SISTEMAS OPERACIONAIS I

Prof. Renato Jensen

THREAD

- **O que significa essa palavra “thread”?**

  - No que se refere a hardware, essa palavra apareceu no Brasil quando surgiram os primeiros modelos de processador com múltiplos núcleos. A princípio era fácil compreender que um dual-core tinha dois núcleos.

  - Entretanto, com a evolução das arquiteturas nas CPUs, surgiu o suporte para múltiplos threads (multithreading). E é aí que muitas pessoas se perguntaram o que realmente mudava.

  - Ao efetuar o carregamento de um programa, o sistema operacional trabalha com processos. Estes “processos” são módulos executáveis, que contêm linhas de código para que a execução do programa seja realizada apropriadamente. Isso quer dizer que o processo é uma lista de instruções a serem executadas pelo processador.

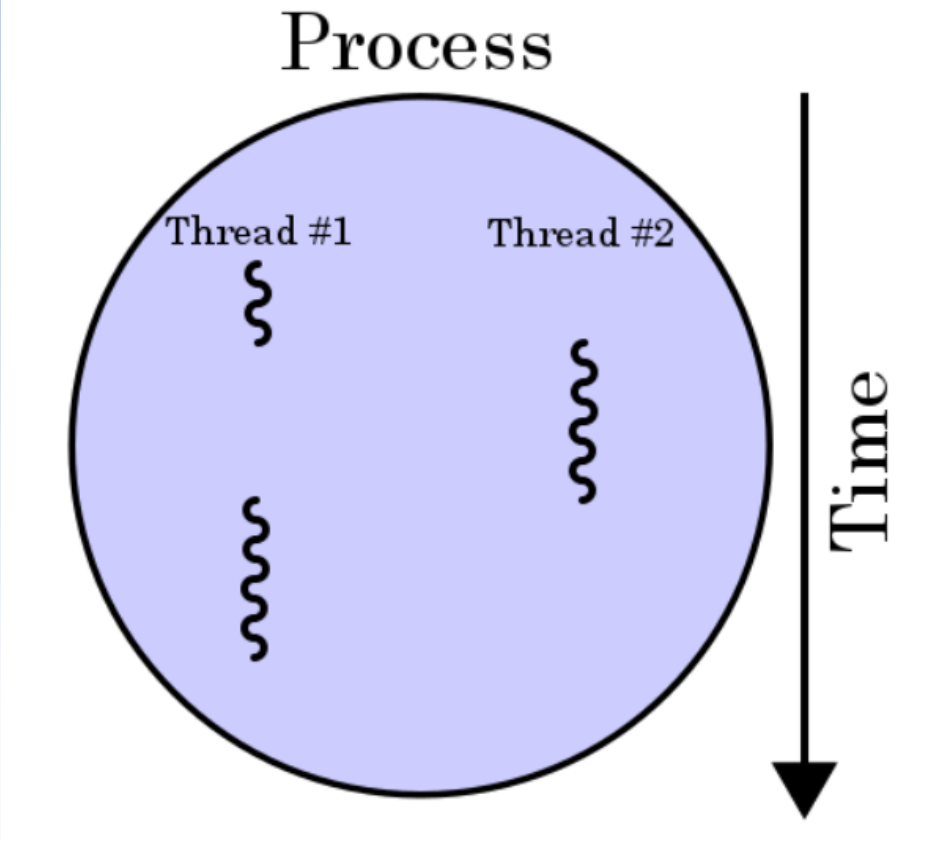
THREAD

- **O que significa essa palavra “thread”?**

  - Em sistemas operacionais tradicionais, cada processo tem um espaço de endereçamento próprio e um único fluxo de controle. Porém, existem situações em que é desejável ter múltiplos fluxos de controle (threads) **no mesmo espaço de endereçamento** executando em paralelo, como se fossem processos separados.

