Universidade Federal do Rio Grande do Norte
Unidade Acadêmica Especializada em Ciências Agrárias
Escola Agrícola de Jundiaí
Curso de Análise e Desenvolvimento de Sistemas
TAD0006 - Sistemas Operacionais - Turma 01

Threads

Antonino Feitosa

antonino.feitosa@ufrn.br

Macaíba, abril de 2025

Recapitulação

- Modelo: programa x processo
- Multiprogramação
- Criação: eventos, chamada de sistema, PID
- Término: eventos
- Hierarquia
- Estados: pronto, bloqueado e execução
- Implementação e Interrupções
- Modelagem de Multiprogramação: modelo probabilístico

Roteiro

- Conceito de Threads
- Casos de Uso
- Modelo de Thread
- Threads POSIX
- Implementações
  - Espaço de Usuário
  - No Núcleo
  - Modelos 1:1, N:1, N:M

Introdução

Introdução: Escalonador

- Principal tarefa no gerenciamento de processos: **escalonamento**.
  - Responsável pelo compartilhamento do processador.
  - Diferentes estratégias.

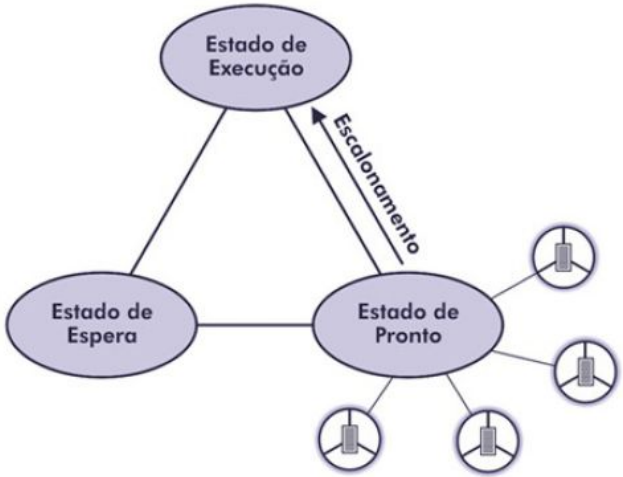


Fig. 8.1 Escalonamento.

Introdução

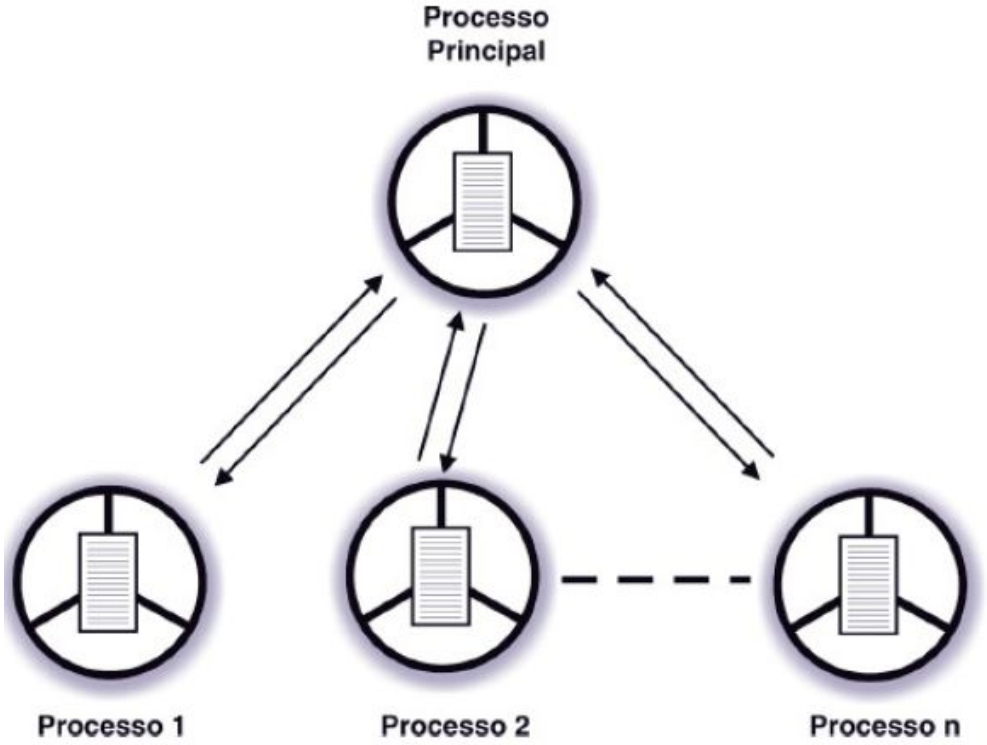


Fig. 7.2 Concorrência em programas.

Introdução: Sinais

- Sinais é um mecanismo que permite notificar processos de **eventos** gerados pelo sistema operacional ou por outros processos.
  - Exemplo: comando para encerrar uma aplicação via teclado (Ctrl+C);
- Eventos que geram sinais podem ter diferentes causas.
  - Sistema operacional ou hardware: ocorrência de exceções, interrupções, alarmes de tempo, etc.
  - Outros processos: sincronização e comunicação.

Introdução: Sinais

- Processo são independentes.
- Recursos podem ser compartilhados.
  - Envolvem controle e sincronização.

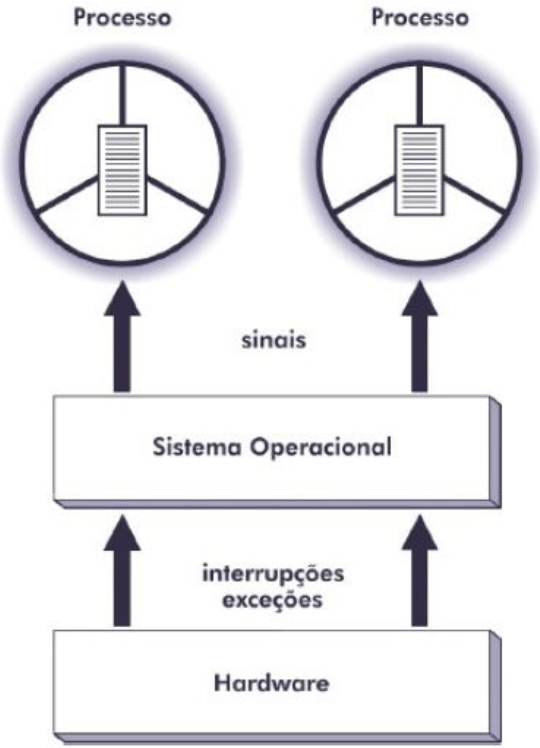


Fig. 5.18 Sinais, interrupções e exceções.

