Programação Concorrente

Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Threads

Objetivo da aula

- Apresentar o conceito de *threads*
- Observar as diferenças entre *threads* e processos

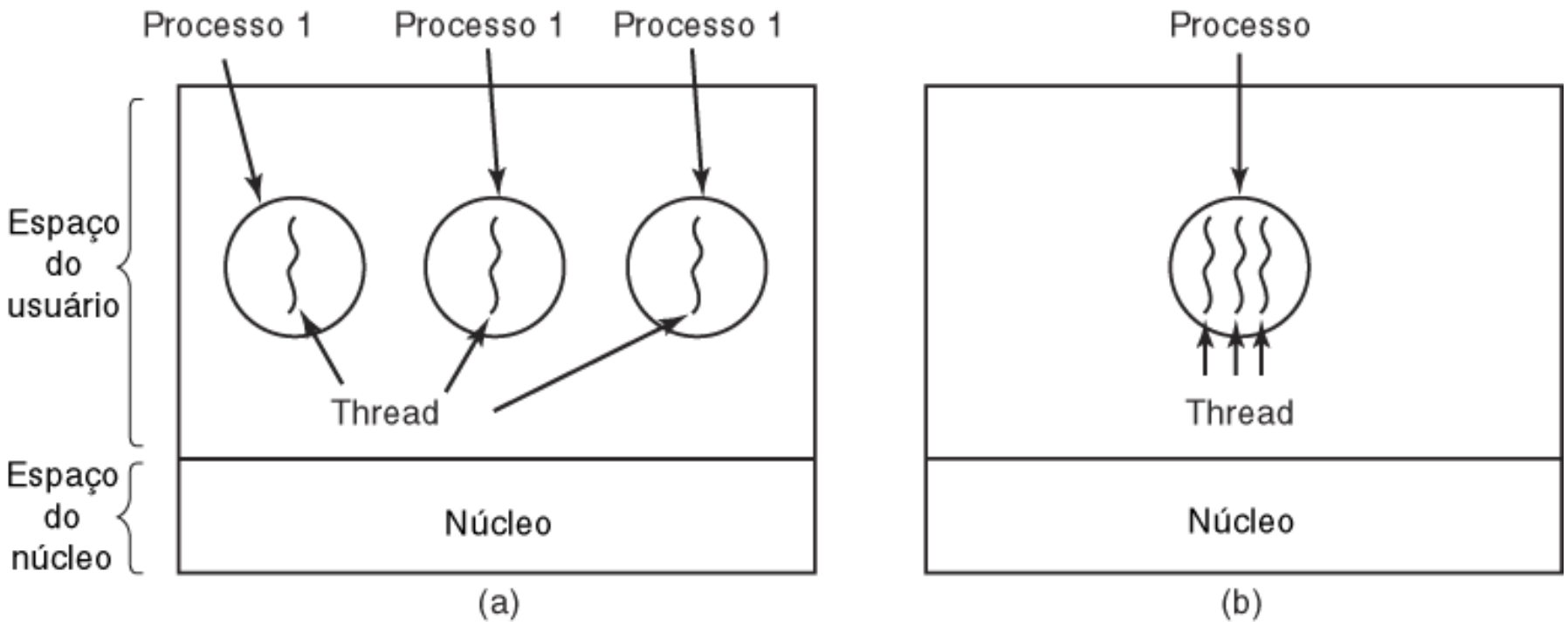
Threads

- São tarefas de um processo executadas concorrentemente
  - Ao ser criado, um processo possui apenas uma *thread*
  - Novas *threads* podem ser criadas no código do programa

- Em sistemas com múltiplos núcleos de processamento, um processo pode ter várias threads em execução simultânea (paralelismo real)

- Threads compartilhar recursos do mesmo processo, sem necessidade de duplicatas dos dados

