



Universidade Federal do Ceará
Campus Quixadá

Disciplina: Sistemas Operacionais

REVISÃO DE PROCESSOS E THREADS

PROF. SIDARTHA CARVALHO

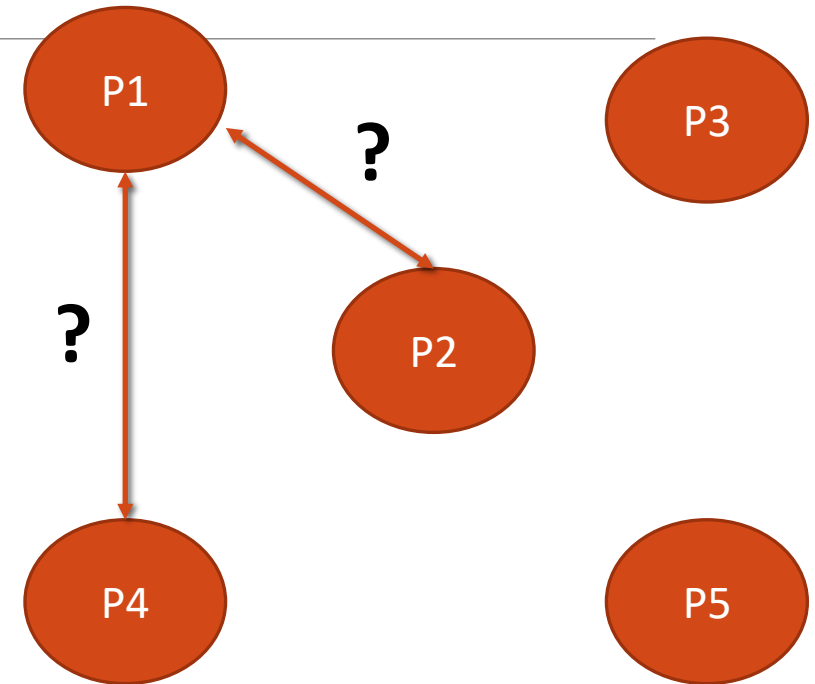
PROCESSOS



www.shutterstock.com · 760273996

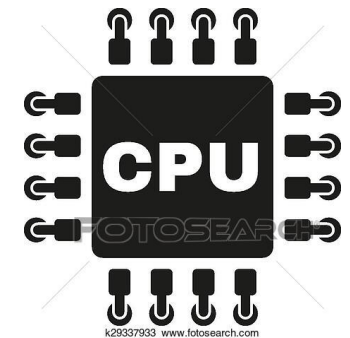
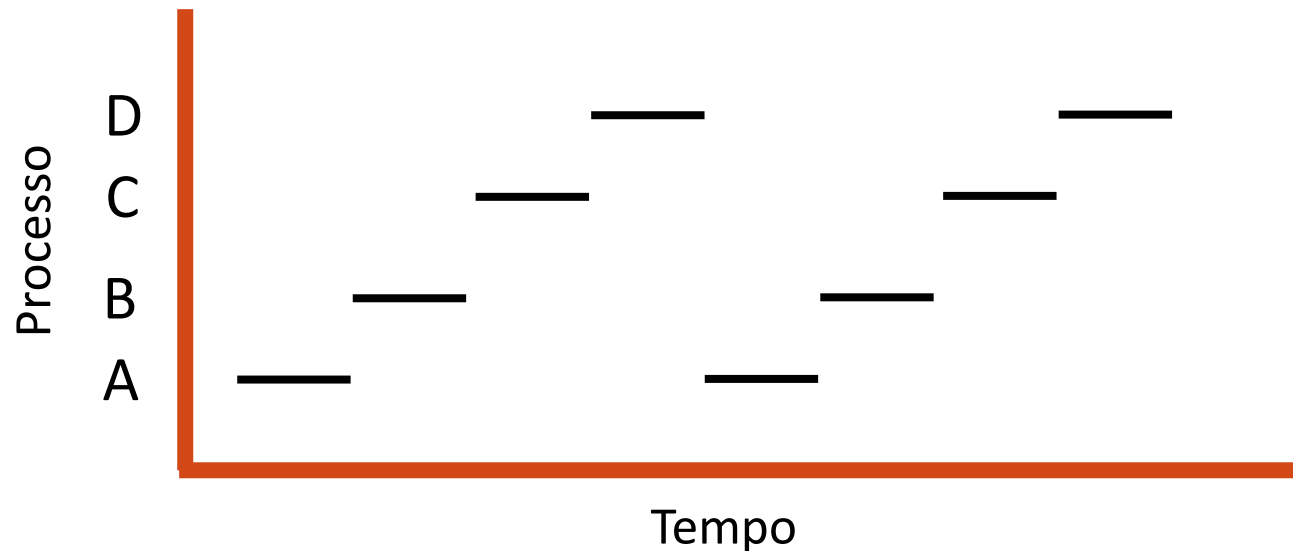
Processos

- Abstração de programa
- Execução “simultânea” de programas
- Código fonte em execução
- Dois programas em execução são dois processos
 - Mesmo que seja o mesmo programa!



