



Sistemas Operacionais

Processos e Threads

Prof. Fernando Parente Garcia



Processos Introdução

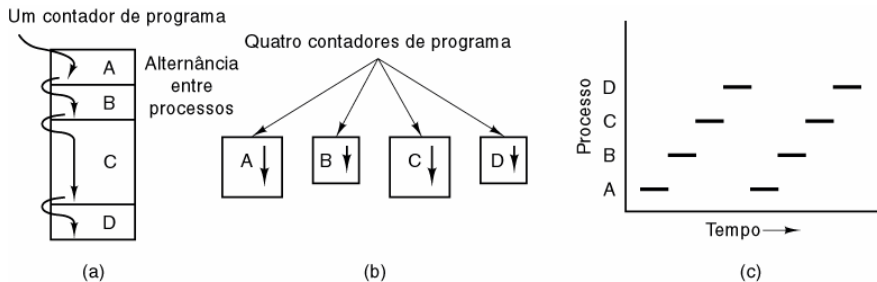
- Um processo é basicamente um programa em execução.
- Associado a cada processo está:
 - O seu espaço de endereçamento;
 - Um conjunto de registradores, que inclui o contador de programa (PC), o ponteiro para pilha (SP), outros registradores de hardware e todas as demais informações necessárias para executar um programa.



Processos

O Modelo de Processo


- Nesse modelo, todos os softwares que podem executar em um computador, são organizados em vários processos seqüenciais.
- A CPU efetua um troca-troca de processos. Esse mecanismo de trocas rápidas é chamado de **Multiprogramação**.



Processos

Criação de Processos


- Principais eventos que levam à criação de processos:
 - Início do sistema
 - Quando um S.O. é carregado, em geral criam-se vários processos;
 - Na execução de uma chamada ao sistema de criação de processo por um processo em execução;
 - Solicitação do usuário para criar um novo processo;
 - Início de um job em lote.



Processos

Término de Processos

- Condições que levam ao término de processos:
 - Saída normal (voluntária)
 - Os processos terminam porque fizeram seu trabalho;
 - Saída por erro (voluntária)
 - Fechamento de telas;
 - Erro fatal (involuntário)
 - Erro causado pelo processo. Ex: divisão por zero;
 - Cancelamento por um outro processo (involuntário)
 - Quando um processo executa uma chamada ao sistema dizendo ao SO para cancelar outro processo.



Processos

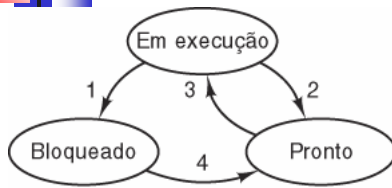
Hierarquias de Processos

- Processo pai cria um processo filho;
- O próprio processo filho pode gerar mais processos, formando uma hierarquia;
 - UNIX chama isso de **grupo de processos**;
- Windows não possui o conceito de hierarquia de processos



Processos

Estados dos Processos



1. O processo bloqueia aguardando uma entrada
2. O escalonador seleciona outro processo
3. O escalonador seleciona esse processo
4. A entrada torna-se disponível

- Possíveis estados de processos
 - **Em execução** ou **rodando**: usando a CPU naquele instante;
 - **Bloqueado**: incapaz de executar enquanto um evento externo não ocorrer (Ex.: aguardando I/O);
 - **Pronto** (executável, temporariamente parado para dar lugar a outro processo).



Processos

Implementação de Processos

- O processo é implementado pelo SO através de uma **tabela de processos** (PCB – **Process Control Block**), com um entrada para cada processo;
- Na tabela de processos, o SO mantém todas as informações sobre o **contexto de hardware**, o **contexto de software** e o **espaço de endereçamento**;
- A tabela de processos reside na memória principal em uma área exclusiva do SO;
- Informações armazenadas na tabela de processos:
 - Identificador do processo;
 - Estado do processo;
 - Prioridade do processo;
 - Registradores;
 - Limites de memória
 - Lista de arquivos abertos;
 - Etc...



