Procesos e threads

Sistemas Operacionais
Gerência de processos

Agenda

- Contextualização
- Processo
  - Visão geral
  - Estados de um processo
  - Troca de contexto e PCB
- Threads
  - Visão geral
  - Modelos de multithreading
  - Threads em Java

Lembrando...

* Software
  * Sinônimos : Programa, sistema de software, aplicativo...
  * Tipos : Software do usuário, software do SO, software utilitário
* Multiprogramação vs. multiprocessamento
  * Multiprogramação é um pseudoparalelismo: coleção de softwares sendo executados alternadamente na CPU
  * Multiprocessamento é a execução de instruções em 2 ou mais processadores

Lembrando...

* Software
  * Programa, sistema de software, aplicativo...
  * Software do usuário, software do SO, software utilitário
* SO também gerencia os softwares em execução
  * Independente do tipo de SO (multitarefa/ multiprogramado ou monotarefa, tempo real...)

Agenda

- Contextualização
- Processo
  - Visão geral
  - Estados de um processo
  - Troca de contexto e PCB
- Threads
  - Visão geral
  - Modelos de multithreading
  - Threads em Java

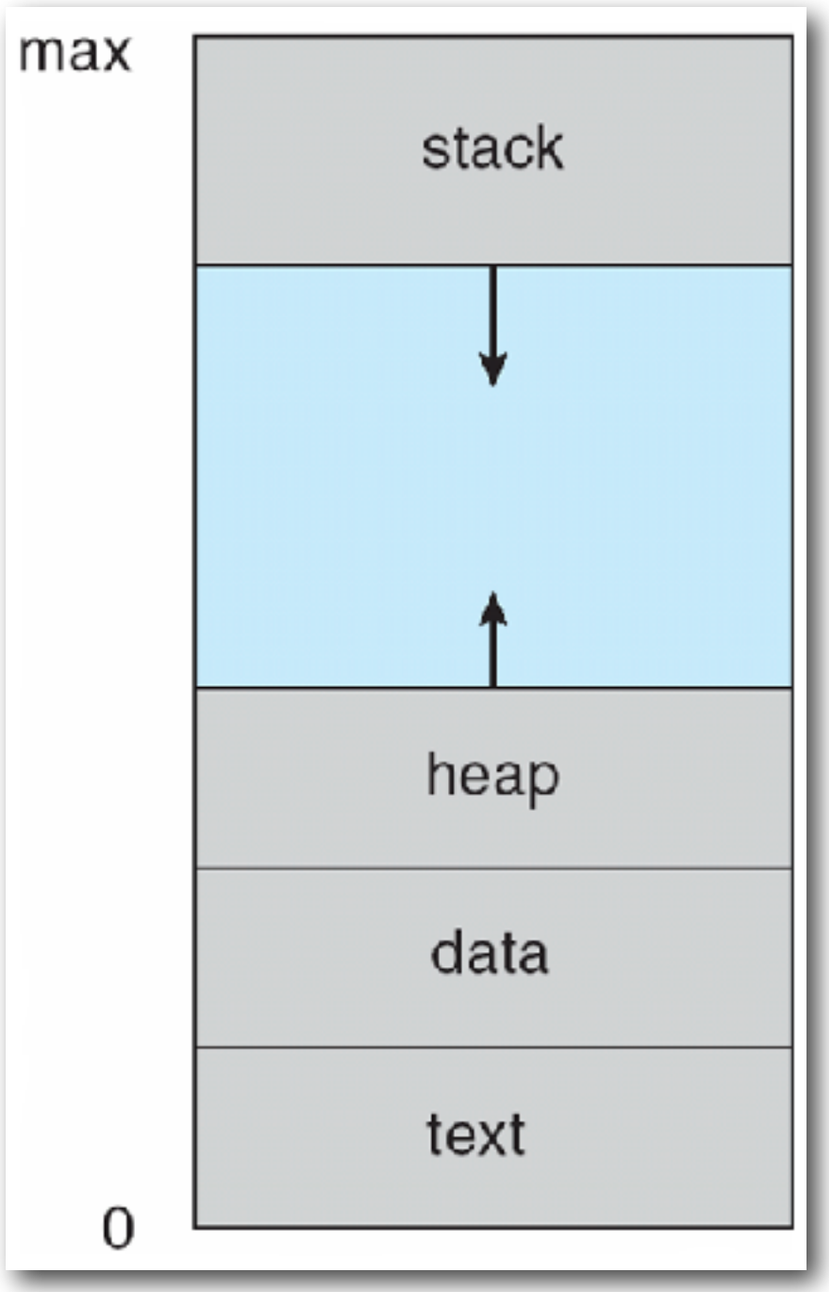
Visão geral de processo

* Um sistema operacional executa diversos programas
  * Sistemas batch – jobs
  * Sistemas compartilhados no tempo – programas ou tarefas do usuário
* Normalmente job e processo são sinônimos
* Ou seja, software (programa, aplicativo, sistema de software...) é a informação em disco, enquanto que processo é o software na memória ("em execução")

Visão geral de processo

- Conceito de processo
  - Um software em execução forma a "base" de toda a computação
    - A execução do processo deve progredir de modo seqüencial
- Um processo inclui, entre outras coisas:
  - Identificador, contador de programa, pilha, seção de dados (ver mais adiante PCB)

Exemplo de representação de um processo na memória



Ciclo de vida de um processo

* O sistema operacional
* Cria um processo
* Controla a execução dos processos
* Finaliza um processo

Criar um processo

* Processos são criados
  * Na inicialização do sistema
  * [Subprocesso] Execução de uma chamada ao sistema de criação de processo realizada por algum processo em execução
  * Requisição de usuário para criar um novo processo
  * Inicialização de processos em lote

Finalizar um processo

* Condições de finalizar
  * Execução normal
  * Por erros
    * Exemplo: Proteção, aritméticos, E\S, tentativa de execução de instruções inválidas, falta de memória, exceder tempo de limite
* Intervenção de outros processos (kill)
* Log off de usuários

