

Sistemas Operacionais

Threads

Lesandro Ponciano

2025

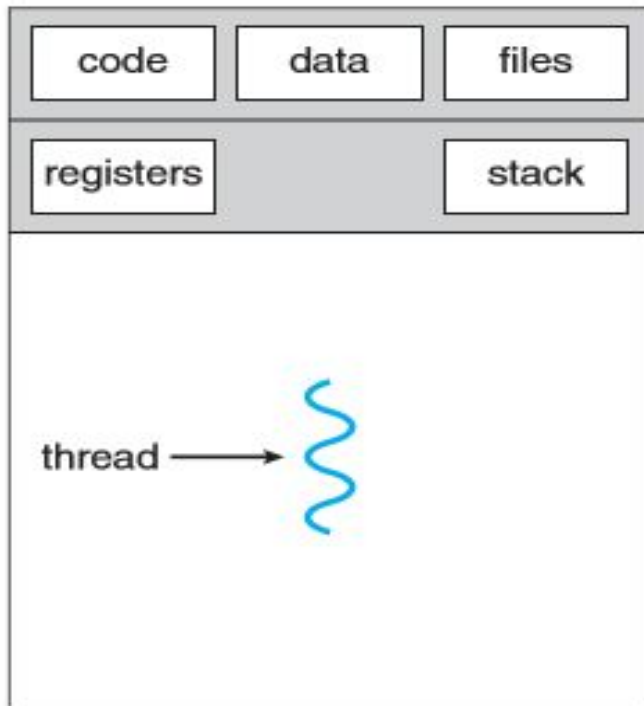
Objetivos da Aula

- **Apresentar** os principais conceitos relacionados a threads
 - Estrutura, Tipos, Relações, Criação e Encerramento
- **Analisar** detalhes da implementação de threads
- **Analisar** detalhes da execução de threads

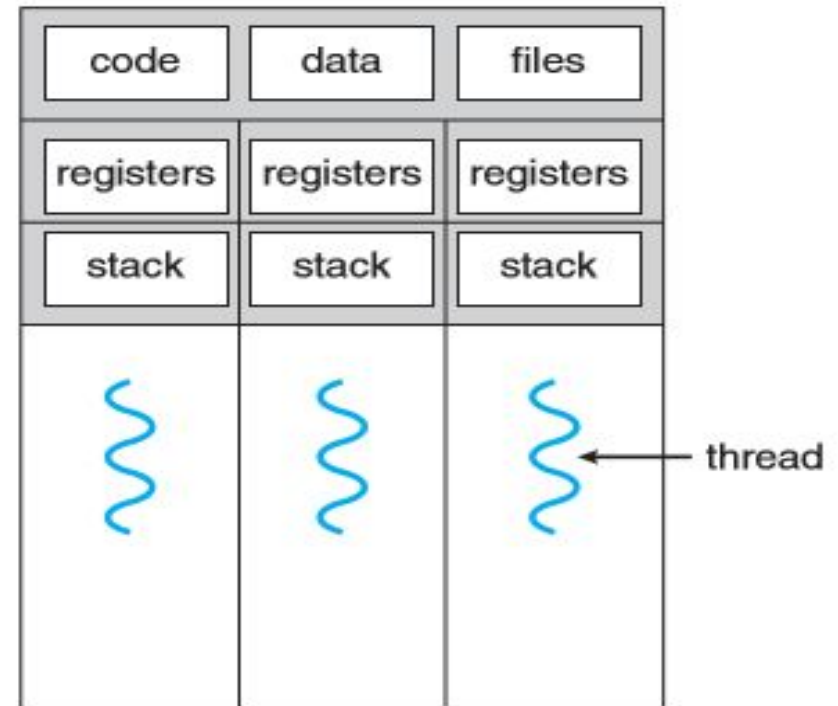
Threads

- Um processo possui um ou mais fluxos (threads) de execução
 - Todos os threads de um processo compartilham o espaço de endereçamento do processo
- Threads são processos leves
 - Unidades básicas de utilização de CPU
 - Possuem contador de programa (PC), registradores e pilha
- Threads de um mesmo processo compartilham
 - dados, como variáveis globais
 - recursos do sistema, como arquivos abertos e sinais