Processos

Implementação de Processos

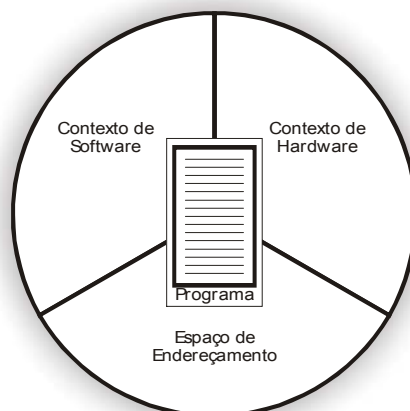
Campos da entrada de uma tabela de processos:

Gerenciamento de processos	Gerenciamento de memória	Gerenciamento de arquivos
Registradores Contador de programa Palavra de estado do programa Ponteiro de pilha Estado do processo Prioridade Parâmetros de escalonamento Identificador (ID) do processo Processo pai Grupo do processo Sinais Momento em que o processo iniciou Tempo usado da CPU Tempo de CPU do filho Momento do próximo alarme	Ponteiro para o segmento de código Ponteiro para o segmento de dados Ponteiro para o segmento de pilha	Diretório-raiz Diretório de trabalho Descritores de arquivos Identificador (ID) do usuário Identificador (ID) do grupo



Processos

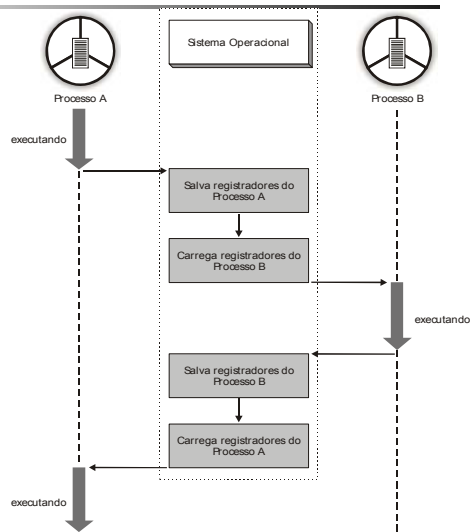
Estrutura do Processo



Processos

Contexto de Hardware

- Armazena o conteúdo dos registradores gerais da CPU, os registradores de uso específicos, como o PC, SP e o registrador de status.
- Troca de contexto:** troca de um processo por outro comandada pelo S.O.



Processos

Contexto de Software

- São especificadas características e limites dos recursos que podem ser alocados pelo processo;
- Composto por 3 grupos de informações sobre o processo: Identificação, quotas (time-slice) e privilégios (prioridade);





Processos

Espaço de Endereçamento

- Área de memória pertencente ao processo onde as instruções e os dados do programa são armazenados para execução;
- Cada processo possui seu espaço de endereçamento, que deve ser devidamente protegido do acesso dos demais processos;