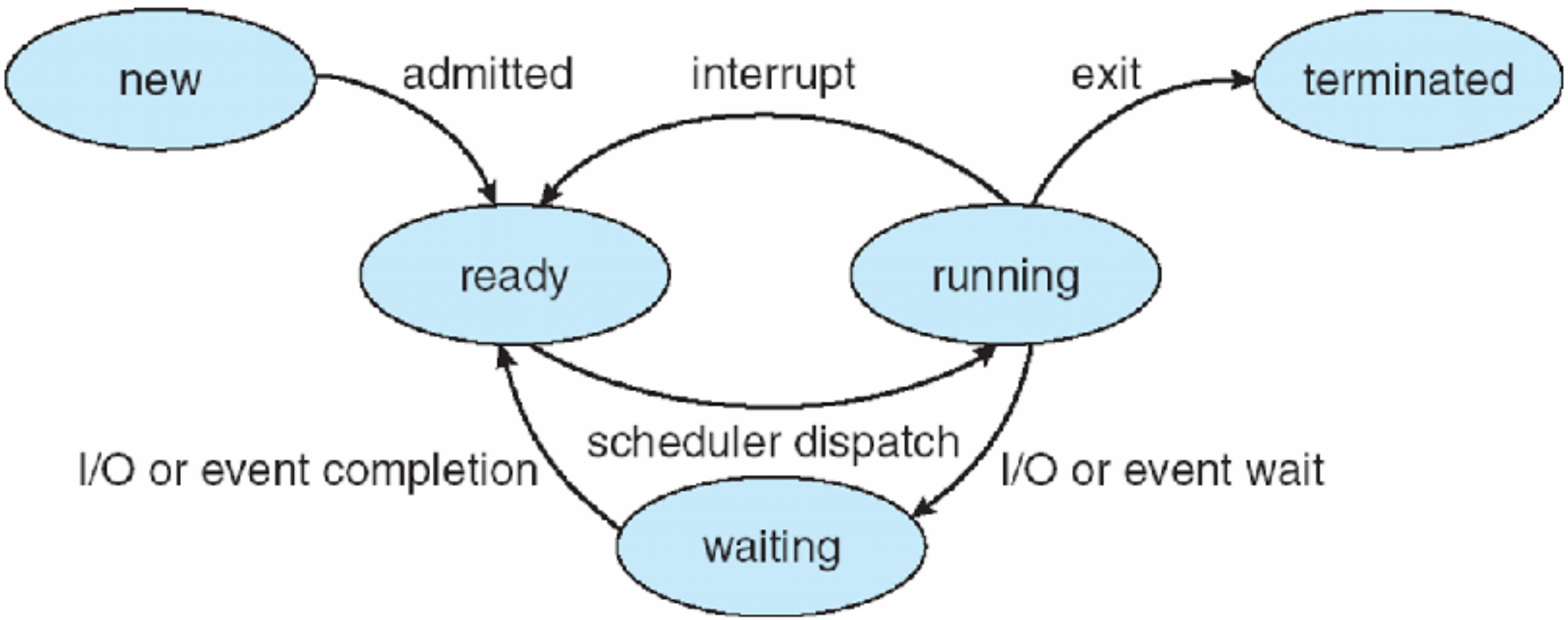
Relacionamento entre processos

- Processos independentes
  - Não apresentam relacionamento com outros processos
- Grupo de processos
  - Apresentam algum tipo de relação
  - Compartilham recursos
  - Podem ter uma hierarquia
- Hierarquia de processos
  - Grupo de processos que apresentam hierarquia
  - Processo criador é nomeado como processo pai
  - Processos criados são nomeados como processos filhos

Agenda

- Contextualização
- **Processo**
  - Visão geral
  - **Estados de um processo**
  - Troca de contexto e PCB
- Threads
  - Visão geral
  - Modelos de multithreading
  - Threads em Java