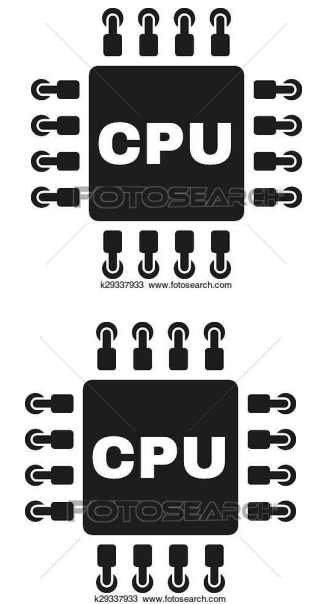
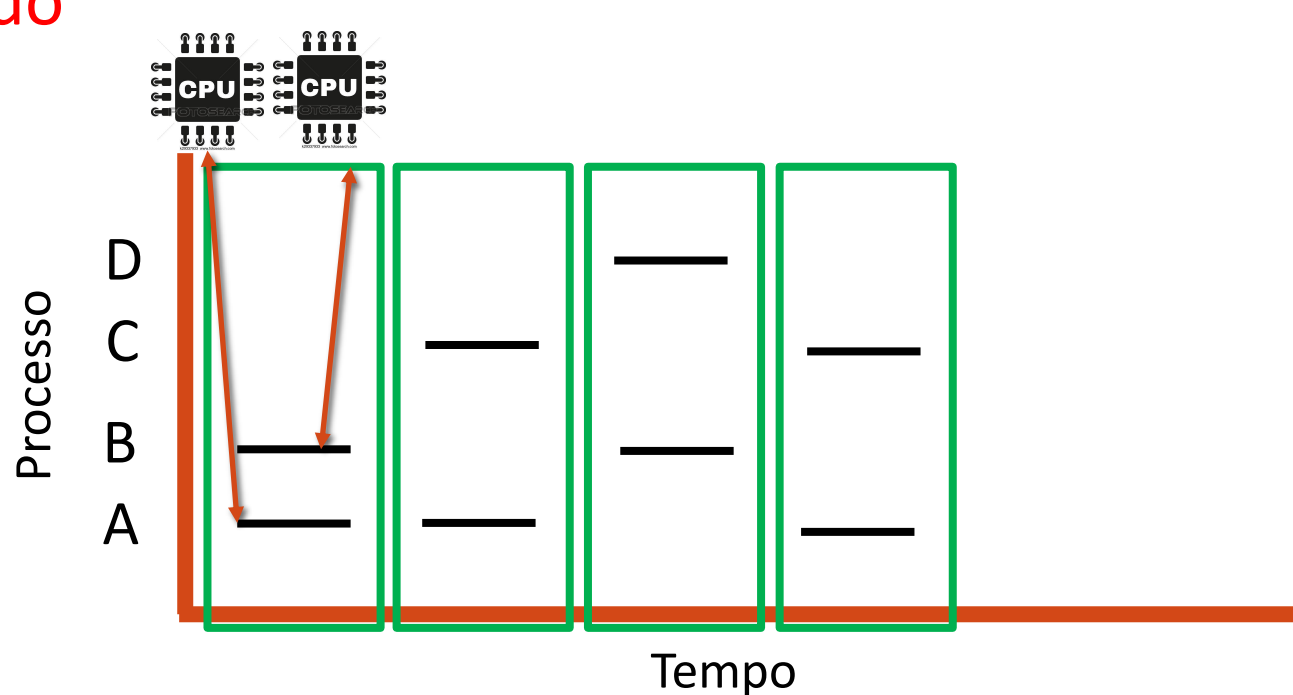
O modelo de Processo

Por um intervalo de tempo, todos os processos estão avançando, mas a cada instante, **apenas um único processo está realmente executando**



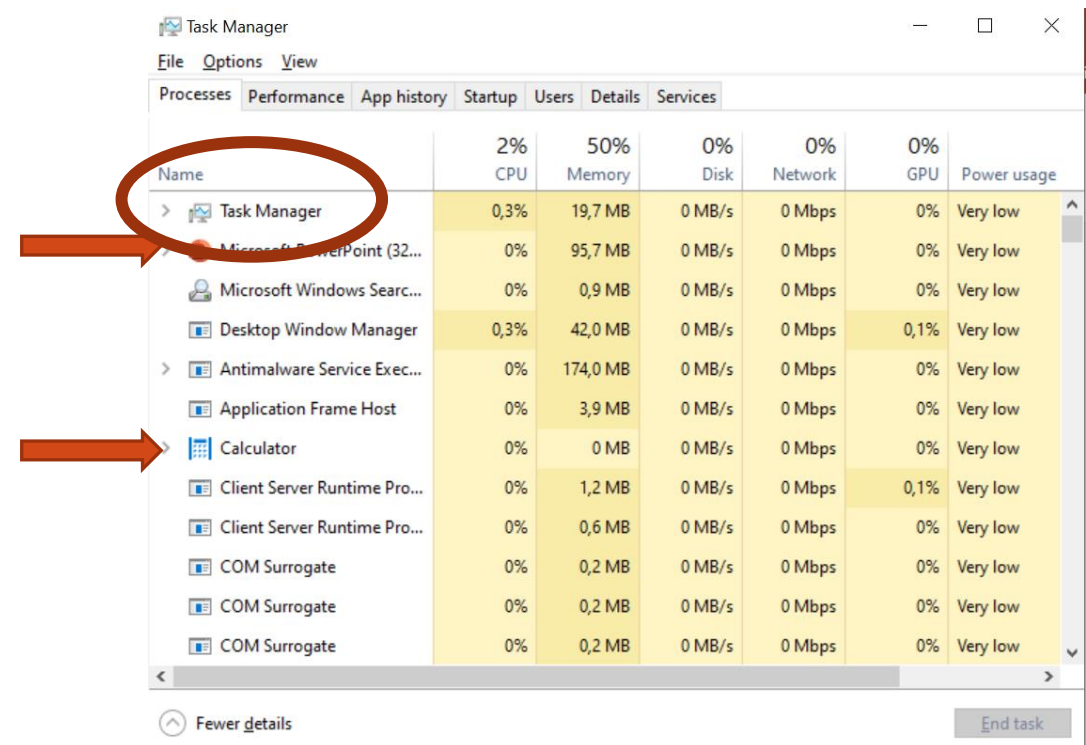
O modelo de Processo

Por um intervalo de tempo, todos os processos estão avançando, mas a cada instante, **apenas um único processo está realmente executando**