  - Thread significa **linha ou encadeamento de execução**. É uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

THREAD



Threads como linhas de execução

THREAD

- Um exemplo simples seria um jogo, que pode ser modelado com duas linhas de execução diferentes:
  - Uma para desenho de imagem
  - Outra para áudio.
- No ponto de vista do usuário, a imagem é desenhada ao mesmo tempo em que o áudio é emitido pelos alto-falantes.

- Vamos tomar como exemplo o processador Intel Core i7 8809G. Verificando no site da fabricante, temos a informação de que esse modelo vem com quatro núcleos e tem suporte para trabalhar com até oito threads.
- Isso quer dizer que essa CPU pode trabalhar com quatro processos indivisíveis simultaneamente (um em cada núcleo) ou com até oito threads — as quais podem ou não ser de um mesmo processo.

THREAD

- Em *hardwares* equipados com uma única CPU, cada thread é processada de forma aparentemente simultânea, pois a mudança entre uma thread e outra é feita de forma tão rápida que para o utilizador, isso está acontecendo paralelamente.

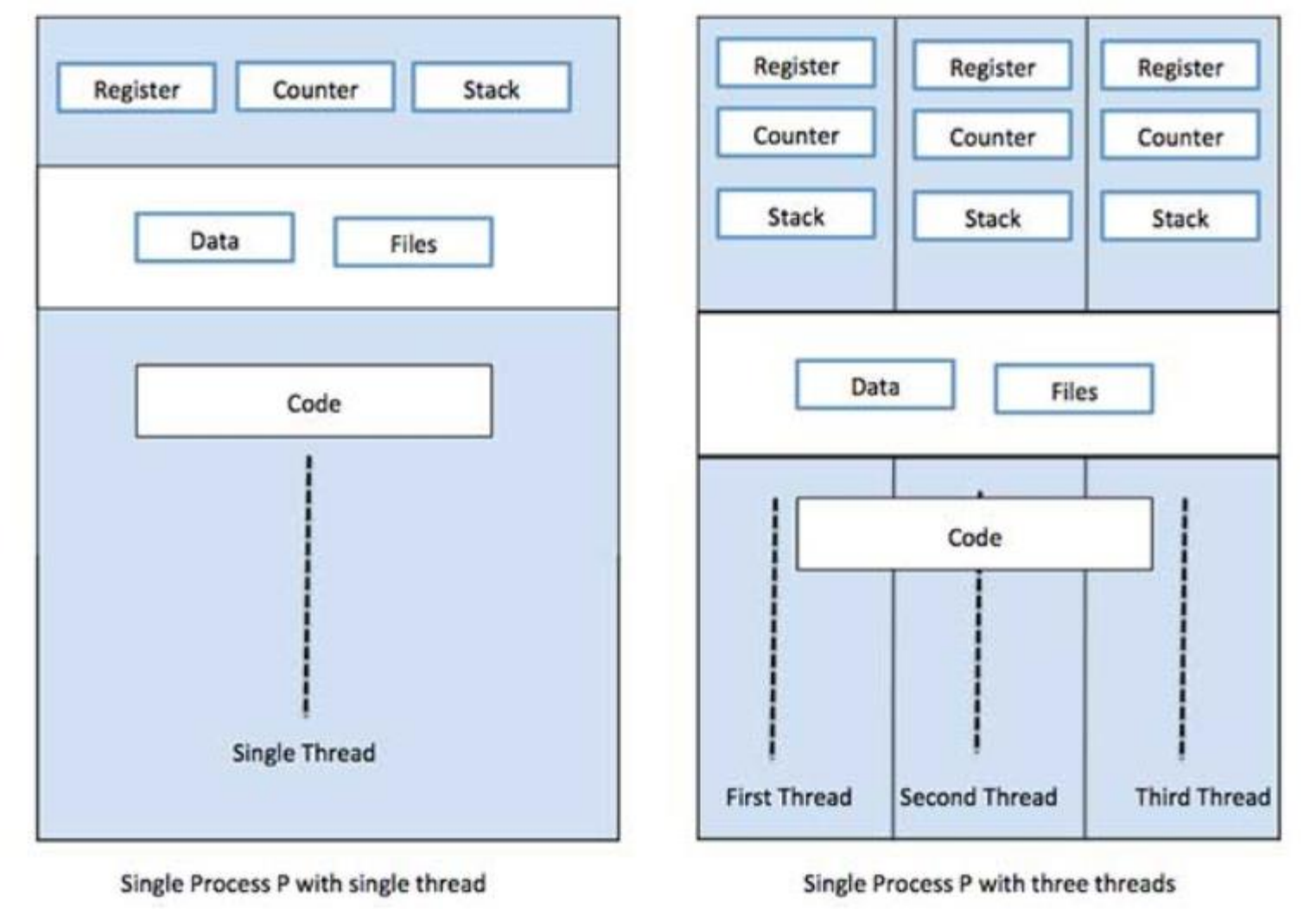
- Em *hardwares* com múltiplos CPUs ou multicores, as threads são realizadas realmente de forma simultânea.