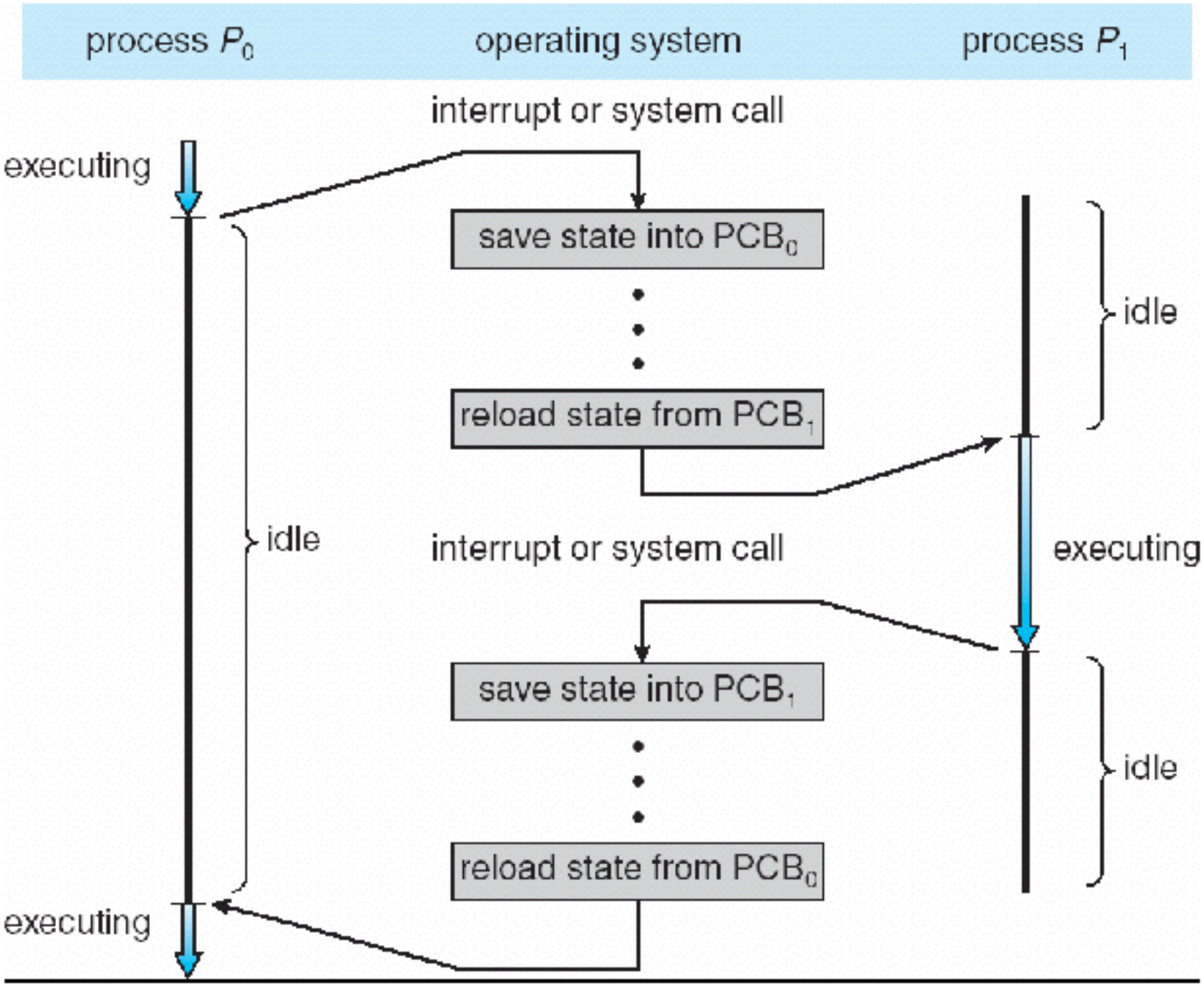
Estados de processo



Agenda

- Contextualização
- **Processo**
  - Visão geral
  - Estados de um processo
  - **Troca de contexto e PCB**
- Threads
  - Visão geral
  - Modelos de multithreading
  - Threads em Java

Troca de contexto



PCB - Process Control Block

- Informações associadas a cada processo
- Estado do processo
- Contador de programa
- Registradores da CPU
- Informação de escalonamento da CPU
- Informação de gerenciamento de memória
- Informação de contabilidade
- Informação de estado de E/S

PCB - Process Control Block

https://github.com/torvalds/linux/blob/master/include/linux/sched.h

```
struct desc_proc{
     char       estado_atual;
     int        prioridade;
     unsigned   inicio_memoria;
     unsigned   tamanho_mem;
     struct     arquivos arquivos_abertos[20];
     unsigned   tempo_cpu;
     unsigned   proc_pc;
     unsigned   proc_sp;
     unsigned   proc_acc;
     unsigned   proc_rx;
     struct     desc_proc *proximo;
}

struct desc_proc tab_desc[MAX_PROCESS];
```

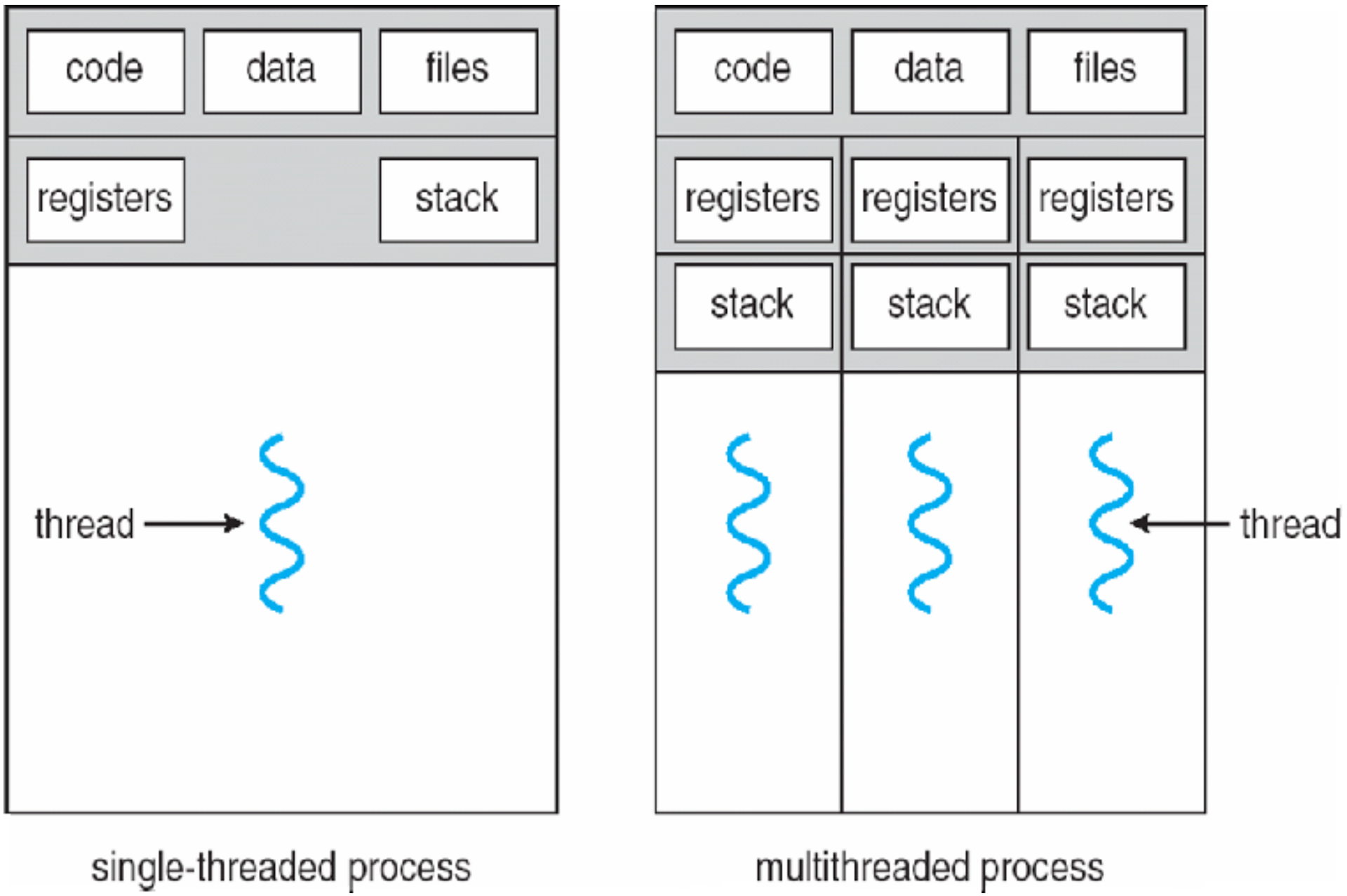
Algumas outras informações sobre processos

- Como organizar os processos em Estado de pronto?
  - Escalonamento de processo
- Relacionamento entre processos
  - Como fazer os processos trocarem informações?
    - Processos em "paralelo" e/ou "concorrentes"

Agenda

* Contextualização
* Processo
  * Visão geral
  * Estados de um processo
  * Troca de contexto e PCB
* **Threads**
  * **Visão geral**
  * Modelos de multithreading
  * Threads em Java

