AULA 2 – PROCESSOS, THREADS E GERÊNCIA DO PROCESSADOR

OBJETIVO DA AULA

Definir processo e escalonamento de processos no processador.

Apresentação

Olá!

Sabemos que os computadores atualmente trabalham atendendo a várias demandas do usuário simultaneamente. Por exemplo, você pode estar digitando um texto enquanto ouve música pela internet ou vinda de um arquivo no disco rígido e, ao mesmo tempo, estar realizando um download.

Todas essas atividades ativam programas na memória do computador e esses programas em execução são chamados de processos.

Vamos ver nesta aula exatamente o que é um processo e como o sistema operacional permite que vários processos compartilhem o tempo do processador.

1. INTRODUÇÃO

Imagine que você está usando seu computador para digitar um texto e, enquanto trabalha, coloca um vídeo de uma música no YouTube. Neste momento você ativou dois programas: o editor de texto e o browser.

Mas, além desses programas, há outros rodando no computador e na maioria das vezes você sequer sabe o que eles estão fazendo. Porém, eles usam o mesmo processador e esse uso precisa ser dividido eficientemente por todos esses programas.

Quem exerce esse controle e de que forma isso acontece? É o que vamos começar a ver agora.

2. PROCESSO

O primeiro conceito que precisamos conhecer é o de *processo*. Como vimos anteriormente, uma definição simples para processo é *programa em execução*. Mas processo é algo muito maior do que isso.

Podemos definir processo como uma estrutura criada pelo sistema operacional para que um programa possa ser executado. Assim, toda vez que você aciona um programa como um editor de textos ou um jogo, o sistema operacional entra em ação e cria um processo.

Um processo possui as seguintes informações:

- Espaço de endereçamento: é uma faixa de endereços de memória reservada para que o programa possa usar;
- Contexto de *hardware*: é composto pelo conteúdo do banco de registradores a cada momento. Esse contexto é uma espécie de status do programa e deve ser salvo a cada interrupção. Ele é dinâmico, pois seu conteúdo é alterado a cada instrução executada;
- Contexto de *software*: é composto de quatro informações:

1) PID (*Process IDentification*) – é um número de identificação do processo;

2) *Owner* – é a identificação de quem solicitou a criação do processo, que pode ser um usuário ou outro processo;

3) *Quotas* – definem as quantidades de cada recurso que o processo pode utilizar. Por exemplo, a quantidade de arquivos que ele pode abrir simultaneamente ou a quantidade de subprocessos que ele pode criar.

4) Privilégios – definem as ações que o processo pode executar e as que não pode. Em suma, refletem as permissões dadas ao processo.

Quanto aos estados que um processo pode assumir, eles são os seguintes:

- Executando – é quando o processo está com a posse do processador e, portanto, tem suas instruções sendo executadas;
- Pronto – é quando o processo reúne todas as condições para entrar em execução e apenas aguarda sua vez de pegar o processador;
- Espera ou bloqueado – é quando o processo possui alguma pendência – por exemplo, uma variável compartilhada que está sendo usada por outro processo – e, por isso, não pode entrar em execução ainda que o processador esteja ocioso.

As mudanças de estado acontecem mediante a ocorrência de alguns eventos. Vejamos quais são esses eventos:

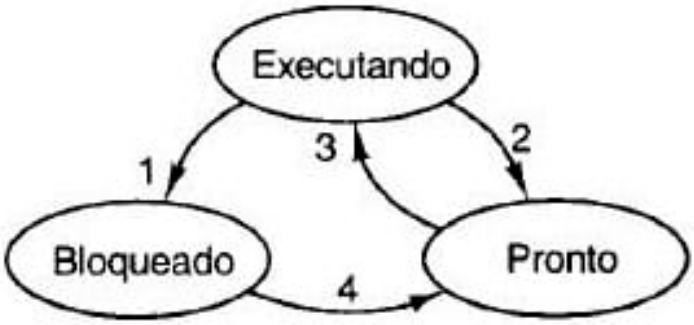
1) De **executando** para **bloqueado**: ocorre quando o processo detecta a necessidade de um recurso (variável ou *hardware*) que está bloqueado por outro processo e sem ele não pode prosseguir;

2) De **executando** para **pronto**: ocorre quando o tempo que o processo tem para usar o processador (*time slice* ou *quantum*) termina ou quando um processo de maior prioridade é criado;

3) De **pronto** para **executando**: ocorre quando chega a vez do processo, que aguarda numa fila (fila de prontos), entrar em execução;

4) De **bloqueado** para **pronto**: ocorre quando a pendência que impedia o processo de entrar em execução é solucionada.

FIGURA 1 | **Mudanças de estado de processo**



Fonte: https://www.oficinadanet.com.br/post/12786-sistemas-operacionais-o-que-e-deadlock.

3. THREADS

As threads, também conhecidas como *processos leves*, são unidades, ou blocos básicos, contendo fluxo de instruções independentes em um mesmo processo que podem ser executados em paralelo, uma vez que não apresentam dependências entre si.