Exemplos de processos no Windows

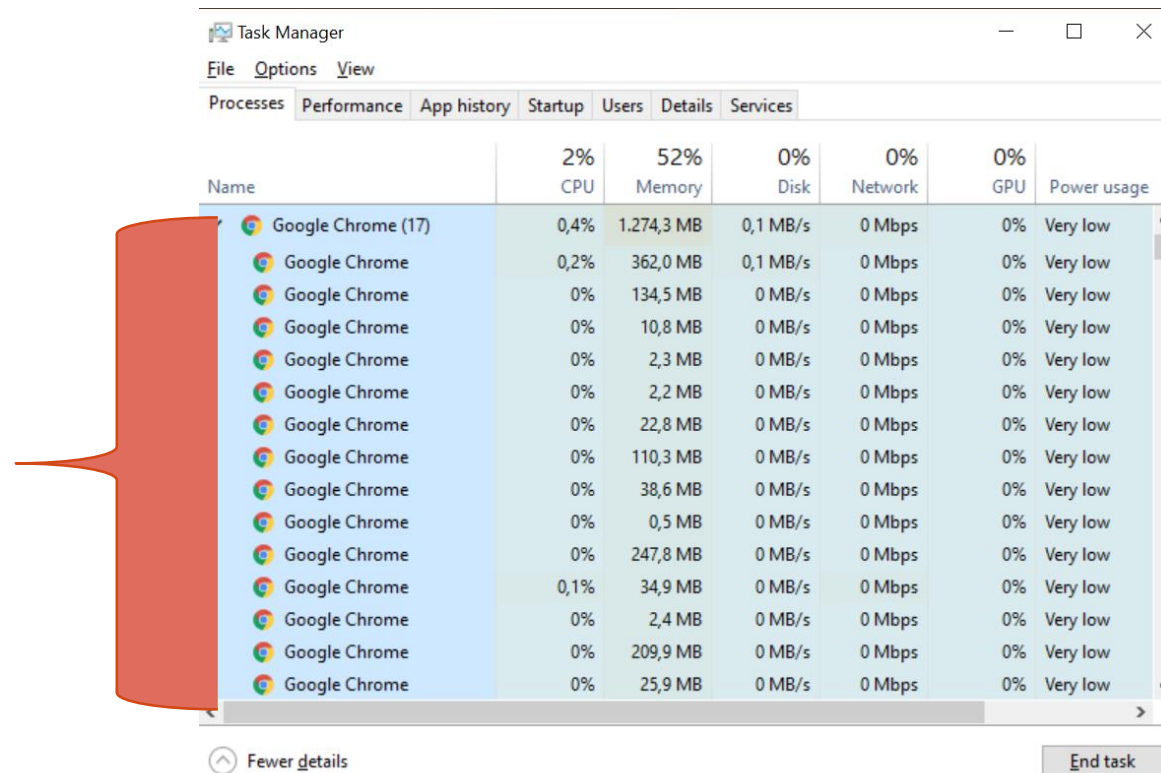
- Aperte CTRL + ALT + DELETE para entrar no gerenciador de tarefas
- Clique em exibir mais detalhes no canto inferior esquerdo



Exemplos de processos no Windows

- O navegador Chrome usa um processo para cada aba, extensão ou plugin aberto, isso ajuda a encerrar um componente se houver falha sem afetar os demais recursos do navegador

Processos do Google Chrome



Task Manager						
File Options View						
Processes Performance App history Startup Users Details Services						
Name	2% CPU	52% Memory	0% Disk	0% Network	0% GPU	Power usage
Google Chrome (17)	0,4%	1.274,3 MB	0,1 MB/s	0 Mbps	0%	Very low
Google Chrome	0,2%	362,0 MB	0,1 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	134,5 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	10,8 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	2,3 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	2,2 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	22,8 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	110,3 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	38,6 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	0,5 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	247,8 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0,1%	34,9 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	2,4 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	209,9 MB	0 MB/s	0 Mbps	0%	Very low
Google Chrome	0%	25,9 MB	0 MB/s	0 Mbps	0%	Very low

Fewer details End task

Criação de Processos

Os Sistemas Operacionais precisam de mecanismos para a **criação de processos**

- Em sistemas muito simples, pode ser possível que todos os processos necessários sejam criados quando o sistema é ligado
 - Geladeira inteligente, Televisão, etc...
- Já em **sistemas de propósito geral** é necessário algum mecanismo para criar e terminar os processos
 - Smartphones, PCs e Notebooks, Tablets, etc...

