Capítulo 4: Threads

Sobre a apresentação (About the slides)

Os slides e figuras dessa apresentação foram criados por Silberschatz, Galvin e Gagne em 2005. Esse apresentação foi modificada por Cristiaino Costa (cac@unisinos.br). Basicamente, os slides originais foram traduzidos para o Português do Brasil.

É possível acessar os slides originais em http://www.os-book.com

Essa versão pode ser obtida em http://www.inf.unisinos.br/~cac

The slides and figures in this presentation are copyright Silberschatz, Galvin and Gagne, 2005. This presentation has been modified by Cristiano Costa (cac@unisinos.br). Basically it was translated to Brazilian Portuguese.

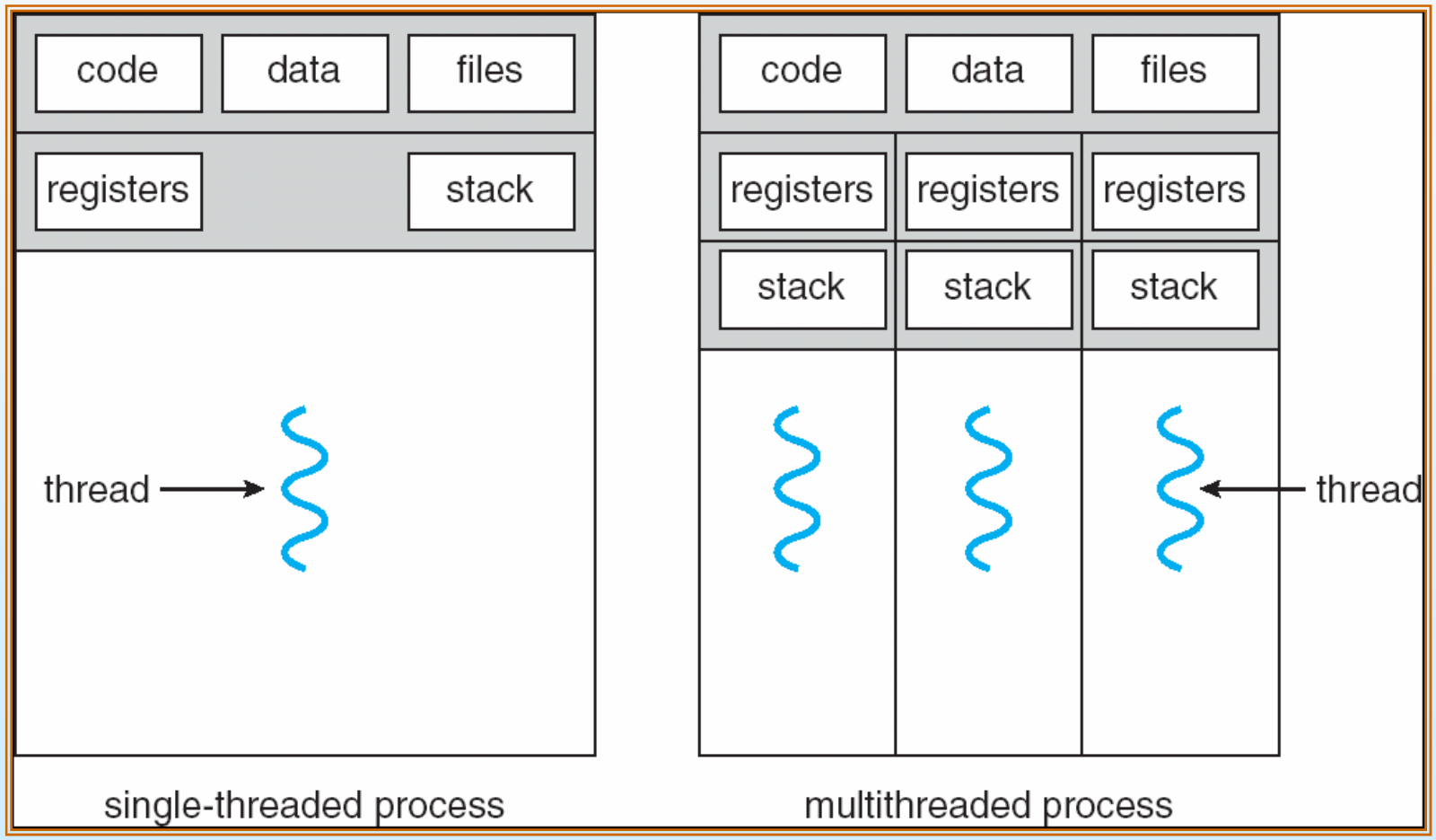
You can access the original slides at http://www.os-book.com

