TEXTO DE APOIO

AULA 2

Introdução aos Sistemas Operacionais

Professor Jean Marcos Laine

EaD | Universidade Presbiteriana Mackenzie

Sumário

PROCESSOS E THREADS

Hoje em dia, todos os sistemas operacionais conseguem executar várias tarefas ao mesmo tempo. Esses sistemas são chamados de multiprogramados. A CPU muda de um programa para outro rapidamente, executando cada um por dezenas ou centenas de milissegundos apenas. Esse gerenciamento faz parte dos serviços implementados pelo sistema operacional; nesse caso, o módulo responsável é o escalonador de processos.

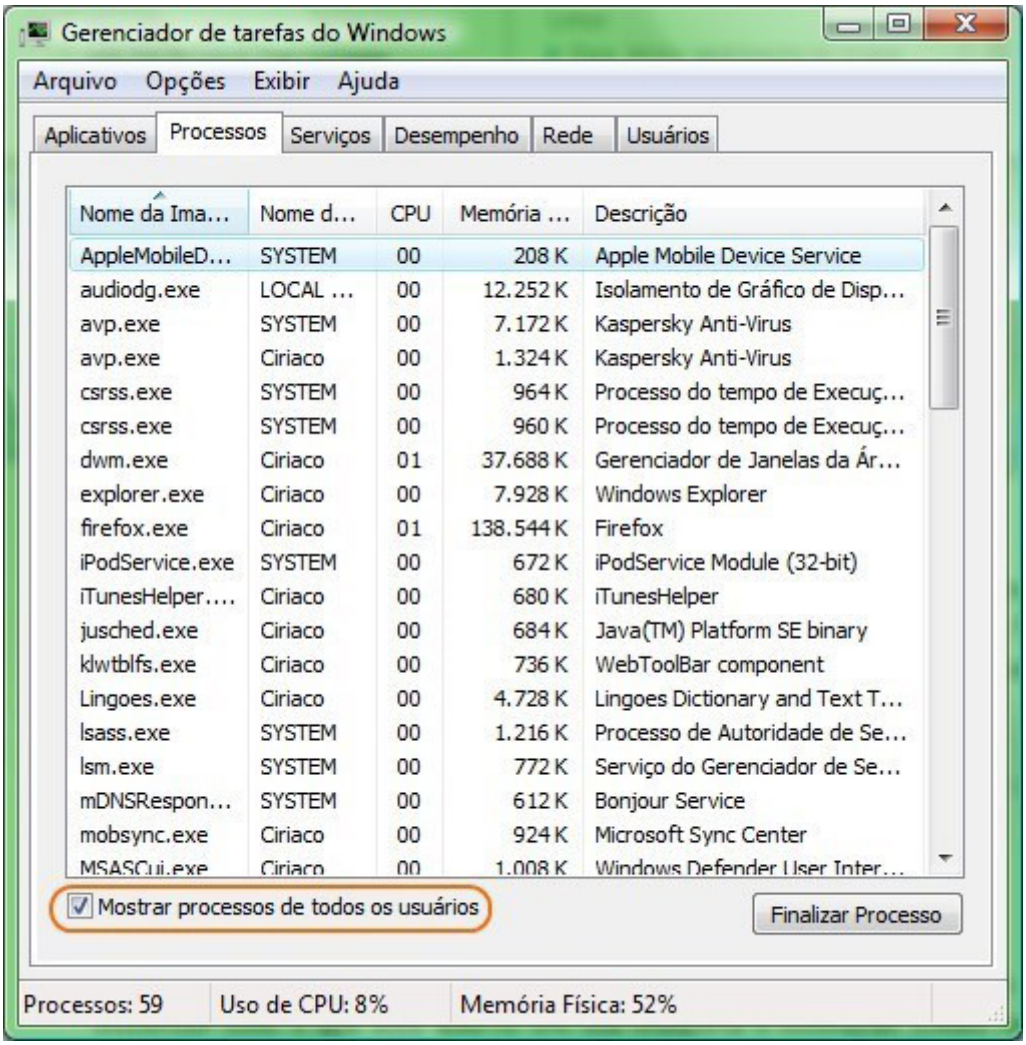
Nos computadores que possuem um processador com vários núcleos, o sistema operacional realiza este controle em cada núcleo, permitindo que, ao mesmo tempo, vários programas sejam executados em paralelo. Nos processadores com um único núcleo os programas concorrem pelo uso da CPU, disputando a chance de entrarem em execução.

Nesse contexto, é conveniente diferenciar os conceitos de programa e processo.

Programa: é uma entidade estática composta por uma sequência de instruções escrita para realizar algum processamento. Exemplo: um programa para ler as notas e calcular a média final de um aluno.