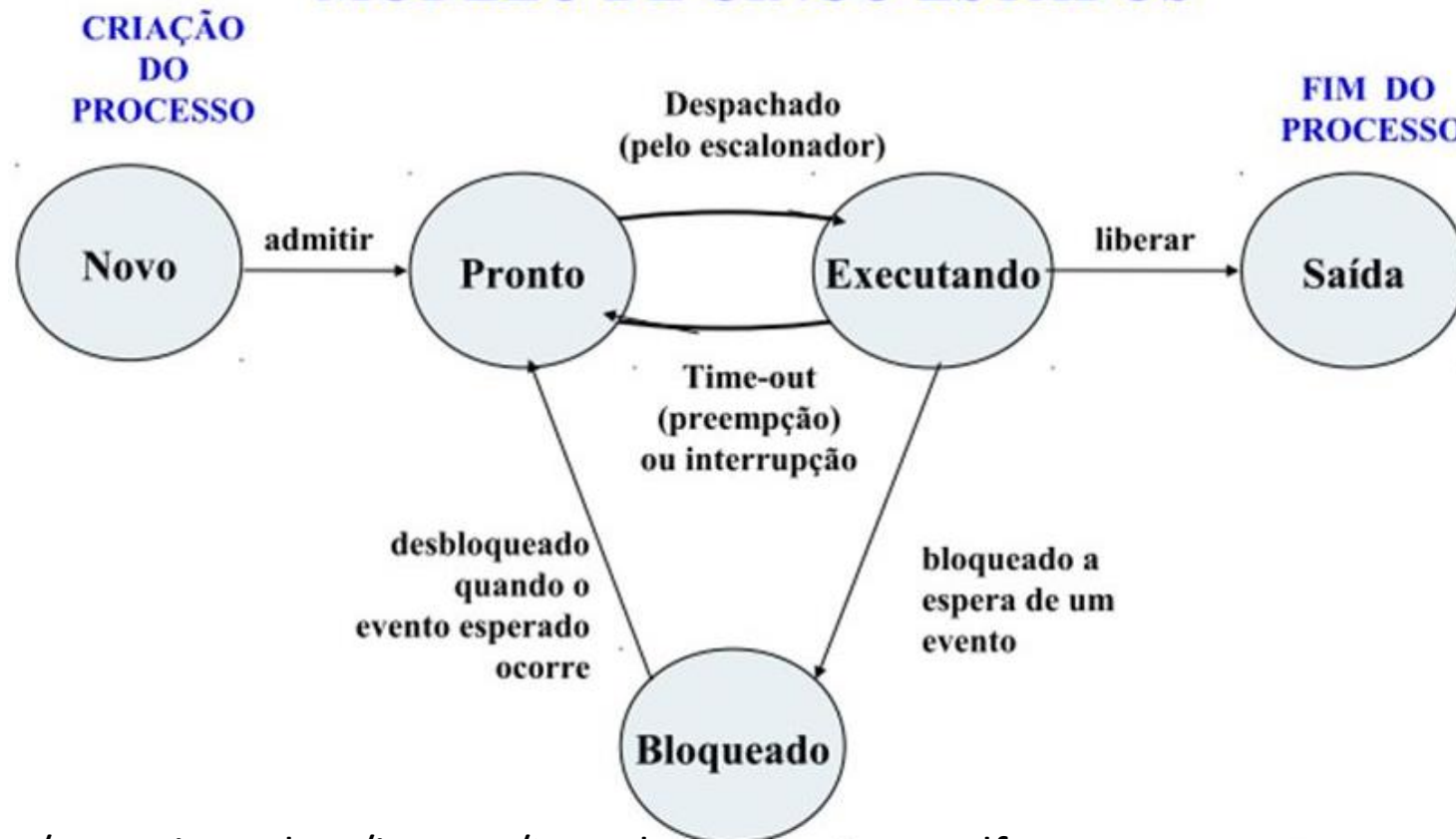
Hierarquia de Processos

Em alguns sistemas, quando um processo cria outro, o processo pai e o processo filho continuam, de certa forma, associados.

O processo que foi criado, por sua vez, pode gerar novos processos, criando assim uma hierarquia de processos.

Estados de Processos

MODELO DE CINCO ESTADOS



Fonte: <http://ctd.ifsp.edu.br/~marcio.andrey/images/Introducao-Processo.pdf>

Estados de Processos

Os três principais estados de um processo (pronto, execução e bloqueado), prove um modelo sistemático para **modelar o comportamento dos processos** como também para ajudar na implementação de processos nos sistemas operacionais.

Porém, estes estados são suficientes?

Quando um processo vai executar, ele deve estar na memória principal (**loadable**).

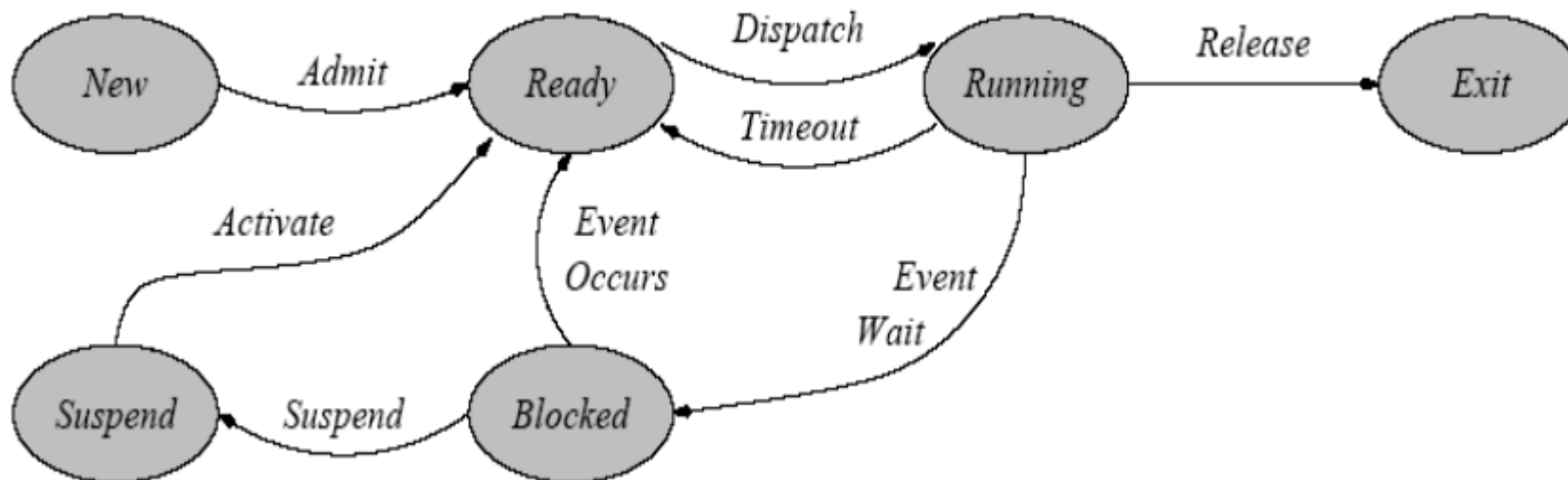
- **O que aconteceria se não houvesse mais espaço disponível na memória principal?**