Um processo pode conter várias threads que compartilham todos os seus recursos (quotas e privilégios, além do espaço de endereçamento).

4. ESCALONAMENTO DE PROCESSOS

Essa função executada pelo sistema operacional consiste em gerenciar a fila de processos que disputam o tempo do processador.

Para isso há algumas políticas ou algoritmos de escalonamento.

Antes de estudarmos esses algoritmos, precisamos saber que eles estão divididos em dois tipos: algoritmos preemptivos e algoritmos não preemptivos. No primeiro caso, quando um processo "pega" o processador, ele pode sofrer uma *preempção*, que é um tipo de interrupção causada exclusivamente para dar a vez a outro processo. No segundo caso, a preempção não ocorre, o que significa que o processo que está com o processador fica com ele até terminar todas as suas instruções.

Vale ressaltar que em ambos os casos as interrupções causadas por dispositivos de E/S ocorrem normalmente.

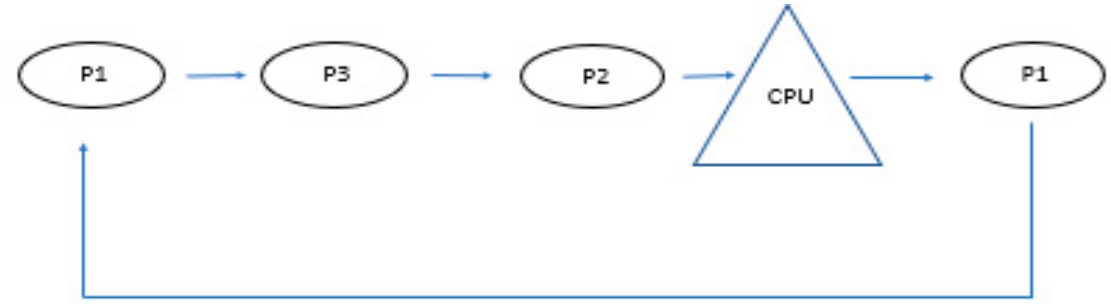
Vamos agora ver os principais algoritmos de escalonamento:

1) FIFO (*First In First Out*) – neste algoritmo os processos são encaminhados ao processador pela ordem de chegada. Como ele é não preemptivo, os processos só "devolvem" o processador quando terminam sua execução.

2) SJF (*Shortest Job First*) – neste algoritmo, também não preemptivo, os processos são enfileirados de acordo com seu tempo de execução, do mais curto. O problema deste algoritmo é que processos com tempo de execução muito grande podem ficar por tempo indeterminado na fila, uma vez que podem ser criados processos com menor tempo de duração. Outro problema, que torna a implementação desse algoritmo difícil, é que nem sempre é possível estimar o tempo que o processo gastará, já que as circunstâncias podem torná-lo mais longo ou mais curto.

3) *Round Robin* (circular) – algoritmo preemptivo, o *Round Robin* atende os processos pela ordem de chegada, dando a cada um uma fatia de tempo (*time slice* ou *quantum*) ao final da qual o processo sofre uma preempção dando lugar ao próximo e retornando ao final da fila onde aguardará a próxima vez. Esse algoritmo é muito usado, pois, além evitar que algum processo não seja atendido, permite um tempo de resposta médio mais interessante quando há vários processos disputando o processador.

FIGURA 2 | **Escalonamento Round Robin**



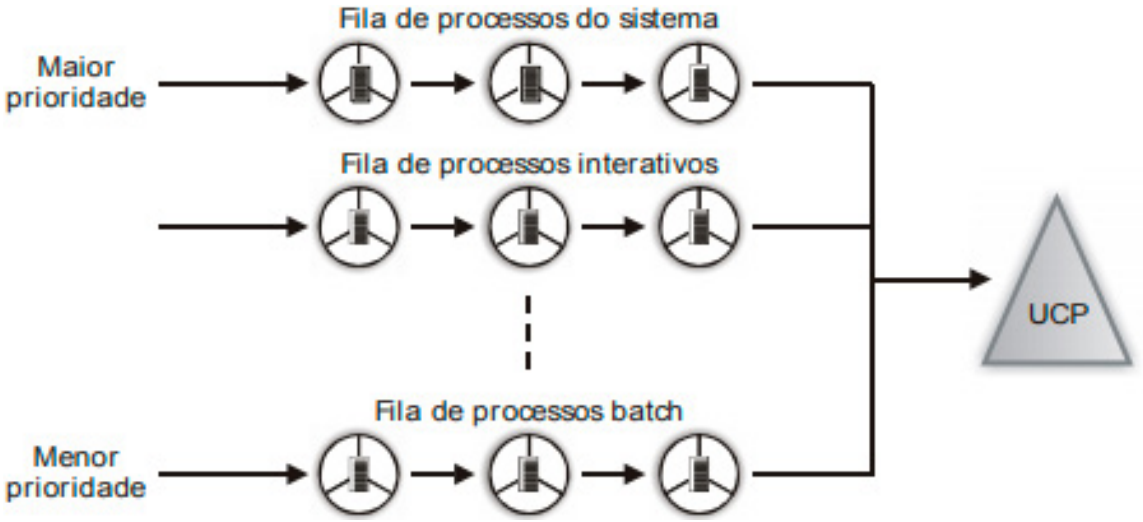
Elaborado pelo autor.

