

# SISTEMAS DISTRIBUÍDOS

## Threads em Java

Prof. Guilherme C. Kurtz



# Processos e Threads

## ▶ Definições:

- “Processo é apenas um **programa em execução** acompanhado dos valores atuais do contador de programa, registradores e variáveis” (Tanenbaum)
- CPU troca de um **processo para o outro** simulando o paralelismo (**troca de contexto**). Cada processo é executado em um determinado momento;
- O contador de programa contém o **endereço de memória da próxima instrução** a ser executada;

# Processos e Threads

## ▶ Definições:

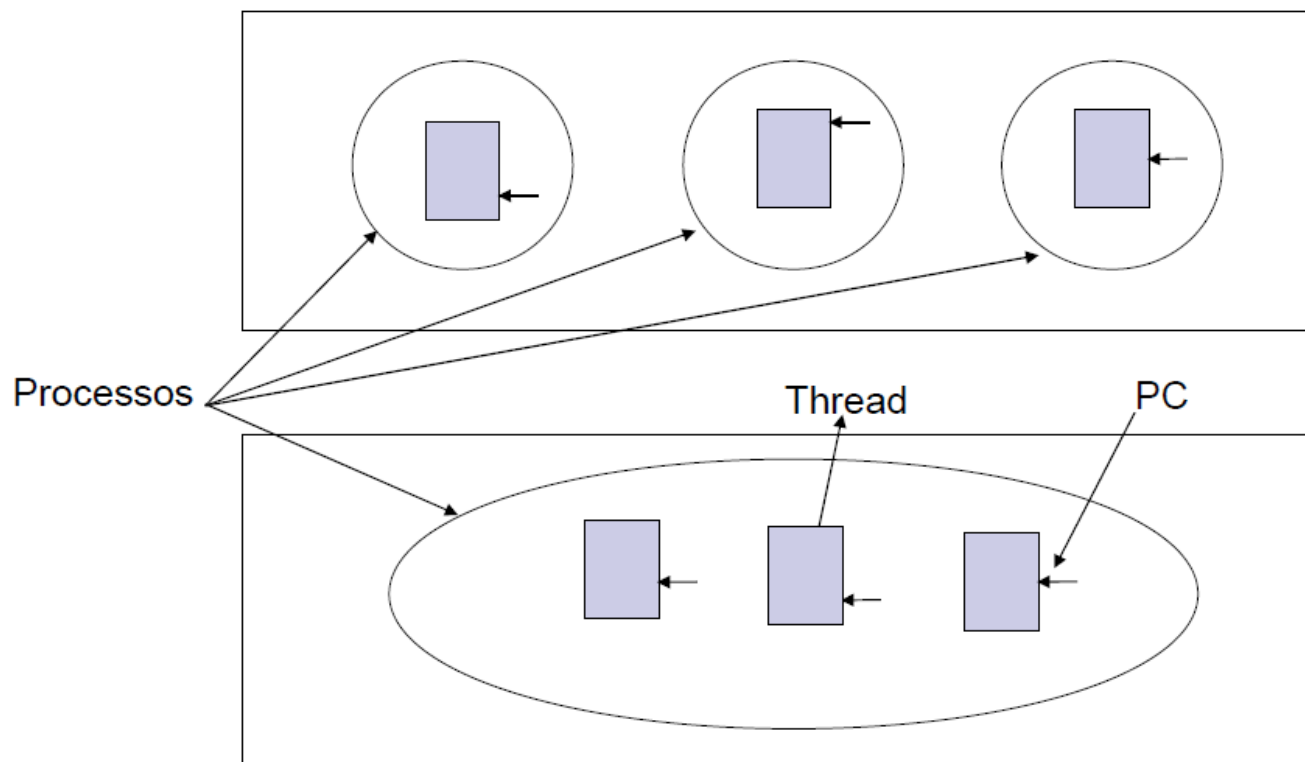
- “Processo consiste em um **ambiente de execução**, junto com uma ou mais threads” (Coulouris)
- **Ambiente de execução**: unidade de **gerenciamento de recursos locais**, sendo gerenciados pelo núcleo (kernel) aos quais as threads tem acesso;
- Os ambientes de execução representam um **domínio** onde as **threads são executadas**;

