Processos e Threads

Processos

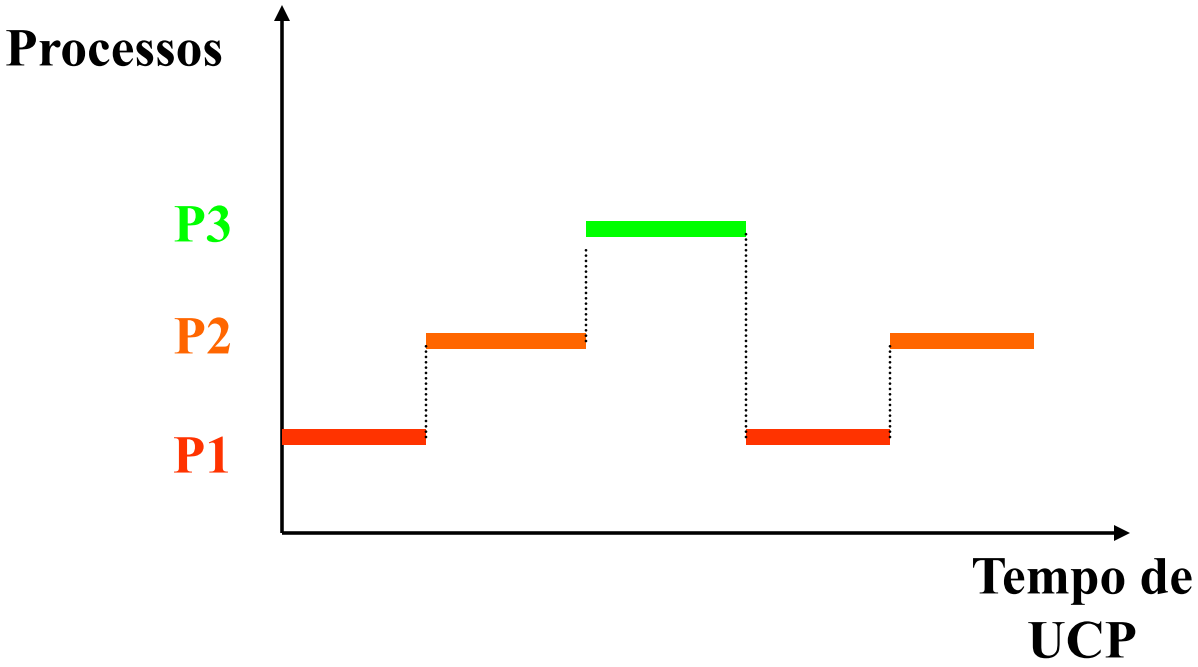
Sistemas de Processamento Paralelo

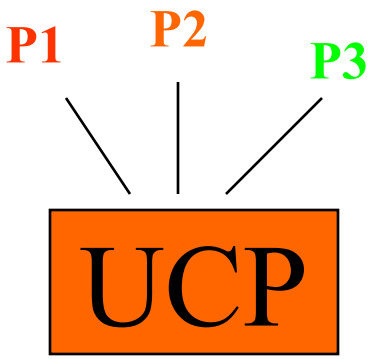
- **Sistemas simples de computação → limitados**

Necessidade de melhoria no desempenho → **criação dos sistemas paralelos de processamento**

Processos

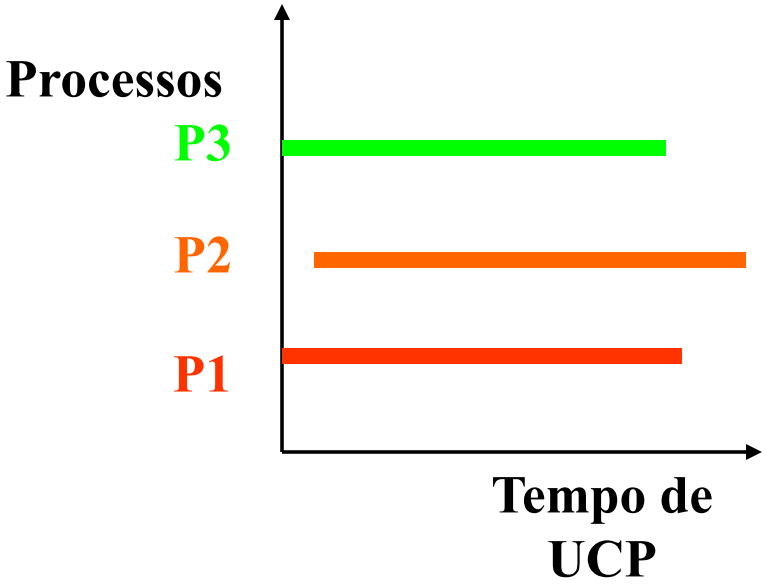
- **Sistemas Logicamente Paralelos**
  - **Intercalam os processamentos de vários programas em uma única UCP**

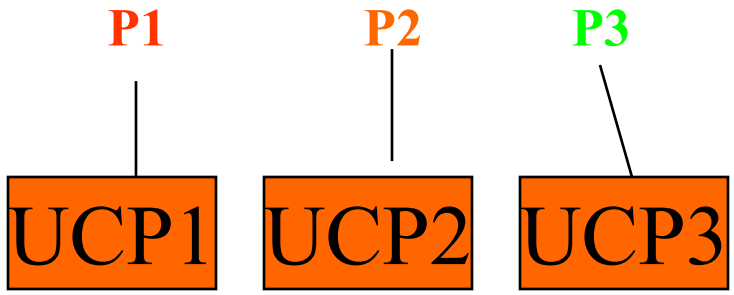




Processos

- **Sistemas Fisicamente Paralelos**
  - **Executam as suas tarefas em múltiplos computadores, ou em múltiplas UCPs, compartilhando os demais recursos de máquina, tais como memória principal e unidades de E/S**





Processos

Modelo de Processo

- **Sistemas de processamento paralelo** → **difícil controle**

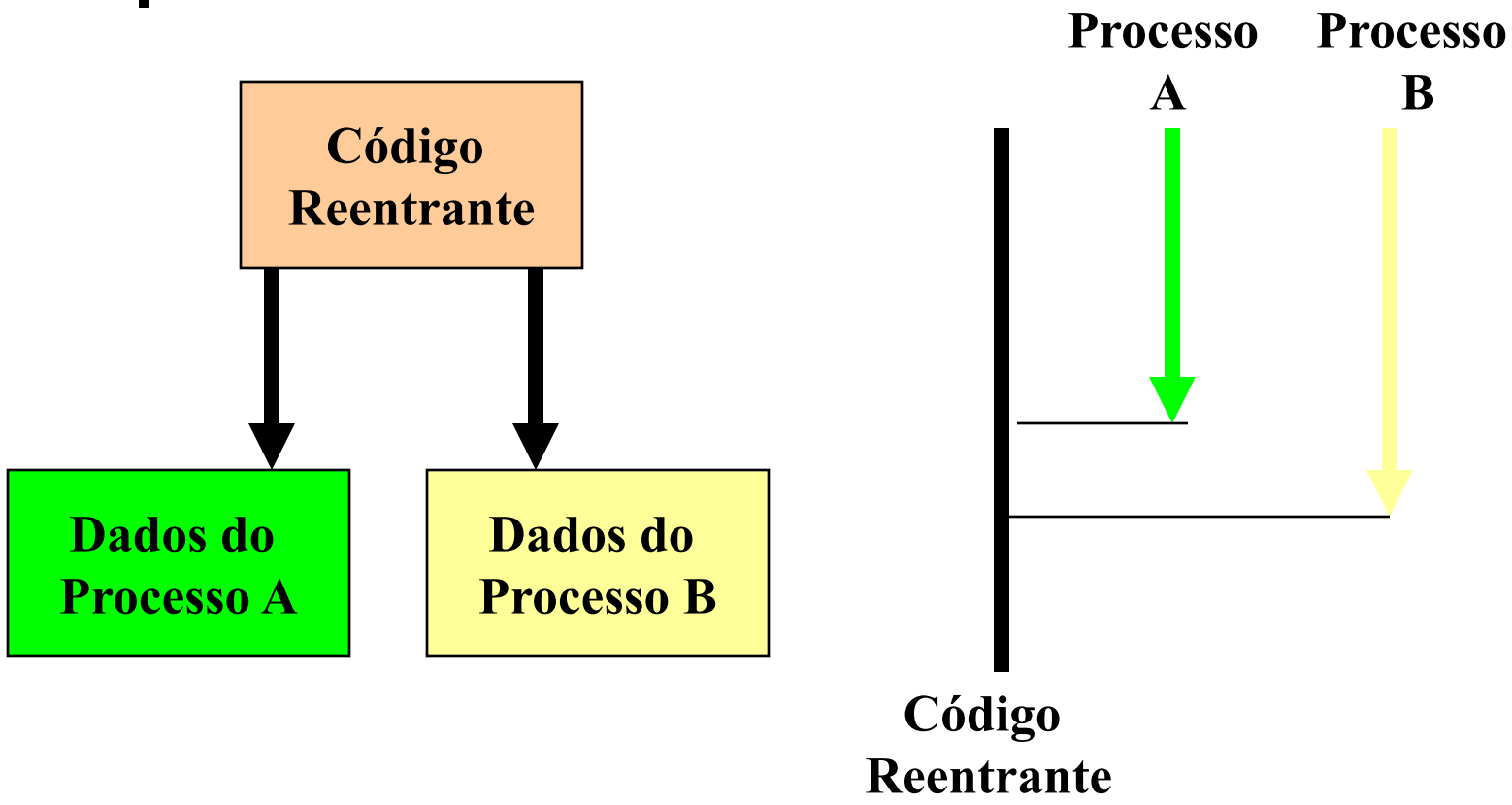
Facilidade de construção de Sistemas Operacionais → **Modelo de Processo**