Quanto ao tamanho da fatia de tempo, ela é definida com base na importância do processo e não necessariamente é igual para todos os processos.

4) Escalonamento por prioridades – algoritmo preemptivo em que a cada processo é associada uma prioridade e o atendimento aos processos é feito de acordo com essa prioridade, de forma que os mais importantes são atendidos primeiro. Nesse caso, se um processo de maior prioridade chega na fila, ele "toma" o processador daquele que está executando e só o libera se chegar outro processo com maior prioridade ou quando termina todas as suas tarefas.

5) Escalonamento por múltiplas filas – nesse algoritmo, que também é preemptivo, os processos são organizados em filas por nível de prioridade de forma que uma fila de menor prioridade só é atendida se todas as filas com maior prioridade estiverem vazias. Para evitar que processos das filas de mais baixa prioridade fiquem muito tempo sem serem atendidos, há mecanismos de incremento de prioridade para esses processos para que eles mudem de fila eventualmente.

FIGURA 3 | **Escalonamento por múltiplas filas**



Fonte: https://www.passeidireto.com/arquivo/69218857/escalonamento-por-multiplas-filas.

Na Figura 3 vemos três filas de processos. Na fila de mais alta prioridade estão os processos do próprio sistema operacional, que, como vimos, gerenciam os recursos de *hardware* e *software*.

Na segunda fila estão os processos interativos, ou seja, processos do usuário, que precisam de um tempo de resposta razoável.

Finalmente, na fila de mais baixa prioridade estão os processos cujo resultado não precisa ser tão rápido e por isso são executados somente quando as filas superiores estiverem vazias.

Nesse tipo de escalonamento cada fila pode funcionar com um algoritmo diferente. Assim, por exemplo, na fila de menor prioridade pode ser usado o algoritmo FIFO, na segunda fila o Round Robin e na primeira fila o escalonamento por prioridades.

Cada algoritmo desses que estudamos atende em maior ou menor nível aos seguintes parâmetros:

- Justiça – todo processo deve ser atendido em algum momento;
- Taxa de utilização da CPU – o algoritmo deve evitar ao máximo deixar a CPU ociosa, minimizando a quantidade de trocas de contexto, que ocorrem cada vez que um processo dá lugar a outro no uso da CPU;
- Tempo de resposta – cada processo deverá responder às solicitações do usuário no menor tempo possível;
- *Turnaround time* – é o tempo total em que o processo existe, desde o momento em que foi criado até o momento em que foi encerrado. Isso é um somatório dos tempos em que passou em cada um dos três estados que vimos (executando, pronto e bloqueado) e deve ser minimizado;

- *Throughput* – é a quantidade de processos que são concluídos em uma quantidade específica de tempo. Este parâmetro deve ser maximizado.

O atendimento melhor ou pior a esses parâmetros varia entre os algoritmos estudados. Não é possível atender bem a todos, já que muitas vezes eles são conflitantes entre si.

CONSIDERAÇÕES FINAIS

Chegamos ao fim de mais uma aula.

Nela conhecemos um dos conceitos mais importantes para o estudo de sistemas operacionais: processo. Um processo é o ambiente ou infraestrutura que o sistema operacional cria para que um programa possa ser executado.

Os processos podem estar em três estados: executando, pronto e bloqueado e suas informações são divididas em contexto de *hardware*, contexto de *software* e espaço de endereçamento.

Vimos também que a função de escalonamento de processos consiste em organizar a fila de processos sob algum critério de forma que o uso do processador seja o melhor possível atendendo aos parâmetros de justiça, tempo de resposta, taxa de utilização da CPU, *turnaround time* e *throughput*.

Agora é hora de vermos como o sistema operacional gerencia a memória do computador.

MATERIAIS COMPLEMENTARES

Neste vídeo você entenderá um pouco mais sobre threads: https://www.youtube.com/watch?v=xNBMNKjpJzM.

REFERÊNCIAS

STALLINGS, William. *Arquitetura e Organização de Computadores: projeto para o desempenho*. 8ª edição. Editora Pearson. Livro. (642 p.). ISBN 9788576055648. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788576055648>. Acesso em: 16 out. 2022.

TANENBAUM, Andrew S. *Sistemas Operacionais Modernos*. 3ª edição. Editora Pearson. Livro. (674 p.). ISBN 9788576052371. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788576052371>. Acesso em: 16 out. 2022.

TANENBAUM, Andrew S. *Organização estruturada de computadores*. 6ª edição. Editora Pearson. Livro. (628 p.). ISBN 9788581435398. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788581435398>. Acesso em: 16 out. 2022.