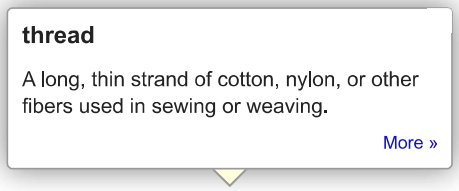
VOLTAR



Threads

Neste tópico iremos estudar o conceito de Thread e a fundamental importância para os SO atuais.

NESTE TÓPICO

Marcar tópico

Introdução

Processos (programas em execução) podem ser divididos em instruções ou grupo de instruções e, cada uma delas ser executada em processadores diferentes. Esse é o conceito de Thread.

Segundo Machado, a utilização comercial de sistemas operacionais multithread é crescente em função do aumento de popularidade dos sistemas com múltiplos processadores, do modelo cliente-servidor e dos sistemas distribuídos.

Thread

A execução de um programa acontece ou por meio de uma ação do SO ou por meio de uma intervenção do usuário (que “abre” o aplicativo”). Quando o usuário abre um aplicativo, o SO interpreta a ação e requisita tudo o que é necessário para que esse aplicativo (software) seja executado, nesse momento ele é carregado na memória RAM.

Todo programa em execução gera um processo e, alguns programas possuem uma árvore de processos, ou seja, cada qual com uma lista de instruções a serem executadas pelo processador. A lista é composta por linhas de instruções (código fonte) módulos executáveis, os quais contêm linhas de código para que a execução do programa seja realizada apropriadamente, ou seja, informam ao processador os passos a serem executados

O Thread é uma divisão do processo principal, são linhas de instruções de um processo principal. Essa divisão de linhas de instruções dentro dos processos dividem-se em 2 tipos:

- Monothread;
- Multithread.

Monothread

Nesse ambiente um processo suporta apenas um programa no seu espaço de endereçamento. Aplicações concorrentes são implementadas apenas com o uso de múltiplos processos independentes ou subprocessos.

Para os processos independentes (são processos que não interferem na execução de outros processos) e subprocessos, torna-se viável dividir uma aplicação em partes e fazer com que executem concorrentemente.

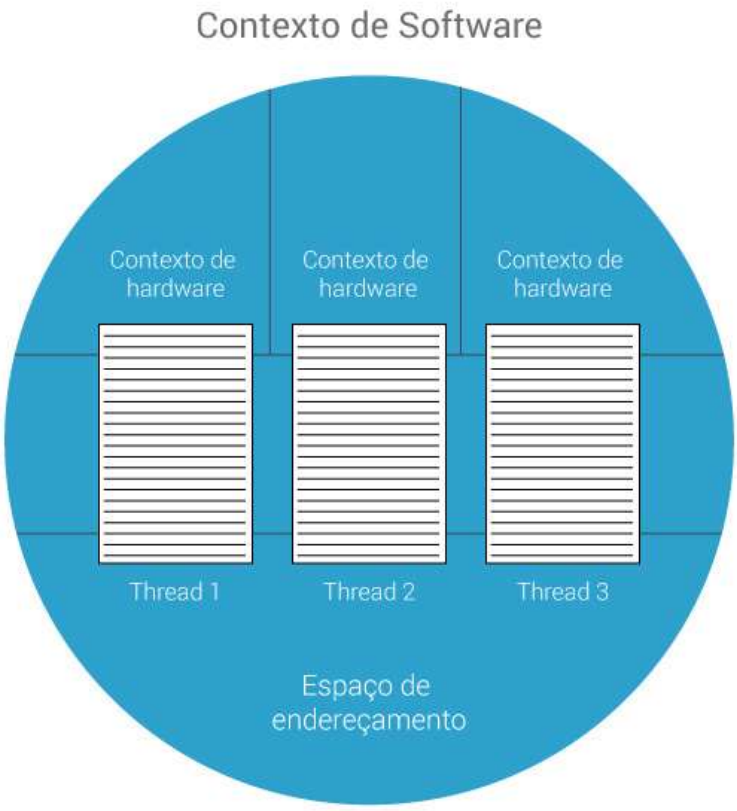
MS-DOS e as primeiras versões do Ms Windows utilizavam ambiente monothread.