Processos

- **Definição de Processo**
  - **É um programa em execução, incluindo os valores correntes do contador de programa, registradores, e variáveis. O conceito de processo é dinâmico, em contraposição ao conceito de programa, que é estático.**
  - **Nem sempre um programa equivale a apenas um processo**
  - **Em sistemas que permitem a reentrância, o código de um programa pode gerar diversos processos.**

Processos

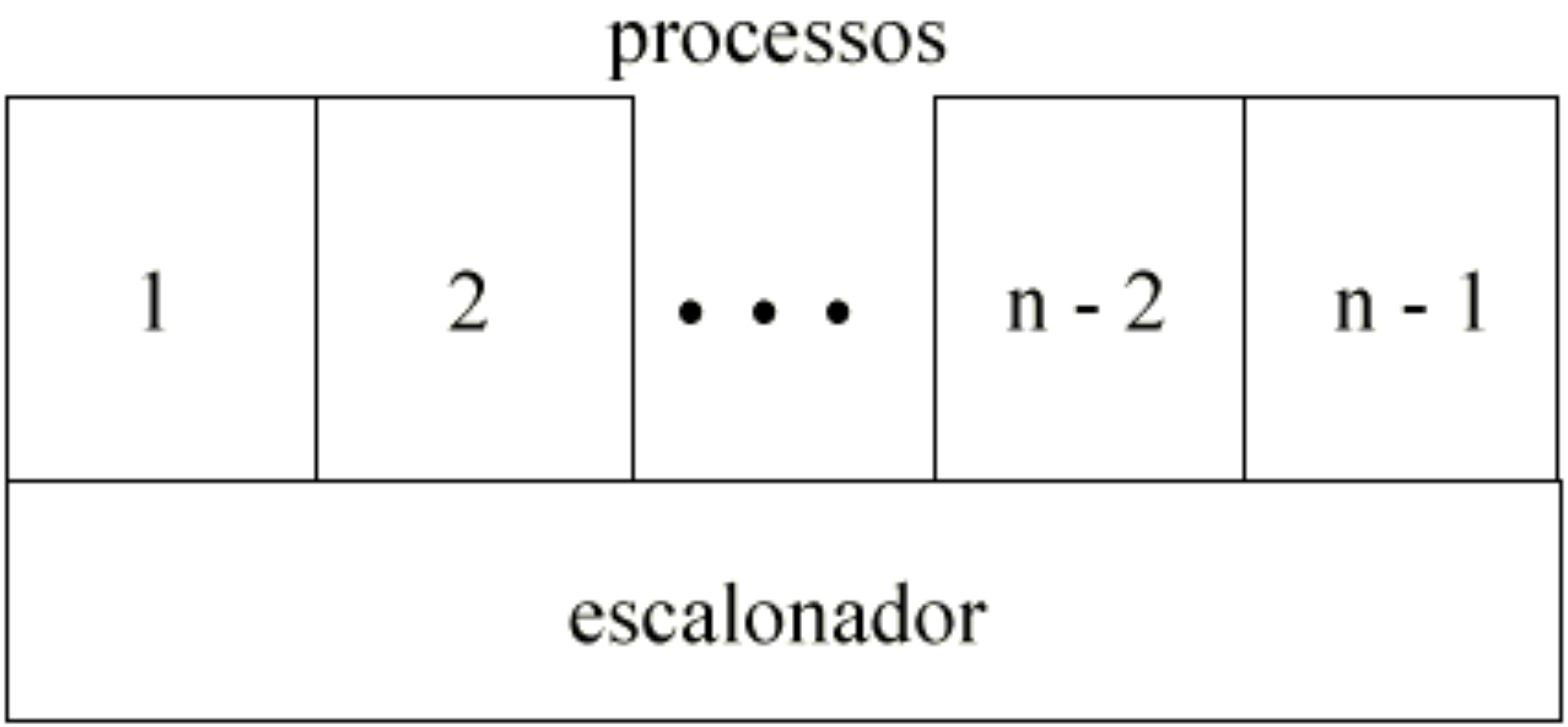
- **Exemplo de Reentrância**



Processos

- **Outras características dos processos**
  - **Os processos não podem ser programados com suposições embutidas sobre temporização**
  - **Em geral, um sistema multiplexa um só processador central entre vários processos. Os instantes em que o processador é realocado de um processo para outro são, em geral, imprevisíveis**
  - **Um algoritmo de escalonamento determina quando parar o trabalho com um processo e atender um diferente**

Processos



Criação de Processos

Principais eventos que causam a criação de um processo

1. Inicialização do Sistema
2. Execução de uma chamada ao sistema de criação de processo por um processo em execução
3. Uma requisição de usuário para a criação de um novo processo
4. Início de um job em lote

Término de um Processo

Condições que podem provocar o término de um processo

1. Saída normal (voluntária)
2. Saída por erro (voluntária)
3. Erro Fatal (involuntária)
4. Destruído por outro processo (involuntário)

Hierarquias de Processos

- Pai cria um processo filho, processo filho pode criar seus próprios processos
- Formação de hierarquia
  - UNIX chama a isto um “grupo de processos” ("process group”)
- Windows não possui conceito de hierarquia de processos
  - Todos os processos são criados da mesma forma

Processos

- **Os Processos do Ponto de Vista do S.O.**

  Um sistema operacional deve incluir funções para o gerenciamento de processos, como por exemplo:

  - **criação e remoção (destruição) de processos;**
  - **controle do progresso dos processos;**
  - **agir em condições excepcionais que acontecem durante a execução de um processo, incluindo interrupções e erros aritméticos;**
  - **alocação de recursos de hardware entre os processos;**
  - **fornecer um meio de comunicação através de mensagens ou sinais entre os processos.**

Processos

Existem 2 tipos de sistemas, com relação aos processos:

- **Sistemas Estáticos**

  Geralmente são sistemas projetados para executar uma única aplicação. Permite que todos os processos necessários já estejam presentes quando o sistema é iniciado.

- **Sistemas Dinâmicos**

  Possuem um número variável de processos. Necessita de uma forma de criar e destruir processos durante a execução.