- Os sistemas que suportam uma única thread (em real execução) são chamados de ***monothread*** enquanto que os sistemas que suportam múltiplas threads são chamados de ***multithread***.

THREAD

- Cada thread tem o mesmo contexto de software compartilha o mesmo espaço de memória.
- Dentro de um processo, todas as threads possuem informações em comum, mas cada thread tem informações pertencentes somente a ela. Por exemplo:
  - **Itens compartilhados entre as threads:**
    - Espaço de endereçamento
    - Variáveis globais
    - Arquivos abertos
    - Processos filhos
    - Informação de contabilidade
  - **Itens privados por thread:**
    - Contador de programa
    - Registradores
    - Pilha
    - Estado

THREAD


Register
Counter
Stack
Data
Files
Code
Single Thread
Single Process P with single thread
Register
Counter
Stack
Register
Counter
Stack
Register
Counter
Stack
Data
Files
Code
First Thread
Second Thread
Third Thread
Single Process P with three threads

THREAD

- **ULT – User-Level Thread**

  - São suportadas pela aplicação, sem conhecimento do núcleo e geralmente são implementadas por pacotes de rotinas fornecidas por uma determinada biblioteca de uma linguagem, como é o caso da **thread.h** (biblioteca padrão da linguagem C).

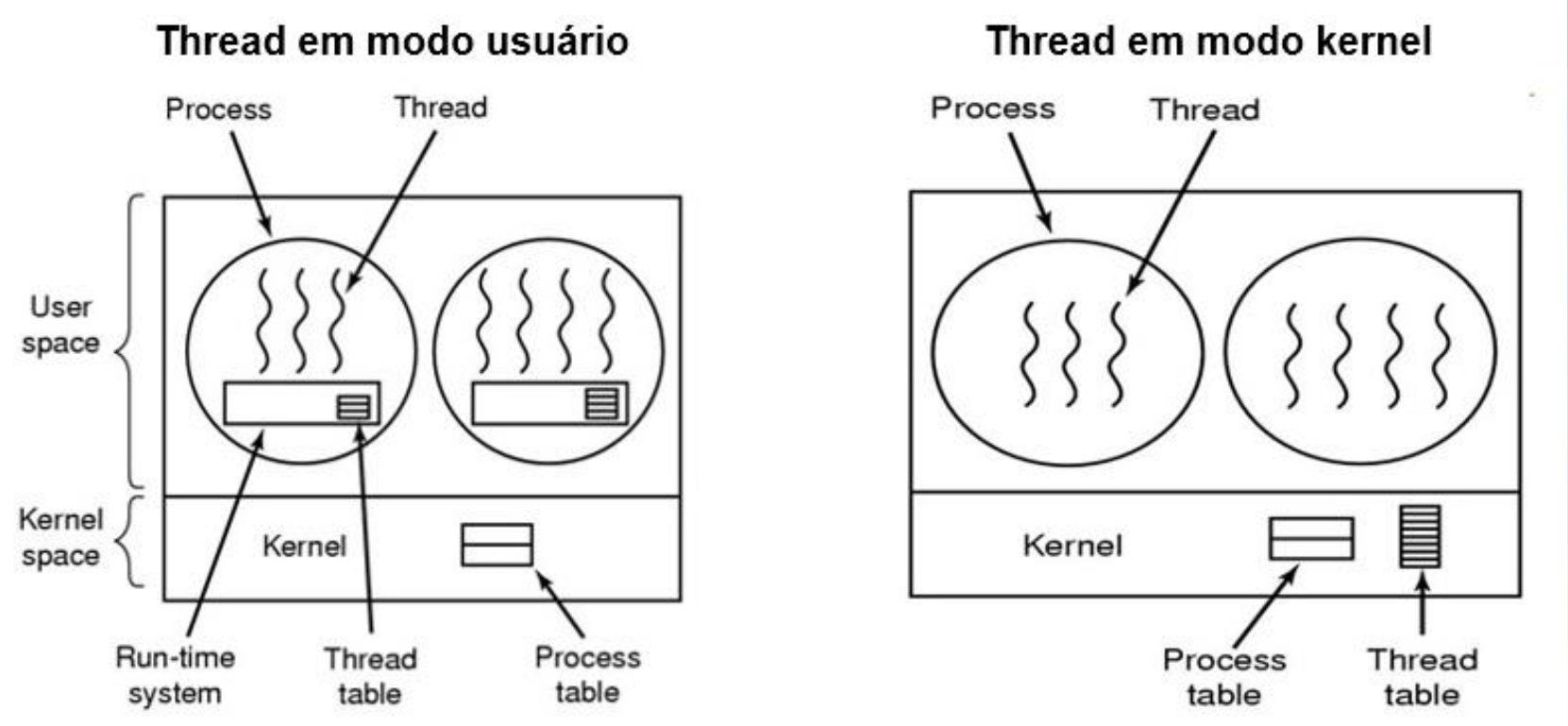
  - Possuem como vantagens a possibilidade de implementação em sistemas operacionais que não suportam nativamente este recurso, sendo geralmente mais rápidas e eficientes pois dispensam o acesso ao núcleo.

  - Evita assim mudança no modo de acesso, e a estrutura de dados fica no espaço do utilizador, levando a uma significativa queda de overhead, além de poder escolher entre as diversas formas de escalonamento em que melhor se adequa.

THREAD

- **KLT – Kernel-Level Thread**