Representação de Threads



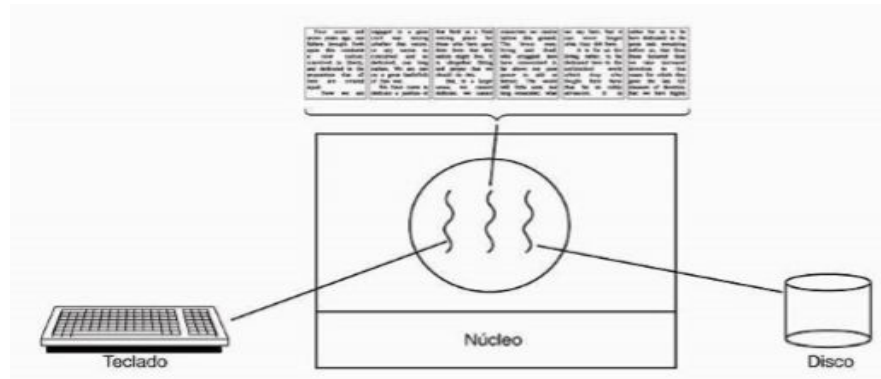
single-threaded process



multithreaded process

Exemplo

- Editor de texto em execução (processo) com múltiplos fluxos (multithread), cada fluxo (thread) é responsável por uma atividade
 - interagir com o teclado
 - realizar backups esporádicos
 - formatar o texto à medida que mudanças são efetuadas pelo usuário