Fonte: <http://ctd.ifsp.edu.br/~marcio.andrey/images/Introducao-Processo.pdf>

Estados de Processos - Swapping

Uma solução é a adição de um novo estado, o estado **Suspenso**

Quando um processo está bloqueado, ele pode ser movido pelo sistema operacional para o estado suspenso em disco, efetuando assim um **swapping**



Fonte: <http://ctd.ifsp.edu.br/~marcio.andrey/images/Introducao-Processo.pdf>

Estados de Processos - Swapping

Quando o sistema operacional executa esta operação (**swapping out**), ele tem duas escolhas para fazer na hora de inserir o processo na memória principal:

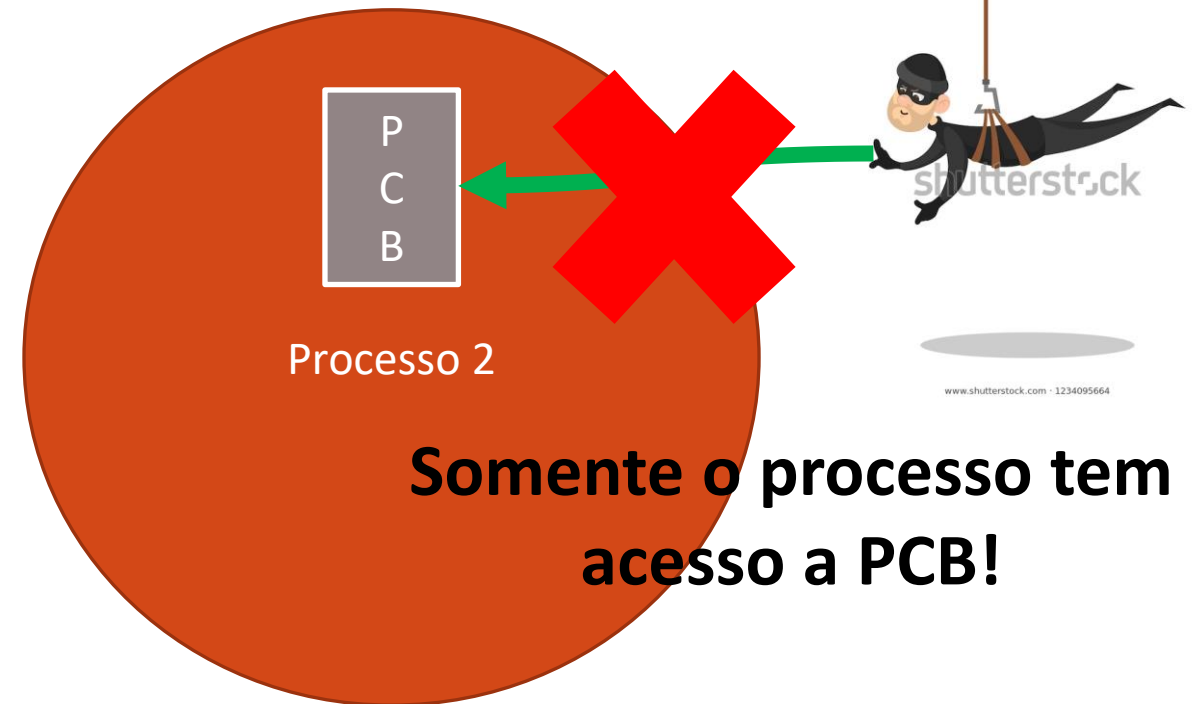
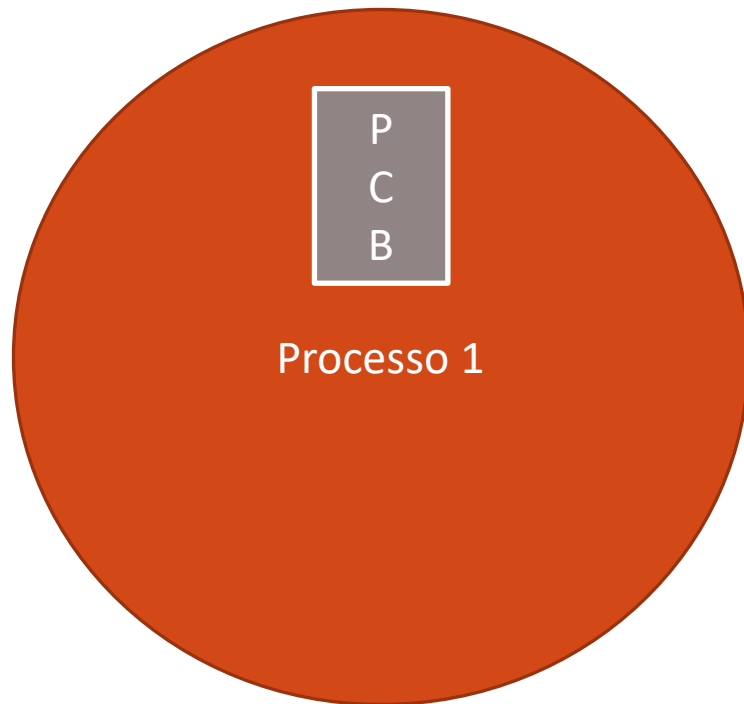
1. O sistema pode admitir um processo novo (que acabou de ser criado)
2. Ele pode carregar um processo previamente suspenso.