  - A gestão da thread não é realizada através do código do próprio programa; todo o processo é subsidiado pelo SO.
  - Esse modelo tem a vantagem de permitir o suporte a multiprocessamento e o fato do bloqueio de uma linha de execução não acarretar bloqueio de todo processo.
  - Não obstante, temos a desvantagem de ter que mudar o tipo de acesso sempre que o escalonamento for necessário, aumentando assim o tão temido overhead.

THREAD


Thread em modo usuário
Process
Thread
User space
Kernel space
Kernel
Run-time system
Thread table
Process table
Thread em modo kernel
Process
Thread
Kernel
Process table
Thread table

THREAD

- **Escalonamento**

  - As threads **ULT** são escalonadas pelo programador, tendo a grande vantagem de cada processo usar um algoritmo de escalonamento que melhor se adapte a situação. O sistema operacional neste tipo de thread não faz o escalonamento, em geral ele não sabe que elas existem. Neste modo o programador é responsável por criar, executar, escalonar e destruir a thread.

  - As threads **KLT** são escalonadas diretamente pelo sistema operacional, comumente são mais lentas que as threads ULT pois a cada chamada elas necessitam consultar o sistema, exigindo assim a mudança total de contexto do processador, memória e outros níveis necessários para alternar um processo.

THREAD

- Os sistemas operacionais executam de maneiras diferentes os processos e threads.
- No caso do **Windows**, ele trabalha com maior facilidade para gerenciar programas com apenas um processo e diversos threads do que quando gerencia vários processos e poucos threads.
- Isso acontece porque no sistema da Microsoft a demora para criar um processo e alterná-los é muito grande.
- Enquanto isso, o **Linux** e os demais sistemas baseados no **Unix** podem criar novos processos de maneira muito mais rápida.