Programação paralela usando threads

Programação Orientada a Objetos

2023-03-14

- O hardware está evoluindo para processadores que possuem diversos núcleos para a execução de instruções
- Algumas abordagens
  - Um executável diferente para cada núcleo
  - Usar o recurso de *threads* do sistema operacional

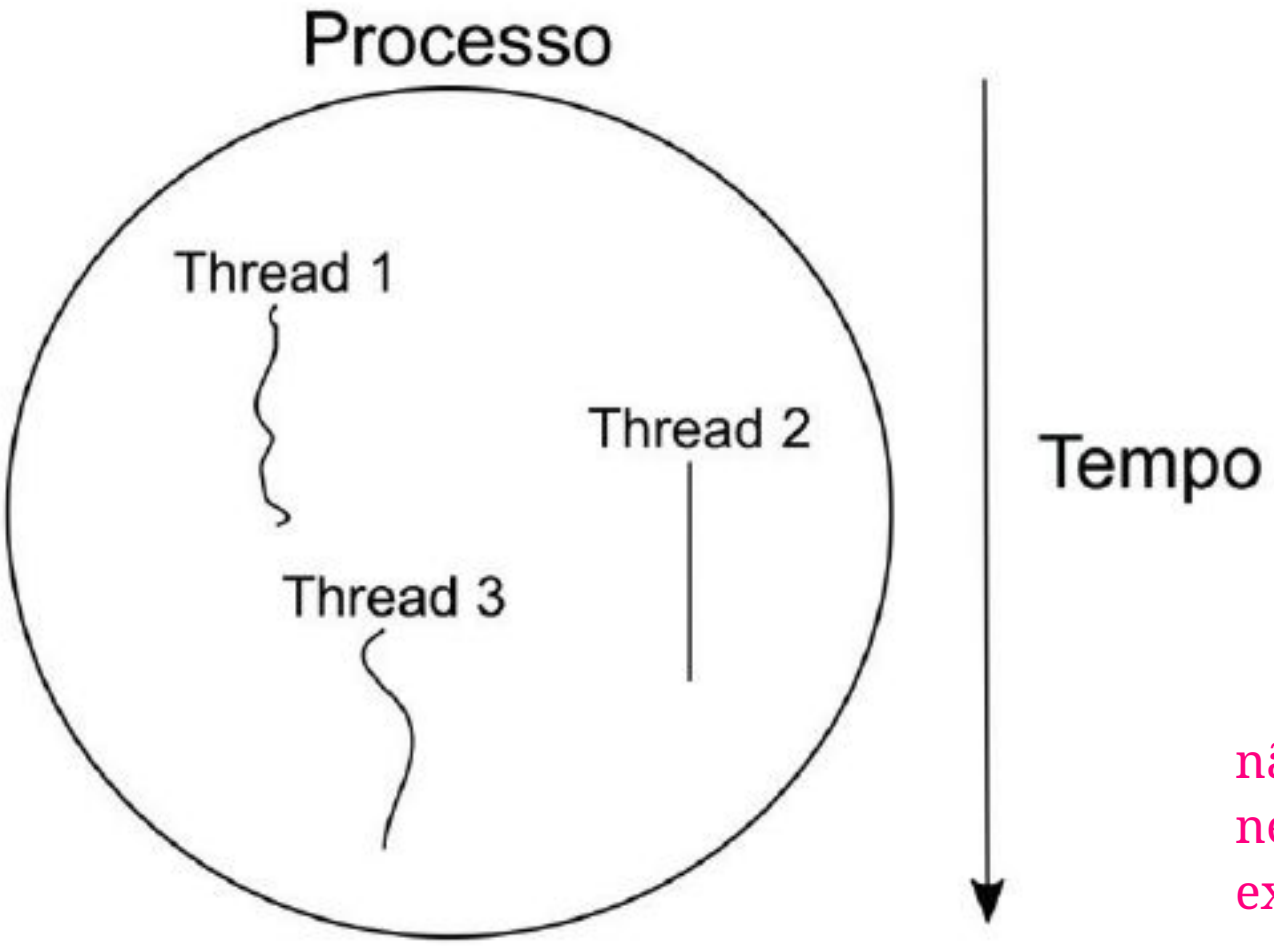
Processo e Thread

https://learn.microsoft.com/pt-br/windows/win32/procthread/about-processes-and-threads