Threads



(a) Três processos cada um com um *thread*
(b) Um processo com três *threads*

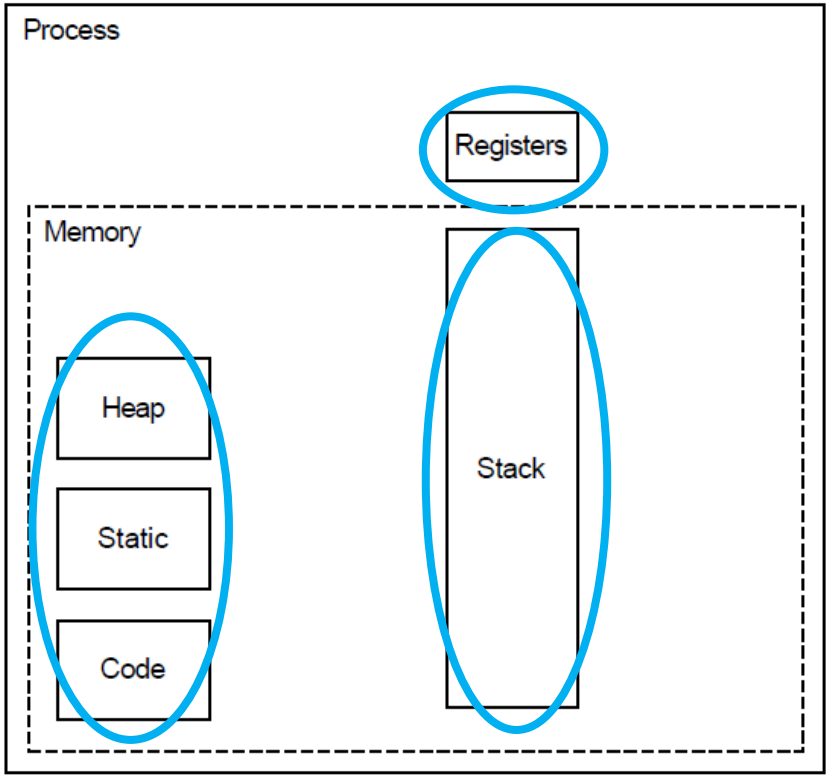
Threads

- Cada *thread* possui
  - Contador de programa (PC – *Program Counter*)
  - Registradores
  - Pilha de execução (*stack*)
  - Estado

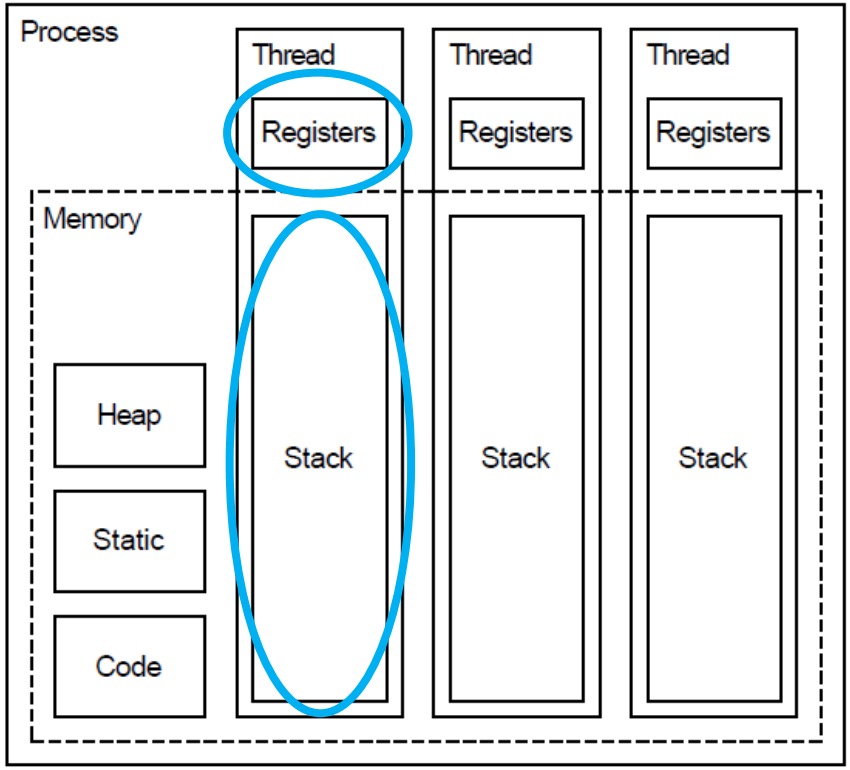
- Threads de um mesmo processo compartilham:
  - Espaço de endereçamento
  - Variáveis globais
  - Arquivos abertos, sinais
  - Informações de contabilidade do processo