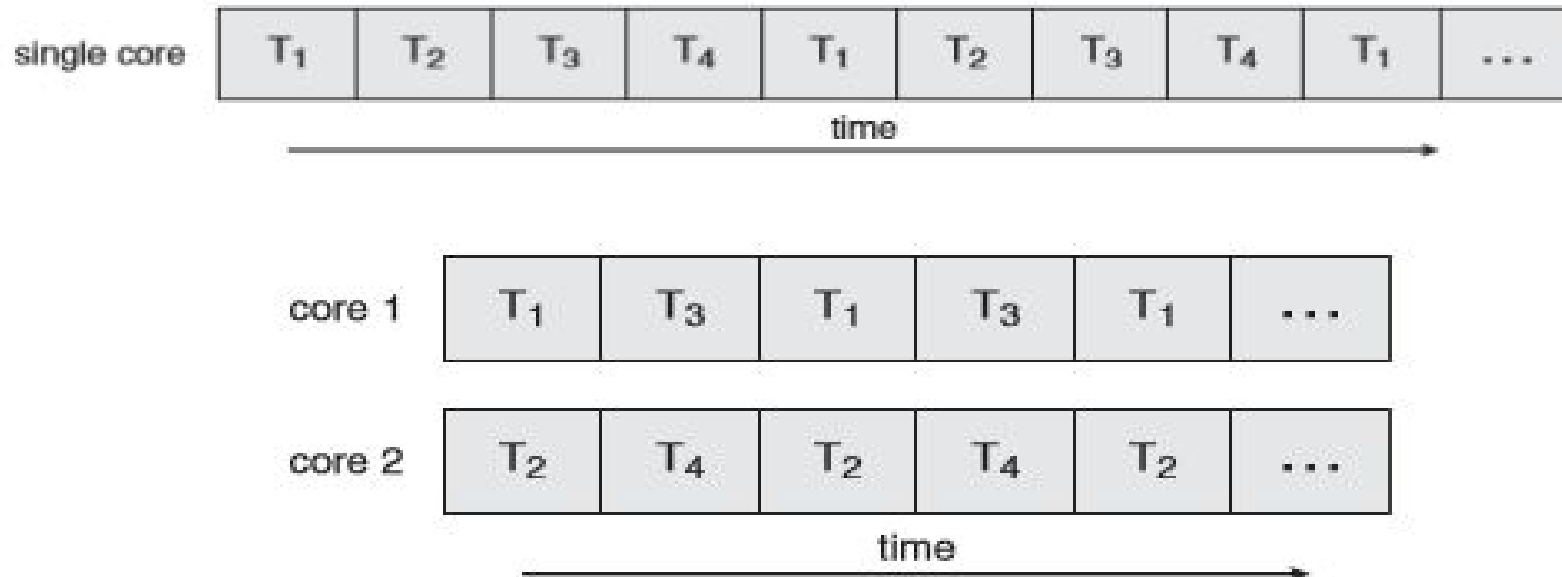
Benefícios de *Multithread*

- Quatro categorias principais:
 - **Capacidade de resposta:** Mesmo que uma parte do processo esteja bloqueada ou em uma execução demorada, outra parte do processo é capaz de continuar em operação
 - **Compartilhamento de recursos:** Por definição, threads compartilham a memória e os recursos dos processos aos quais pertencem
 - **Economia:** Demora muito mais para se criar e gerenciar processos do que para se criar e gerenciar threads
 - **Escalabilidade:** O uso de vários threads em uma máquina com vários CPUs aumenta o paralelismo

Programação *Multicore*

- Programação com vários threads fornece um mecanismo para
 - uso mais eficiente de muitos núcleos
 - aumento da concorrência
- Considere um processo com 4 *threads*
 - Em uma máquina com um único núcleo, tem-se uma execução sequencial dos threads em cada momento
 - Em uma máquina com dois núcleos, tem-se dois threads em execução em cada momento

Exemplo



É importante desenvolver sistemas com múltiplos threads para se tirar benefício da existência de diversos núcleos

Dificuldades da Programação *Multithread*

■ Divisão de atividades

- Identificar partes que podem ser threads diferentes

■ Equilíbrio

- Detectar qual processo vai se beneficiar da implementação de determinada atividade em um thread

■ Divisão de dados

- Determinar as dependências de dados entre dois threads e a sincronização no uso desses dados

■ Teste e depuração

- Múltiplos caminhos de execução
- Testar esses caminhos é uma tarefa complexa

Threads do Usuário e do Kernel

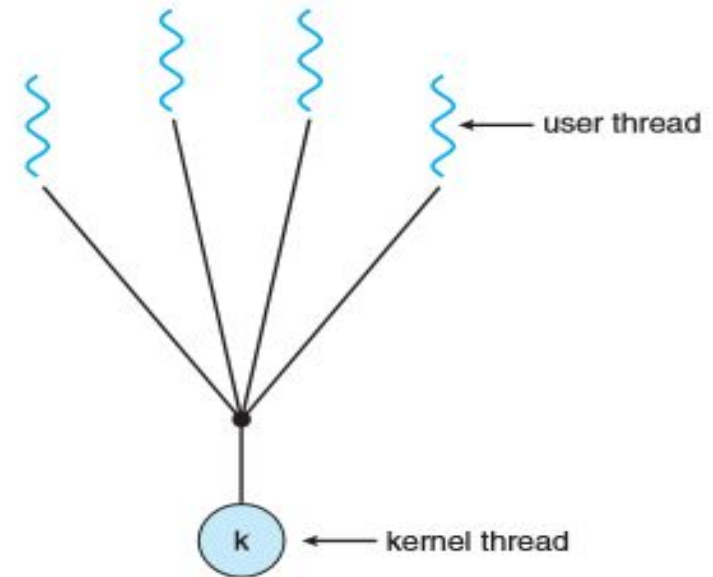
- Threads do usuário
 - São suportados acima do núcleo e gerenciados sem o suporte do kernel
 - São definidos por uma biblioteca
 - Podem ser implementados em sistemas sem suporte à multithread
- Threads do kernel
 - São suportados e gerenciados pelo kernel
 - O kernel gerencia threads e não apenas processos

Modelos de Relacionamento

- Quando há threads do usuário e threads do sistema é preciso que exista um relacionamento entre eles
- Esse relacionamento pode ser
 - Muitos-para-Um
 - Um-para-Um
 - Muitos-para-Muitos

