ESCALONAMENTO DE PROCESSOS

Quando inicia-se um programa, como o Word, o processador do computador realiza uma série de tarefas para carregar e executar o programa corretamente. O processador é responsável por executar as instruções do programa em sua unidade de processamento central (CPU) e coordenar o acesso a outros componentes do sistema. Mas como isso acontece? Quais os critérios para a execução?

Um dos componentes mais importantes da gerência de tarefas é o escalonador de tarefas (*task scheduler*), ele decide a ordem de execução das tarefas prontas. O algoritmo utilizado no escalonador define o comportamento do sistema operacional, permitindo obter sistemas que tratam de forma mais eficiente e rápida as tarefas para execução, que podem ter características diversas: aplicações interativas, processamento de grandes volumes de dados, programas de cálculo numérico, etc.

Supondo que o usuário abra na sua máquina em primeiro plano o Word, nesse sentido, o processador realiza as seguintes tarefas:

1. Carregamento do programa; em que lê o código do disco rígido ou da memória RAM e carrega-o em registradores de alta velocidade próximos a CPU, permitindo que o processador acesse as instruções do programa.
2. Execução de instruções; o processador executa as instruções do programa uma por uma, podendo envolver a manipulação de dados, exibição de elementos na tela, interação teclado/mouse, entre outras que utilizam o cálculo de prioridade.
3. Acesso a recursos do software e memória; durante a execução do Word, o processador pode precisar dos dados que estão na memória principal, como o conteúdo dos documentos. Pode interagir também com outros componentes do sistema, como o disco rígido para salvar alterações no documento ou a placa gráfica para exibir a interface do Word.

Dessa forma, calcula-se a prioridade o tempo todo, até outro programa chegar. Dito isso, o escalonamento trabalha com a tomada de três decisões:

- Quais trabalhos serão admitidos pelo sistema?
- Quais processos serão mantidos na memória principal?
- Que processo utilizará a CPU quando ela estiver livre?

Para ajudar na resolução dessas questões, há três tipos de escalonadores:

1. Escalonador de longo prazo:
   - Responsável por controlar o grau de multiprogramação do sistema;
   - Número de processos que serão executados “ao mesmo tempo”;
   - Admite novos trabalhos no sistema, convertendo-os em processos.

2. Escalonador de médio prazo:
   - Responsável por escolher os processos que serão removidos totalmente ou parcialmente da memória para serem levados ao disco (suspensos);
   - Manter o rendimento do sistema.

3. Escalonador de curto prazo:
   - Responsável por alocar à CPU os processos alocados na memória.

Contudo, há avaliações que o escalonador define, pois em todo processo há um objetivo, verifica-se então diversos parâmetros, como, por exemplo:

- Justiça: chances iguais de uso do processador;
- Throughput: maximizar o número de jobs processados em uma unidade de tempo;
- Tempo de Resposta: minimizar o tempo de resposta para os usuários interativos;
- Previsível: Rodar as tarefas com igualdade;
- Turnaround: Minimizar o tempo;
- Menor overhead: Tempo de gerência.

Repare que o uso dos recursos deve ser balanceado, o escalonador tenta encontrar um equilíbrio entre a resposta e a utilização, a fim de evitar a postergação indefinida e garantir de fato a prioridade do processo. Por isso, quanto melhor os serviços oferecidos, melhor será o comportamento exibido ao usuário.

Uso do algoritmo de escalonamento e funcionalidades

Existem situações nas quais um processo de escalonamento é necessário. Dentre elas estão ocasiões onde um novo processo é criado.

Quando um processo termina sua execução e um próximo processo pronto deve ser executado ou quando um processo é bloqueado (semáforo, dependência de E/S), resultando que outro processo deve ser executado.

Quando uma interrupção de E/S ocorre, o escalonador deve optar por executar o processo que estava esperando essa interrupção, continuar executando o processo que já estava sendo executado ou executar um terceiro processo que esteja pronto para ser executado.

A política pode ser preemptiva ou não preemptiva.