Bloco de Controle de Processo (PCB)

Quando um processo é suspenso temporariamente, ele deverá ser **reiniciado** mais tarde, exatamente do mesmo ponto em que estava quando foi interrompido.

Como realizar esse procedimento de retornar exatamente onde ele estava?

Bloco de Controle de Processo (PCB)



Somente o processo tem acesso a PCB!

Todas as **informações relativas ao processo** devem ser salvas na PCB durante a suspensão.

Bloco de Controle de Processo (PCB)

O Sistema Operacional guarda vários pedaços de informações associadas a cada processo, dentre elas, destacam-se:

- Estado do processo
- Contador de programa
- Registradores da CPU
- Informações para escalonamento da CPU
- Informações para gerência de memória
- Informação sobre métricas em geral
- Estados das operações de E/S

Descrição dos Processos

Para gerenciar os recursos, as informações mínimas que o Sistema Operacional deve ter é o estado corrente de cada processo e cada recurso

A abordagem universal para esta questão é: o sistema operacional constrói e mantém **tabelas de informações** sobre cada entidade gerenciada

- Memória
- Dispositivos de E/S
- Arquivos
- Processos

Descrição dos Processos

Tabela de Memória:

- São usadas para guardar informações de ambas as memórias, primária e secundária.
- Uma parte da memória é reservada para o sistema operacional, o restante é disponibilizada para os processos.

A tabela de memória deve guardar as seguintes informações:

- A alocação da memória **principal e secundário** para cada processo
- **Atributos de proteção a blocos de segurança** para a memória principal e secundária, tal que os processos possam acessar somente **certas regiões compartilhada** da memória
- Informações necessária para o **gerenciamento da memória** virtual;

Atributos dos Processos

Em relação aos atributos dos processos, estes podem ser organizados em três **categorias**:

- A **identificação** do processo
- A informação do **estado** do **processador**
- A informação do estado do **processo**

Atributos dos Processos

Identificação do processo:

- O identificador do processo
- O identificador do processo que o criou (processo pai)
- O identificador do usuário que criou o processo

Atributos dos Processos

Identificação do estado do processador:

- Consiste nos conteúdos dos registradores do processo , tal como contador de programa (PC), apontadores de pilha, etc.

Identificação do estado dos processos:

- Estas são as informações necessárias para o sistema operacional controlar e coordenar os vários processos que estão em execução.

Tipos de Processo

- Há dois **tipos** principais de processo baseado onde ele gasta mais tempo executando.
 - O processo pode ter mais operações que necessitam da **CPU** ou da **Entrada e Saída (E/S)** de dados.
-
- **CPU-bound** (orientado a CPU) ou
 - **E/S-bound** (orientado a E/S)

Tipos de Processo

Processos CPU-bound (orientados à CPU):

- Fazem muito uso da CPU
- O tempo de execução é definido pelos ciclos de processador usados
- **Exemplo:** jogos eletrônicos, cálculos matemáticos, **edição de vídeo aulas**, dentre outros.