Threads

Processo com uma única thread



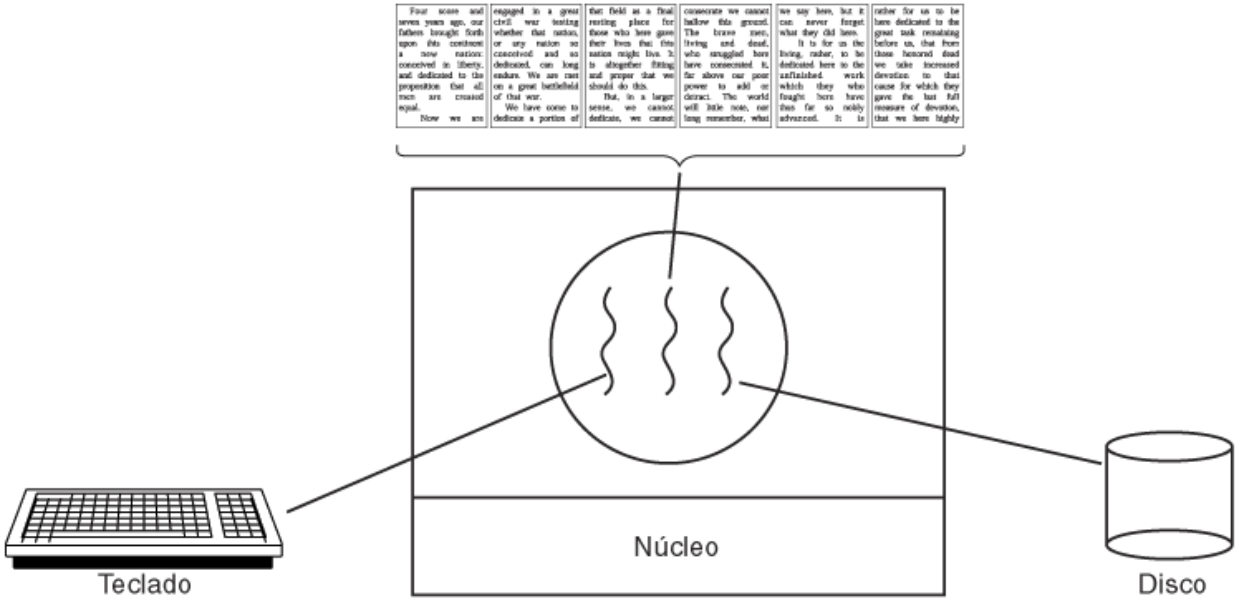
Processo com múltiplas threads



Threads: Exemplo de uso

- Várias funcionalidades paralelas em um mesmo programa