Introdução: Sinais

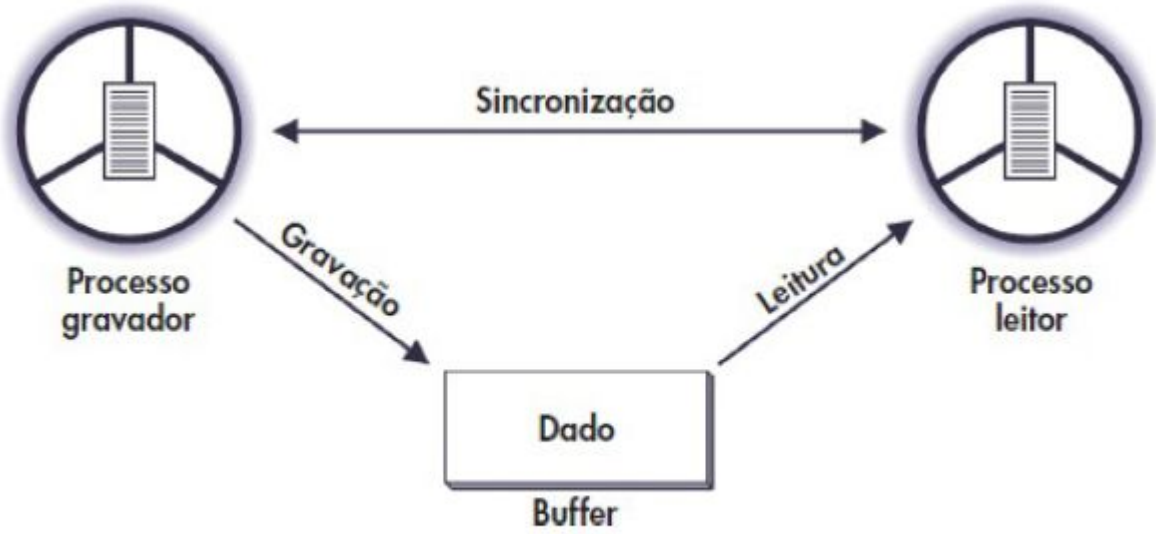
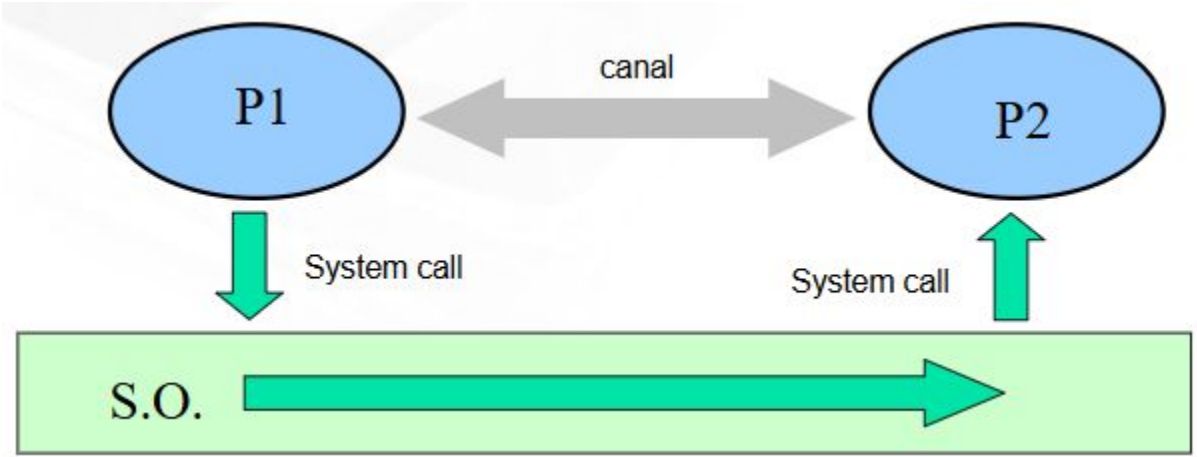


Fig. 7.1 Sincronização e comunicação entre processos.

Introdução

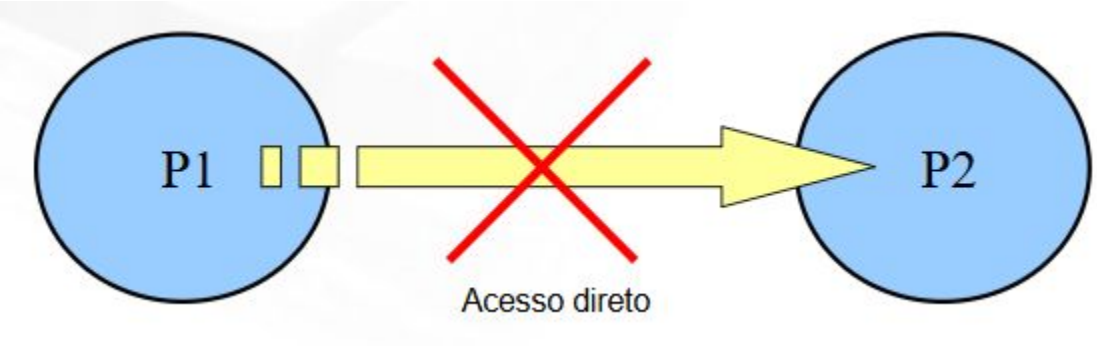
- A comunicação entre processo envolve chamadas de sistemas.
  - Canais diretos ou indiretos controlados pelo sistema operacional.



https://slideplayer.com.br/slide/367577/

Introdução

- Não existe acesso direto aos dados.