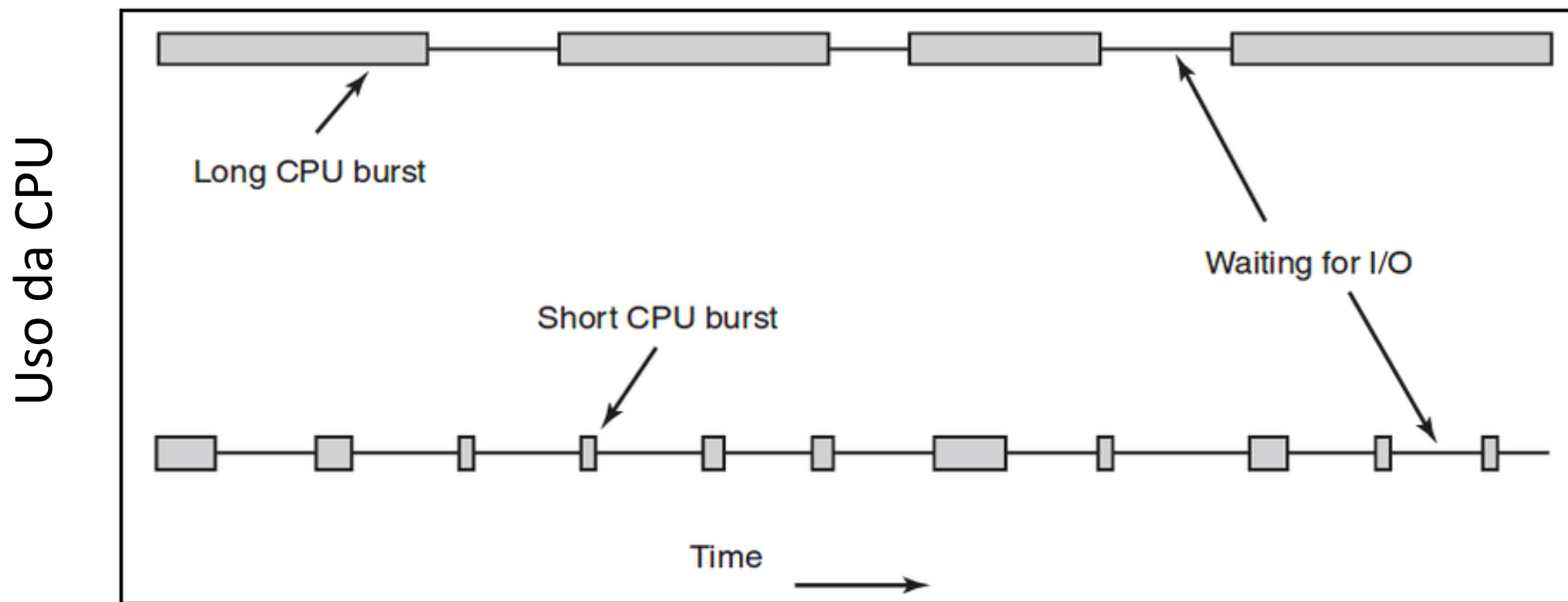
Tipos de Processo

O desempenho de muitos computadores atualmente está restrito a velocidade do disco (E/S-bound), possuindo um processador e memória muito rápida, mas um armazenamento lento.

Processos E/S-bound (orientados à E/S):

- Fazem pouco uso da CPU
- Realizam **muitas operações de entrada e saída** de dados
- O tempo de execução é definido pela duração das operações de E/S
- **Exemplo:** cópia de arquivos, leitura de vídeos, contar linhas de um programa ou arquivo, dentre outros.

CPU-bound vs E/S-bound



THREADS



www.shutterstock.com · 659858677

THREADS

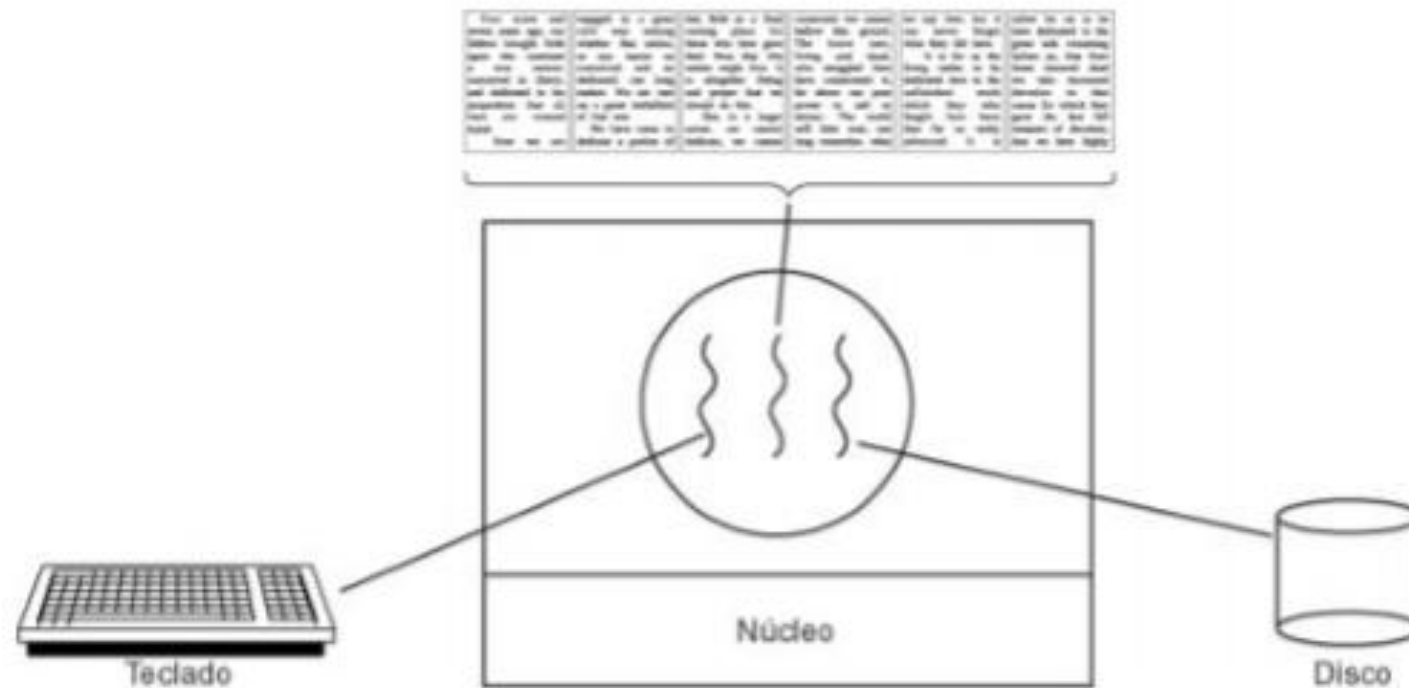
- Fluxo de execução
 - 2 threads = 2 fluxos de execução independentes
- Executam dentro de um processo
 - Fluxos de execução independentes dentro do processo!
- Permite compartilhar os recursos do processo
 - Entre as threads!
- Criação de threads é mais leve que de processos
 - Usa menos recursos e a instanciação é menos complexa e mais rápida

O conceito de Thread

“Uma thread é uma unidade básica de uso de CPU. Cada thread é composta por um **identificador**, um **contador de programa**, um **conjunto de registradores** e uma **pilha**. Eles **compartilham** com as demais threads o código-fonte, os dados e outros **recursos** de alto nível atribuídos ao processo dono das threads em questão.”