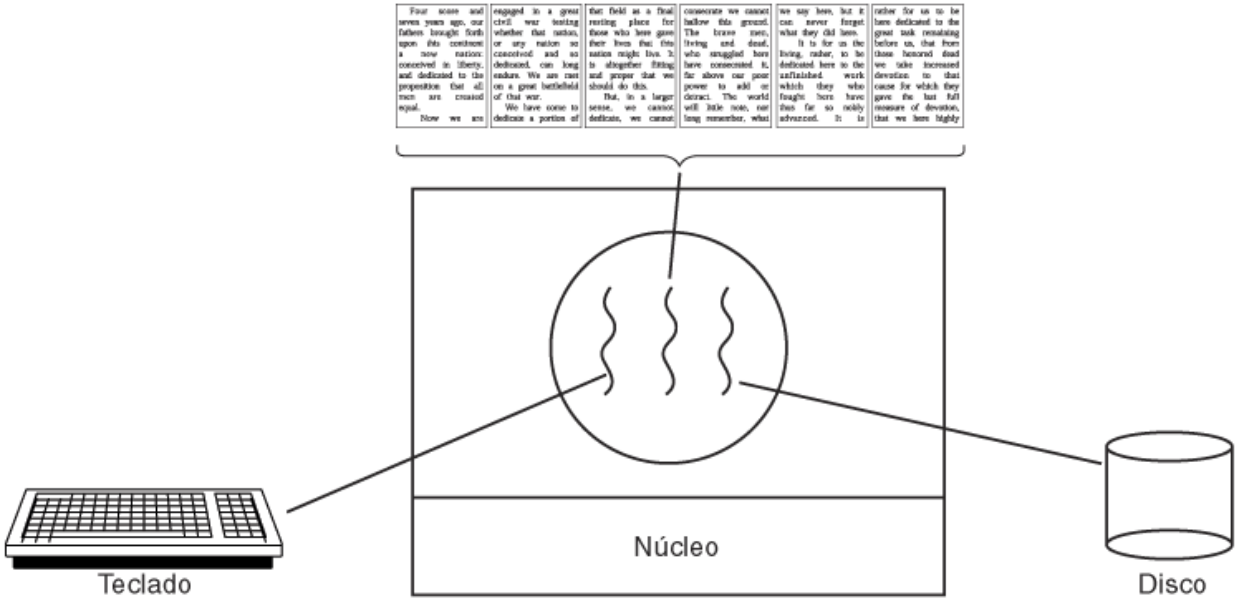
Editor de texto



Threads: Exemplo de uso

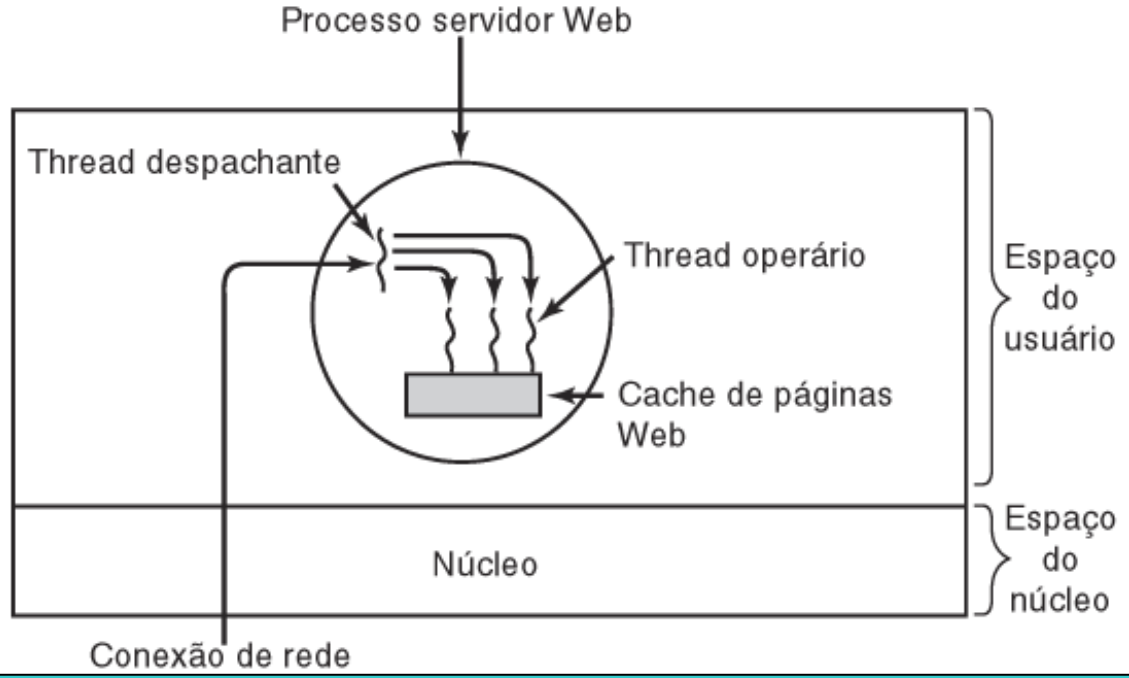
- Várias funcionalidades paralelas em um mesmo programa