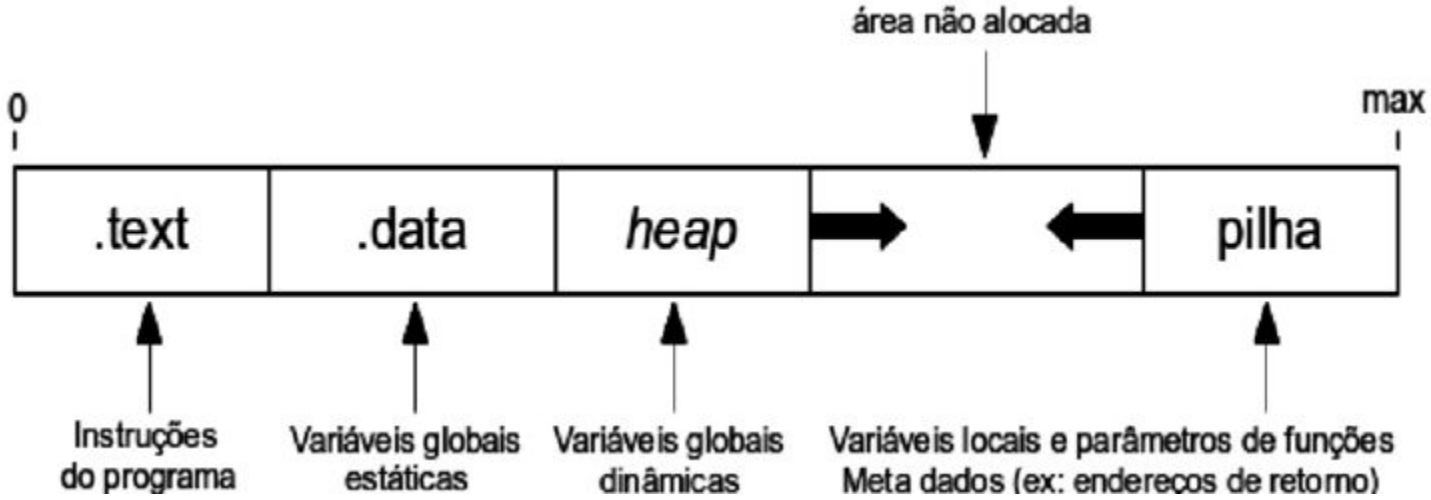
https://slideplayer.com.br/slide/367577/

Conceito de Threads

Processos: Informações Armazenadas

Fig. 5.4 Características da estrutura de um processo.

Processo: Espaço de Endereçamento



https://www.researchgate.net/figure/Figura-12-Organizacao-do-espaco-de-memoria-de-um-processo-Maziero-2011_fig1_247162627

Threads

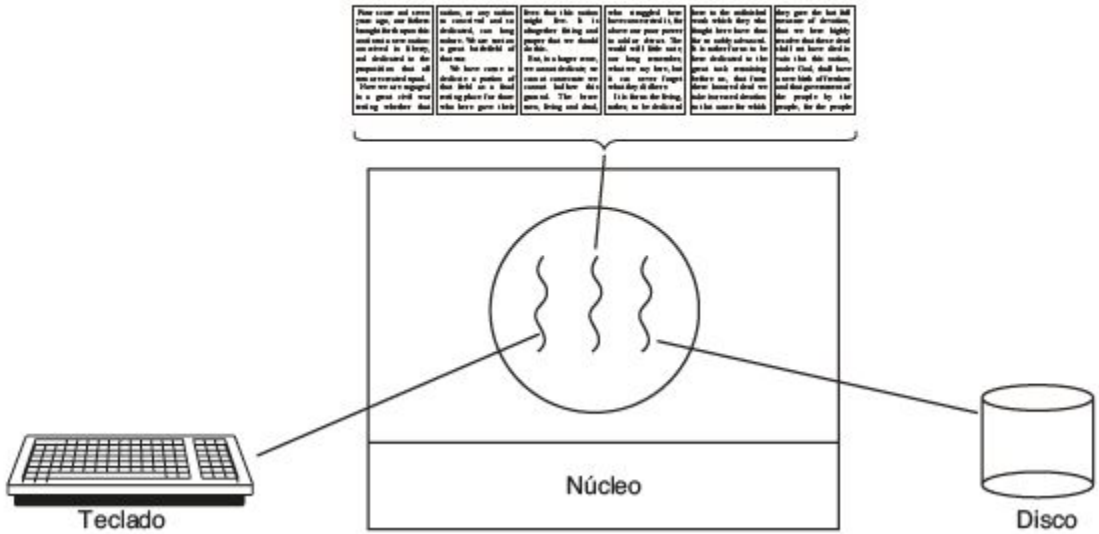
- Cada processo possui um espaço de endereçamento e uma única thread controle.
  - Thread (linha de execução): linha de execução de um processo.
- Em muitas situações é desejável ter múltiplos threads.
  - Controle no mesmo espaço de endereçamento executando em paralelo.
  - Compartilhamento do espaço de endereçamento.

Threads: Exemplo - Processador de Textos

- Processador de textos
  - Thread para interação com usuário;
  - Thread para formatação do texto;
  - Thread para salvar o texto a cada intervalo fixo de tempo;

Threads: Exemplo - Processador de Textos

FIGURA 2.7 Um processador de texto com três threads.



Threads: Exemplo - Processador de Textos

- Em um processo, cada operação precisaria ser feita sequencialmente, bloqueando o processo em cada chamada do sistema.
- O uso de 3 processos também seria inviável, pois seria necessário comunicar todas as alterações, como também sincronizar as ações para evitar inconsistências.

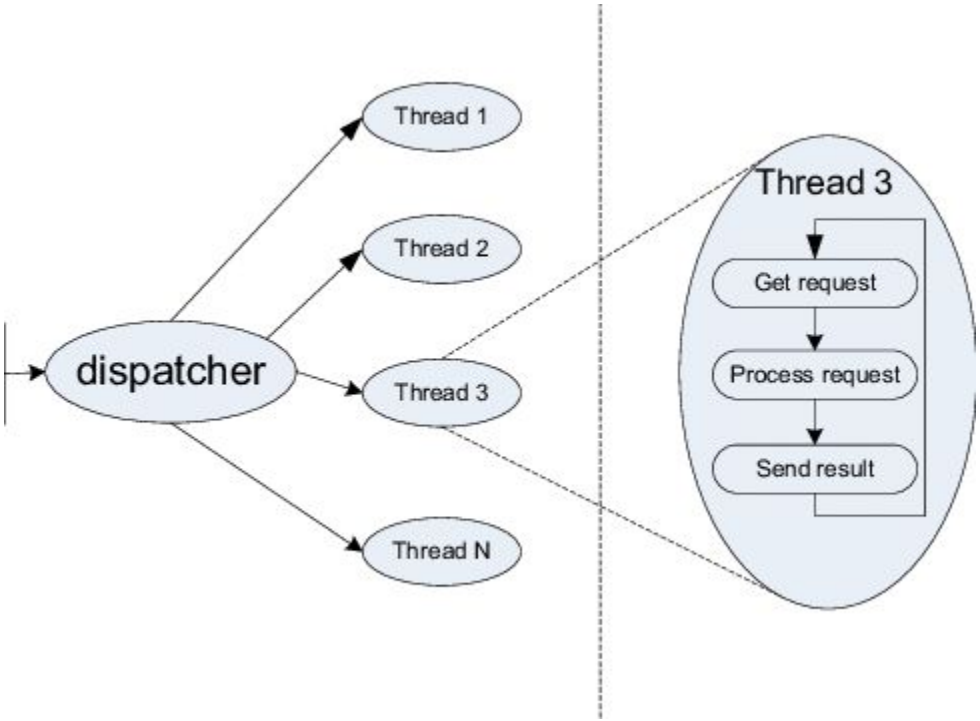
Threads - Exemplo - Alto Processamento

- Aplicações que processam grande quantidade de dados.
  - Leitura de um bloco de dados, processamento e então escrita do bloco processado.
  - Processo é bloqueado na leitura e na escrita dos dados.
- Solução: thread de entrada de dados, thread de processamento e thread de saída.
  - Os dados são armazenados em regiões temporárias assim que uma atividade é finalizada, executadas em paralelo.

