Programação Concorrente

Threads

André Luis Martinotto

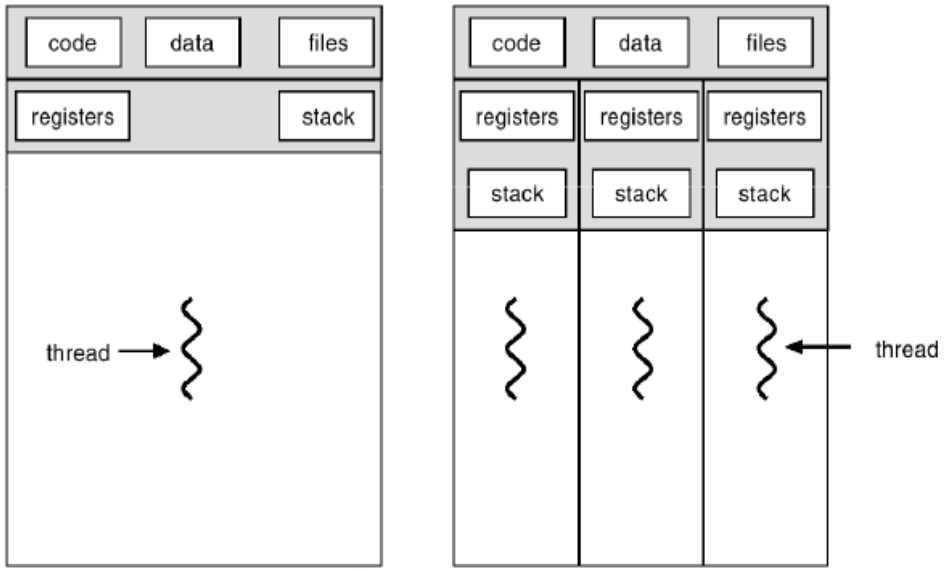
UCS UNIVERSIDADE DE CAXIAS DO SUL

Threads

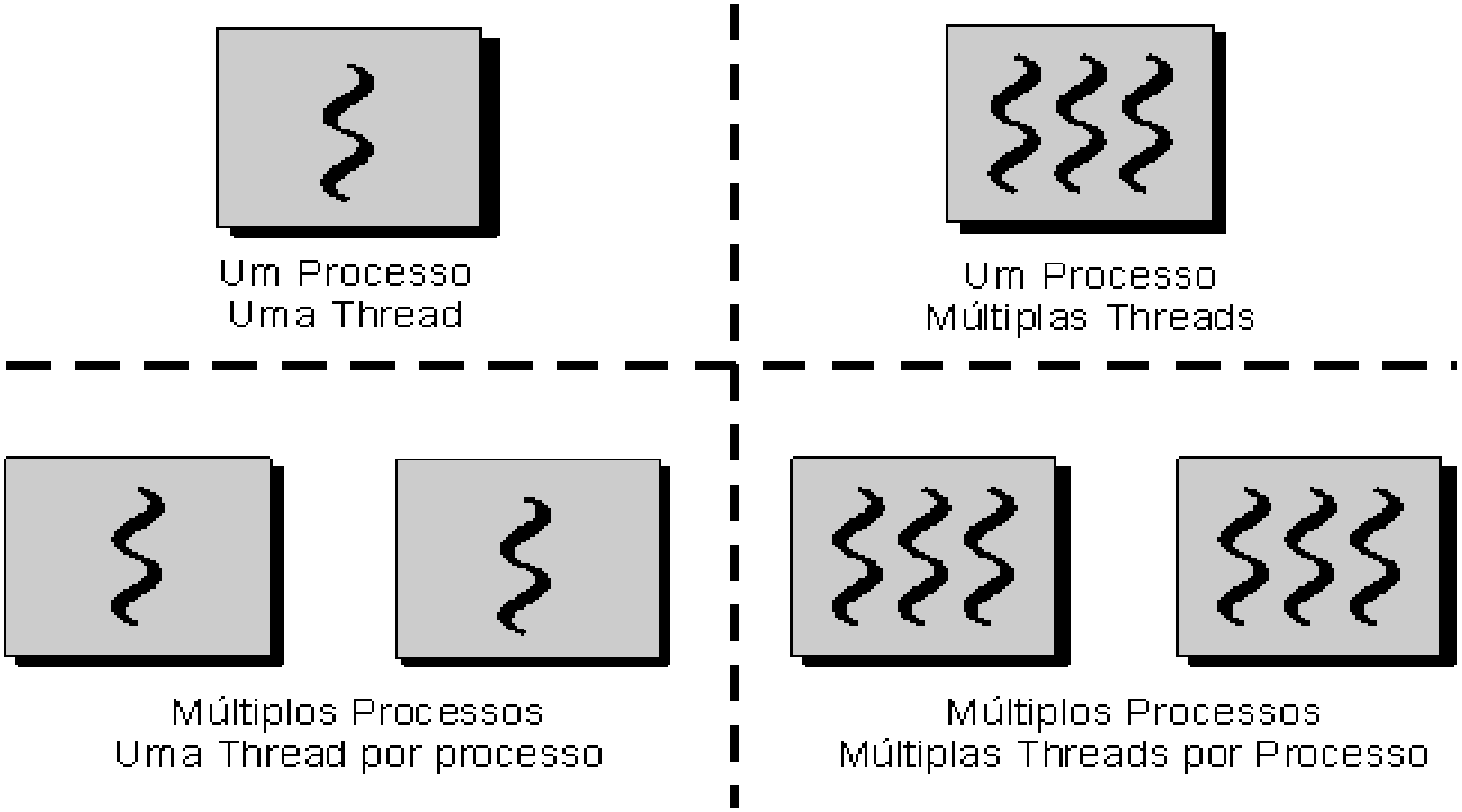
- **Processo:**
  - Programa em execução
- **Processo tradicional:**
  - Tem um único fluxo de execução
- **Thread:**
  - É uma forma de um processo dividir a si mesmo em duas ou mais tarefas (fluxos) que podem ser executadas concorrentemente