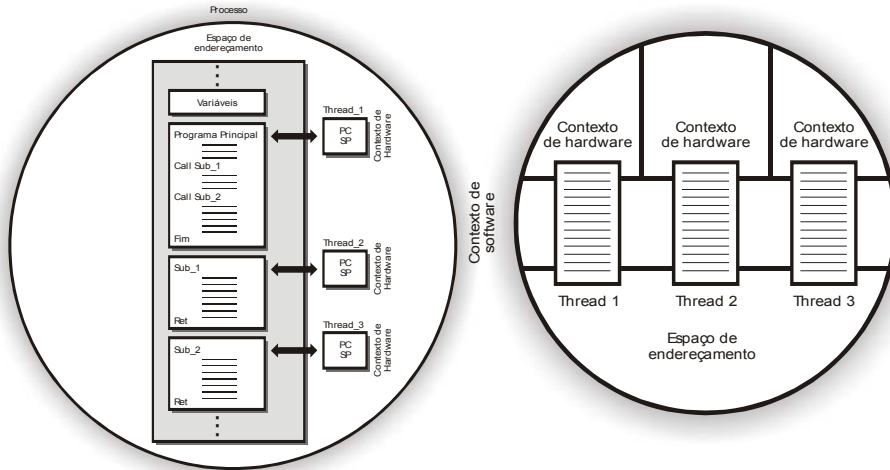
Threads

- Em sistemas operacionais tradicionais, cada processo tem um espaço de endereçamento e um único fluxo de controle (thread);
- O modelo de processo está baseado em dois conceitos independentes: agrupamento de recursos e execução;
- A separação destes conceitos constitui as **threads**;
- Processos são usados para agrupar recursos e as threads são as entidades escalonadas para a execução sobre a CPU;
- Threads permitem que múltiplas execuções independentes ocorram no mesmo ambiente do processo;
- Uma thread pode ser definida como uma sub-rotina de um programa que pode ser executada de forma assíncrona, ou seja, executada paralelamente ao programa chamador;
- O programador deve especificar as threads, associando-as às subrotinas assíncronas.



Threads

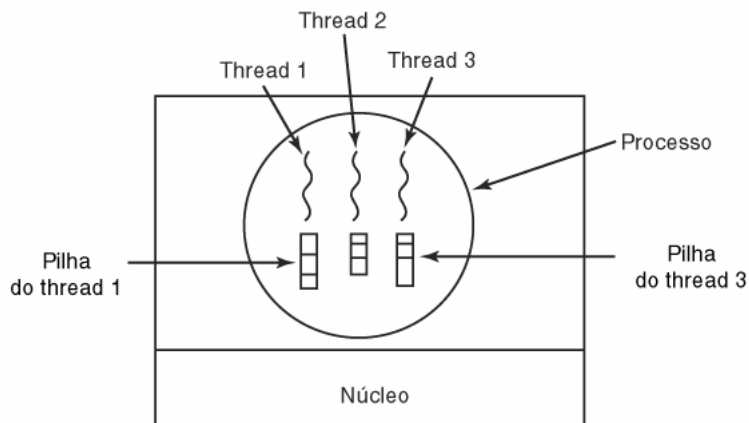
Aplicação multithread



Threads

Aplicação multithread

Cada thread tem sua própria pilha de execução.





Threads

Aplicação multithread

- No ambiente multithread não existe programas associados a processos, mas sim, a threads;
- A grande vantagem do uso de threads é a possibilidade de minimizar a alocação de recursos do sistema, além de diminuir o overhead na criação, troca e eliminação de processos;
- Threads compartilham o processador da mesma maneira que processos e passam pelas mesmas mudanças de estado;
- Dentro de um mesmo processo, threads compartilham o mesmo contexto de software e espaço de endereçamento, porém cada thread possui seu contexto de hardware individual;
- Threads são implementados internamente através de uma estrutura de dados denominada **Thread Control Block (TCB)**.
- O TCB armazena, além do contexto de hardware, mais algumas informações relacionadas exclusivamente ao thread, como prioridade, estado de execução, etc.



Processos x Threads

- Threads são mais fáceis de criar e destruir que os processos, pois não têm quaisquer recursos associados a elas;
- A comunicação entre threads de mesmo processo é mais rápida e eficiente.