Threads: Exemplo - Servidor WEB

- Servidor WEB
  - Solicitações para páginas chegam e a página solicitada é enviada de volta para o cliente.
  - Páginas mais acessadas são mantidas na memória principal.
    - Estratégia de cache.
  - Demais páginas estão no armazenamento secundário.
    - Demandam tempo para obtenção dos dados.
    - Acesso ao dispositivo bloqueia o processo.

Threads: Exemplo - Servidor WEB

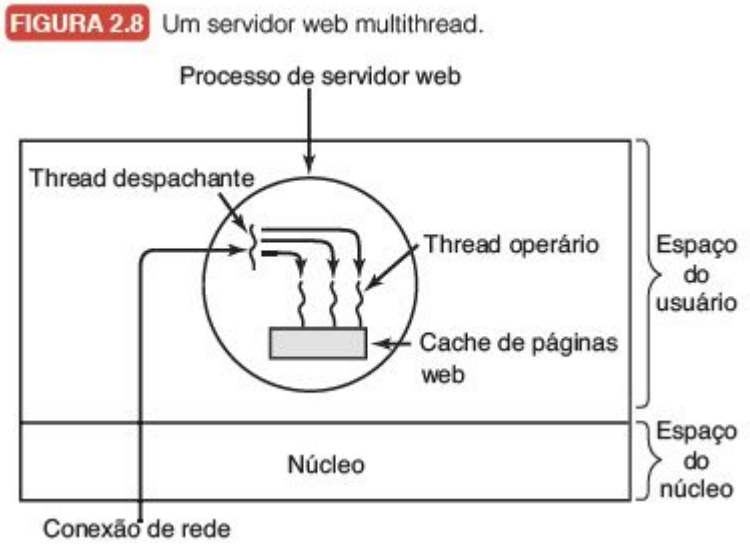


https://www.researchgate.net/figure/Multi-threaded-server-design_fig2_221244103

Threads: Exemplo - Servidor WEB 1

- Thread despachante: lê as requisições de trabalho que chegam da rede e escolhe uma thread operário ocioso (bloqueado) para atender a solicitação.
- Thread operário verifica se a solicitação pode ser atendida por acesso ao cache.
  - Cache é compartilhado entre todos os operários.
  - Se a página não estiver na cache, os dados são acessados na memória secundária.
    - A thread operário é bloqueada, mas o despachante não.

Threads: Exemplo - Servidor WEB 1



Threads: Exemplo - Servidor WEB 2

- O mesmo poderia ser feito com um processo.
  - Bloqueio do processo a cada acesso à memória secundária.
    - Perda de desempenho.
  - Uma única thread.

Threads: Exemplo - Servidor WEB 3

- Terceira opção: depende da existência de chamadas de sistema **read** não bloqueantes.
  - Situação em que múltiplos threads não estão disponíveis.
- Continua atendendo as solicitações das páginas do cache diretamente.
- Porém, o acesso ao disco é iniciada sem bloqueio.

Threads: Exemplo - Servidor WEB 3

- O servidor registra o estado da solicitação ao disco e segue para o próximo evento.
  - Solicitação de nova página.
  - Resposta do disco sobre um acesso anterior.
- O registro da solicitação é acessado e então processadas.
  - A resposta do disco deve ser modelada como um sinal ou interrupção.

Threads: Exemplo - Servidor WEB 3

- Perdemos o conceito de modelo de processo sequencial.
- Ocorre uma simulação das threads.
  - O contexto de cada thread precisa ser registrado e recuperado quando os dados estiverem prontos.
- Conceito de máquina de estados finitos.
  - Estado de espera de acesso; espera dos dados; processamento.

Threads: Exemplo - Servidor WEB

- Chamadas de sistema bloqueantes tornam o desenvolvimento mais simples.
- Paralelismo melhora o desempenho.
- Única thread: é simples, mas de baixo desempenho
- Simulação de threads: alto desempenho, porém é difícil de programar por usar chamadas não bloqueantes e interrupções.

Threads: Exemplo - Servidor WEB

FIGURA 2.10 Três maneiras de construir um servidor.

Modelo	Características
Threads	Paralelismo, chamadas de sistema bloqueantes
Processo monothread	Não paralelismo, chamadas de sistema bloqueantes
Máquina de estados finitos	Paralelismo, chamadas não bloqueantes, interrupções

Modelo de Thread

Modelo de Thread

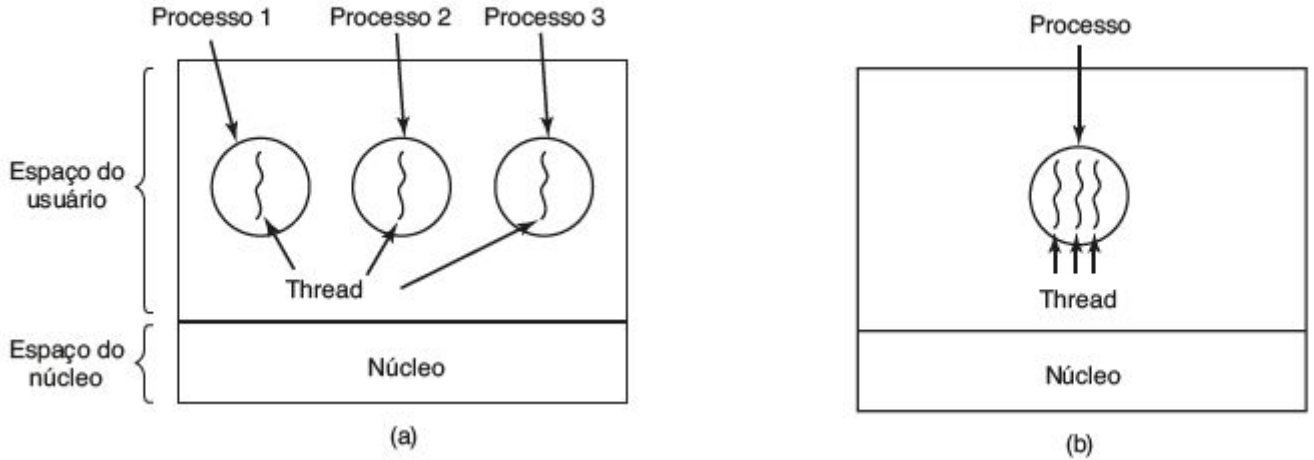
- O modelo de processo é baseado em dois conceitos independentes: agrupamento de recursos e execução.
  - A separação pode ser útil (threads).
- Processo possui:
  - Espaço de endereçamento (código e dados do programa).
  - Recursos: arquivos, alarmes pendentes, tratadores de sinais, etc.
  - Uma única thread: contador de programa, registradores, pilha (histórico de execução, com uma estrutura para cada rotina chamada, mas ainda não retornada).