Processo: é uma entidade dinâmica e efêmera, que altera seu estado à medida que avança sua execução. O processo representa um programa em execução, basta olharmos a aba Processos no gerenciador de tarefas do Windows (Figura 1) que enxergamos uma lista de processos relacionados a todos os programas que estão sendo executados naquele instante no computador. Sempre que pedimos para executar um programa, o sistema operacional cria um processo para representá-lo. Por meio desse processo é que o sistema operacional gerencia a execução do programa na máquina.

Figura 1 – Gerenciador de tarefas do Windows



Fonte: https://www.tecmundo.com.br/windows-xp/3567-fique-craque-com-estas-dicas-para-o-gerenciador-de-tarefas-do-windows.htm

O sistema operacional é responsável pelas seguintes atividades relacionadas ao gerenciamento dos processos:

- Executar o escalonamento de processos nas CPUs.
- Criar e excluir processos de usuário e do sistema.
- Suspender e retomar a execução dos processos.
- Fornecer mecanismos de sincronização de processos.
- Fornecer mecanismos de comunicação entre processos.

Um processo precisa de certos recursos, incluindo tempo de CPU, espaço em memória, arquivos e dispositivos de entrada e saída para cumprir sua tarefa. Esses recursos são fornecidos ao processo quando ele é criado, ou são alocados a ele durante sua execução. Para melhorar tanto a utilização dos recursos da máquina, como a CPU, quanto para melhorar o tempo de resposta do computador para seus usuários, os computadores devem manter vários programas na memória prontos para serem executados. Cabe ao sistema operacional realizar todas essas funções.

ESTRUTURA DE UM PROCESSO

Associado ao processo, existe um conjunto de informações relevantes que permite controlar o estado e o momento de execução do programa, como o conteúdo dos registradores da CPU, os valores das variáveis existentes no programa, a pilha de execução, os endereços de memória reservados ao programa etc. Essa estrutura é representada na Figura abaixo.

Figura 2 – Estrutura de um processo

Fonte: MACHADO; MAIA (2007).

Pela figura, é possível compreender que um processo contempla (i) o código do programa que ele representa mais (ii) um conjunto de informações de controle necessárias para o gerenciamento de sua execução.

Cada processo é representado, no sistema operacional, por um bloco de controle de processo (PCB – Process Control Block), também chamado bloco de controle de tarefa. Essa estrutura contém muitas informações associadas a um processo específico, como:

- Estado do processo.

- Contador do programa: indica o endereço da próxima instrução a ser executada para esse processo.

- Registradores da CPU: os registradores variam em número e tipo, dependendo da arquitetura do computador. Eles incluem acumuladores, registradores de índice, ponteiros de pilhas e registradores de uso geral etc. Junto com o contador do programa, essas informações de estado devem ser salvas quando ocorre uma interrupção, para permitir que o processo seja retomado corretamente mais tarde.

- Informações de escalonamento da CPU: incluem a prioridade de um processo, ponteiros para filas de escalonamento e quaisquer outros parâmetros necessários.

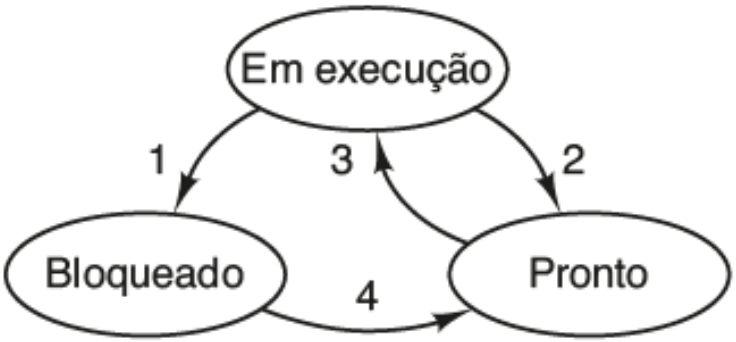
- Informações de gerenciamento da memória: podem incluir itens como o valor dos registradores base e limite e as tabelas de páginas, ou as tabelas de segmentos, dependendo do sistema de memória usado pelo sistema operacional.

ESTADOS DE UM PROCESSO

Durante sua existência, um processo passa por vários estados. Após ser criado, o processo é adicionado em uma fila de execução e fica esperando o escalonador o escolher para que ele entre em um ciclo de CPU e comece a execução das instruções que compõem o programa. No entanto, devemos entender que vários processos são criados e todos disputarão o uso da CPU. Nesse caso, o que o sistema operacional deve fazer?