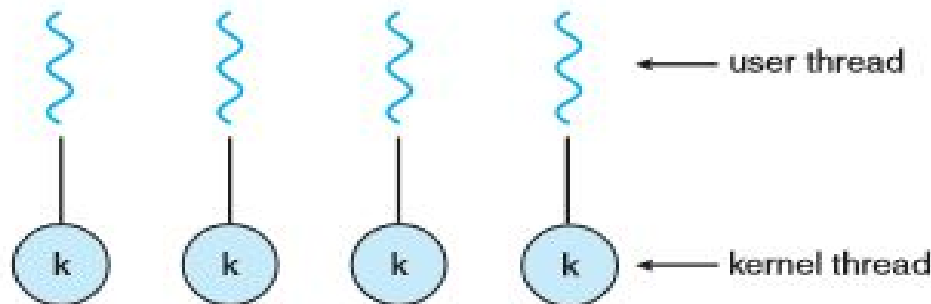
Modelo Muitos-para-Um

- Mapeia muitos threads no nível do usuário para um thread no nível do kernel
 - O processo inteiro é bloqueado se um de seus threads fizer uma chamada de sistema bloqueadora



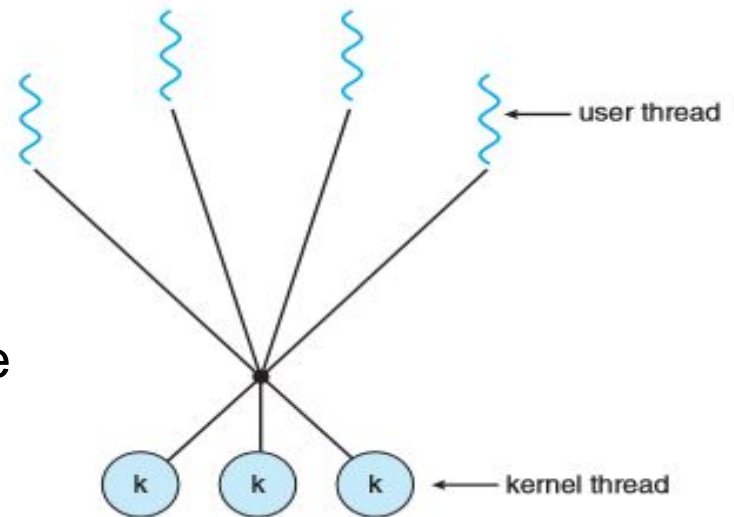
Modelo Um-para-Um

- Cada thread do usuário é mapeado como um thread do kernel
 - **Vantagem:** demais threads de um processo continuam em execução, mesmo se um thread fizer uma chamada de sistema bloqueadora
 - **Desvantagem:** pode existir um *overhead* de se criar threads adicionais no nível do kernel e muitos threads podem sobrecarregar o sistema



Modelo Muitos-para-Muitos

- Multiplexa muitos threads do nível do usuário em uma quantidade menor ou igual de threads do kernel
 - Usuário criar quantos threads quiser que o Kernel controla a quantidade de threads que é criada no nível kernel
 - Há concorrência e paralelismo, se um thread fizer uma chamada bloqueadora, os demais continuarão executando



Criação, Conclusão e Cancelamento


- A **criação** (iniciação) de um thread é a definição de um fluxo de execução
 - realizado por meio de uma chamada de sistema
- Um thread **termina** naturalmente quando sua execução termina
- **Cancelar** um threads é encerrar um thread (thread-alvo) antes que ele seja concluído
 - Assíncrono: Um thread encerra o thread alvo
 - Adiado: O thread alvo verifica periodicamente se deve ser encerrado e encerra a si próprio quando indicado