This version could be downloaded at http://www.inf.unisinos.br/~cac

Capítulo 4: Threads

- Visão Geral
- Modelos de Múltiplas Threads
- Questões sobre Threads
- Pthreads
- Threads no Windows XP
- Threads no Linux
- Threads em Java

Processos com uma e múltiplas Threads



Benefícios

- Responsividade
- Compartilhamento de Recursos
- Economia
- Utilização de arquiteturas multiprocessadas (MP)

Threads em Nível Usuário

- Gerência de Threads é feito por bibliotecas em nível de usuário
- Trê bibliotecas de threads principais:
  - POSIX Pthreads
  - Win32 threads
  - Java threads

Threads em Nível Kernel

- Suportada pelo Kernel
- Exemplos
  - Windows XP/2000
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

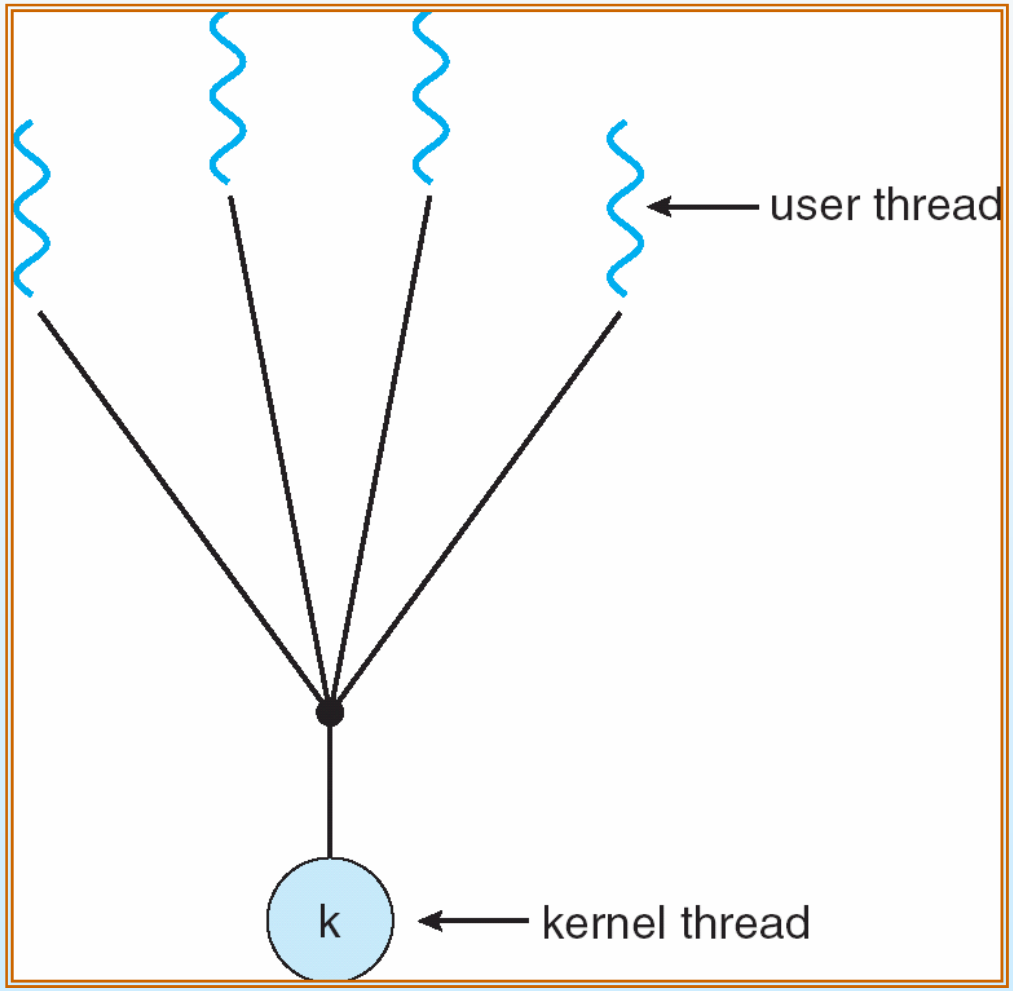
Modelos de Múltiplas Threads

- Muitos-para-Um
- Um-para-Um
- Muitos-para-Muitos

Modelo Muitos-para-Um

- Muitas threads em nível usuário são mapeadas para uma única thread no kernel
- Exemplos:
  - Solaris Green Threads
  - GNU Portable Threads

