu tempo e passam para filas de menor prioridade. Ou seja, os processos evoluem bem, independentemente da sua natureza (I/O bound ou CPU bound).

Escalonamento com múltiplos processadores

É usado quando existem vários processadores.



Existem três formas para esse tipo de escalonamento:

Forma 1

Uma maneira de escalonar é o processador ser autoescalonável, ou seja, cada processador vai à fila e seleciona um processo. Cuidados a serem tomados:
- não permitir que um processo seja escolhido por mais de um processador;
- não permitir que um processo não seja escolhido por nenhum processador.

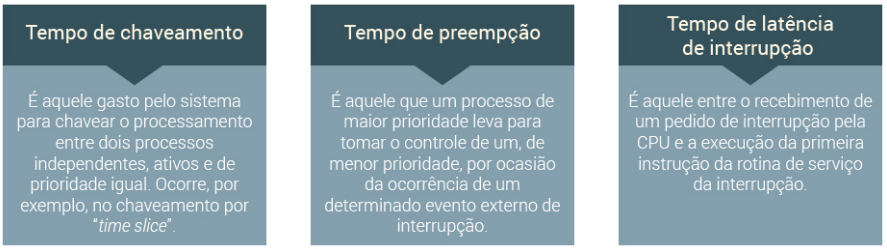
Forma 2

Outra maneira é usar um processador como escalonador de processos para os outros processadores, criando uma estrutura mestre-escravo.

Forma 3

A terceira possibilidade é o multiprocessamento assimétrico, onde um processador é responsável por todas as decisões de escalonamento, processamento de E/S etc. Os demais processadores só executam código do usuário.

Para sistemas de tempo real, é importante definir alguns conceitos críticos:



Escalonamento de tempo real

Podemos dividir o escalonamento de tempo real em dois tipos de sistemas. São eles:

Sistemas de tempo real hard

Nestes sistemas, como as tarefas têm que terminar em tempos definidos, o SO tem que admitir o processo somente quando ele poderá terminar na hora certa; caso contrário, o SO deve rejeitar o pedido como impossível.

Isto é conhecido como "Reserva de recurso". Para ter tempos garantidos, recursos tais como memória virtual e armazenamento em disco, estão descartados. Normalmente, este tipo de sistema possui hardware específico.

Sistema de tempo real soft

Neste tipo, os processos de tempo real devem ter a mais alta prioridade. Essa prioridade não deve ser degradada com o tempo. Para esses processos não deve haver time *sharing*. O tempo de latência deve ser pequeno para que o processo possa começar a executar o mais rapidamente possível.

Em todos os casos, o tempo de chaveamento e de latência devem ser pequenos, o tempo de preempção deve ser adequado aos propósitos do sistema e, em alguns casos, podem existir primitivas de bloqueio da preempção.

ATIVIDADE

Faremos mais uma atividade utilizando o Sosim, sistema que já utilizamos na atividade da aula passada.

Clique **aqui (glossário)** para acessar a atividade.

Após realizar a atividade, clique **aqui (glossário)** e compare com o gabarito.

Glossário

FIFO

First in First out, ou seja, o primeiro a entrar é o primeiro a sair.