Editor de texto



- Atendimento de requisições em paralelo (criação de *worker threads* conforme a demanda)

Servidor Web



Threads – Troca de contexto

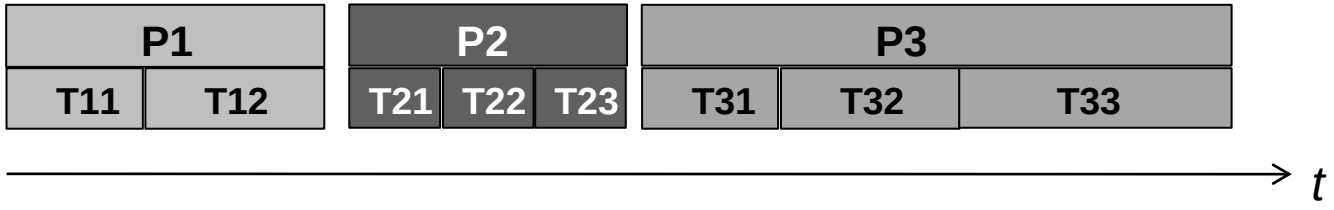
- Quando duas *threads* de um mesmo processo alternam o uso do processador, ocorre uma troca de contexto parcial:
  - O contador de programa, registradores e a pilha devem ser salvos
  - Uma troca de contexto parcial é mais rápida que uma troca de contexto entre processos

- Uma troca de contexto completa é necessária quando uma *thread* de um processo que não estava em execução ganha o processador

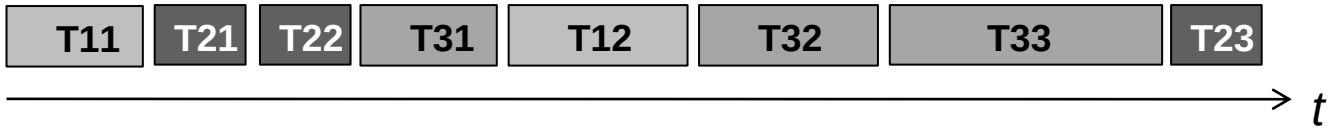
Itens por processo	Itens por thread
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	Estado
Alarmes pendentes	
Sinais e tratadores de sinais	
Informação de contabilidade	

Threads – Escalonamento

- Escalonamento por processo: escalonador aloca tempo para execução dos processos, que definem como usar este tempo para executar suas threads