Modelo de Thread

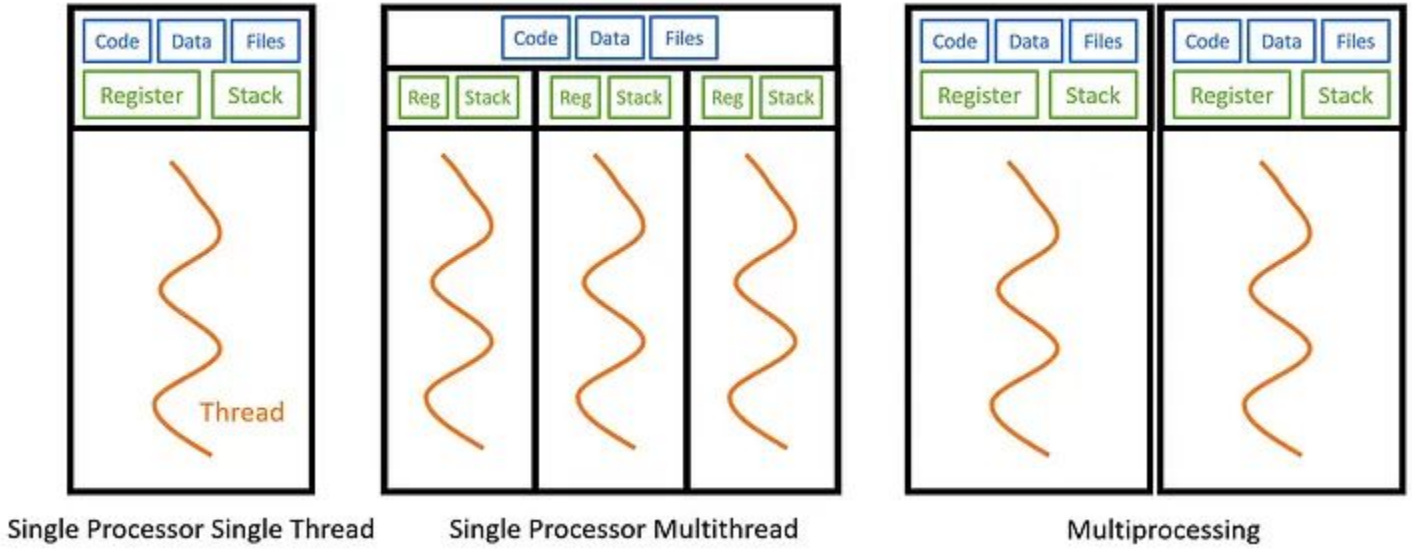
- Threads permitem múltiplas execuções no mesmo ambiente.
  - Múltiplas threads executando em paralelo em um processo.
    - Compartilham espaço de endereçamento e recursos.
  - Múltiplos processos executando em paralelo em um computador.
    - Compartilham memórias físicas, discos, impressoras e outros recursos.
- Também chamadas de processos leves.
- Processo multithread: possui múltiplas threads.
  - Pode ser implementado em hardware.

Modelo de Thread

FIGURA 2.11 (a) Três processos, cada um com um thread. (b) Um processo com três threads.



Modelo de Thread



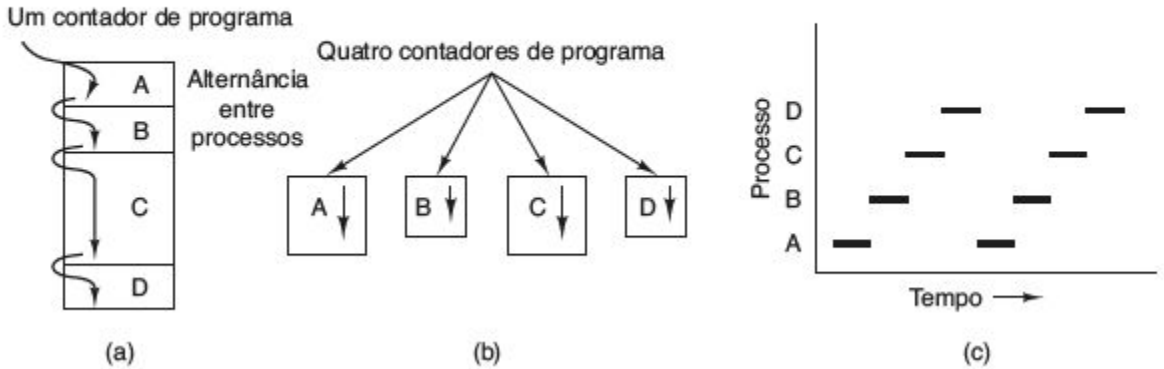
Single Processor Single Thread Single Processor Multithread Multiprocessing

https://medium.com/@amitvsolutions/java-concurrency-part1-05b1e4c280eb

Modelo de Thread

- Quando um processo multithread é executado em um sistema de CPU única, os threads se revezam executando.

FIGURA 2.1 (a) Multiprogramação de quatro programas. (b) Modelo conceitual de quatro processos sequenciais independentes. (c) Apenas um programa está ativo de cada vez.



Modelo de Thread

- Independência limitada.
  - Compartilham o espaço de endereçamento.
    - Possuem as mesmas variáveis globais.
  - Não há proteção entre threads.
    - Uma thread pode modificar dados de outra.
    - Não é possível implementar.
    - Desnecessário: pois pertence a um único processo, ou seja, pertencem a um único usuário.

Modelo de Thread

FIGURA 2.12 A primeira coluna lista alguns itens compartilhados por todos os threads em um processo. A segunda lista alguns itens específicos a cada thread.