Processos

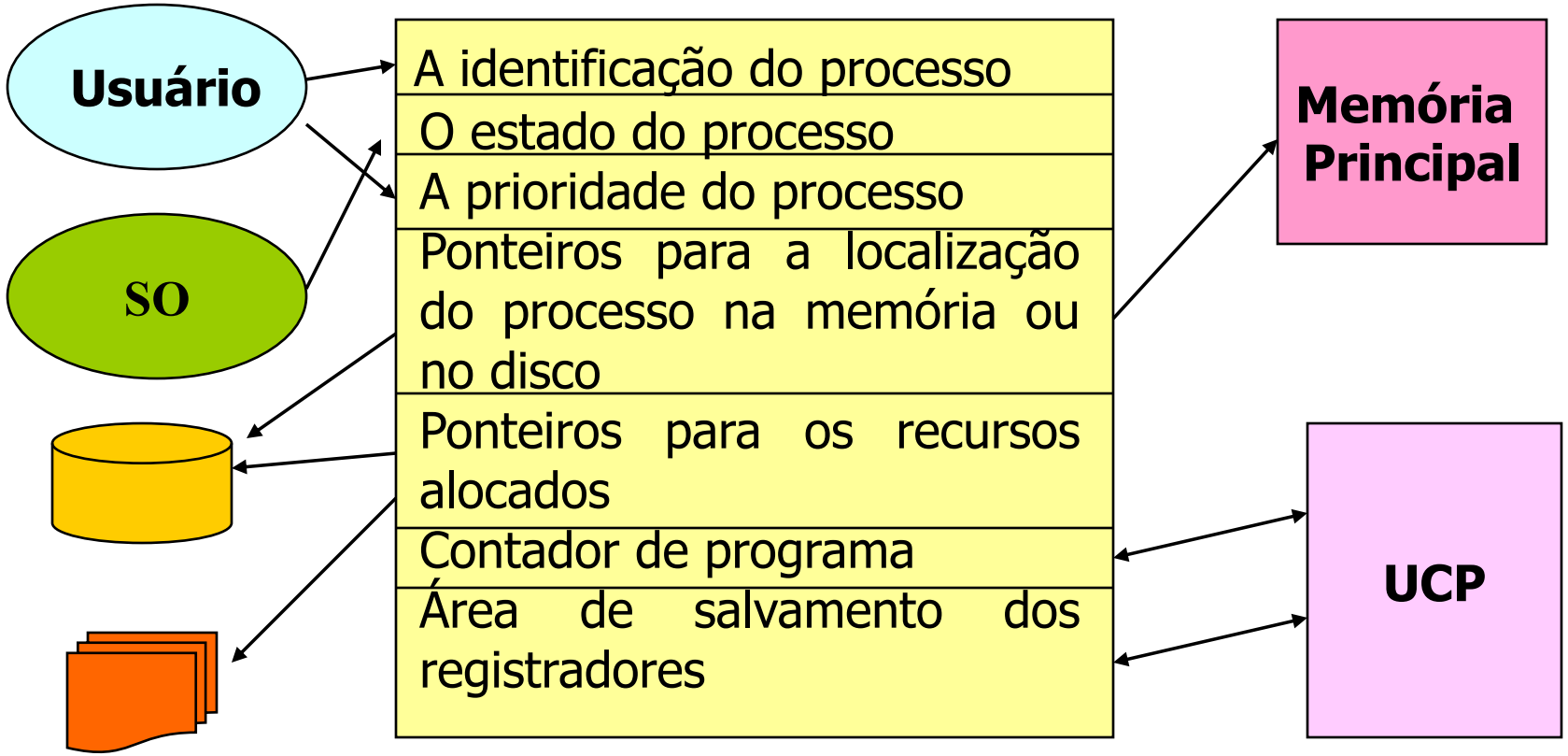
- **O Bloco de Controle de Processos**

  No sistema, cada processo será representado por seu resumo, que consiste no Bloco de Controle de Processo (BCP), também conhecido por Bloco de Controle de Programa ou Descritor de Processo. O BCP consiste de uma estrutura de dados contendo informações importantes sobre o processo, incluindo:

  - **A identificação do processo;**
  - **O estado do processo;**
  - **A prioridade do processo;**
  - **Ponteiros para a localização do processo na memória ou no disco;**
  - **Ponteiros para os recursos alocados;**
  - **Contador de programa;**
  - **Área de salvamento dos registradores;**
  - **etc.**

Processos

O Bloco de Controle de Processos



Processos

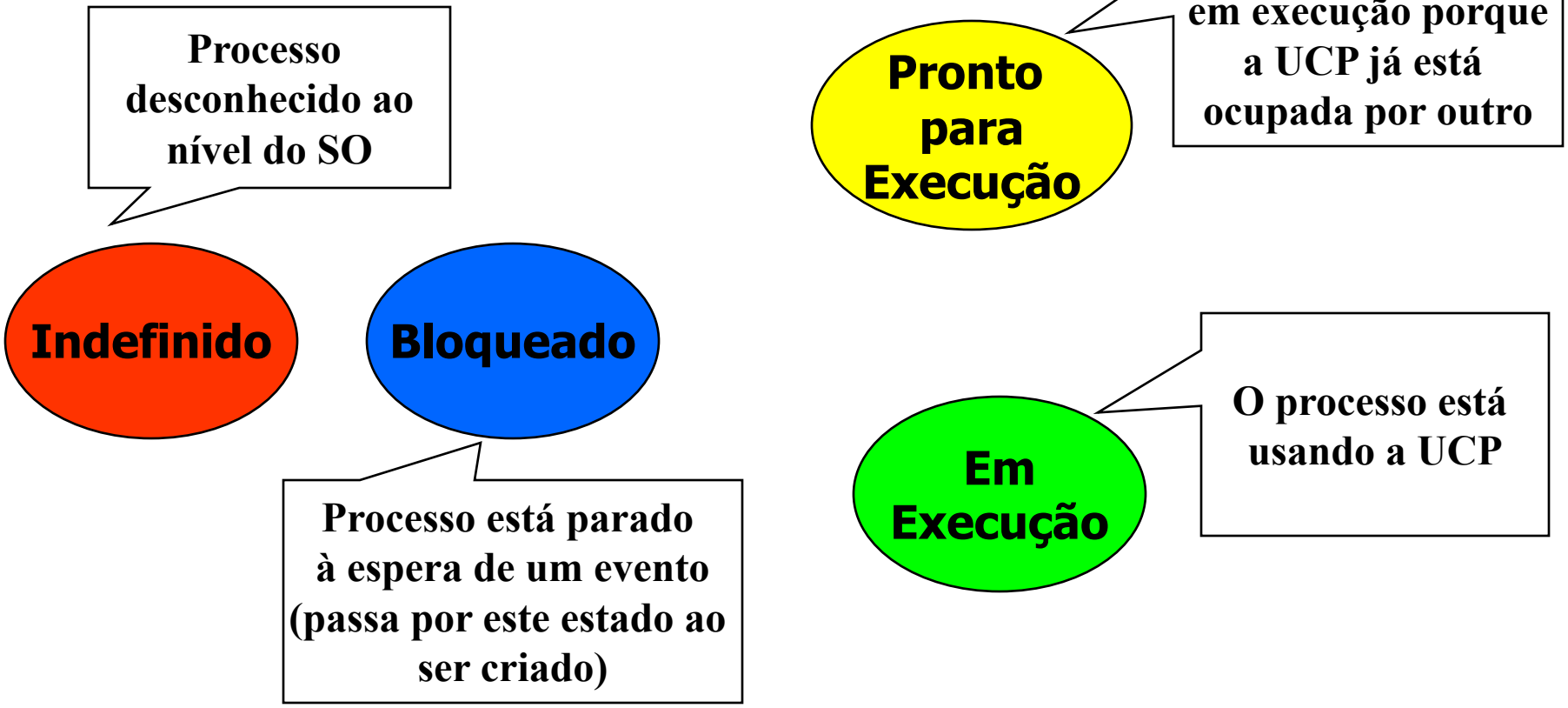
Os Estados de um Processo

Desde o momento em que um processo codificado pelo programador for colocado em memória ou disco, até o momento de sua destruição, ele poderá passar por quatro estados:

- **Indefinido:** quando o processo é desconhecido ao S.O. Um processo estará neste estado antes de ser criado e depois de ser destruído. (Neste ponto, ele será apenas um bloco de código no disco ou na memória).
- **Bloqueado:** quando o processo estiver parado à espera da ocorrência de um evento, equivalente a não estar em andamento progressivo. Ao ser criado, o processo passará por este estado.
- **Pronto para a execução:** quando o processo só não estiver em execução pelo fato da UCP estar sendo utilizada por outro processo.
- **Em execução:** quando o processo estiver tendo andamento progressivo normal, usando a UCP.