Threads vs Processos

- **Processos:**
  - Completamente separados com suas próprias variáveis, pilha, alocação de memória, etc.
- **Threads:**
  - Compartilham o mesmo espaço de memória e variáveis globais



Threads vs Processos



Threads vs Processos

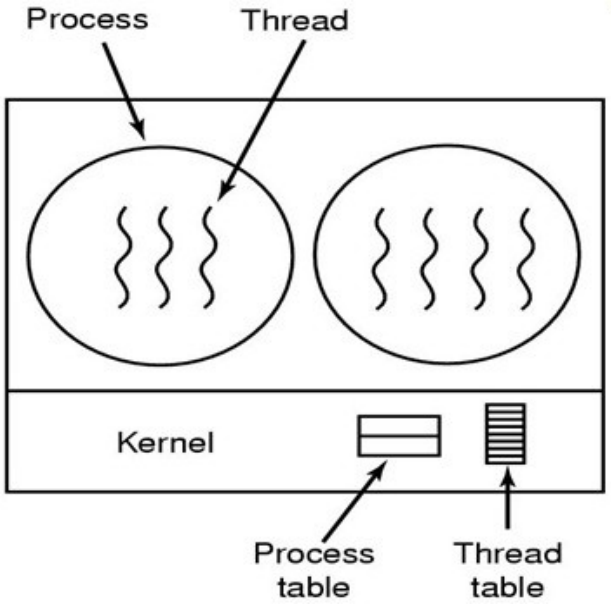
- **Vantagens das threads em relação aos processos:**
  - A criação e terminação é mais rápida
  - A troca de contexto é mais rápida
  - A comunicação entre threads é mais rápida
    - Threads compartilham tudo: espaço de endereçamento, variáveis globais etc.
    - Permite a comunicação por memória compartilhada sem interação com o núcleo
  - Multi-programação usando o modelo de threads é mais portável do que multi-programação usando múltiplos processos

Tipos de Threads

- **Kernel-Level Thread (KLT):**
  - O suporte à thread é fornecido pelo próprio sistema operacional
- **User-Level Thread (ULT):**
  - O suporte à thread é implementado através de uma biblioteca de uma determinada

Kernel-Level Thread

- As threads são consideradas na implementação do SO
  - Gerenciamento das threads é feito pelo núcleo
  - O núcleo mantém a informação de contexto dos processo e das threads
  - Escalonamento e chaveamento das threads é feito pelo núcleo



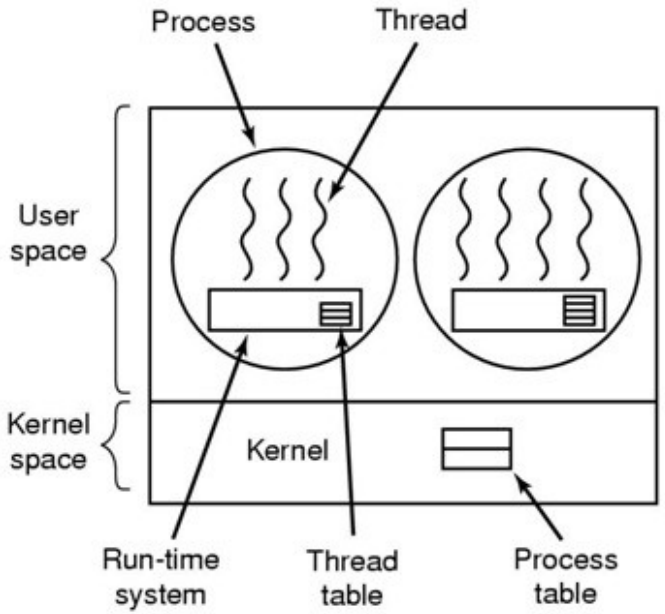
Kernel-Level Thread

- **Vantagens:**
  - Podem aproveitar a capacidade de multiprocessamento
  - O kernel pode simultaneamente escalonar várias threads do mesmo processo em vários processadores

- **Desvantagens:**
  - A troca entre threads implica acções do kernel e isto tem um custo que pode ser significativo

User-Level Thread

- O gerenciamento de threads é feito em nível da aplicação
  - São implementadas por uma biblioteca que é ligada ao programa
  - O sistema operacional não “enxerga” a presença das threads
  - A troca de contexto é realizada em modo usuário pelo escalonador embutido na biblioteca



User-Level Thread

- **Vantagens:**
  - A troca de contexto entre as threads não envolve o kernel
    - Não há o custo adicional de execução do kernel
    - A biblioteca pode oferecer vários métodos de escalonamento
      - Escalonamento pode ser especifico para uma aplicação
  - São portáveis podendo executar em qualquer SO

- **Desvantagens:**
  - Não explora paralelismo em máquinas multiprocessadas
    - O kernel vai atribuir o processo a uma CPU portanto duas threads de um mesmo processo não podem executar simultaneamente

Latência de Operações

	Tempo de Criação (µs)	Tempo de Sincronização (µs)
Processos	11300	1840
KLTs	948	441
ULTs	34	37