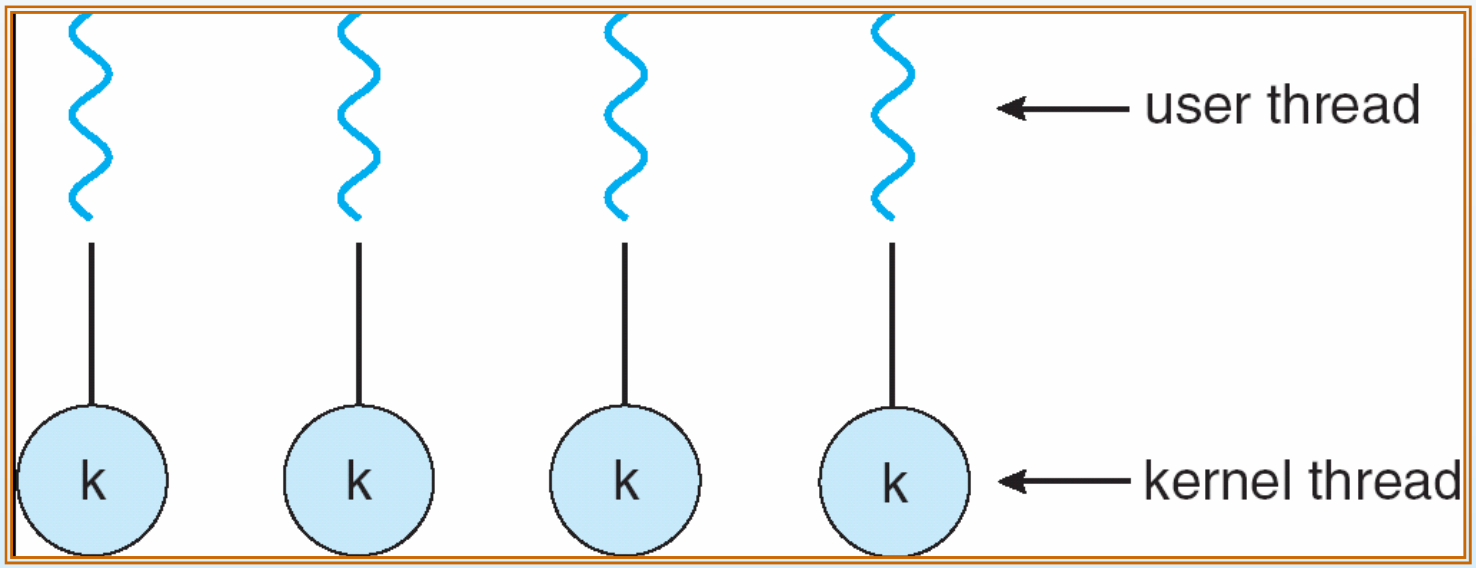
Modelo Muitos-para-Um (cont.)



Modelo Um-para-Um

- Cada thread em nível usuário é mapeada para uma thread em nível kernel
- Exemplos
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 e posteriores