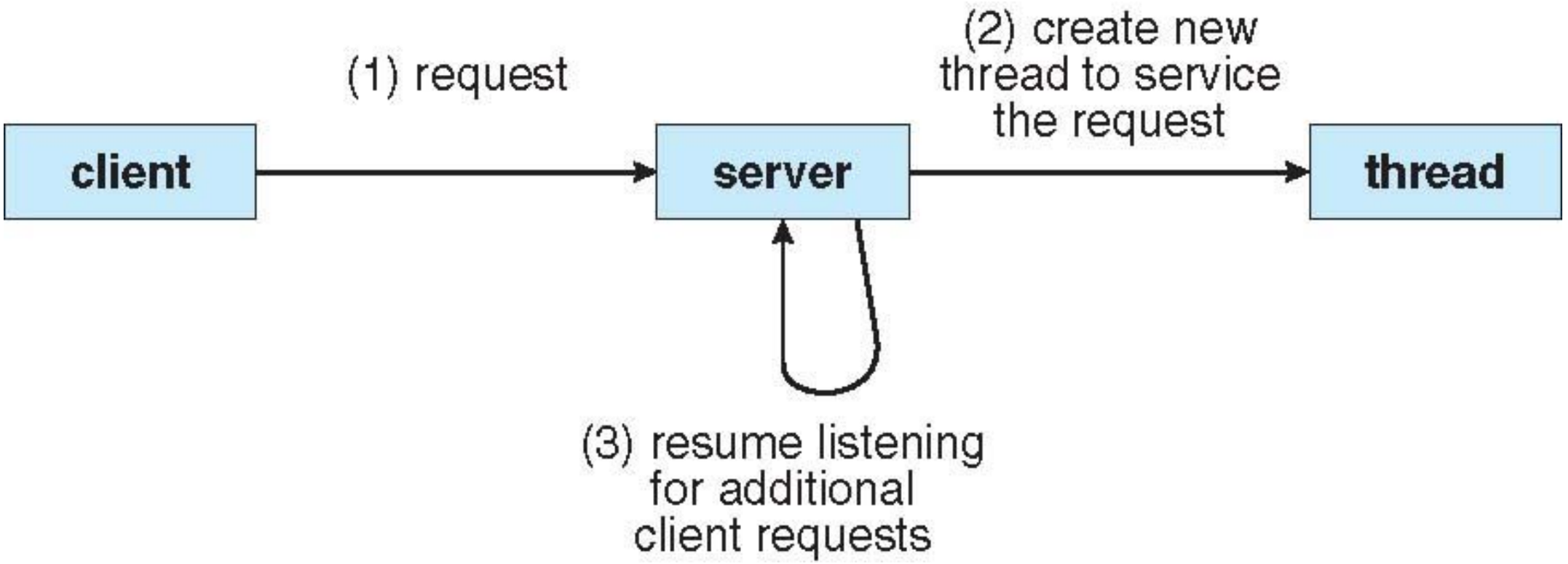
Visão geral


code
data
files
registers
stack
thread
single-threaded process
code
data
files
registers
registers
registers
stack
stack
stack
thread
multithreaded process

Benefícios Thread vs. Processos

* Responsividade das interfaces com o usuário
* Compartilhamento de recursos entre threads
* Economia de recursos entre threads
* Utilização de arquiteturas com concorrência interna
* Escalabilidade do software