Itens por processo	Itens por thread
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	Estado
Alarmes pendentes	
Sinais e tratadores de sinais	
Informação de contabilidade	

Threads

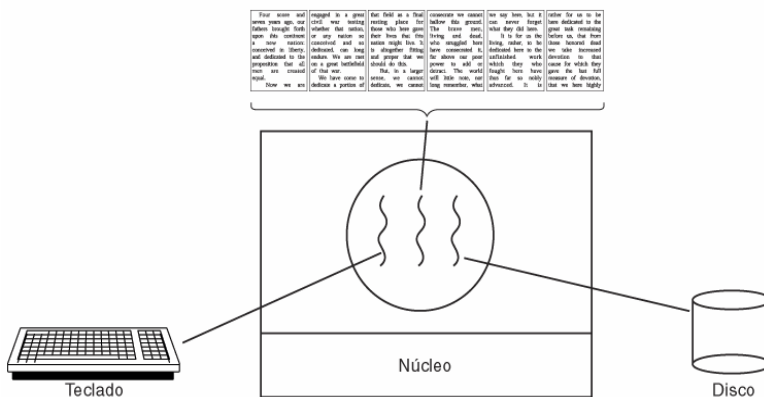
Uso de Threads

- Deve-se utilizar threads em aplicações que ocorram múltiplas atividades ao mesmo tempo, pois algumas dessas atividades podem ser bloqueadas de tempos em tempos.
- O uso de threads não resulta em um ganho de desempenho quando todas elas são orientadas à CPU. No entanto, quando há grande quantidade de computação e de E/S, as threads permitem que essas atividades se sobreponham e, desse modo, aceleram a aplicação.

Threads

Uso de Threads

Um processador de texto com três threads.

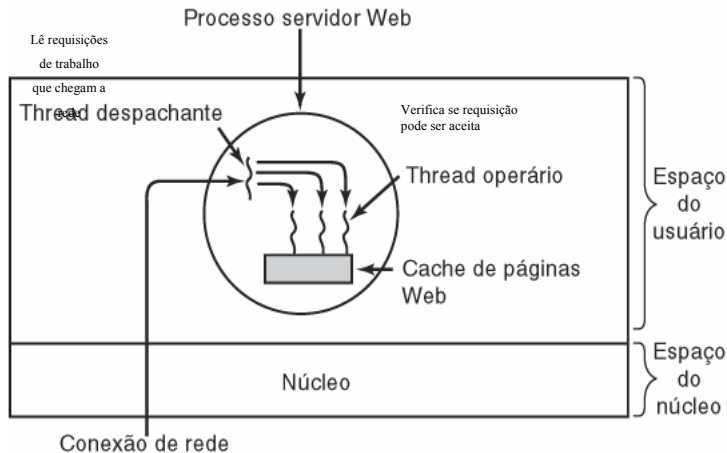




Threads

Uso de Threads

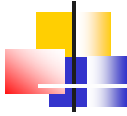
Um servidor web com múltiplos threads.



Threads

Vantagens

- O tempo de criação e destruição de uma thread é menor que o tempo de criação e destruição de um processo;
- A troca de contexto entre threads é mais rápida do que a troca de contexto entre processos;
- As threads de um mesmo processo dividem o mesmo espaço de endereçamento, o que permite a comunicação por memória compartilhada sem interação com o núcleo do SO.



Implementação de Threads

- Há dois modos de implementar um pacote de threads:
 - No espaço do usuário;
 - No espaço do sistema (núcleo).



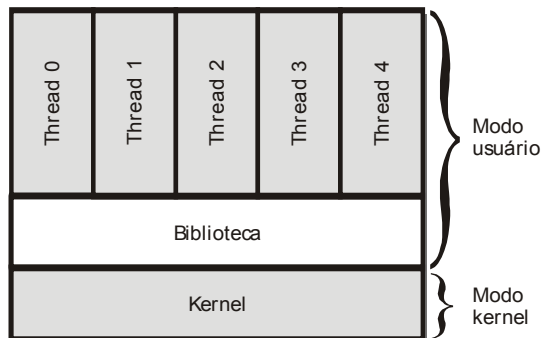
Implementação de threads Threads em modo usuário

- São implementados pela aplicação e não pelo SO;
- Existência de uma biblioteca de rotinas para gerenciamento das threads (**sistema supervisor**) provida pela aplicação;
- O SO não sabe da existência das threads;
- São rápidas e eficientes por dispensarem acessos ao Kernel do SO, evitando assim a mudança de modo de acesso;
- O SO gerencia cada processo como se existisse apenas uma única thread;
- Cada processo precisa de sua própria tabela de threads para manter o controle das suas threads;
- As threads do usuário permitem que cada processo tenha seu próprio algoritmo de escalonamento personalizado.