- Processo
  - Fornece os recursos necessários para executar um programa
  - Espaço de endereço virtual, código executável, contexto de segurança, variáveis de ambiente, etc.
  - Pelo menos uma **thread** de execução
- Thread
  - É uma entidade dentro de um processo
  - Todos os threads de um processo compartilham seu espaço de endereço virtual e recursos do sistema

- Para um software efetuar diversas tarefas ao mesmo tempo
  - É preciso utilizar os diversos núcleos do processador de forma paralela, ou seja,
  - É preciso criar um programa que possua várias threads ou simplesmente *multithread*
- “thread” também pode ser chamado de “linha de execução”, significando a mesma coisa

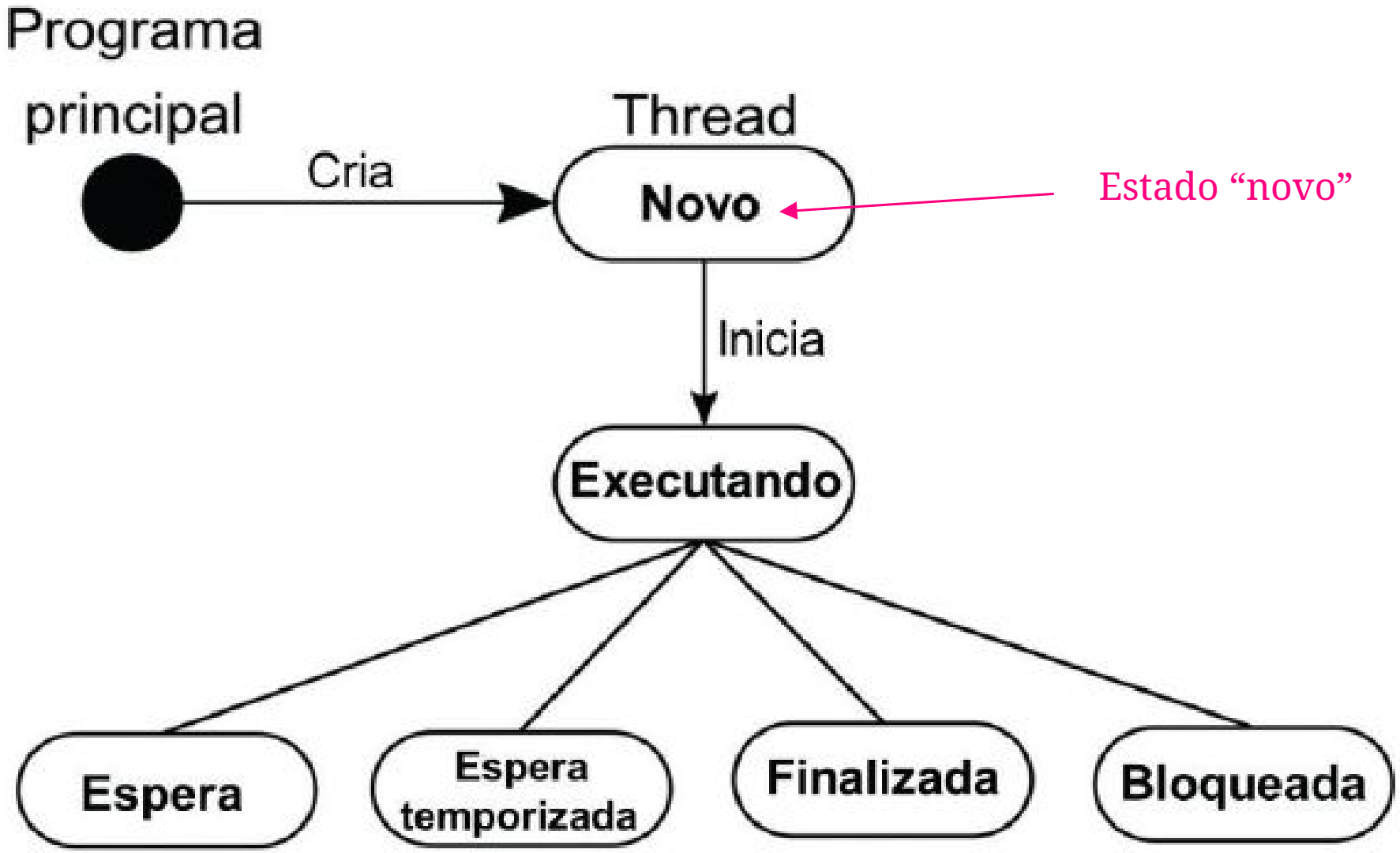
Figura 2.1 | Representação de um processo com threads em um sistema operacional



não necessariamente executa em paralelo

- Na linguagem Java, para que seja possível gerar essas linhas de execução, é necessário utilizar uma Application Programming Interface (API)
- Usa os recusos do sistema operacional para criar as threads
- As threads possuem estados de execução relativos ao tempo de processamento concedido pelo sistema operacional