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	

Modelo de Thread

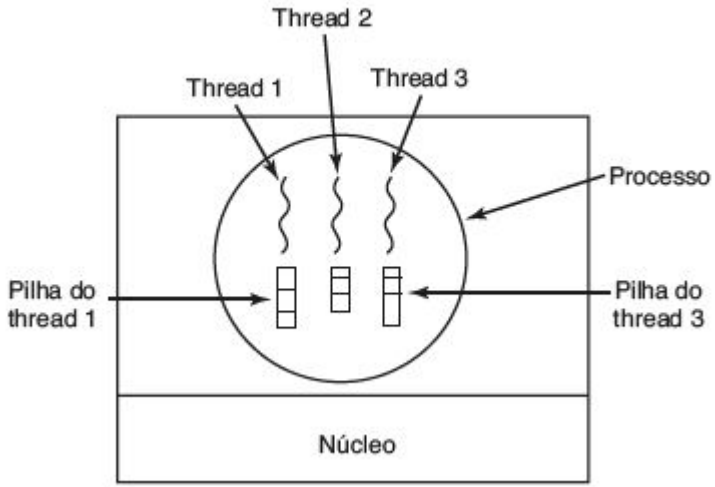
- **Objetivo**: gerar a capacidade para múltiplos threads de execução de compartilhar um conjunto de recursos de maneira que possam trabalhar juntos intimamente para desempenhar alguma tarefa.

Modelo de Thread

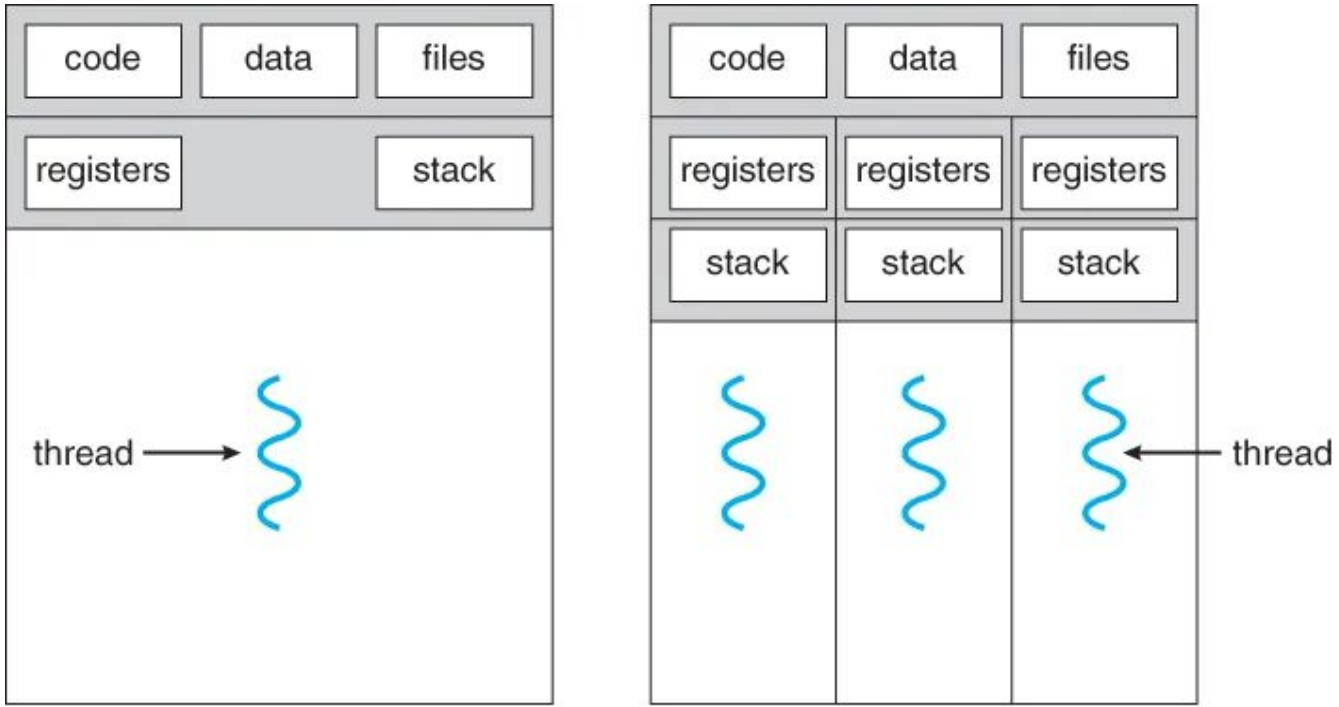
- Threads possuem os mesmos estados que os processos.
- Cada thread possui sua própria pilha.
  - Histórico de execução, com uma estrutura para cada rotina chamada, mas ainda não retornada.
    - Variáveis locais da rotina e o endereço de retorno para usar quando a chamada de rotina for encerrada.

Modelo de Thread

FIGURA 2.13 Cada thread tem a sua própria pilha.



Modelo de Thread



single-threaded process

multithreaded process

https://42bits.medium.com/o-pequeno-artigo-sobre-threads-5aa22a18a89e

Modelo de Thread

- Os processos normalmente começam com um único thread presente.
- Esse thread tem a capacidade de criar novos.
  - thread_create especifica o nome de uma rotina para o novo thread executar.
  - Cada thread possui um identificador.
  - thread_exit chamada que encerra a thread atual.
    - Não é mais escalonável.

Modelo de Thread

- Em alguns sistemas, um thread pode esperar pela saída de um thread (específico) chamando uma rotina.
  - thread_join: bloqueia o thread que executou a chamada até que um thread (específico) tenha terminado.
- thread_yield: permite que um thread abra mão voluntariamente da CPU para deixar outro thread ser executado.
  - Não há uma interrupção de relógio para realmente forçar a multiprogramação como há com os processos.

Modelo de Thread: Problemas

- O que acontece em uma chamada fork para um processo multithread?
- Introduzem complicações de paralelismo.
  - O que acontece se um thread fecha um arquivo enquanto outro ainda está lendo dele?
  - Suponha que um thread observe que há pouca memória e comece a alocar mais memória. No meio do caminho há um chaveamento de threads, e o novo também observa que há pouca memória e também começa a alocar mais memória. A memória provavelmente será alocada duas vezes.

Threads POSIX

Threads POSIX

- Possibilita a escrita de programas com threads portáteis.
  - Padrão IEEE 1003.1c.
  - Pacote Pthreads.
    - Presente na maioria dos sistemas UNIX.

FIGURA 2.14 Algumas das chamadas de função do Pthreads.