Multithread

Nessa configuração o processo tem pelo menos um thread de execução, mas pode compartilhar o seu espaço de endereçamento com outros threads.

Para Machado, thread pode ser definida como uma sub-rotina de um programa e pode ser executada de forma assíncrona, ou seja, executada concorrentemente ao programa chamador. O programa deve especificar os threads, associando-os às sub-rotinas assíncronas. Dessa forma, um ambiente multithread possibilita a execução concorrente de sub-rotinas dentro de um mesmo processo.

Em ambiente com múltiplos processadores, cada processo pode responder a várias solicitações concorrentemente ou simultaneamente. A vantagem no uso de threads é a possibilidade de minimizar a alocação de recursos do sistema, tais como processador e memória. As threads compartilham contexto de software e espaço de endereçamento, mas o contexto de hardware cada um tem o seu.



Multithread.

A principal diferença entre os ambientes monothread e multithread está no uso do espaço de endereçamento (alocação de memória).

Threads compartilham o espaço dentro de um único processo. Dessa forma, permite que o compartilhamento de dados entre threads de um mesmo processo seja mais simples e rápido, tendo como base os ambientes monothread.

Conforme acima, threads de um único processo compartilham o mesmo espaço de endereçamento, portanto, não há proteção no acesso à memória, permitindo que um thread possa alterar facilmente dados alheios. Por isso, é fundamental que a aplicação implemente mecanismos de comunicação e sincronização entre threads, com o objetivo de garantir o acesso seguro aos dados compartilhados na memória.

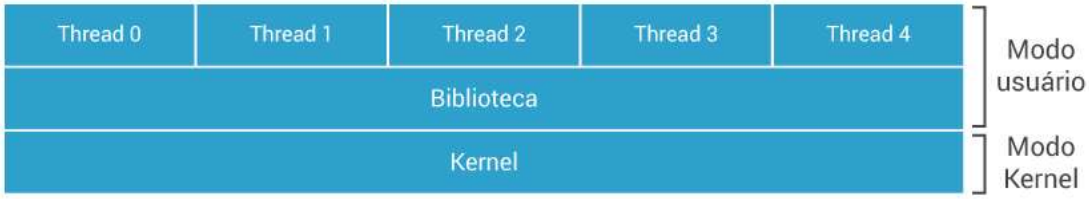
O uso de threads traz para o sistema operacional muitos benefícios, pois programas concorrentes com múltiplos threads são mais rápidos do que implementados com múltiplos processos, pois operações de criação e troca geram menor overhead. Como dito anteriormente, threads dentro de um processo dividem o mesmo espaço de endereçamento, logo, a comunicação entre eles pode ser realizada de forma rápida, causando menos interrupção no sistema operacional.

Segundo Machado (2007, pg. 96), existem diversas abordagens na implementação dos pacotes de threads em um sistema operacional, o que influenciará no desempenho, na concorrência e na modularidade das aplicações multithread. Threads podem ser oferecidos por uma biblioteca de

rotinas fora do núcleo do sistema operacional (modo usuário), pelo núcleo do sistema operacional (modo kernel), uma combinação de ambos (modo híbrido).

Thread de Usuário

Essa modalidade de thread é implantada por aplicações e não pelo sistema operacional. Para que isso seja possível, deverá existir uma biblioteca de rotinas que permita à aplicação realizar a criação / eliminação de threads, a comunicação e o escalonamento. O sistema operacional não tem a responsabilidade de gerenciamento e sincronização entre as threads.



Thread de Usuário.

Nesse modelo é possível implementar aplicações multithreads, caso o sistema operacional não dê suporte para threads, por meio de bibliotecas, múltiplos threads podem ser criados, compartilhando o mesmo espaço de endereçamento do processo, além de outros recursos como o processador, por exemplo.