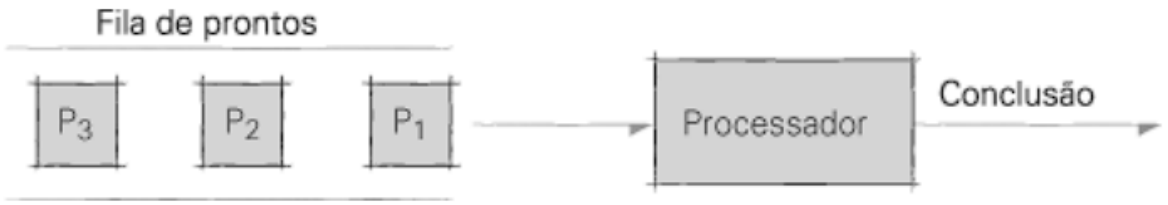
Não preemptivo: O processo que possui a CPU somente a libera quando finaliza a sua execução e não necessita suporte de hardware adicional. Um

processo pode monopolizar a CPU e não é conveniente para ambientes de tempo compartilhado. Exemplo: Windows 3.1 e Apple Macintosh OS.

- Fisrt-Came, Fisrt-Served – FCFS (FIFO):

O processador é alocado seguindo a ordem de chegada dos processos à fila de processos prontos (Figura 1). Em português: "primeiro que entra, primeiro que sai". Processos que passam para o estado de pronto vão para o final da fila e são escalonados quando chegam no início. Possui a vantagem de ser o mais simples entre os processos de escalonamento, todos os processos tendem a ser atendidos. Entretanto, é muito sensível à ordem de chegada, se processos maiores chegarem primeiro aumentarão o tempo médio de espera, não garante um tempo de resposta rápido.

Figura 1: Escalonamento FIFO



Fonte: UFSCAR.

- Shortest-Job-First – SJF:

O processador é alocado ao processo com etapa de CPU mais breve, onde o menor processo ganhará a CPU e atrás do mesmo formar uma fila de processos por ordem crescente de tempo de execução. Possui a desvantagem de baixo aproveitamento ao ter poucos processos prontos para serem executados.

Modo preemptivo: O escalonador pode desalocar um processo da CPU em qualquer instante de tempo. Maior custo, porém evita-se que um processo tenha 100% da CPU.

- Por prioridades:

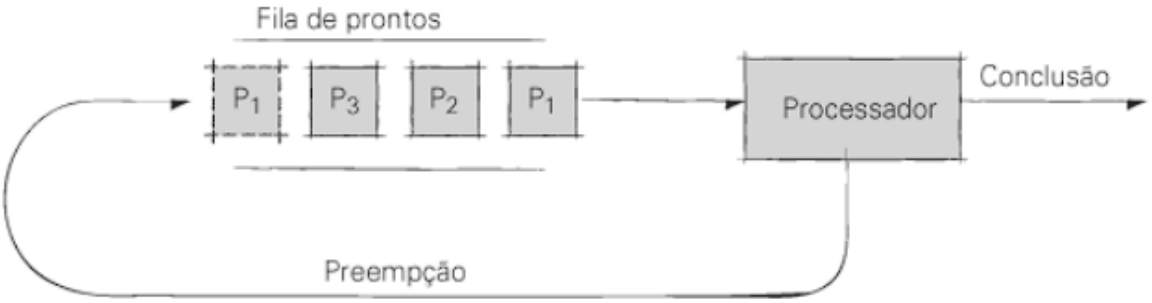
Quando observamos o escalonamento por prioridades, nota-se que cada processo associa um valor inteiro que representa sua prioridade de execução. O escalonador escolhe o processo da fila de processos prontos que tenha a maior prioridade e a fila de processos prontos é ordenada pela prioridade dos processos. Existem duas maneiras de ligar uma prioridade a um processo, podendo ser estática ou dinâmica. A prioridade estática é associada no momento da criação do processo e não é alterada durante a sua existência, o problema é gerar tempos de resposta elevados. A prioridade dinâmica pode ser modificada pelo escalonador durante a

execução do processo, o grau da sua prioridade é gerado conforme a execução desse processo anteriormente.

- Turno rotativo ou Circular (Round-Robin):

O escalonamento Round Robin consiste em dividir o tempo de CPU entre todos os processos prontos para serem executados, se o processo não for concluído no tempo limite (quantum) o sistema operacional interrompe sua execução, inicia o próximo processo da fila e o processo que até então estava sendo executado é colocado em último da fila com uma nova fatia de tempo, assim como na figura 2:

Figura 2: Escalonamento Round Robin.



Fonte: UFSCAR.

Na figura 2 temos os processos P1, P2 e P3 prontos para serem executados, se P1 está sendo processado e não for concluído até o tempo limite, este é retirado do processador e colocado ao final da fila, P1 só irá retornar para o processador assim que P2 e P3 tiverem sido concluídos ou atingido o tempo limite.