# Threads

- ▶ Threads podem ser vistas como **mini-processos**, em que cada thread irá executar sua **própria porção de código**;
- ▶ Threads **compartilham a CPU** do mesmo modo que diferentes processos (denominado *timesharing*);
- ▶ Threads que fazem parte de um mesmo processo **não são independentes** como no caso de diferentes processos:
  - Todas as threads de um mesmo processo possuem a mesma **região de memória, compartilhando as mesmas variáveis globais**;
  - Uma thread pode ler, escrever ou mudar os dados de uma outra thread;
  - Portanto, os dados não são protegidos uns dos outros:
    - A proteção deve ser feita pela aplicação!

# Threads

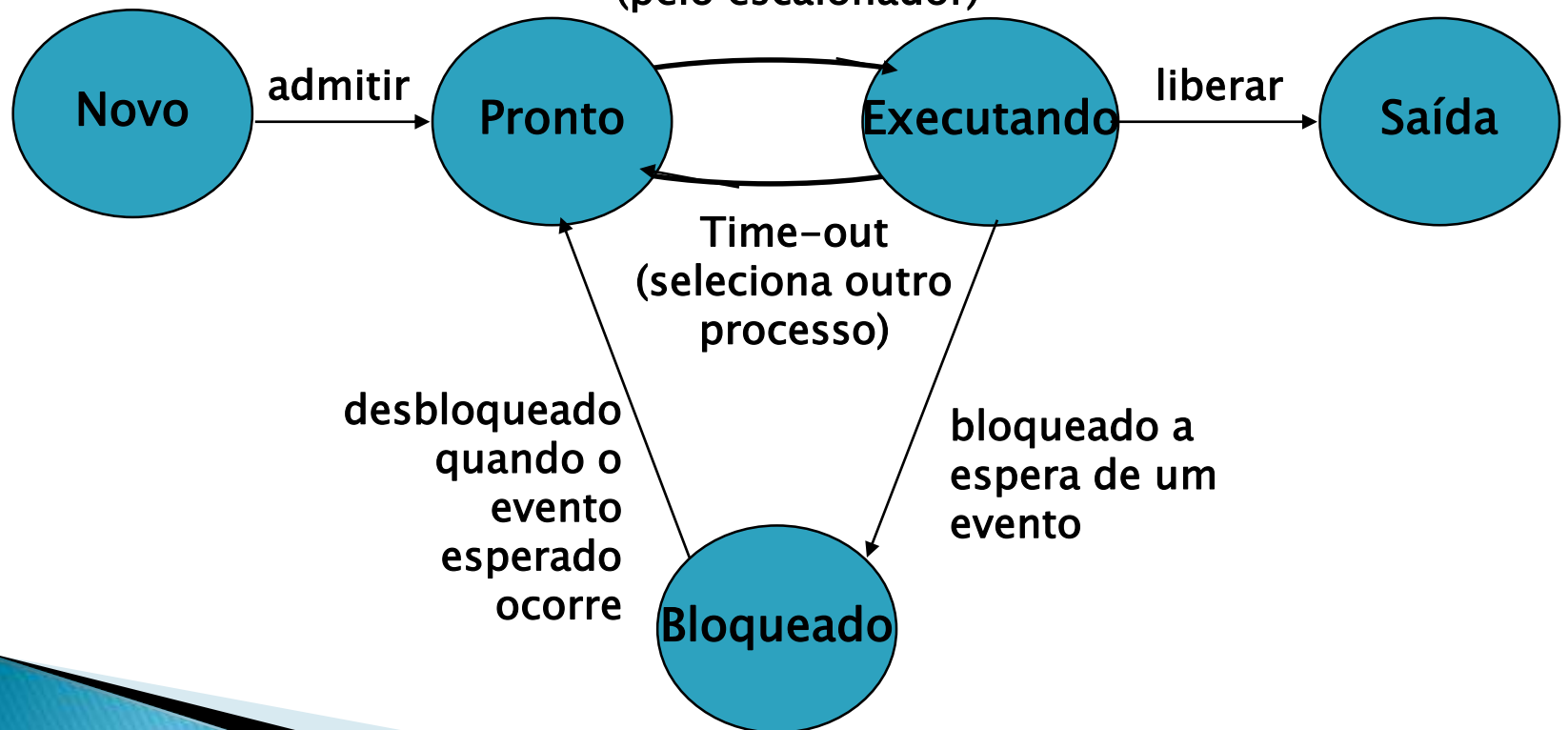
- ▶ Diferença entre threads e processos em relação a memória:



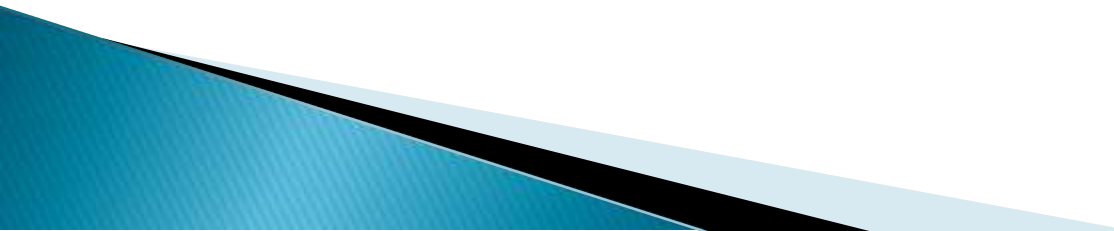
# Threads

- Threads/Processos podem estar em diferentes estados: novo, pronto, executando, bloqueado e finalizado;

CRIAÇÃO DO  
PROCESSO



# Threads em Java

- ▶ Uma thread pode ser implementada:
    - Pelo próprio **sistema operacional**
    - Por uma **biblioteca**;
  - ▶ A linguagem **Java** fornece o suporte a threads ao nível de programação;
  - ▶ O método **main()** é a **principal thread** na execução de uma aplicação Java.
- 

# Threads em Java

- ▶ Uma forma de criar threads em java é criar uma classe derivada (**extends**) da classe **Thread**, e redefinir o método **run()** dessa classe;
- ▶ Um objeto dessa classe executará como uma thread de controle separado na máquina virtual;
- ▶ A **criação do objeto** derivado da classe Java só **especifica** a thread;
- ▶ A **criação e inicialização** da thread fica por conta do método **start()**;
- ▶ O método **start()** aloca memória e inicializa uma nova thread na máquina virtual;



# Considerações

- ▶ Antes da chamada ao método **t.start()**
  - Antes de realizar a chamada a este método, a thread está no estado **novo**;
  - Nós temos o objeto thread, mas ainda não temos uma thread real, pois não foi colocada em execução;
- ▶ Após a chamada ao método **start()**
  - Uma nova thread de **execução** é iniciada;
  - A thread passará do estado **novo** para o estado **executável**;
  - Quando a thread é executada, o método **run()** é executado.

# Considerações importantes

- ▶ O comportamento das threads não é garantido, pois a documentação **java não fala** nada a respeito sobre a **ordem de execução das threads**;
- ▶ Também não há garantias de que, uma vez iniciada, ela será concluída;
- ▶ A ordem de inicialização não afeta a ordem de execução;
- ▶ Uma thread deixará de ser thread quando o seu método **run()** é concluído;
- ▶ Uma thread **jamaís** poderá ser reinicializada, ou seja, se for feita uma chamada ao **start()** novamente, será gerada uma exceção.

# Métodos de suspensão

- ▶ Permitem interromper ou retomar o funcionamento de uma Thread
- ▶ Deve-se ter cuidado ao realizar uma chamada a estes métodos.
- ▶ **Métodos de Suspensão:**
  - **suspend()**
    - Suspende a execução de uma thread que estiver em processamento naquele momento;
  - **sleep(long millis)**
    - Suspende a execução de uma thread por um determinado período de tempo (em milissegundos);
  - **resume()**
    - Retorna a execução de uma thread que foi suspensa;
  - **stop()**
    - Interrompe permanentemente a execução de uma thread;