Processos

Os Estados de um Processo



Processos

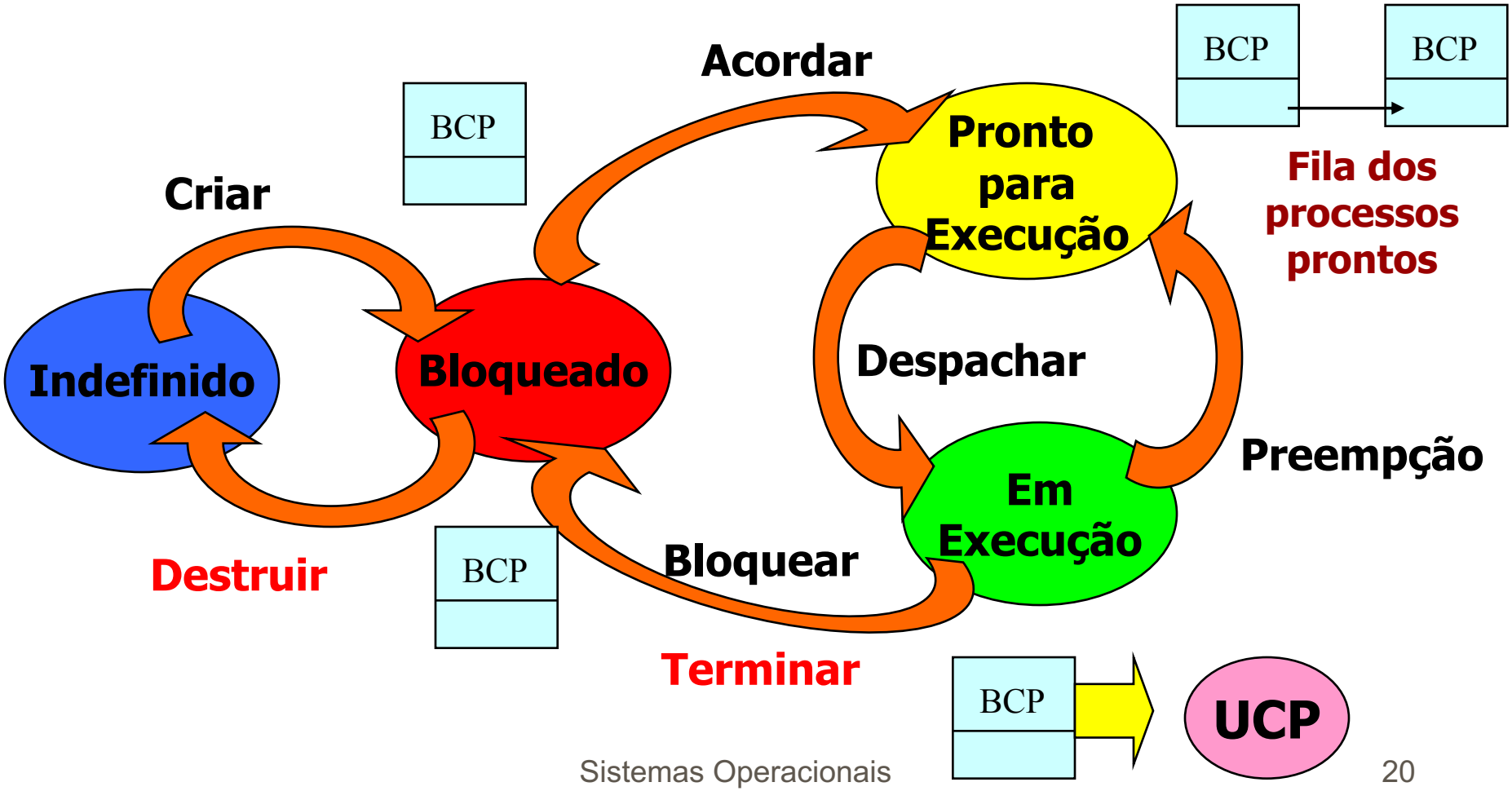
As Ações que provocam mudanças de estado

As ações responsáveis pelas transições tem o seguinte significado:

- **Criar:** colocar o processo a memória, tornando-o conhecido ao nível do sistema.
- **Acordar:** liberar o andamento do processo. Esta ação será desempenhada quando ocorrer algum evento ou o disparo inicial do processo, fazendo com que haja a transição do estado bloqueado para o pronto para execução.
- **Despachar:** dar seqüência imediata ao andamento do processo. O processo deverá estar em uma fila, e a ocorrência dessa ação resultará na retirada do processo da fila e sua colocação no estado em execução.
- **Bloquear:** parar o andamento do processo para esperar a ocorrência de um evento. A ação bloquear será desempenhada quando uma determinada condição não for satisfeita, ou quando ocorrer o término do processo.
- **Preempção (Suspender):** parar o andamento do processo por motivos alheios ao mesmo, como acontece na comutação forçada de fatias de tempo. Esta ação retirará o processo da UCP e o colocará numa fila, no estado pronto para a execução, liberando a utilização da UCP.
- **Destruir:** liberar a área de memória ocupada pelo processo, tornando-o desconhecido ao nível do sistema.