Figura 2.2 | Criação e estados de uma thread



- O programa principal cria a thread e essa nova linha de execução fica em estado “Novo”
- Quando o sistema operacional escalona essa thread para execução, ela entra no estado “Executando”
- A partir desse ponto é possível que ela passe para quatro outros estados
  - Esperando
  - Espera temporizada
  - Finalizada
  - Bloqueada

- Espera
  - Após executar por um certo tempo, o sistema operacional interrompe o processamento para permitir que as outras threads ou processos possam realizar suas tarefas
- Espera temporizada
  - A própria thread pede para interromper a execução por certo tempo

- Bloqueada
  - Ex.: entrada e saída do sistema
    - Enquanto não está pronto e disponibilizado pelo sistema operacional, deixa a thread no estado “Bloqueada”
- Finalizada
  - O processo que controla a thread determina seu fim e pede para o sistema operacional terminar a execução

- Para criar threads na linguagem Java, é possível utilizar diversas classes que fazem a abstração das linhas de execução de dentro de um processo

- Dois mais comuns:

Quadro 2.1 | Classe que podem ser utilizadas para criar threads

Classe	Descrição
java.lang.Thread	Classe inicial que fornece a utilização básica de threads.
java.util.Timer	Classe que fornece a utilização de thread com elementos de repetição e controle encapsulados.

- Para se criar uma thread, são necessários 3 passos
  - Criar uma classe que implementa
    - a interface Runnable
    - o método run()
  - Criar uma instância tipo Thread
    - indica a classe que implementa a interface Runnable
  - Iniciar a thread com o método start()

```
package U2S1;
1.  public class Processador implements Runnable{
2.      private Thread th;
        public Processador()
3.      {
            th = new Thread(this);
4.          th.start();
5.      }
        public void run() {
6.          for (int i = 0; i < 1000; i++) {
                System.out.println("Processando
dados "+i);
            }

        }
            public static void main(String[] args)
7.          Processador p = new Processador();

        }
}
```