Pools de Threads

- Em sistemas dinâmicos, criar uma quantidade ilimitada de threads pode exaurir o desempenho do sistema
- Geralmente define-se um pool de threads
 - Threads são definidos na inicialização do sistema e ficam armazenados em um pool
 - Quando chega uma requisição, um thread no pool é iniciado
 - Quando um thread termina a execução, ele retorna para o pool e aguarda novas requisições
 - Se surgir uma requisição e o pool estiver vazio, a requisição deve aguardar até que um thread possa atendê-la

Criação de Threads em C#

```
using System;
using System.Threading;

class ThreadIdentidade
{
    private int id = 0;
    public ThreadIdentidade(int id)
    {
        this.id = id;
    }
    public void ApresentarThread()
    {
         Console.WriteLine("Olá mundo, eu sou o thread " + id);
    }
}
```

Continua ...

Continuação ...

```
class Program
{
    static void Main()
    {
        int numThreads = 20;
        Thread[] pool= new Thread[numThreads];
        for (int i = 0; i < numThreads; i++)
        {
            ThreadIdentidade b = new ThreadIdentidade(i);
            pool[i] = new Thread(b.ApresentarThread);
        }

        for (int i = 0; i < numThreads; i++)
            pool[i].Start();

        Console.ReadKey();
    }
}
```

Resultados de Três Execuções


```
Olá mundo, eu sou o thread 0
Olá mundo, eu sou o thread 1
Olá mundo, eu sou o thread 2
Olá mundo, eu sou o thread 3
Olá mundo, eu sou o thread 4
Olá mundo, eu sou o thread 5
Olá mundo, eu sou o thread 6
Olá mundo, eu sou o thread 7
Olá mundo, eu sou o thread 8
Olá mundo, eu sou o thread 10
Olá mundo, eu sou o thread 9
Olá mundo, eu sou o thread 11
Olá mundo, eu sou o thread 12
Olá mundo, eu sou o thread 13
Olá mundo, eu sou o thread 14
Olá mundo, eu sou o thread 16
Olá mundo, eu sou o thread 15
Olá mundo, eu sou o thread 17
Olá mundo, eu sou o thread 18
Olá mundo, eu sou o thread 19
```

```
Olá mundo, eu sou o thread 0
Olá mundo, eu sou o thread 1
Olá mundo, eu sou o thread 3
Olá mundo, eu sou o thread 2
Olá mundo, eu sou o thread 4
Olá mundo, eu sou o thread 5
Olá mundo, eu sou o thread 6
Olá mundo, eu sou o thread 8
Olá mundo, eu sou o thread 7
Olá mundo, eu sou o thread 9
Olá mundo, eu sou o thread 12
Olá mundo, eu sou o thread 11
Olá mundo, eu sou o thread 10
Olá mundo, eu sou o thread 14
Olá mundo, eu sou o thread 13
Olá mundo, eu sou o thread 15
Olá mundo, eu sou o thread 16
Olá mundo, eu sou o thread 18
Olá mundo, eu sou o thread 17
Olá mundo, eu sou o thread 19
```

```
Olá mundo, eu sou o thread 0
Olá mundo, eu sou o thread 1
Olá mundo, eu sou o thread 2
Olá mundo, eu sou o thread 5
Olá mundo, eu sou o thread 3
Olá mundo, eu sou o thread 4
Olá mundo, eu sou o thread 6
Olá mundo, eu sou o thread 8
Olá mundo, eu sou o thread 7
Olá mundo, eu sou o thread 11
Olá mundo, eu sou o thread 9
Olá mundo, eu sou o thread 12
Olá mundo, eu sou o thread 10
Olá mundo, eu sou o thread 13
Olá mundo, eu sou o thread 14
Olá mundo, eu sou o thread 17
Olá mundo, eu sou o thread 15
Olá mundo, eu sou o thread 16
Olá mundo, eu sou o thread 18
Olá mundo, eu sou o thread 19
```

Encerramento de Threads

```
using System;
using System.Threading;

class ThreadIdentidade
{
    private int id = 0;
    public ThreadIdentidade(int id)
    {
        this.id = id;
    }
    public void ApresentarThread()
    {
        while(true)
         Console.WriteLine("Olá mundo, eu sou o thread " + id);
    }
}
```