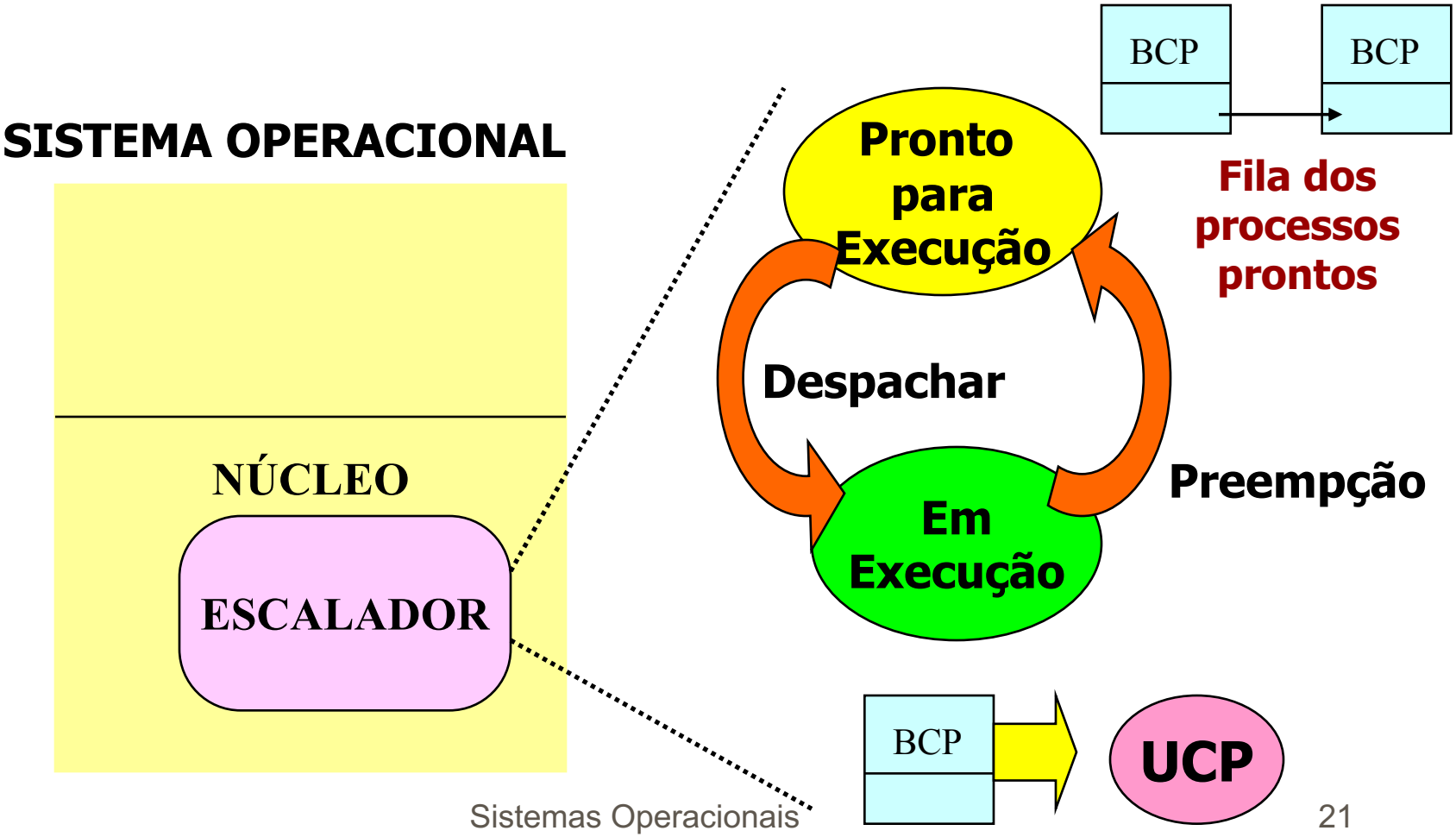
Processos

- As Ações que provocam a transição entre estados



Processos

- As Ações de Responsabilidade do Escalador



Threads

- Sistemas tradicionais – cada processo tem um espaço de endereçamento e um único thread (fluxo) de controle
- Situações frequentes: necessidade de múltiplos threads de controle no mesmo espaço de endereçamento executando em quase-paralelo, como se fossem processos separados

Threads
Modelo de Thread

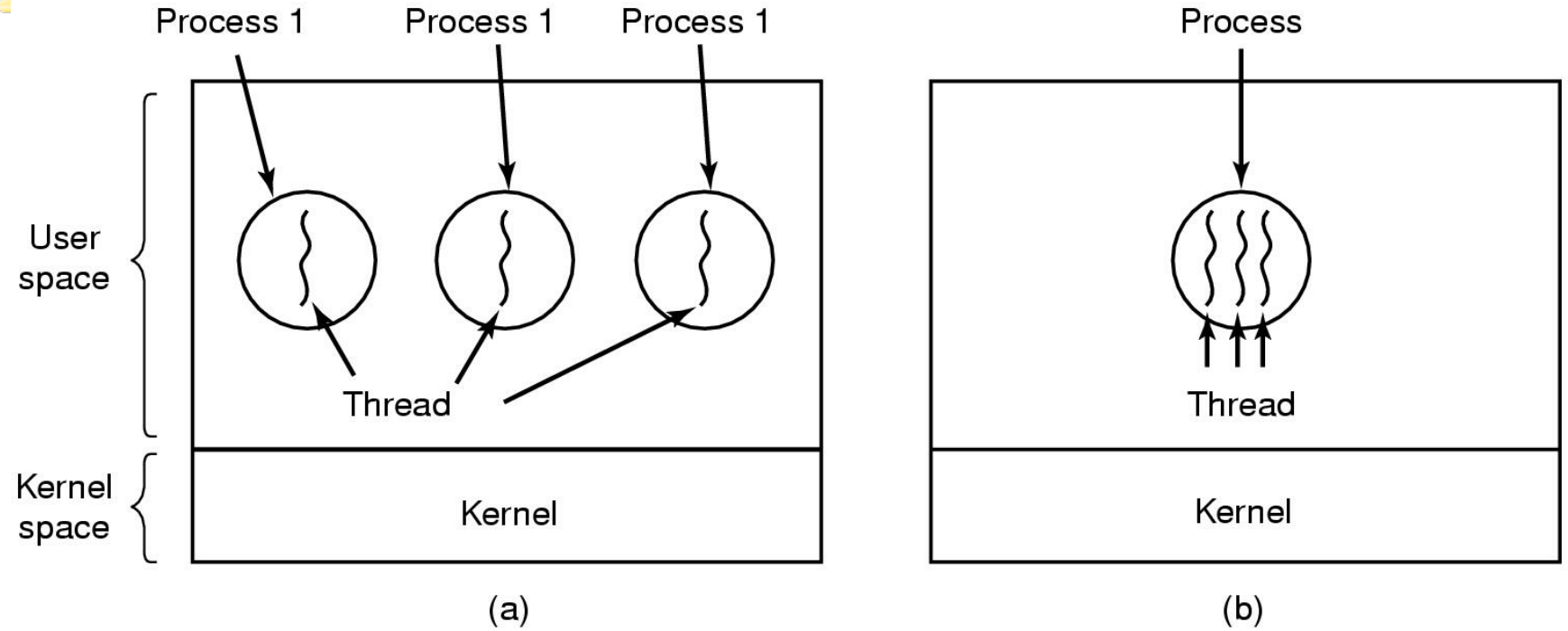
- O Modelo de Processo– baseado em dois conceitos independentes: agrupamento de recursos e execução
- Algumas vezes é interessante separá-los: este é o caso dos threads
- A execução é representado pelo **thread**, também conhecido por **thread de execução** ou **processo leve** (*lightweight process*)

Threads
Modelo de Thread

- Utilidade dos threads – o que acrescentam ao modelo de processo é permitir que múltiplas execuções ocorram no mesmo ambiente do processo com um grande grau de independência uma da outra
- Os threads possuem independência, uma vez que possuem, privadamente: contador de programa, registradores, pilha e estado
- Multithread: situação em que se permite a existência de vários threads no mesmo processo