- Filas multiníveis:

As filas multiníveis dividem os processos em diferentes filas de prioridade. Cada fila segue sua própria regra de escalonamento. Os processos começam a ser executados nas filas de maior prioridade. Processos de baixa prioridade só são executados quando não há mais processos nas filas de maior prioridade. Nos quadros 1 e 2 um exemplo de como seria o algoritmo de escalonamento de filas multiníveis.

Quadro 1: Escalonamento filas multiníveis.

Processos	T. Chegada	T. Execução	Prioridade
A	0	4	1
B	1	5	1
C	5	5	0
D	8	2	0
E	10	3	2
F	13	2	1

Fonte: Acervo pessoal.

O quadro 1 indica o tempo de chegada, tempo de execução e prioridade de cada processo, sendo os processos C e D de prioridade 0 inseridos na primeira fila; os processos A, B e F de prioridade 1 inseridos na segunda fila; e o processo E de prioridade 2 inserido na terceira fila. O quadro 2 apresenta a ordem de execução de cada processo.

Quadro 2: Escalonamento filas multiníveis.

Processo/Tempo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	A	A	A	A																	
B					B								B	B	B	B					
C						C	C	C	C	C											
D											D	D									
E																			E	E	E
F																	F	F			

Fonte: Acervo pessoal.

Os primeiros processos a serem executados são os processos A e B que possuem prioridade 1, contudo com a chegada de um novo processo C com prioridade 0, o processo B que até então estava sendo executado é retirado do processador e a partir deste ponto os processos C e D de prioridade 0 são executados. Ao finalizar os processos de prioridade 0, o processo B volta a ser executado e em seguida o processo F também de prioridade 1. Por último o processo E de prioridade 2 é executado, vale ressaltar que mesmo o processo E chegando na fila antes que o processo F este foi executado por último devido a sua prioridade ser inferior.

- Múltiplos processadores:

O escalonamento em múltiplos processadores envolve dividir as tarefas entre vários processadores em um sistema operacional com mais de um processador. Existem diferentes formas de fazer isso, como atribuir tarefas a processadores

específicos, equilibrar a carga de trabalho ou considerar a compatibilidade entre tarefas e processadores, o objetivo é otimizar o desempenho do sistema.

- Tempo Real:

Tempo real é um tipo de escalonamento de processos usado em sistemas em que o cumprimento de prazos é essencial. Cada processo possui um prazo associado e o escalonador deve garantir que todos os processos cumpram seus prazos.

Threads e prioridades

Processos podem ser divididos em “pedaços” para que eles não deixem de responder por algum motivo externo, como isso poderia atrapalhar a sua execução, ou para agilizar a programação e execução. Quando programas são divididos em threads, podemos ter partes do processo rodando em paralelo, pois as threads também são escalonáveis e dentro de um processo compartilham o mesmo espaço de endereçamento.

Windows (2000 em diante): processos e threads são associados a classes de prioridade (6 classes para processos e 7 classes para threads), a prioridade final de uma thread depende de sua prioridade, de sua própria classe de prioridade e da classe de prioridade do processo ao qual está associada, assumindo valores entre 0 e 31. As prioridades dos processos, apresentadas aos usuários no gerenciador de tarefas, apresentam os seguintes valores default:

- 24: tempo real;
- 13: alta;
- 10: acima do normal;
- 8: normal;
- 6: abaixo do normal;
- 4: baixa ou ociosa.

