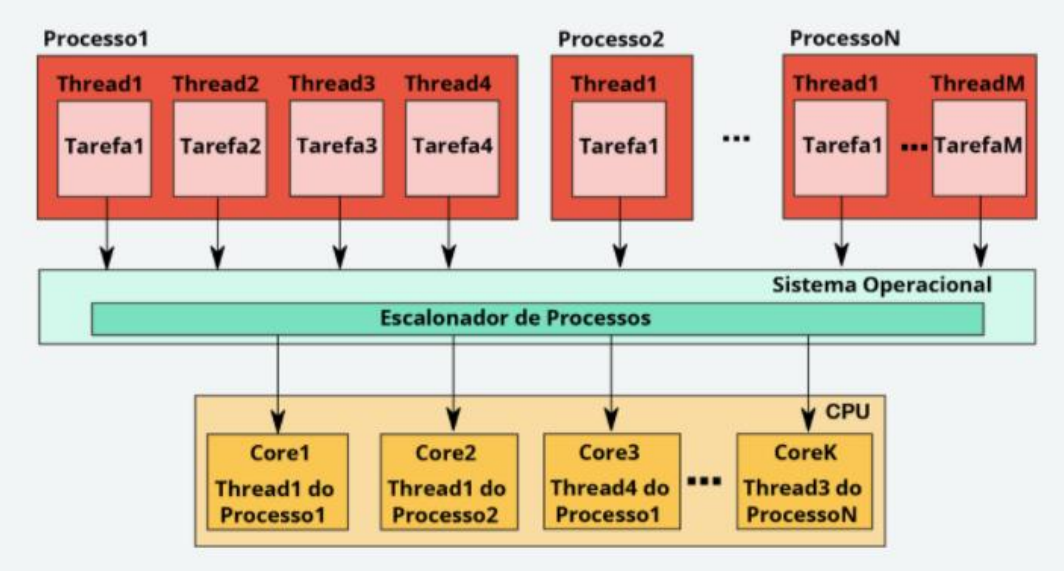
Programação em Java usando *threads*

Fluxo de execução/escalonamento

O esquema a seguir ilustra os processos, suas *threads* e como são escalonados pelo sistema operacional para serem executados pelos núcleos do processador.



Fonte: adaptado de AndroidBerry.

- **PROCESSOS E THREADS**: existem N processos em execução no computador; cada processo tem, no mínimo, uma *thread* e, no máximo, M *threads*. É importante destacarmos que cada processo pode estar vinculado a uma aplicação diferente. Por exemplo: o Processo1 pode ser um navegador de internet, o Processo2 pode ser um editor de texto aberto e o ProcessoN pode ser um IDE aberto.
- **SISTEMA OPERACIONAL**: Todos esses processos e threads são executadas sobre um Sistema Operacional (SO). Uma das partes do SO é o escalonador de processos, que pega um conjunto de processos e threads e o direciona para que o hardware o execute.
- **CPU**: O hardware do computador em questão possui K núcleos (cores), dessa maneira, o SO direciona cada uma das threads para um núcleo (core) diferente executar; é o escalonador de processos que decide quem vai executar, por quanto tempo vai executar e em qual núcleo vai ser executado.

Pesquise Mais

A classe Thread possui diversos métodos para manipulação da *thread*, destacando-se: *start*, *sleep*, *isAlive* e *join*, que são extremamente importantes para a realização da manipulação de threads. Pesquise mais sobre esses métodos nas seguintes videoaulas.

- LOIANE GRONER. **Curso de Java 67**: criando threads + métodos start, run e sleep. 2016.
- LOIANE GRONER. **Curso de Java 68**: threads: interface runnable. 2016.
- LOIANE GRONER. **Curso de Java 69**: criando várias threads + métodos isAlive e join.