Threads

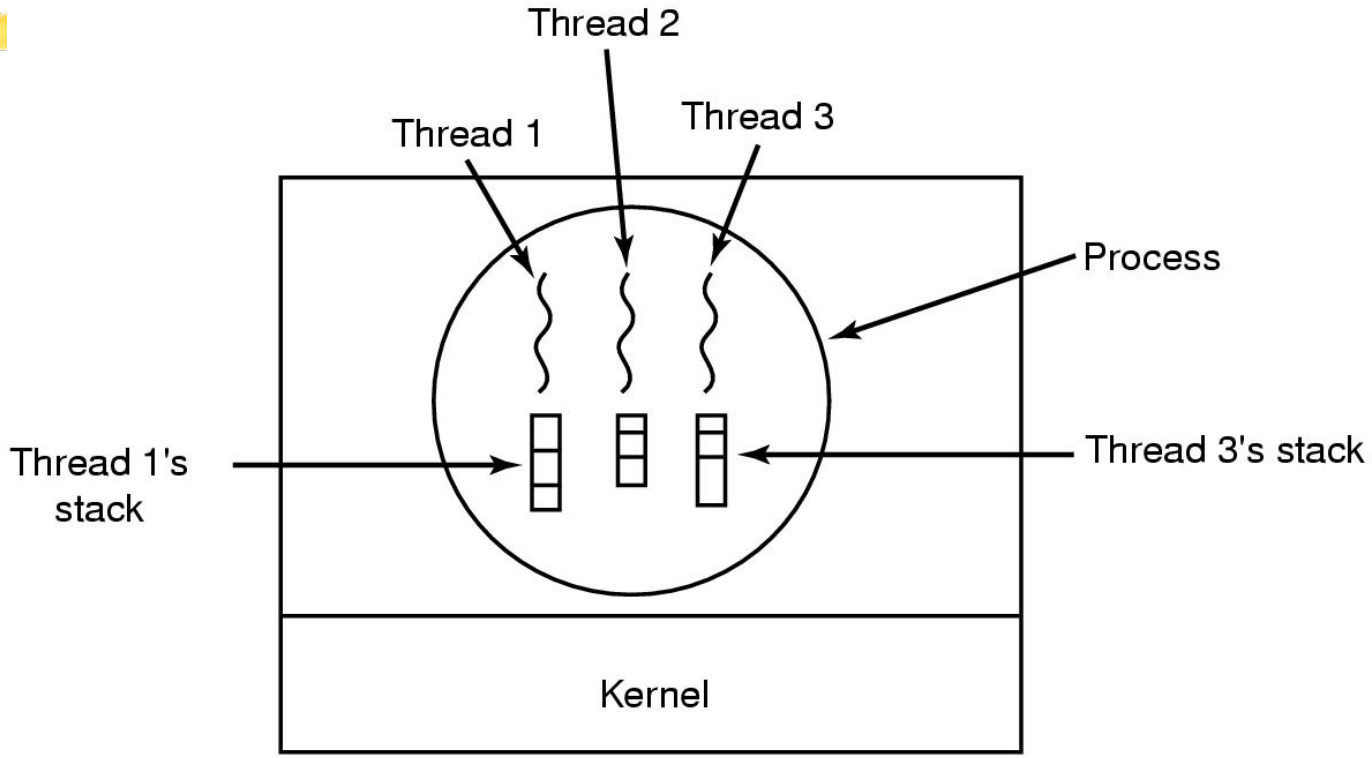
O Modelo Thread



(a) Três processos, cada um com um thread

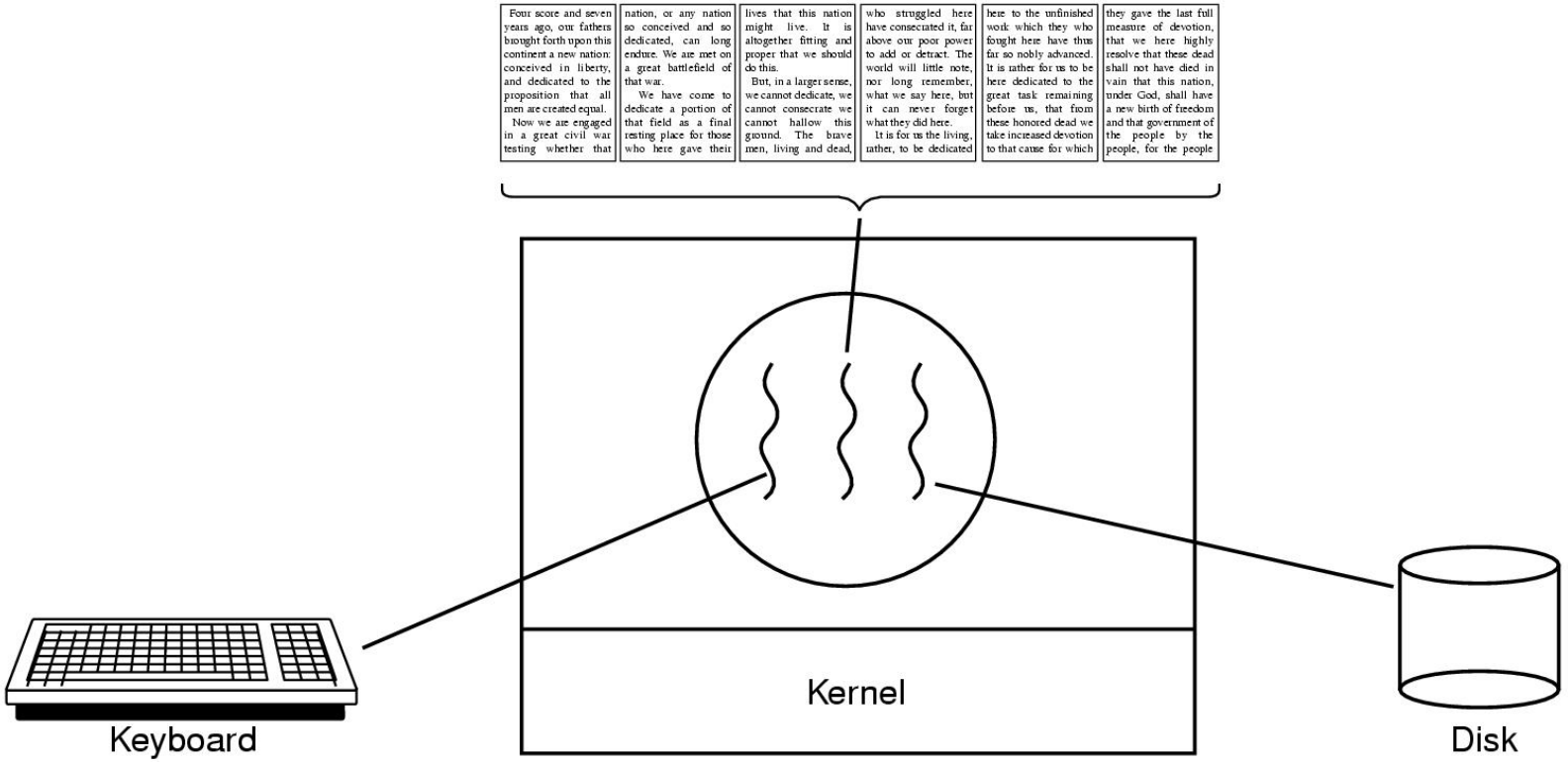
(b) Um processo com três threads

O Modelo Thread



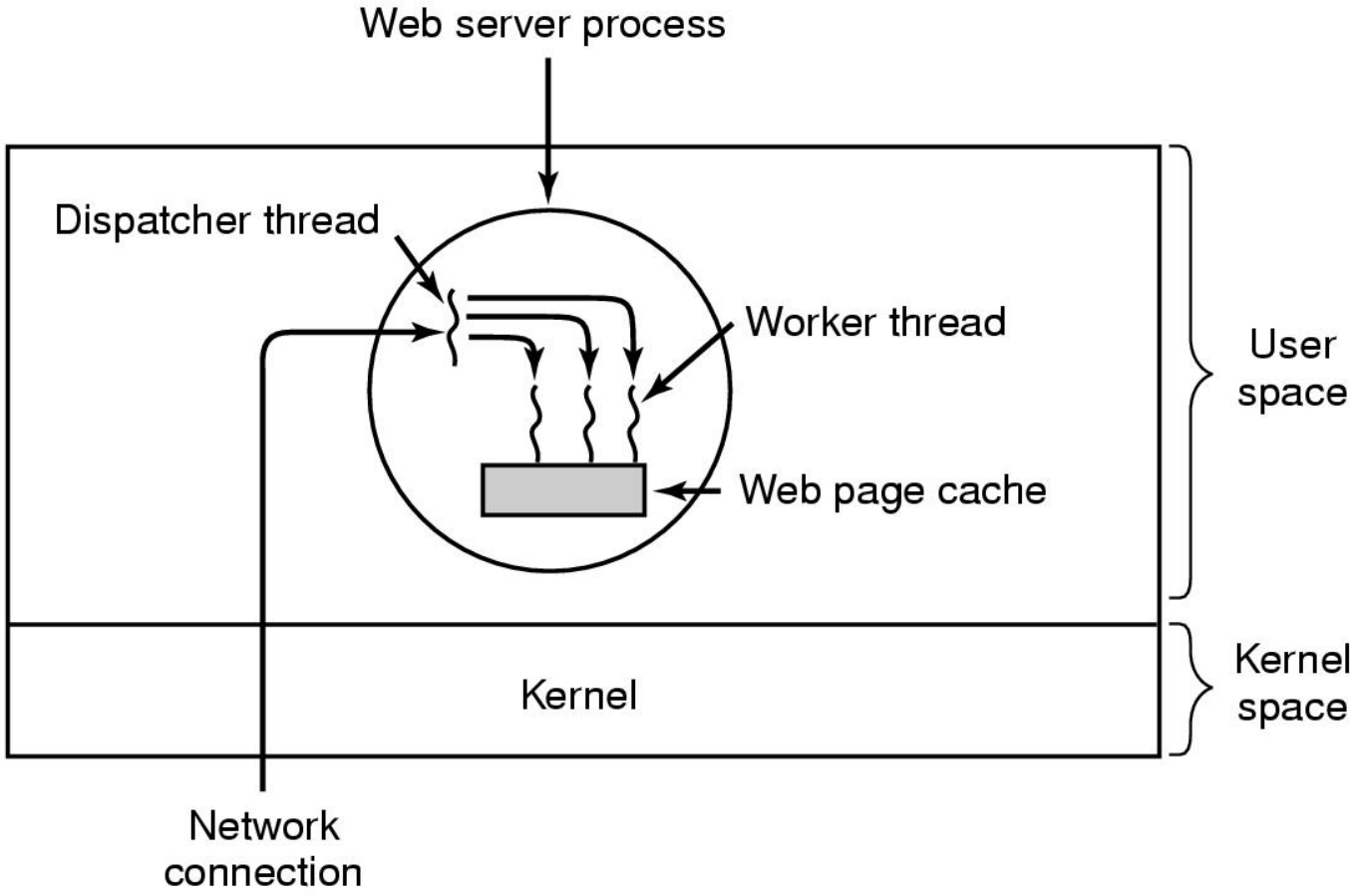
Cada thread tem sua própria pilha

Uso dos threads



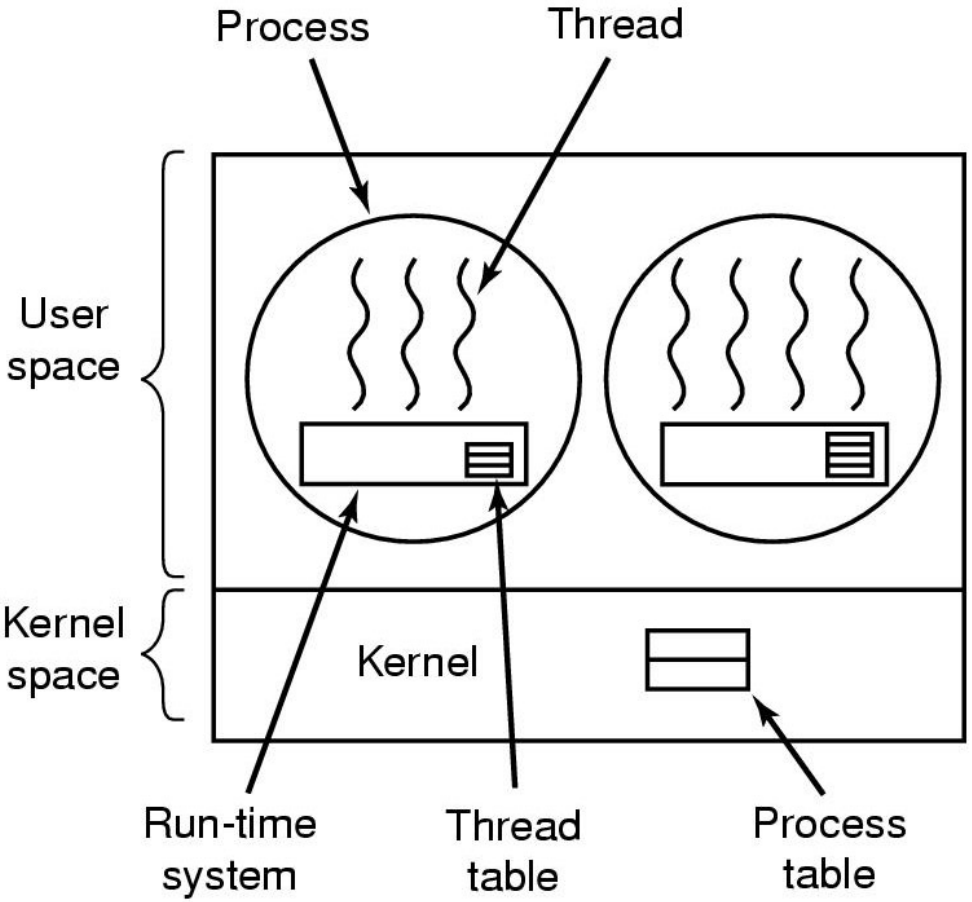
Um processador de textos com 3 threads

Uso dos Threads



Um Servidor Web multithreaded

Implementando Threads no Espaço do Usuário



Um pacote de Threads ao nível do Usuário

Implementando Threads no Espaço do Usuário

- O pacote de threads é inserido totalmente dentro do espaço do usuário (threads de usuário)
- O núcleo não é informado sobre eles