Continuação ...

```
class Program
{
    static void Main()
    {
        ThreadIdentidade ti1 = new ThreadIdentidade(1);
        Thread t1 = new Thread(ti1.ApresentarThread);
        t1.Start();

        ThreadIdentidade ti2 = new ThreadIdentidade(2);
        Thread t2 = new Thread(ti2.ApresentarThread);
        t2.Start();

        t1.Abort();
        t2.Abort();
        Console.WriteLine("Abortados");
        Console.ReadKey();
    }
}
```

esandro Ponciano

Resultados em Duas Execuções

[illegible][illegible]

Mostra apenas as últimas linhas

Atividade de Fixação

1. O que são threads?
2. Qual a relação entre threads e processos?
3. Qual a relação entre threads dos usuários e threads do kernel? Quais os três principais modelos dessa relação?
4. Apresente e discuta dois benefícios e duas dificuldades de se implementar um programa multithread.
5. Por que a vantagem de se implementar um programa multithread é maior em sistemas com múltiplos cores?

Atividade de Fixação 2

Sobre threads e programas multithread é incorreto afirmar

- A) Em programas multithread, as threads são definidas pelo programador.
- B) Se o sistema operacional não implementa multithread, então várias threads no nível do usuário se tornarão apenas uma thread no nível do kernel.
- C) Threads compartilham o espaço de endereçamento do processo que as criou.
- D) Pool de threads é um mecanismo de controle para evitar que uma thread interfira na outra.

POSCOMP 2010

35) Embora ambos tenham seu escalonamento feito pelo gerenciamento de processos, *threads* e processos são estruturalmente distintos.

Qual é a principal diferença entre eles?

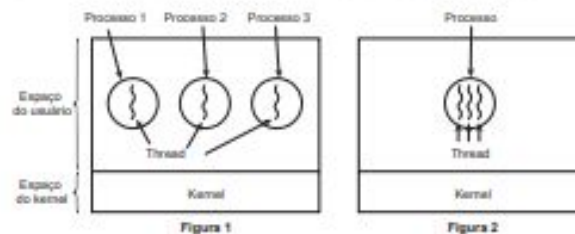
- a) Apenas *threads* podem ser executados em paralelo.
- b) *Threads* possuem contexto simplificado.
- c) Processos executam mais rapidamente.
- d) Processos apenas podem ocorrer em sistemas de grande porte.
- e) *Threads* apenas podem ocorrer em processadores multicore.

ENADE 2014

Computação

QUESTÃO 20

Um processo tem um ou mais fluxos de execução, normalmente denominados apenas por *threads*.



TANENBAUM, A. D. *Sistemas operacionais modernos*. 3. ed. São Paulo: Pearson Prentice Hall, 2010 (adaptado).

A partir das figuras 1 e 2 apresentadas, avalie as afirmações a seguir.

- I. Tanto na figura 1 quanto na figura 2, existem três *threads* que utilizam o mesmo espaço de endereçamento.
- II. Tanto na figura 1 quanto na figura 2, existem três *threads* que utilizam três espaços de endereçamento distintos.
- III. Na figura 2, existe um processo com um único espaço de endereçamento e três *threads* de controle.
- IV. Na figura 1, existem três processos tradicionais, cada qual tem seu espaço de endereçamento e uma única *thread* de controle.
- V. As *threads* permitem que várias execuções ocorram no mesmo ambiente de processo de forma independente uma das outras.

É correto apenas o que se afirma em

- A** I, II e III.
- B** I, II e IV.
- C** I, III e V.
- D** II, IV e V.
- E** III, IV e V.

Referências

TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson Prentice Hall, 2009. xvi, 653 p. ISBN 9788576052371

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais: princípios básicos. Rio de Janeiro, RJ: LTC, 2013. xvi, 432 p. ISBN 9788521622055

Sistemas Operacionais

Prof. Dr. Lesandro Ponciano

<https://orcid.org/0000-0002-5724-0094>