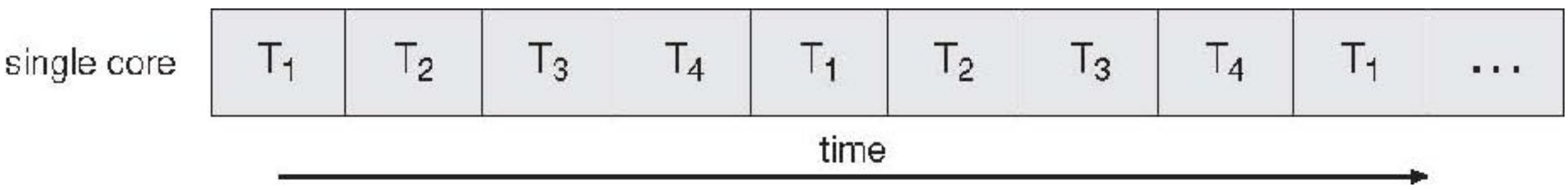
Exemplo de arquitetura

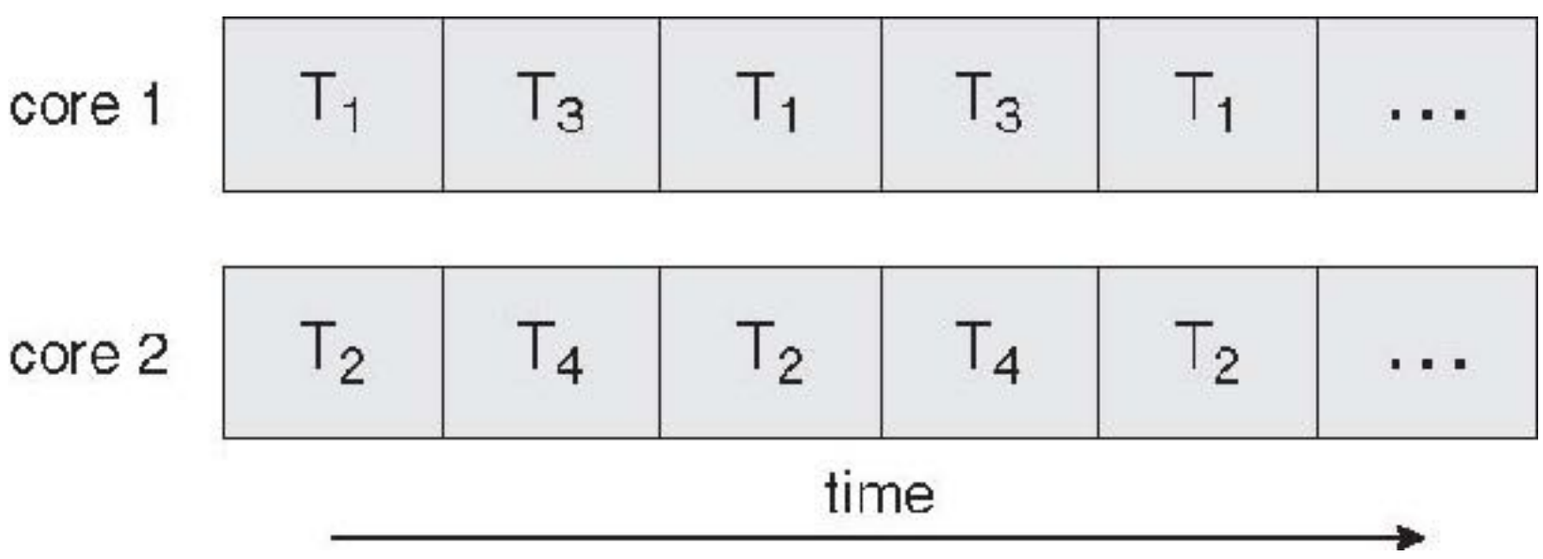


Desafios da programação de threads

* São os mesmos desafios da programação paralela e concorrente (concorrência e sincronização)
* Alguns desafios
  * Divisão de atividades, balanceamento das atividades, dependência de dados, e testar/depurar

Execução concorrente





Agenda

- Contextualização
- Processo
  - Visão geral
  - Estados de um processo
  - Troca de contexto e PCB
- **Threads**
  - Visão geral
  - **Modelos de multithreading**
  - Threads em Java

Tipos de threads

* Threads do usuário
  * Gerenciamento de thread feito pela biblioteca de threads da plataforma
  * Threads em nível de usuário
* Threads do kernel
  * Threads admitidos diretamente pelo kernel
  * Gerenciamento de threads feito pelo SO

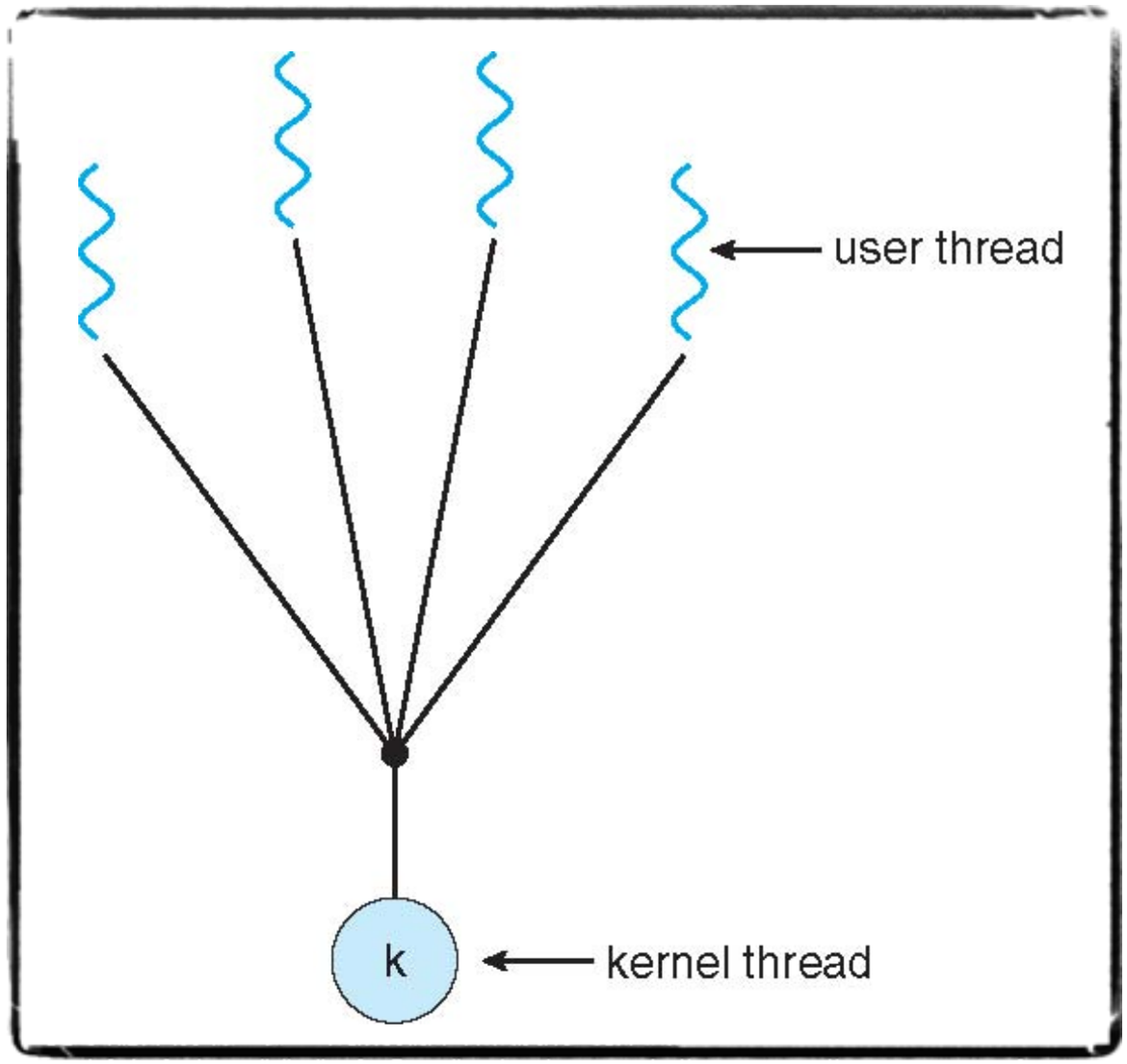
Modelos de multithreading

* Define o mapeamento entre threads de usuário e threads do kernel
* São modelos
  * Muitos para um
  * Um para um
  * Muitos para muitos

Muitos para um

* Muitas threads de nível de usuário mapeadas para uma única thread de nível kernel