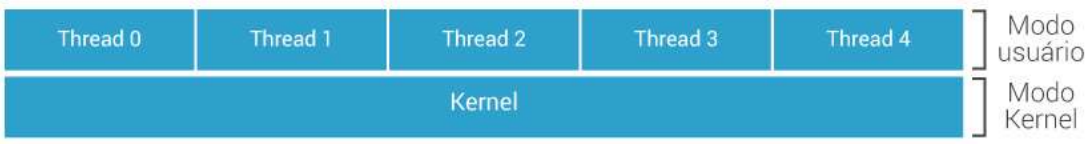
Os threads de modo usuário são mais rápidos porque não necessitam acessar o núcleo do sistema operacional, evitando assim, a mudança de modo de acesso (usuário-kernel-usuário).

Na ótica de escalonamento em ambientes com mais processadores não é possível que múltiplos threads de um processo possam ser executados em diferentes processadores simultaneamente, porque o sistema operacional executa apenas processos e não threads. Com isso existe um limite com relação ao grau de paralelismo da aplicação, pois as threads de um mesmo processo podem ser executadas em somente um processador por vez.

Thread de Kernel

Diferente das threads de modo usuário, as threads de modo kernel são implementadas pelo kernel (núcleo do sistema operacional) com as system calls (chamadas do sistema), que disponibilizam todo o gerenciamento e comunicação.

O problema desses pacotes em modo kernel é a velocidade de execução, pois em modo usuário todo o tratamento é feito sem a ajuda do sistema operacional, ou seja, sem a mudança do modo de acesso (usuário-kernel-usuário). Nos pacotes em modo kernel são necessárias chamadas a rotinas do sistema (system calls) e, consequentemente, várias mudanças no modo de acesso.



Thread de Kernel.

Threads em modo híbrido

Segundo Machado (2007, pg.), a arquitetura de threads em modo híbrido combina as vantagens de threads implementados em modo usuário e modo kernel. Um processo pode ter vários threads de modo usuário e, por sua vez, um thread de modo kernel pode ter vários threads de modo usuário. O núcleo do sistema operacional reconhece os threads de modo kernel e pode escaloná-los individualmente. Um thread de modo usuário pode ser executado em modo kernel, em um determinado momento, e no instante seguinte ser executado em outro.

Quando o programador desenvolve a aplicação em termos de threads de usuário, ele deve especificar quantos threads de modo kernel estão associados ao processo. Os threads de modo usuário são marcados em modo kernel, no momento em que o processo está sendo executado.

Nesse modelo, apesar da maior flexibilidade, há problemas, por exemplo: quando um thread de modo kernel realiza uma chamada bloqueante, todos os threads de modo usuário são colocados no estado de espera. Threads de modo usuário que desejam utilizar vários processadores devem utilizar diferentes threads de modo kernel, o que influenciará no desempenho.

Bem, este tópico finaliza aqui, esperamos que você tenha gostado e tenha absorvido as informações passadas.

Lembrem-se, a área de TI é técnica e possuir conhecimento técnico é fundamental para se dar bem no mercado, ok?

Bom estudo!

Quiz

Exercício

Threads

INICIAR

Quiz

Exercício Final

Threads

INICIAR

Referências

TANENBAUM, A. S.. Sistemas Operacionais Modernos. 2. ed. São Paulo: Pearson / Prentice Hall, 2005.

MACHADO, F. B.; MAIA, L. P. Arquitetura de Sistemas Operacionais. 2. ed. LTC, 2002.

DEITEL, H. M.; DEITEL, P.J.; CHOFFNES, D.R. Sistemas Operacionais. 3. ed. Pearson Prentice Hall, 2005.

Avalie este tópico

ANTERIOR
Processos

Índice

PRÓXIMO
Escalonamento de CPU

Ajuda?
(https://ava.un
idCurso=)

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site