- Núcleo: compete apenas o gerenciamento comum de processos monothread

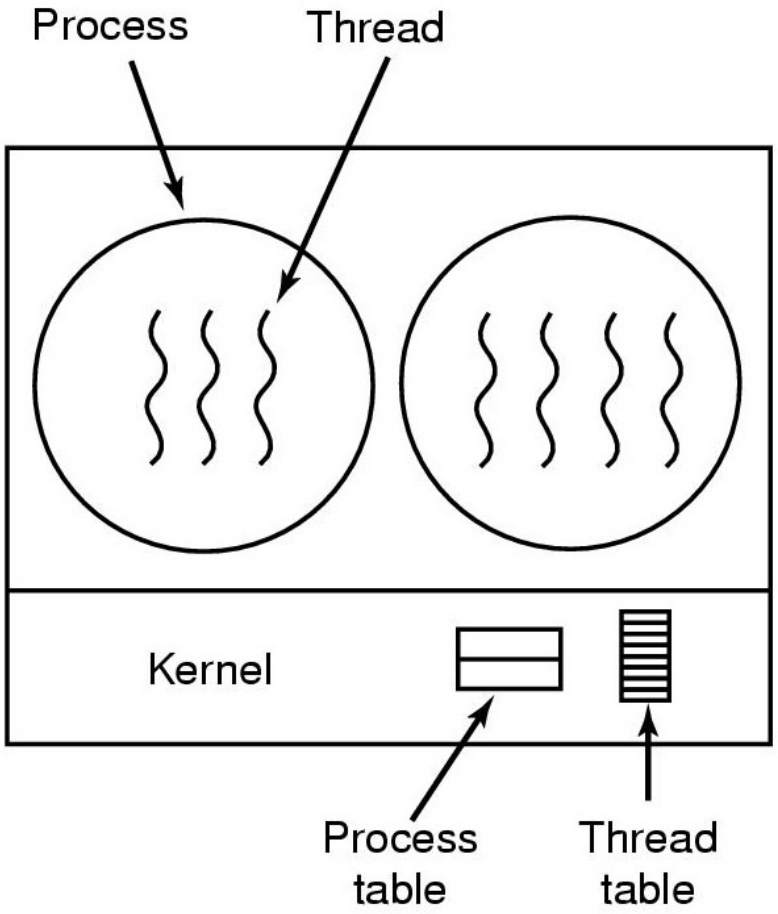
Implementando Threads no Espaço do Usuário

- Vantagens
  - Pode ser implementado em S.O. que não suporta threads, através do uso de um sistema supervisor (run-time)
  - Os threads são gerenciados no espaço do usuário
  - Permite que cada processo tenha seu próprio algoritmo de escalonamento personalizado

Implementando Threads no Espaço do Usuário

- Problemas
  - Como implementar chamadas com bloqueio
  - Problema da falta de página
  - Os threads não podem ser temporizados – devem abrir mão da UCP voluntariamente
  - Não indicados se a aplicação causa muitos bloqueios (justamente quando seriam indicados os threads)