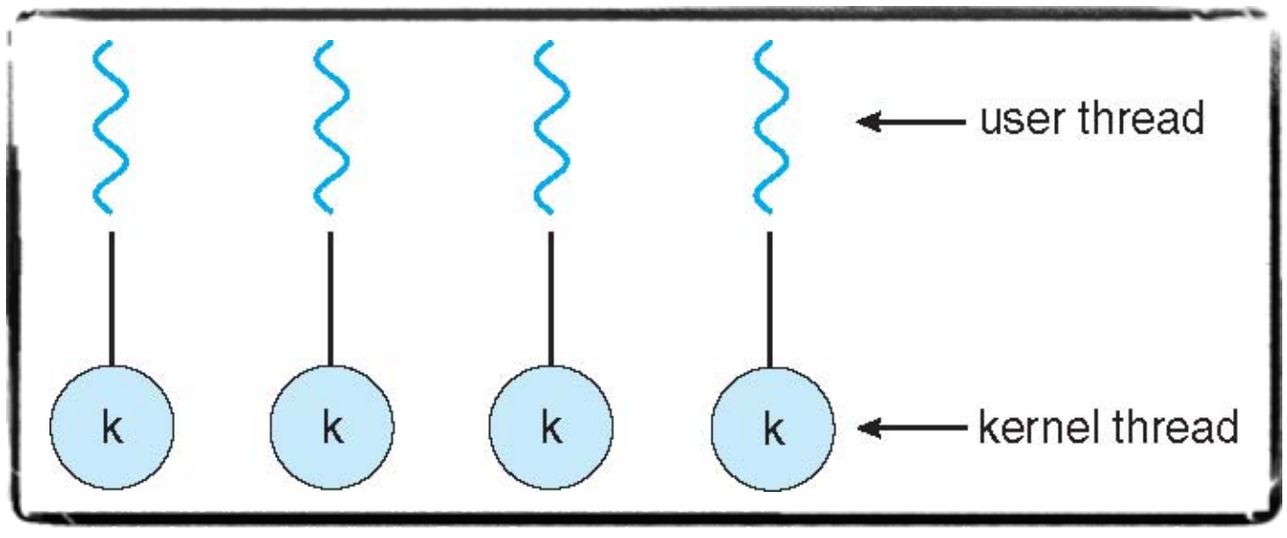
* Exemplos
  * Solaris green threads
  * GNU portable threads



Um para um

* Cada thread de nível usuário é mapeada para uma thread de nível de kernel

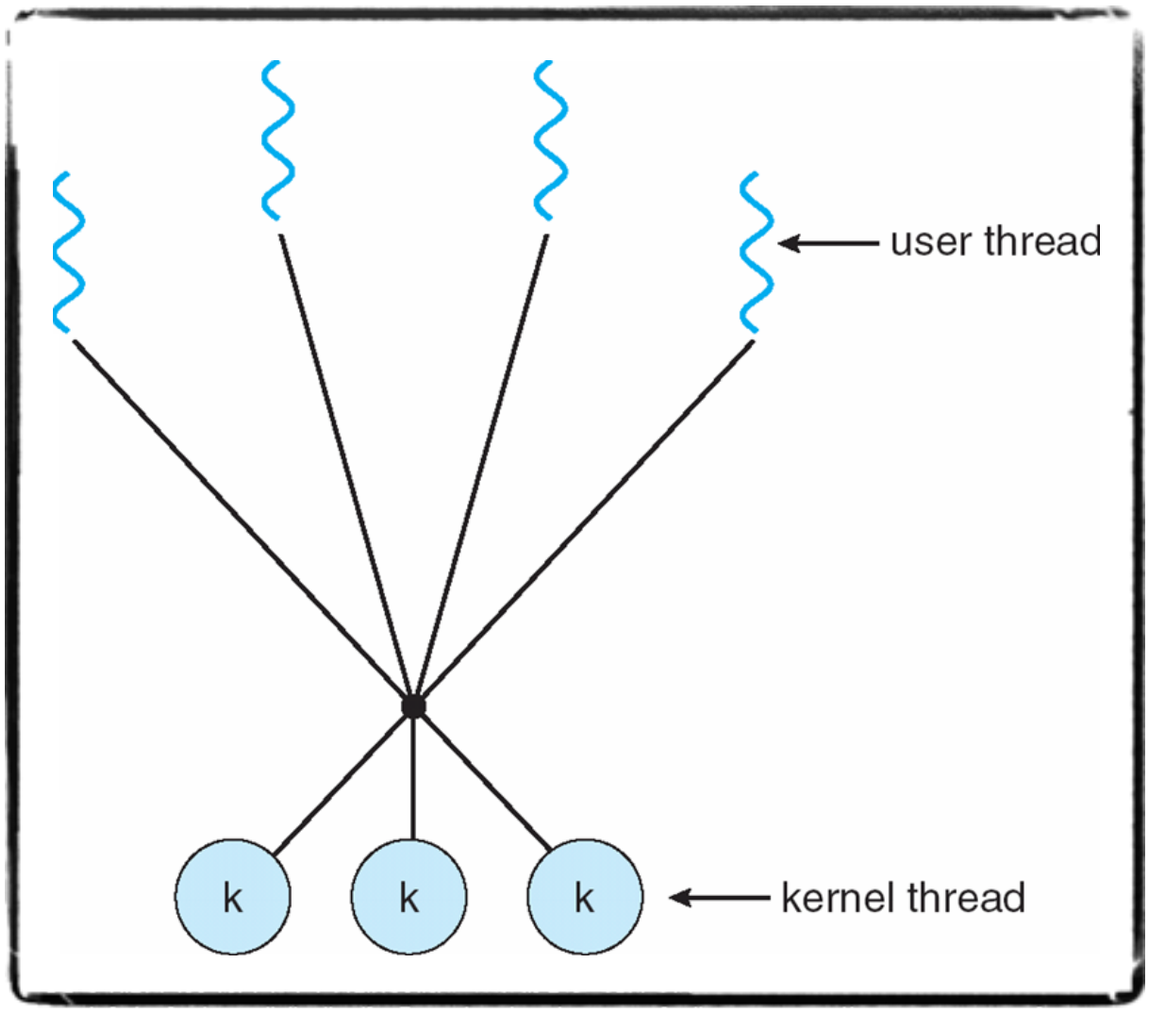
* Exemplos
  * Windows NT/XP/2000
  * Linux
  * Solaris 9 e posterior



Muitos para muitos

* Permite muitas threads em nível de usuário sejam mapeadas à muitas threads de nível de kernel