Chamada de thread	Descrição
Pthread_create	Cria um novo thread
Pthread_exit	Conclui a chamada de thread
Pthread_join	Espera que um thread específico seja abandonado
Pthread_yield	Libera a CPU para que outro thread seja executado

Implementações

Implementações

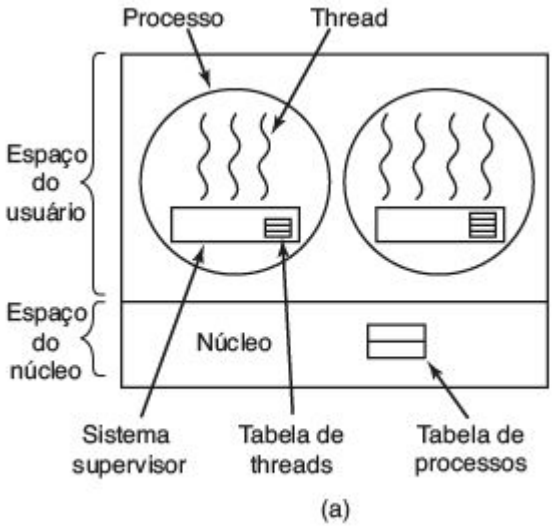
- Threads podem ser implementadas:
  - No espaço do usuário;
  - No núcleo;
  - Híbrido.

Threads do Espaço de Usuário

Threads do Espaço de Usuário

- O núcleo não sabe nada a respeito deles.
  - Gerência de um único processo (única thread).
- Útil em sistemas operacionais que oferecem suporte a threads.
  - Cada vez menos comum.
- Os threads executam em cima de um sistema de tempo de execução, que é uma coleção de rotinas que gerencia os threads.

Threads do Espaço de Usuário



(a)

Threads do Espaço de Usuário

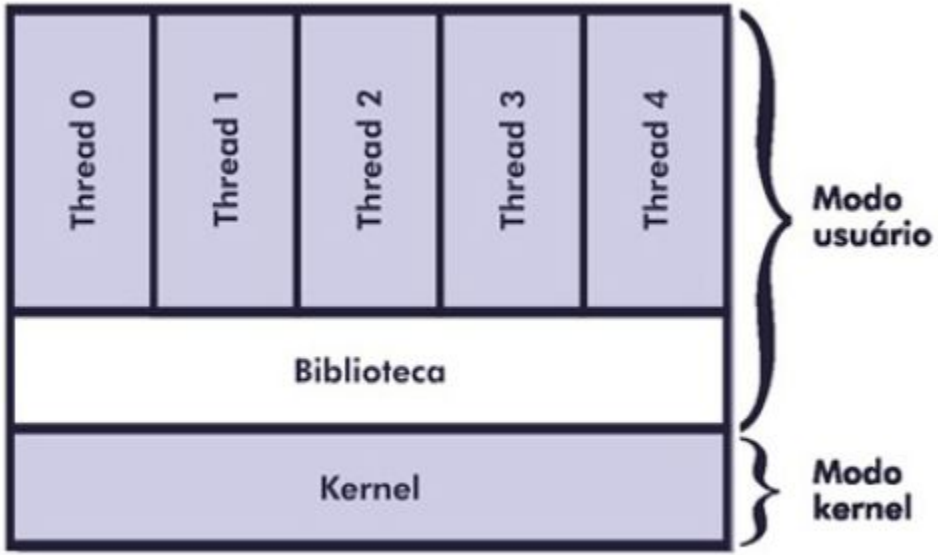


Fig. 6.8 Threads em modo usuário.

Threads do Espaço de Usuário

- Cada processo gerencia a sua própria **tabela de threads**.
- A tabela de threads é gerenciada pelo sistema de tempo de execução.

Threads do Espaço de Usuário: Vantagens

- Realizar um chaveamento de thread no espaço de usuário é pelo menos uma ordem de magnitude — talvez mais — mais rápida do que desviar o controle para o núcleo.
  - Nenhuma chamada do sistema é executada.
- Permitem que cada processo tenha seu próprio algoritmo de escalonamento customizado.
- Possuem maior escalabilidade do que threads no núcleo.
  - Menos recursos são gerenciados.

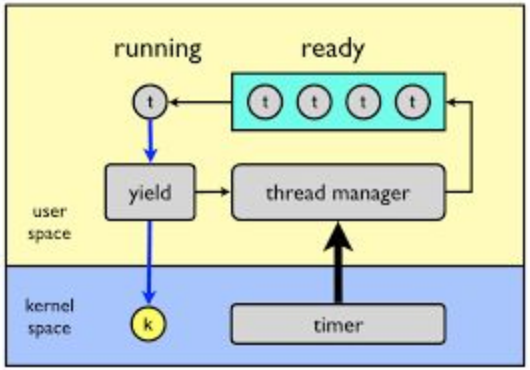
Threads do Espaço de Usuário: Dificuldades

- Necessita de chamadas de sistema não bloqueantes.
  - Não é compatível com a maioria dos sistemas operacionais.
  - Decomposição da chamada do sistema. Exemplo para read:
    - Seleção de dados, verificação se a chamada bloqueará o processo.
      - Ineficiente e deselegante, além de exigir a reescrita das chamadas de sistema.

Threads do Espaço de Usuário: Dificuldades

- Problema 1: se um thread começa a ser executado, nenhum outro naquele processo será executado a não ser que o primeiro thread voluntariamente abra mão da CPU.
- Solicitar interrupções periódicas de relógio.
  - Sobrecarga das chamadas.
  - Uma thread pode necessitar de uma interrupção de relógio em sua lógica.

Threads do Espaço de Usuário: Dificuldades



https://www.it.uu.se/education/course/homepage/os/vt18/module-4/implementing-threads/

Threads do Espaço de Usuário: Dificuldades

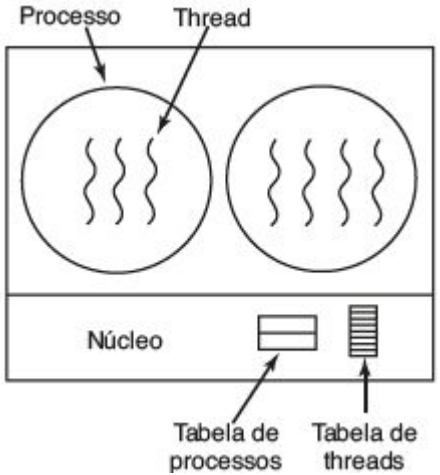
- Problema 2: Programadores geralmente desejam threads precisamente em aplicações nas quais eles são bloqueados com frequência.
- Fazem constantes chamadas de sistema.
  - Já que está no espaço do núcleo, podemos efetuar a troca de threads.
- Para aplicações que são em sua essência inteiramente limitadas pela CPU e raramente são bloqueadas, qual o sentido de usar threads?

Threads no Núcleo

Threads no Núcleo

- Núcleo gerencia as threads.
- Processos não precisam:
  - Sistema de tempo de execução.
  - Tabela de processos.
- Gerenciamento de threads por chamadas de sistema.
  - Tabela de threads do núcleo.
  - Tabela de processos.