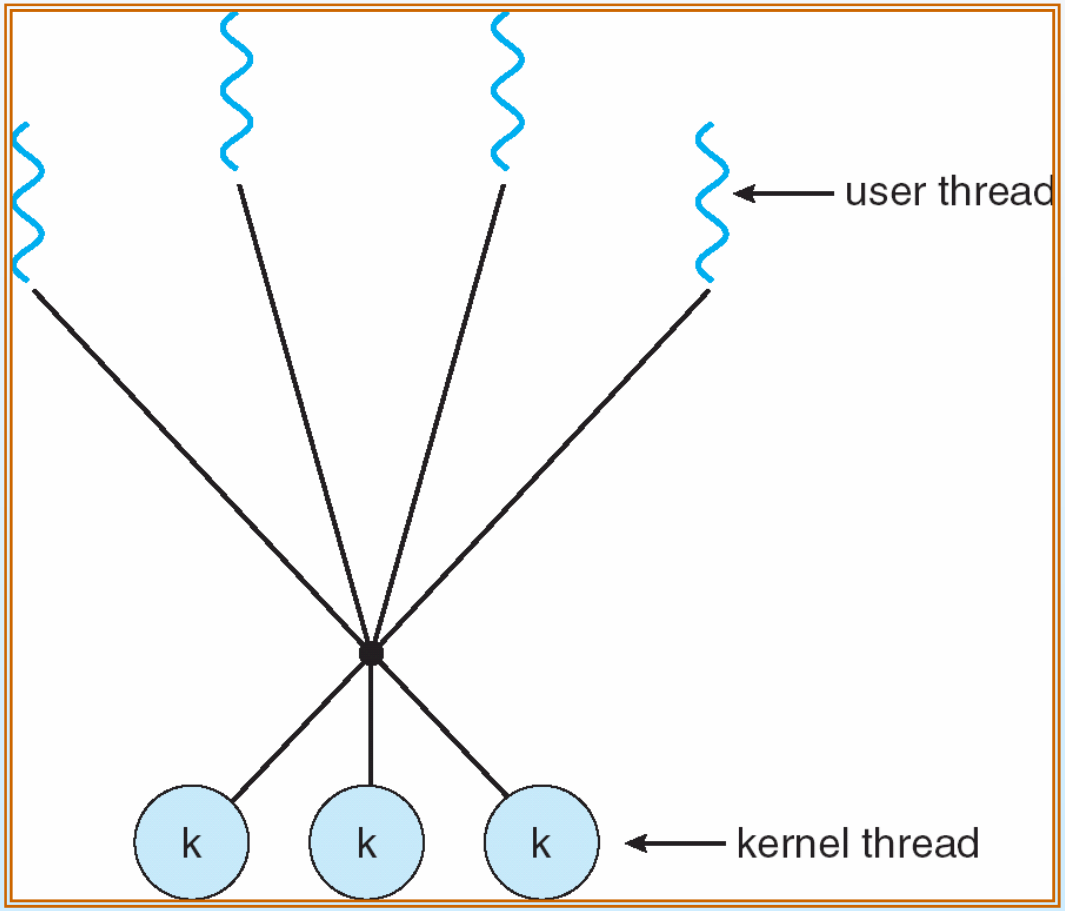
Modelo Um-para-Um (cont.)



Modelo Muitos-para-Muitos

- Permite que muitas threads em nível usuário sejam mapeadas para muitas threads em nível kernel
- Permite que o sistema operacional crie um número suficiente de threads no kernel
- Solaris versão anterior a 9
- Windows NT/2000 com o pacote *ThreadFiber*