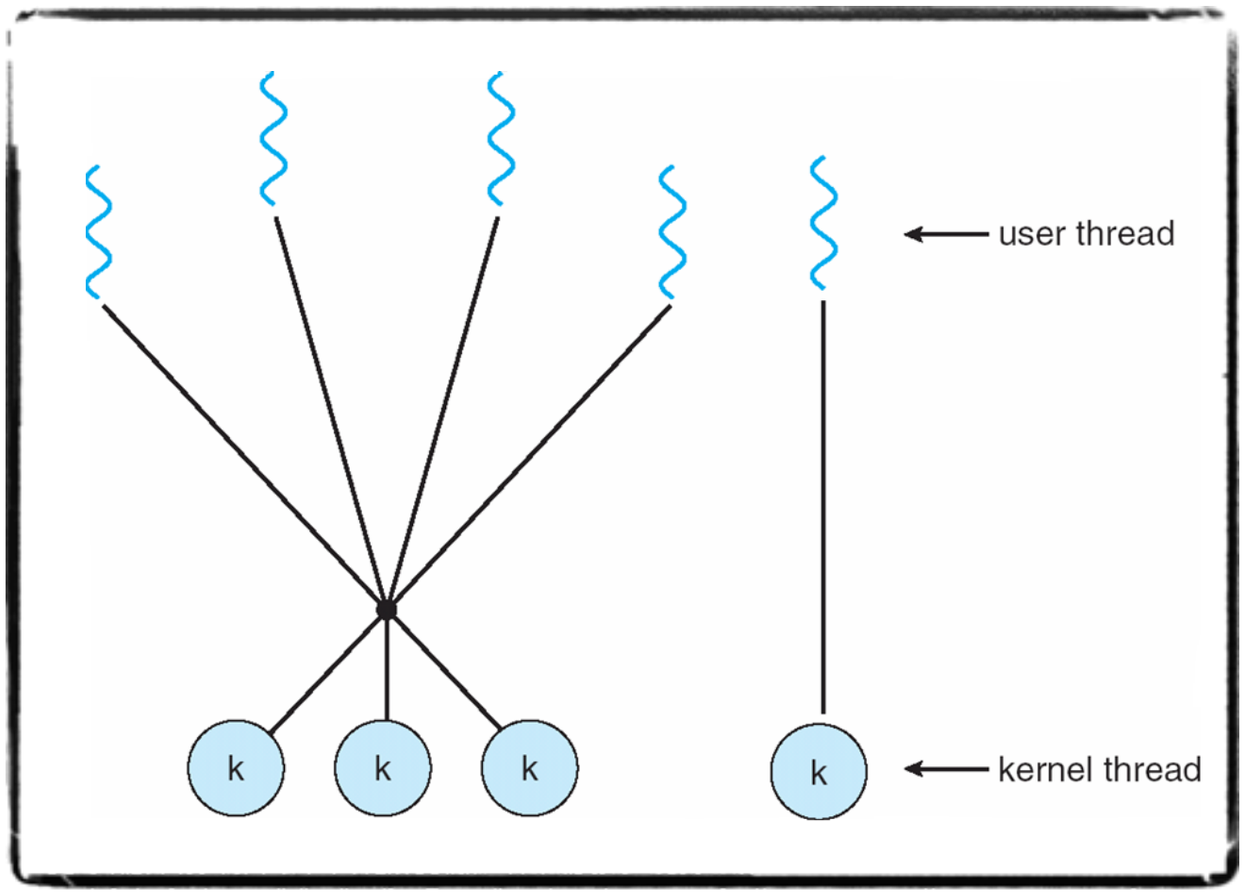
* Exemplos
  * Versões anteriores a Solaris 9
  * Windows NT/2000 com o pacote ThreadFiber



Hibrído em 2 níveis

* Semelhante a "Muitos para Muitos", exceto por permitir que uma thread de nível de usuário seja mapeada a uma thread de nível kernel

* Exemplos
  * IRIX, HP-UX, Tru64 UNIX
  * Solaris 8 e anteriores



Agenda

* Contextualização
* Processo
  * Visão geral
  * Estados de um processo
  * Troca de contexto e PCB
* **Threads**
  * Visão geral
  * Modelos de multithreading
  * **Threads em Java**