Threads no Núcleo



Threads no Núcleo

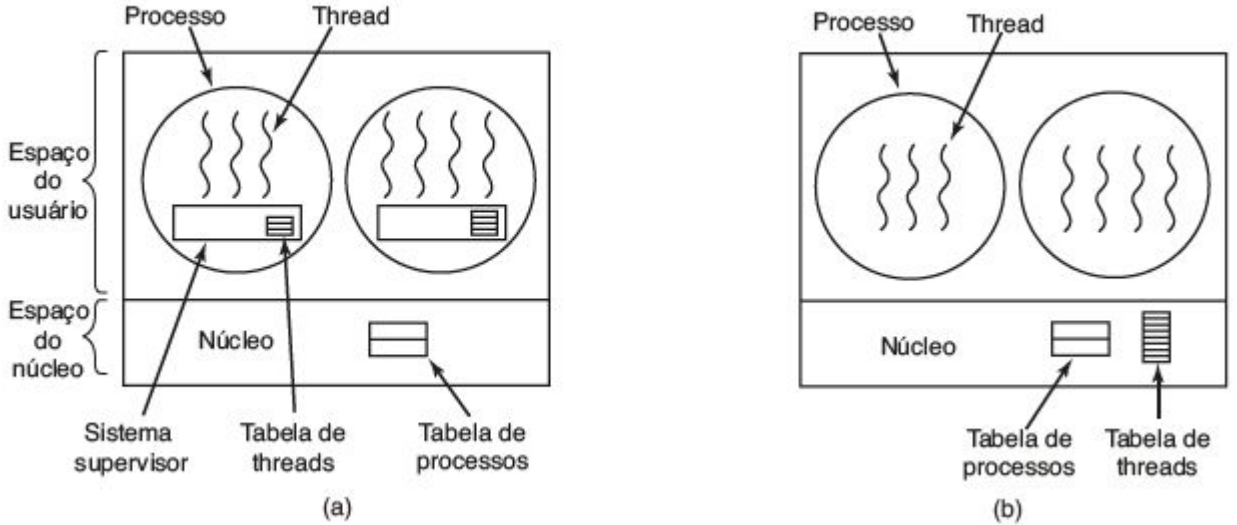
- Uma thread bloqueada pode ser substituída por outra do mesmo processo ou de algum processo diferente.
- Chamadas de sistema para criação e término de threads são custosas.
    - Reciclagem de threads.

Threads no Núcleo: Problemas

- O que acontece quando um processo com múltiplos threads é bifurcado? O novo processo tem tantos threads quanto o antigo, ou possui apenas um?
- Quando um sinal chega, qual thread deve cuidar dele?

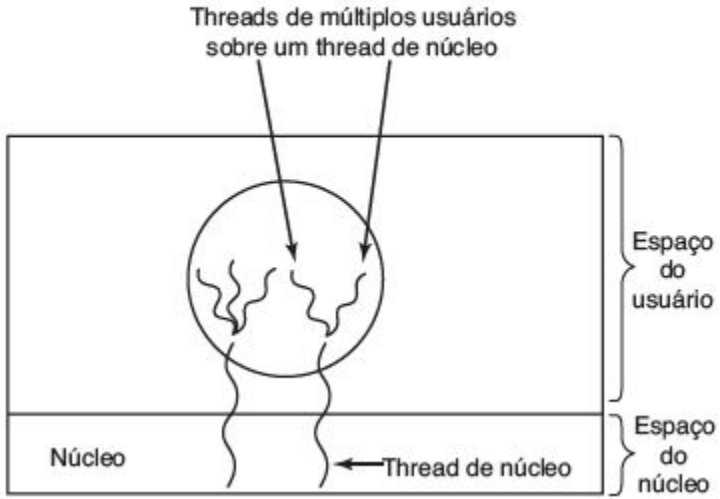
Implementações: Resumo

FIGURA 2.16 (a) Um pacote de threads no espaço do usuário. (b) Um pacote de threads gerenciado pelo núcleo.



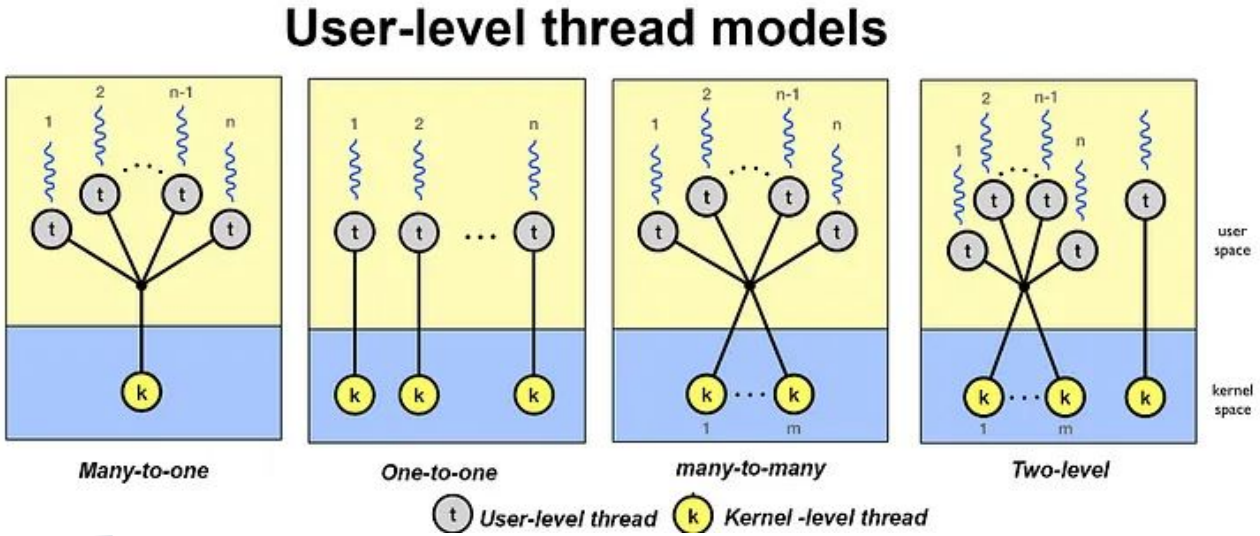
Implementações: Resumo

FIGURA 2.17 Multiplexando threads de usuário em threads de núcleo.



Implementações: Resumo

- Modelo de Threads: modo como as threads de usuário são mapeadas para o núcleo.



https://jyos-sw.medium.com/kernel-and-user-threads-31c2743bc8e3

Resumo

Resumo

- Conceito de Threads
  - Diferenças entre processos e threads
- Modelo de Thread
- Threads POSIX
- Implementações
  - Espaço de Usuário
  - No Núcleo
  - Modelos 1:1, N:1, N:M

Dúvidas?