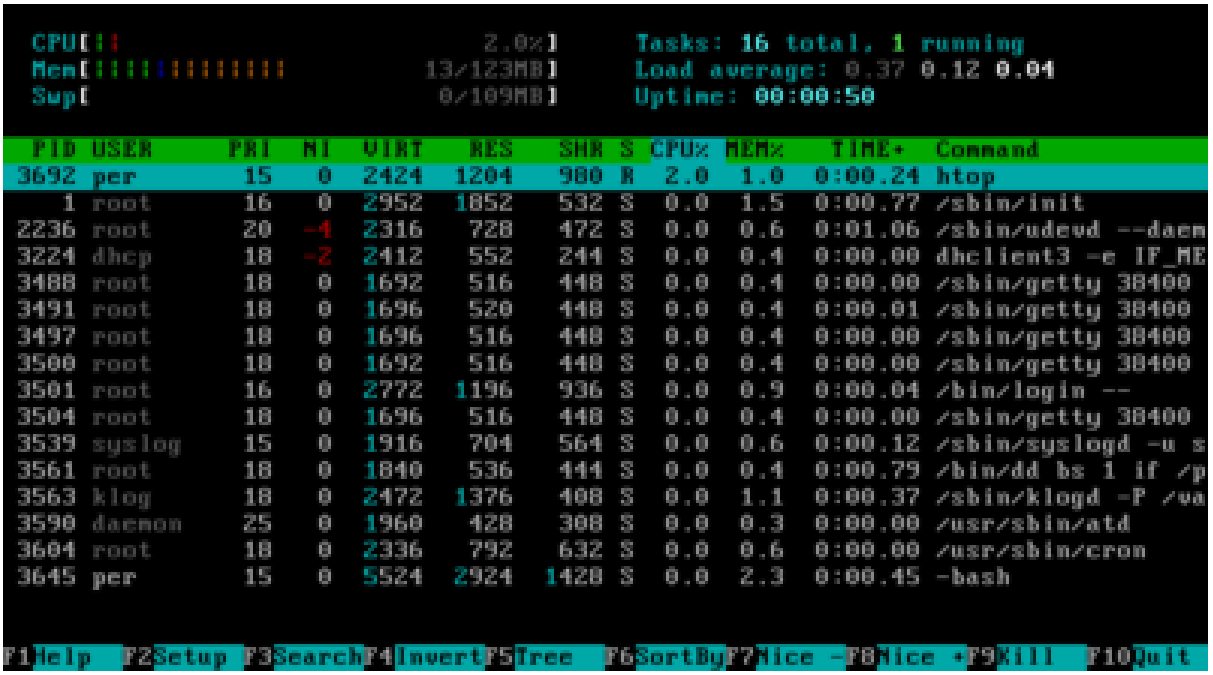
Além disso, geralmente a prioridade da tarefa responsável pela janela ativa recebe um incremento de prioridade +1 ou +2, conforme a configuração do sistema.

Já o Linux (núcleo 2.4 em diante) considera duas escalas de prioridade separadas, as tarefas de tempo real e as demais tarefas. As tarefas de tempo real usam uma escala de 1 a 99 positiva (valores maiores indicam maior prioridade). Somente o núcleo ou o administrador (root) podem lançar tarefas de tempo real. As demais tarefas usam uma escala que vai de −20 a +19 (valores maiores indicam menor prioridade), esta escala denominada nice level, é padronizada em todos os sistemas UNIX-like. A prioridade das tarefas de usuário pode ser ajustada através dos comandos nice e renice.

- Comando nice: altera prioridade do processo escolhido para a execução.
- Comando renice: altera prioridades enquanto o programa já está em execução.

Nas demais tarefas, quanto menor o número maior prioridade, então quando você “desconta” os valores na root, você está aumentando a prioridade do processo.

Figura 3: Lista de processos.



Fonte: https://upload.wikimedia.org/wikipedia/commons/thumb/b/b1/Htop.png/400px-Htop.png

Uma lista de processos como mostrada pelo htop, onde mostra inclusive as prioridades.

Alteração de prioridades no S.O

Existem sistemas que quando um processo inicia sua execução, o sistema garante que este processo terminará, estes são chamados Sistemas Garantidos. Nestes sistemas a inferência do usuário é mínima, ao contrário do que acontece em sistemas de tempo real, como o Windows, em que o usuário interrompe processos a todo instante e, por isso, o sistema não garante que um processo será finalizado.

Vários processos podem ser associados com o mesmo programa, vemos isso quando abrimos várias instâncias do mesmo programa, o que geralmente significa que mais de um processo está sendo executado. Porém, cada CPU executa uma única tarefa por vez, então a partir de um processo em atividade, os próximos podem ter a diminuição do desempenho geral do sistema durante a execução de um determinado programa e de outros programas simultaneamente, como a questão inicial do Word, se ele exigir muitos recursos do processador (operações intensivas, formatação, documentos extensos), poderá consumir uma parte significativa do “poder” de processamento disponível, resultando em um desempenho mais lento dos outros programas que também são dependentes.