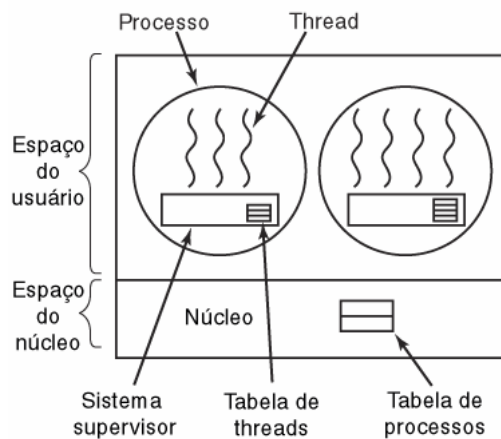
Implementação de threads

Threads em modo usuário



Implementação de threads

Threads em modo usuário





Implementação de threads

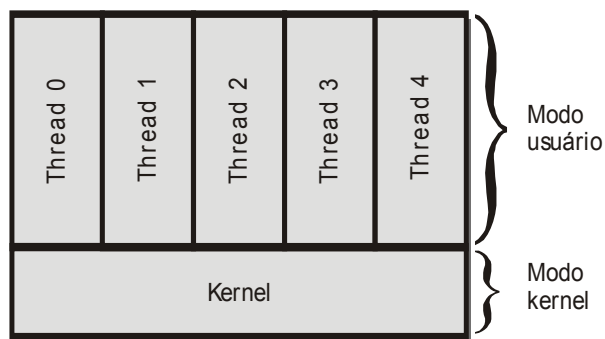
Threads em modo núcleo

- Implementadas diretamente pelo núcleo do SO através de chamadas a rotinas do sistema que oferecem todas as funções de gerenciamento e sincronização;
- O SO sabe da existência de cada thread e pode escaloná-las individualmente;
- **Problema:** baixo desempenho, pois os pacotes em modo kernel utilizam chamadas a rotinas do sistema e, conseqüentemente, ocorrem várias mudanças no modo de acesso.



Implementação de threads

Threads em modo núcleo





Implementação de threads

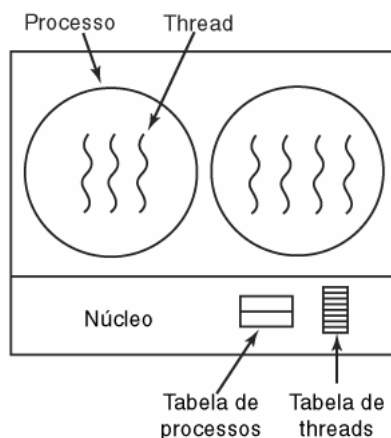
Threads em modo núcleo

- Não é necessário um sistema supervisor, e não há nenhuma tabela de threads em cada processo;
- O núcleo tem uma tabela de threads para gerenciar todas as threads do sistema;
- Quando uma thread quer criar uma nova thread ou destruir uma já existente, ela faz uma chamada ao núcleo, que realiza então a criação ou a destruição atualizando a tabela de threads do núcleo;
- O núcleo também mantém a tabela de processos para acompanhamento dos processos;
- Todas as chamadas que possam bloquear uma thread são implementadas com chamadas ao sistema, com um custo consideravelmente maior do que uma chamada para o procedimento do sistema supervisor.



Implementação de threads

Threads em modo núcleo





Implementação de threads

Threads híbridas

- Usa threads de núcleo e multiplexa threads de usuário sobre alguma ou todas as threads de núcleo.