Implementando Threads no Kernel

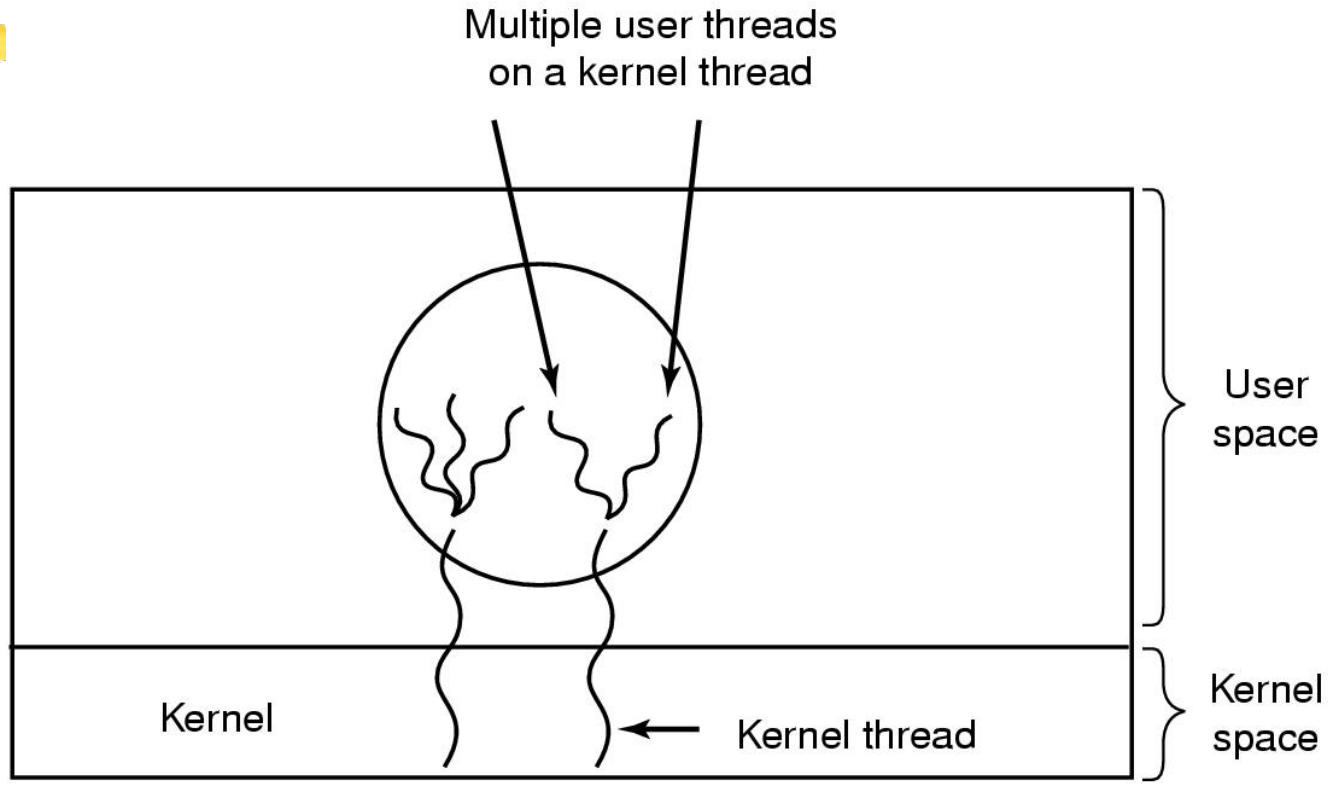


Um pacote de threads gerenciados pelo kernel

Implementando Threads no Kernel

- O núcleo conhece os threads e os gerenciam
- O núcleo possui uma tabela de threads que acompanham todas as threads no sistema
- Todas as chamadas que possam bloquear um thread são implementadas como chamadas ao sistema: custo maior
- Alguns sistemas: existência da reciclagem de threads

Implementações Híbridas



Multiplexando threads ao nível do usuário através de threads ao nível do kernel

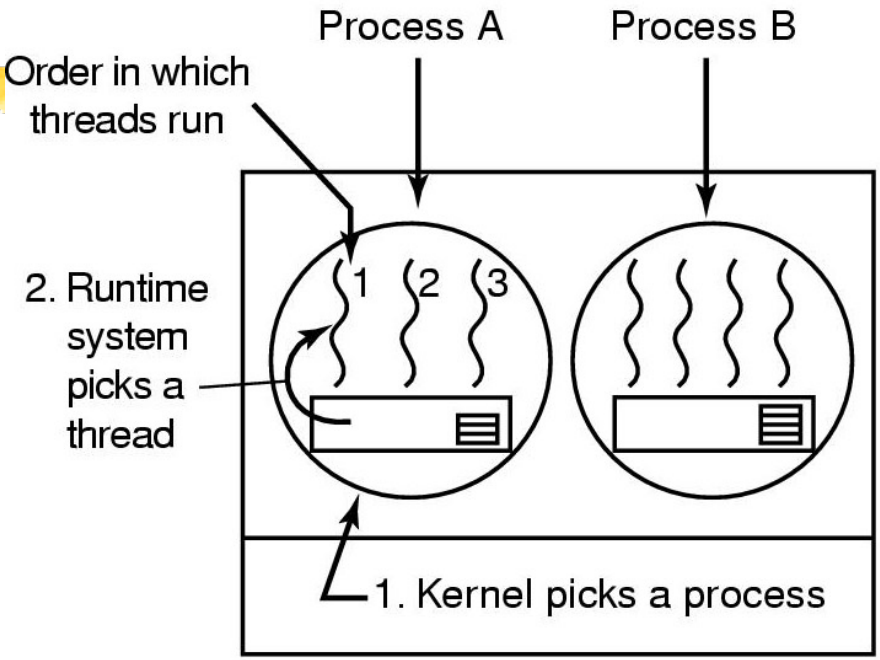
Implementações Híbridas

- Combinar as vantagens das implementações anteriores
- Uma forma: usar threads de núcleo e multiplexar threads de usuário sobre algum ou todos os threads de núcleo
- Núcleo: só sabe dos threads de núcleo e escalona-os
- Alguns destes threads podem ter multiplexados diversos threads de usuário (sistema supervisor)

Ativações do Escalonador

- Objetivo – imitar a funcionalidade dos threads de kernel
  - ganhar desempenho dos threads do espaço do usuário
- Evitar transições desnecessárias entre Usuário/Kernel
- Kernel associa processadores virtuais a cada processo
  - deixa o sistema de runtime alocar threads aos processadores
- Problema:
  Dependência no kernel (baixo nível) chamando procedimentos no espaço do usuário (alto nível)

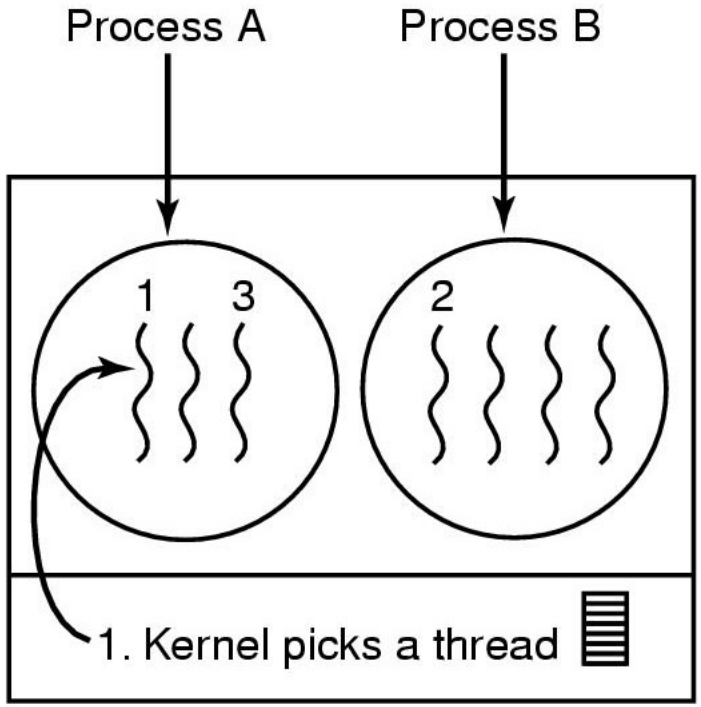
Escalonamento de Thread



Possível escalonamento dos threads de nível de usuário

- quantum do processo é de 50-mseg
- threads executam 5 mseg/CPU no máximo

Escalonamento de Thread



Possible: A1, A2, A3, A1, A2, A3
Also possible: A1, B1, A2, B2, A3, B3

Possível escalonamento dos threads de nível de kernel

- quantum do processo é de 50-mseg
- threads executam 5 mseg/CPU no máximo