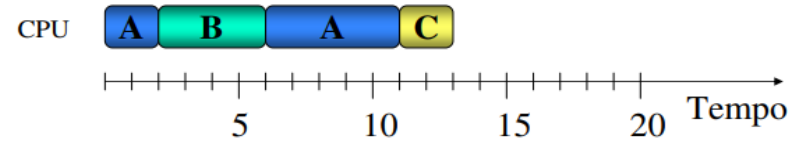
Entretanto, o Word não será executado infinitamente, a partir de outro carregamento, utilizando de exemplo o Chrome, o sistema para o Word automaticamente para ler do disco o próximo evento e, assim, refaz todo o cálculo de prioridades para ver quem será executado. Ambos os programas serão executados pelo sistema operacional, mas a ordem dependerá de diversos fatores que incluem a velocidade relativa de iniciação de cada um e o gerenciamento de recursos do sistema. Se o Word for mais rápido que o Chrome, provavelmente o Word será exibido primeiro em tela de carregamento, o Chrome será iniciado e exibido posteriormente (Vice-versa). Nos dois casos, o sistema estará trabalhando para disponibilizar os recursos.

Caso acrescido mais um programa para execução, supondo o Compilador C++, veremos que esse tipo de programa tende a iniciar mais rápido, pois geralmente não precisa carregar grandiosas bibliotecas ou executar configurações complexas, então nesse caso, se o Compilador C++ for mais rápido do que o Word e o Chrome, ele pode ser exibido primeiro na tela e ser carregado antes dos outros dois, recebendo melhor alocação de recursos. É importante ressaltar que a prioridade depende do estado atual do sistema, ou seja, se o Word ou Chrome estiverem realizando tarefas intensivas, eles podem temporariamente receber uma prioridade mais alta para manter a responsividade. Além disso, se os três exigirem o uso intenso do processador ou de outros recursos, pode haver uma competição entre os programas, o que pode levar a um desempenho mais lento para todos eles e enquanto isso, o cálculo de prioridades é refeito a todo momento.

Portanto, embora um programa com maior prioridade possa receber uma fatia maior do tempo de processamento, ele ainda irá dividir o processador com os demais, o agendamento do S.O garantirá oportunidades de execução e funcionamento equilibrado.

Vamos supor que os processos possuem as prioridades A=5; B=1 e C=6. Considerando 1 como prioridade mais alta e 9 a mais baixa. Conforme a figura 4.

Figura 4: Cálculo de prioridades.



$$TEspera_{medio} = \frac{TEspera_A + TEspera_B + TEspera_C}{n^o\ processos} = \frac{6+0+(11-3)}{3} = \frac{6+0+8}{3} = 4{,}66$$

$$T\,\mathrm{Re}torno_{medio} = \frac{T\,\mathrm{Re}torno_A + T\,\mathrm{Re}torno_B + T\,\mathrm{Re}torno_C}{n^o\ processos} = \frac{11+6+13}{3} = 10$$

Fonte: univasf.edu.br

REFERÊNCIAS:

OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos. Disponível em: <https://www.researchgate.net/profile/Fabiano-Fagundes-3/publication/356223561_OSLIVE_simulacao_do_algoritmo_de_multiplas_filas_do_modulo_de_escalonamento_de_processos/links/6192d9b961f09877209f7737/OSLIVE-simulacao-do-algoritmo-de-multiplas-filas-do-modulo-de-escalonamento-de-processos.pdf>. Acesso em 23 de junho de 2023.

Implementação de um simulador de algoritmos de escalonamento de processos. Disponível em: <https://anais.unicentro.br/siepe/isiepe/pdf/resumo_1326.pdf>. Acesso em 23 de junho de 2023.

Escalonamento de processos. Disponível em: <http://sistemas7.sead.ufscar.br:8080/jspui/bitstream/123456789/2454/1/SO_-_AT1_-_Slides_politicas_de_escalonamento_de_processos_e_threads.pdf>. Acesso em 23 de junho de 2023.

Sistemas operacionais: Conceitos e Mecanismos. Disponível em: <https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-06.pdf>. Acesso em 23 de junho de 2023.

Processo (informática). Disponível em: <https://pt.wikipedia.org/wiki/Processo_(inform%C3%A1tica)#:~:text=V%C3%A1rios%20processos%20podem%20ser%20associados,(UCPs)%20e%20outros%20recursos.>. Acesso em 23 de junho de 2023.

Escalonamento de processos em Sistemas Computacionais distribuídos. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55134/tde-18052005-163302/publico/2-EscalonamentodeProcessos.pdf>. Acesso em 23 de junho de 2023.

Uma abordagem para a avaliação do escalonamento de processos em sistemas distribuídos baseada em monitoramento. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55134/tde-05012018-103739/publico/MarcioAugustodeSouza.pdf>. Acesso em 23 de junho de 2023.