Quando um processo é executado, ele muda de estado e o sistema operacional é quem realiza esse controle. O estado de um processo é definido, em parte, pela atividade corrente do processo. Um processo pode estar em um dos seguintes estados (Figura 3):

Figura 3 – Tipos de kernel dos sistemas operacionais



Fonte: TANENBAUM (2016).

Esses nomes variam entre os sistemas operacionais, mas os estados que eles representam são encontrados em todos os sistemas, algumas vezes com até mais detalhes.

Os processos podem ser criados por diversas razões, tais como: inicialização de um programa, execução de um determinado serviço do sistema operacional ou chamadas de sistemas, por exemplo.

Normalmente, eles são finalizados ao final da execução do programa. Entretanto, algumas outras situações podem ocasionar o encerramento de um processo:

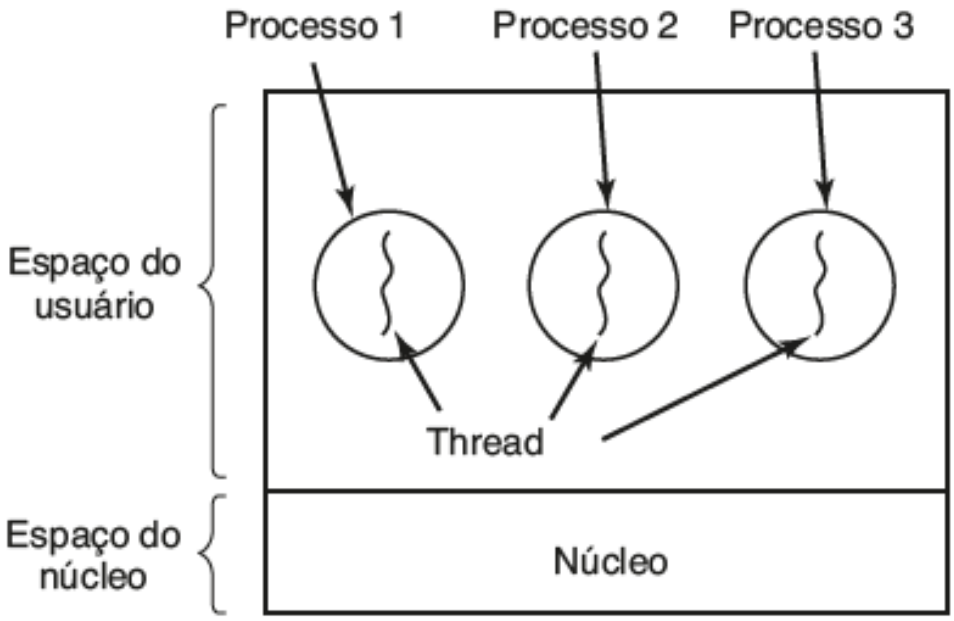
- Situações “anormais”, como falta de memória durante a execução do programa ou por erros como uma divisão por zero.
- Erro em periféricos de E/S.
- Execução de instruções inválidas.
- Intervenção do sistema operacional.
- Log off de usuários.

THREADS

Um thread representa um fluxo de controle e execução dentro de um programa. Todo programa sequencial possui um único thread de execução. Esse thread de controle único permite que o processo execute apenas uma tarefa de cada vez. É o modelo de programação aprendido, por exemplo, na disciplina de algoritmos e programação do primeiro semestre. Um programa monothread não consegue fazer duas ou mais coisas simultaneamente. O usuário de um editor de texto não consegue escrever o texto e, ao mesmo tempo, ter um módulo de corretor ortográfico sendo executado pelo programa.

Hoje em dia, a maioria dos sistemas operacionais estendeu o conceito de processos para permitir que um processo tenha múltiplos threads de execução e, assim, desempenhe mais de uma tarefa de cada vez. Esse recurso é importante em sistemas multicore, em que múltiplos threads podem ser executados em paralelo. Em um sistema que suporte threads, o PCB é expandido de modo a incluir informações para cada thread. Também são necessárias outras alterações no sistema como um todo para que ele suporte threads. A Figura 4 ilustra o conceito de threads e processos.

Figura 4 – Processos e threads



Fonte: TANENBAUM (2016).

Os principais benefícios da programação com múltiplos threads são:

- Capacidade de resposta: uma aplicação multithreaded pode permitir que um programa continue sendo executado mesmo que parte dele esteja bloqueado, aumentando a capacidade de resposta para o usuário.

- Compartilhamento de recursos: os threads compartilham a memória e os recursos do processo ao qual pertencem.

- Economia: alocação de memória e recursos aos processos é custosa. Como os threads compartilham os recursos do processo ao qual pertencem, é mais econômico criar threads e permutar seus contextos.