construtor

cria

roda a thread

executado quando a thread é iniciada

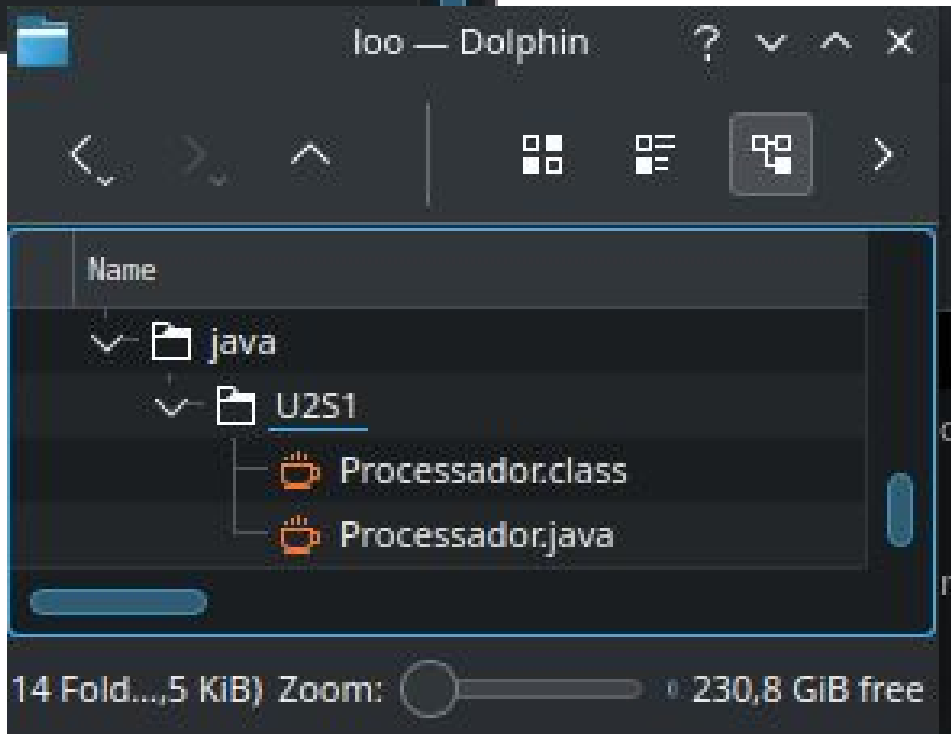
Java inicia a execução por aqui

Processador.java — KWrite

```java
package U2S1;
public class Processador implements Runnable {
  private Thread th;
  public Processador() {
    th = new Thread(this);
    th.start();
  }
  public void run() {
    for (int i = 0; i < 10; i++) {
      System.out.println("Processando dados" + i);
    }
  }
  public static void main(String[] args) {
    Processador p = new Processador();
  }
}
```

java : bash — Konsole

```
$ javac U2S1/Processador.java && java U2S1/Processador
Processando dados0
Processando dados1
Processando dados2
Processando dados3
Processando dados4
Processando dados5
Processando dados6
Processando dados7
Processando dados8
Processando dados9
$
```



Pode ser executado em threads

Jogos utilizam threads para atribuir tarefas de comunicação de redes, controle de mouse, de teclado ou de joystick ou renderização

- Sistemas Enterprise Resources Planning (ERP) utilizam para a geração múltiplos relatórios ou emissão de notas fiscais em paralelo

- Sistemas embarcados podem eventualmente utilizar para leitura de sensores, monitores de atividade, ou alguma outra atividade em paralelo

- Exemplo de uso: watchdogs
  - Softwares que têm o objetivo de monitorar recursos e aplicações
  - Em caso de problemas, podem gerar um alerta ou forçar a reinicialização do software ou do computador
- Programar usando threads pode trazer overhead e complexidade ao código, ao sistema, e à depuração
  - Precisa ser analisado o que vale a pena paralelizar
  - Devido ao overhead, em alguns casos o resultado pode ser pior

Quadro 2.4

```
package U2S1;
import java.util.Scanner;

public class Monitor implements Runnable {
  private Thread th;
  private boolean monitorando;
  public Monitor() {
    monitorando = true;
    th = new Thread(this);
  }
  public void inciar() {
    th.start();
  }
  public void parar() {
    monitorando = false;
    th.interrupt();
    try {th.join(2000);
    } catch (InterruptedException e) {e.printStackTrace();}
  }
```

lê dados do teclado

campo para armazenar o estado

construtor

pede para interromper

espera interromper

caso dê erro e não interrompa, imprime mensagem

(continuação)

executado pela thread

```
public void run() {
    System.out.println("Iniciando monitoramento.");
    while (monitorando == true) {
      // verifica se sistema alvo ainda está em execução
      System.out.println("Monitorando.");
      if (th.isInterrupted() == true) {
        System.out.println("Parando monitoramento.");
        return;
      }
      try {
        Thread.sleep(2000);
      } catch (InterruptedException e) {
        monitorando = false;
      }
    }
  }
```

entra em estado de espera temporizada

caso ocorra interrupção durante o sleep, faz a thread interromper também

(continuação)

trecho principal onde inicia a execução

```
public static void main(String[] args) {
  Monitor monitor = new Monitor();
  monitor.inciar();
  Scanner sc = new Scanner(System.in);
  boolean monitorar = true;
  do {
    System.out.println("Continuar monitoramento S/N?");
    String resp = sc.next();
    if (resp.equalsIgnoreCase("N") == true) {
      monitorar = false;
      monitor.parar();
    }
  } while (monitorar == true);
  sc.close();
  }
}
```

cria

roda

entrada de teclado

este laço de repetição é executado em paralelo à outra thread

- O uso de threads é essencial para seja possível produzir códigos de qualidade que utilizam os recursos do processador de forma otimizada

- A implementação desses recursos cria diversas possibilidades para interações, desempenho e monitoramento

- Como pode ser uma execução paralela, deve-se planejar e manter a organização do código para evitar problemas de alto uso de recursos e problemas de sincronismo

- A programação paralela é um tópico que apresenta certa complexidade, pois é necessário pensar que diversas tarefas serão executadas ao mesmo tempo