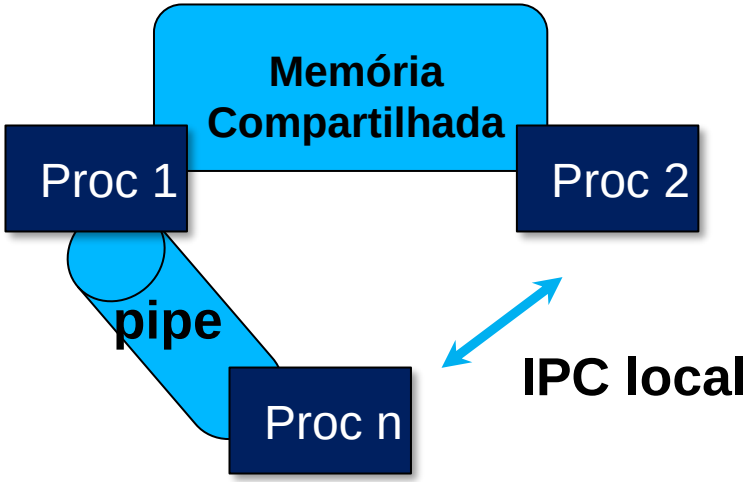


- Escalonamento por *thread*: escalonador define a ordem na qual as threads serão executadas



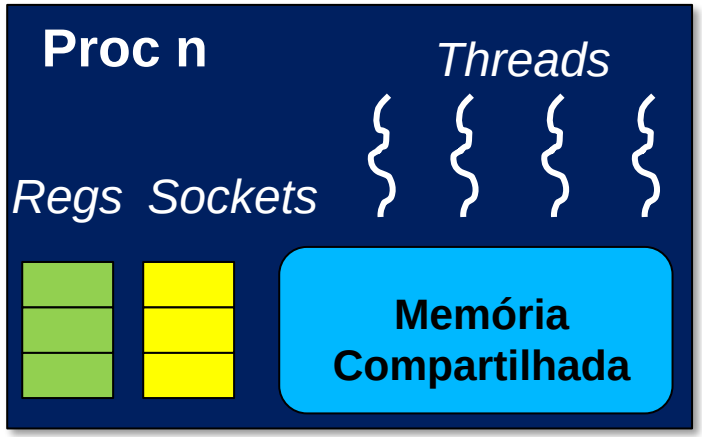
Processos versus *Threads*: Comunicação

Comunicação Multiprocessos



- Cada processo é uma unidade lógica independente das demais:
- Código independente, acesso à memória local, tratadores de E/S, tratadores de sinais, etc.
- Segurança ao acesso de recursos e gerenciamento de memória dado pela MMU (Memory Management Unit)
- Comunicação entre processos é mais custosa (desempenho) e normalmente mais complexa

Comunicação Multi Threads



-*Threads* são unidades de computação executando no contexto de um processo
- O mesmo espaço de endereçamento é compartilhado
- Os mesmos mecanismos de segurança e gerenciamento oferecidos pelo processo são compartilhados
- Comunicação entre processos é mais rápida (desempenho) e normalmente mais simples

Suporte a *Threads*

- *Threads* nativas do sistema
  - São criadas a partir de chamadas ao sistema
    - Ex. Linux (a partir do *kernel* 2.6), Windows (a partir do Windows 95)
- Bibliotecas e APIs
  - *Threads* são fornecidas por bibliotecas ou APIs externas
    - Ex. POSIX *threads*
- Linguagem de programação *multithreaded*
  - A linguagem fornece nativamente mecanismos para criação de threads
    - Ex. Java, Go, etc.

Resumo

	Processos	Threads
Troca de Contexto	Completa	Parcial
Área de Memória	Independente	Compartilhada
Comunicação	Inter-processo	Intra-processo
Código	Independente	Mesmo Código
Suporte em SO	Quase todos	Os mais modernos
Suporte em Ling. Programação	Quase todas	As mais recentes

Referências

Parte destes slides são baseadas em material de aula dos livros:

- *OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva.; TOSCANI, Simão Sirineo. Sistemas operacionais. 4. ed. Porto Alegre: Bookman, 2010. xii, 374p. (Livros didáticos, n.11) ISBN 9788577805211*

- *SILBERSCHATZ, Abraham.; GAGME, Greg; GALVIN, Peter B. Sistemas operacionais com Java. Rio de Janeiro: Elsevier, 2008. 673 p. ISBN 9788535224061*

- *TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. Rio de Janeiro (RJ): Prentice-Hall do Brasil, 2010. xiii, 653p. ISBN 9788576052371*