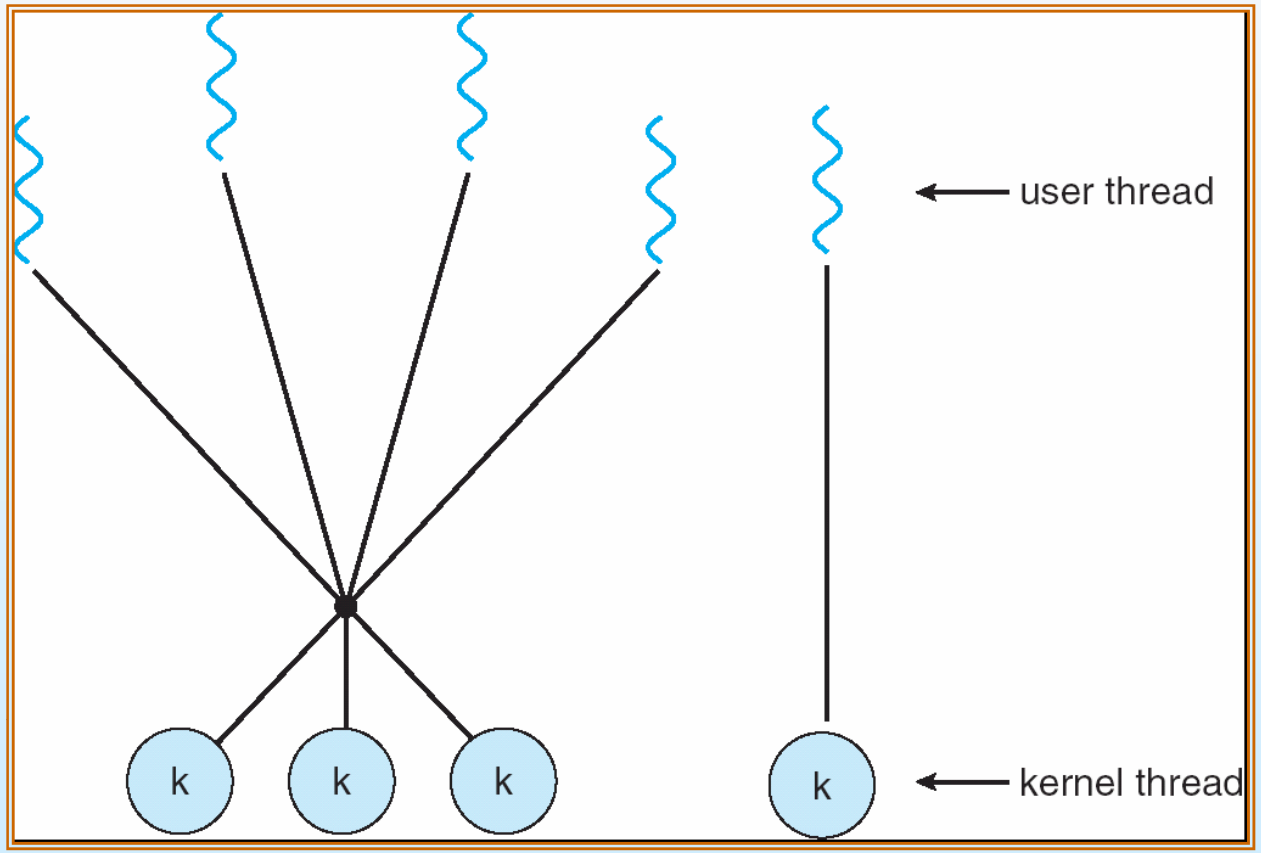
Modelo Muitos-para-Muitos (cont.)



Modelo de Dois Níveis

- Similar ao M:M, exceto que ele permite que uma thread do usuário seja **amarrada** (*bind*) a uma thread no kernel
- Exemplos
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 e anterior

Modelo de Dois Níveis (cont.)



Questões sobre Threads

- Semântica das chamadas de sistemas **fork()** e **exec()**
- Cancelamento de Thread
- Manipulação de Sinais
- Conjunto de Thread (Thread Pools)
- Dados Específicos de Thread
- Ativações de Escalonamento

Semântica de fork() e exec()

- O **fork()** duplica somente a thread chamadora ou todas as threads?

Cancelamento de Thread

- Terminação de uma thread antes dela ter finalizado
- Duas abordagens:
  - **Cancelamento Assíncrono** termina a thread alvo imediatamente
  - **Cancelamento Delegado** permite que a thread alvo seja periodicamente verificada se deve ser cancelada

Manipulação de Sinais

- Sinais são usados nos sistemas UNIX para notificar um processo que um evento particular ocorreu
- Um **manipulador de sinais** (*signal handler*) é usado para processar sinalizações
  1. Sinal é gerado por um evento particular
  2. Sinal é enviado a um processo
  3. Sinal é manipulado
- Opções:
  - Enviar o sinal para a thread para qual ele se aplica
  - Enviar o sinal para cada thread no processo
  - Enviar o sinal para determinadas threads no processo
  - Associar uma thread específica para receber todos os sinais enviados para o processo

Conjunto de Threads

- Cria um número de threads que formam um conjunto para espera de trabalho
- Vantagens:
  - Usualmente torna um pouco mais rápido o atendimento a uma requisição com uma thread existente do que criar uma nova
  - Permite que o número de threads na aplicação seja limitado pelo tamanho do conjunto

Dados Específicos de Thread

- Permite que cada thread tenha seu próprio conjunto de dados
- Útil quando não se tem controle sobre o processo de criação de threads (ex. quando se usa um conjunto de threads)

Ativações de Escalonamento

- Tanto o modelo M:M quanto em dois níveis requer comunicação para manter o número apropriado de threads no kernel alocado para a aplicação
- Ativações de escalonamento fornecem ***upcalls*** - um mecanismo de comunicação do kernel para a biblioteca de threads
- Essa comunicação permite uma aplicação manter o número correto de threads no kernel