Threads em Java

* Introdução
* Ciclo de vida
* Como criar threads
* Exemplos

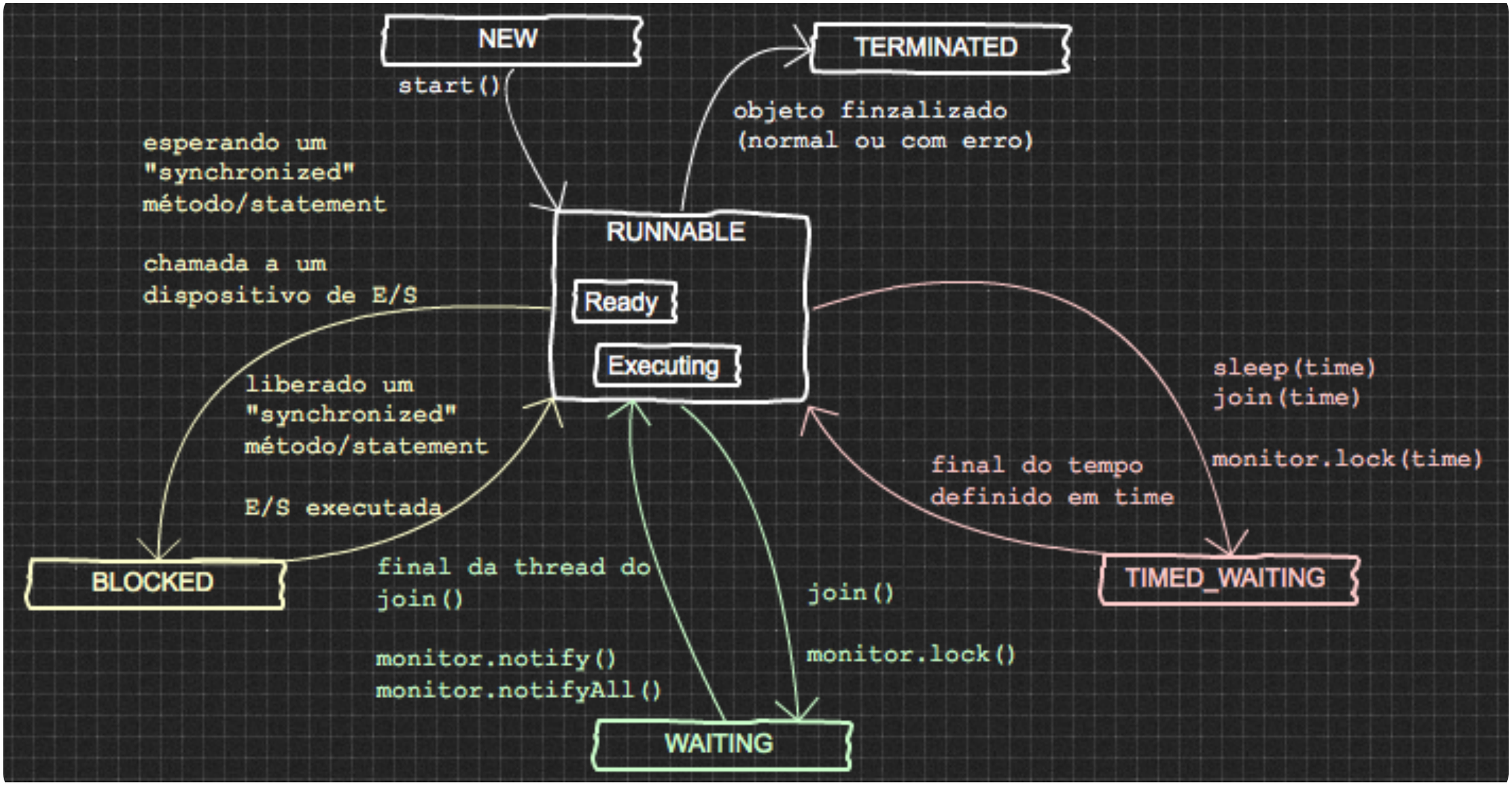
Introdução

* Os mecanismos de gerenciamento, comunicação e sincronização de threads foram incorporados a linguagem
* Threads java são representadas por objetos da classe java.lang.Thread

Introdução

* Quando um software Java é executado
  * A JVM cria um objeto do tipo Thread, cuja tarefa é executar o método main()
  * Este objeto Thread é iniciado automaticamente
  * As instruções descritas no método main() são executadas sequencialmente até que o método termine, finalizando a execução do objeto thread

Ciclo de vida



java.lang.Thread.State

* NEW : um objeto que ainda não foi iniciado
* RUNNABLE : um objeto que esta em execução na JVM
* BLOCKED : um objeto executa chamada a um dispositivo de E/S ou "entra" em um bloco de código "synchronized"
* WAITING : um objeto esta aguardando por eventos externos (monitor.notify* ou final de execução de um objeto thread)
* TIME_WAITING : um objeto aguarda por um tempo determinado
* TERMINATED : um objeto completou sua execução (normal ou com erro)

Como criar threads

* Apresento 2 formas de criar explicitamente um objeto thread em java
  * extend java.lang.Thread
  * implements java.lang.Runnable

Como criar threads

* Apresento 2 formas de criar explicitamente um objeto thread em java
  * extend java.lang.Thread
    * Cria uma classe que herde de Thread
    * Instancia um objeto desta nova classe (nomeada ou anônima)
  * implements java.lang.Runnable

Como criar threads

- Apresento 2 formas de criar explicitamente um objeto thread em java
  - extend java.lang.Thread
  - implements java.lang.Runnable
    - Cria uma classe que implemente a interface java.lang.Runnable
    - Instancia um objeto dessa nova classe
    - Cria um objeto do tipo Thread, passando como parâmetro o objeto de interface Runnable

Como criar threads

* Implementar o método run()
  * Em ambos os casos apresentados, é necessário implementar na nova classe o método run()
* Este método irá descrever o que realmente as instruções a serem executadas da thread

Exemplo [extends Thread]

```java
public class SimpleThread extends Thread {

  final private int MAX = 10;
  final private int INIT = 0;

  public SimpleThread() {
    this("unnamed");
  }
  public SimpleThread(String name) {
    super(name);
  }

  public void run() {
    System.out.println("Thread [" + getName() + "] is started!");
    for (int i = INIT; i < MAX; i++) {
      System.out.println(getName() + " - " + i);
    }
    System.out.println("Thread [" + getName() + "] is done!");
  }
}
```

Exemplo [implements Runnable]

```
1  public class SimpleRunnable implements Runnable {
2
3    final private int MAX = 10;
4    final private int INIT = 0;
5    private String name;
6
7    public SimpleRunnable() {
8      this("unnamed");
9    }
10   public SimpleRunnable(String name) {
11     super();
12     this.name = name;
13   }
14
15   private String getName() {
16     return this.name;
17   }
18
19   public void run() {
20     System.out.println("Thread [" + getName() + "] is started!");
21     for (int i = INIT; i < MAX; i++) {
22       System.out.println(getName() + " - " + i);
23     }
24     System.out.println("Thread [" + getName() + "] is done!");
25   }
26 }
```

Exemplo
Instanciando e executando

```
1 public class SimpleThreadDemo {
2   public static void main(String[] args) {
3     Thread thread = new SimpleThread("SimpleThread");
4
5     Runnable runnable = new SimpleRunnable("SimpleRunnable");
6     Thread threadRunnable = new Thread(runnable);
7
8     thread.start();
9     threadRunnable.start();
10  }
11 }
```

Exemplo
Possível resultado

```
MacBook-Pro-de-Leonardo:threads leo$ javac SimpleThreadDemo.java
MacBook-Pro-de-Leonardo:threads leo$ java SimpleThreadDemo
Thread [SimpleThread] is started!
SimpleThread - 0
SimpleThread - 1
SimpleThread - 2
SimpleThread - 3
SimpleThread - 4
SimpleThread - 5
SimpleThread - 6
SimpleThread - 7
SimpleThread - 8
SimpleThread - 9
Thread [SimpleThread] is done!
Thread [SimpleRunnable] is started!
SimpleRunnable - 0
SimpleRunnable - 1
SimpleRunnable - 2
SimpleRunnable - 3
SimpleRunnable - 4
SimpleRunnable - 5
SimpleRunnable - 6
SimpleRunnable - 7
SimpleRunnable - 8
SimpleRunnable - 9
Thread [SimpleRunnable] is done!
MacBook-Pro-de-Leonardo:threads leo$
```

Bibliografia

Processos e threads
Gerência de processos
Sistemas operacionais

Bibliografia

* **The Java Tutorials: concurrency**. Disponível em http://docs.oracle.com/javase/tutorial/essential/concurrency/ (acessado em 18/01/2013)
* SILBERSCHATZ, G.; GAGNE, G. Sistemas Operacionais com Java. Campus, 7a Ed, 2007.

Procesos e threads

Sistemas Operacionais
Gerência de processos