- Escalabilidade: permitir que os threads possam ser executados em paralelo, em diferentes núcleos de processamento.

ALGORITMOS DE ESCALONAMENTO

O escalonador é a entidade do sistema operacional responsável por selecionar um processo pronto na fila de execução e dividir o tempo do processador de forma justa entre os demais processos que disputam o uso da CPU. Em outras palavras, o objetivo dos escalonadores é implementar uma política de gerenciamento para os processos.

O sistema operacional possui um módulo responsável por efetuar a troca de contexto entre a execução de processos distintos, chamado de Dispatcher. O escalonador, por sua vez, está relacionado com a implementação e aplicação das políticas de seleção adotadas, com os seguintes objetivos:

1. Maximizar a utilização do processador.

2. Privilegiar aplicações que são críticas.

3. Maximizar a produção do sistema (throughput).

4. Minimizar o tempo de execução.

5. Minimizar o tempo de espera.

6. Minimizar o tempo de resposta.

TIPOS DE ESCALONADORES

Os escalonadores são classificados em:

1. ***preemptivos***: escalonadores capazes de suspender processos antes do término de sua execução e passar o controle a outro que aguarda na fila para ser executado.

2. ***não-preemptivo***: esses escalonadores permitem que os processos rodem até o fim de sua execução sem nenhuma interrupção.

Existem vários algoritmos de escalonamento na literatura que são usados na implementação dos sistemas operacionais. Os mais conhecidos são:

- FCFS (First com, first served).
- SJF (Shortest Job First).
- Round-robin.
- Prioridade.

FCFS

É o algoritmo mais simples de implementar. A CPU possui uma fila de execução para armazenar os processos, e a ordem de chegada na fila define a ordem de execução. Nesse escalonamento, o processo que solicita a CPU primeiro poderá usá-la. A implementação da política FCFS é facilmente gerenciada com uma fila FIFO. Quando um processo entra na fila de prontos, seu PCB é inserido no final da fila. Quando a CPU está livre, ela é alocada ao processo que tem sua referência representada no início da fila. O processo em execução é então removido da fila.

SJF

Esse algoritmo associa a cada processo a duração do tempo de execução do processo. Quando a CPU está disponível, o processo que tem o menor tempo de execução é escolhido. Se dois ou mais processos tiverem os tempos iguais, o escalonador pode usar o FCFS para desempate.

Round-robin

Essa política de escalonamento é semelhante ao FCFS, mas a preempção é adicionada para habilitar o sistema a trocar os processos que estão em execução. Uma pequena unidade de tempo, chamada quantum ou timeslice, é definida. Esse tempo tem duração de 10 a 100 milissegundos. A fila de prontos é tratada como uma fila circular. O escalonador percorre a fila alocando a CPU ao primeiro da fila e por um intervalo de no máximo igual ao quantum.

Prioridade

Uma prioridade é associada a cada processo, e a CPU é alocada ao processo com a prioridade mais alta. Processos com prioridades iguais são organizados por ordem de chegada na fila. O algoritmo SJF é simplesmente um algoritmo por prioridades em que a prioridade é o tempo de execução do processo. Quanto maior o tempo de execução, menor a prioridade.

IMPASSES

Em um ambiente multiprogramado, é possível que dois ou mais processos ou threads tentem acessar simultaneamente os mesmos recursos computacionais. Uma situação em que vários processos acessam e manipulam os mesmos dados concorrentemente e o resultado da execução depende da ordem específica em que o acesso ocorre é chamada uma condição de corrida.

Exemplo: um arquivo texto compartilhado por duas pessoas em uma rede.

- Existe algum problema se as duas pessoas apenas leem o arquivo?

- E se alguém estiver modificando o conteúdo do arquivo?

Considere a seguinte situação: dois programas (A e B) acessam os mesmos dois arquivos (R1 e R2) para atualizar informações salvas. Se o programa A acessa primeiro o arquivo R1 e B o arquivo R2, quando eles tentam acessar o outro arquivo não conseguem. Nesse momento, um programa fica esperando pelo outro, aguardando a liberação do recurso que não ocorrerá. Esse impasse é chamado de Deadlock.

Existem quatro condições necessárias para a ocorrência de impasses (deadlocks), são elas:

- Exclusão mútua.
- Posse e espera.
- Não preempção.
- Espera circular.

Os impasses são difíceis de serem detectados e tratados em tempo real e, por isso, uma alocação cuidadosa dos recursos deve ser pensada e realizada quando vários processos ou threads compartilham os mesmos recursos no sistema.

REFERÊNCIAS

MACHADO, F. B.; MAIA, L. P. *Arquitetura de Sistemas Operacionais*. 4. ed. Rio de Janeiro: LTC, 2007.