Pthreads

- Uma API padrão POSIX (IEEE 1003.1c) para criação e sincronização de threads
- A API especifica o comportamento da biblioteca de threads. A implementação está a cargo do desenvolvedor da biblioteca.
- Comum nos sistemas operacionais UNIX (Solaris, Linux, Mac OS X)

Threads no Windows XP

- Implementa o mapeamento um-para-um
- Cada thread contém
  - Um identificador de thread (id)
  - Conjunto de registrados
  - Pilhas separadas para kernel e usuário
  - Área privada de armazenamento de dados
- O conjunto de registradores, pilhas e área de armazenamento privado são denominados **contexto** da thread
- As principais estruturas de dados de uma thread são:
  - ETHREAD (*executive thread block*)
  - KTHREAD (*kernel thread block*)
  - TEB (*thread environment block*)

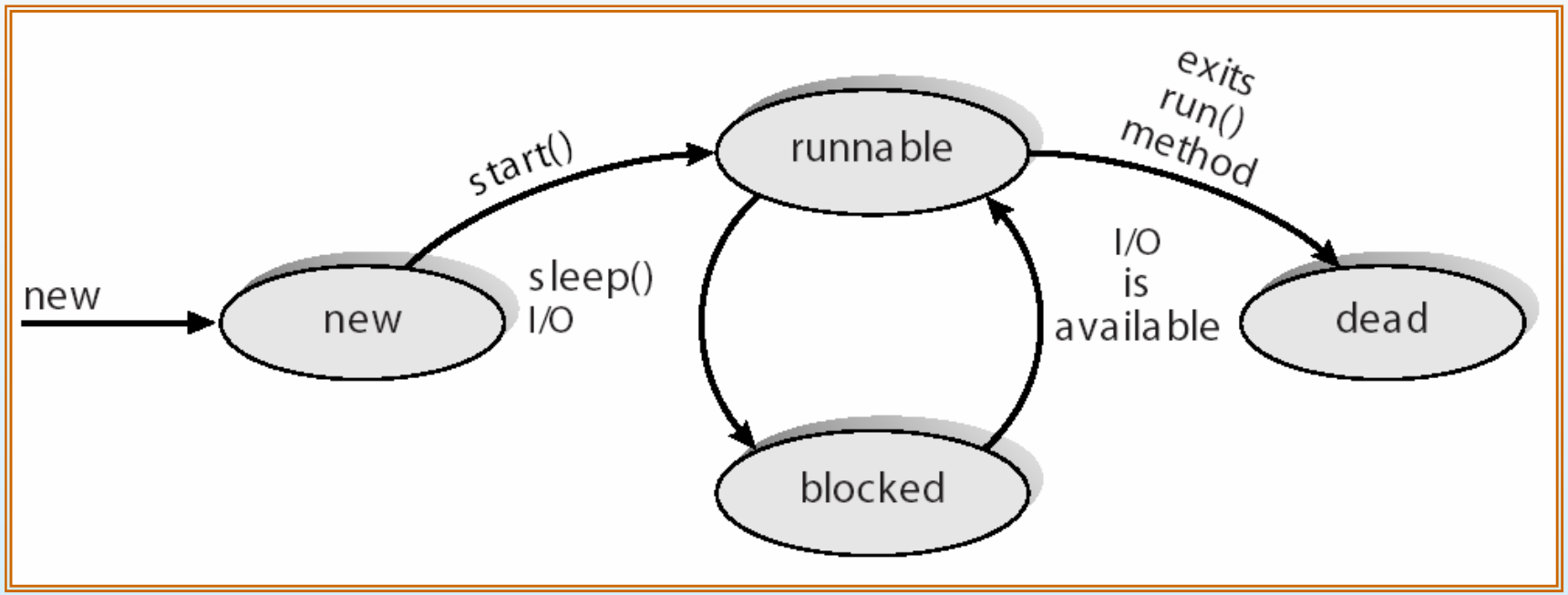
Threads no Linux

- No Linux são denominadas de tarefas (*tasks) ao invés de* threads
- Criação de threads é feita através da chamada de sistemas **clone()**
- **clone()** possibilita que uma tarefa filha compartilha o espaço de endereçamento com a tarefa pai (processo)

Threads em Java

- Java threads são gerenciadas pela JVM
- Java threads podem ser criadas:
  - Estendendo a classe *Thread*
  - Implementando a interface *Runnable*

Estados das Threads em Java



Fim do Capítulo 4