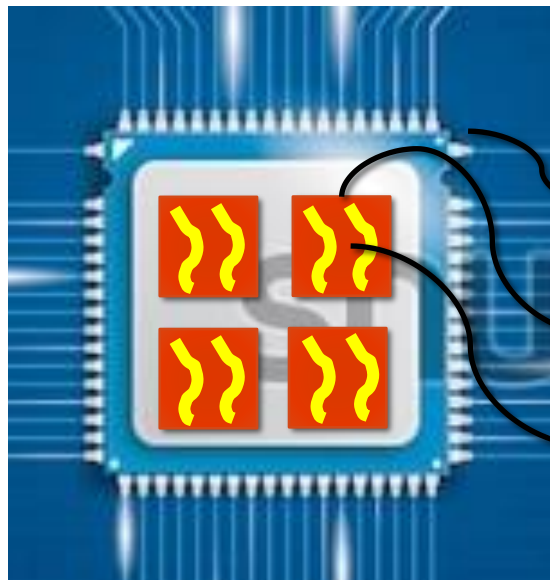
SILBERSCHARTZ, P. B. GALVIN, G. GAGNE (2009)

Exemplo: programa editor de texto



THREADS

- Cuidado para não confundir com thread de processador!



- 1 processador
- 4 núcleos (cores)
- 8 threads de PROCESSADOR



Exemplo: servidor web



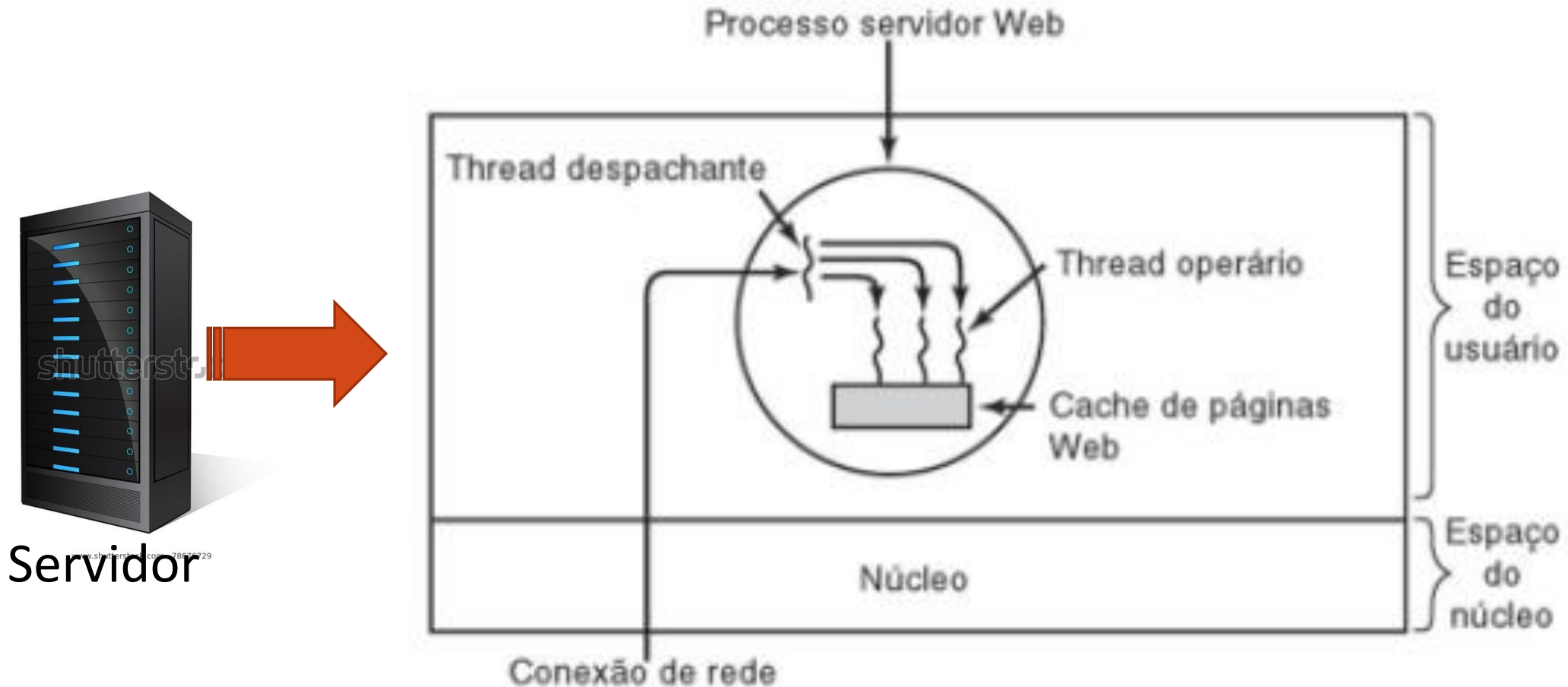
www.shutterstock.com · 78676729

Servidor

Clientes



Exemplo: servidor web

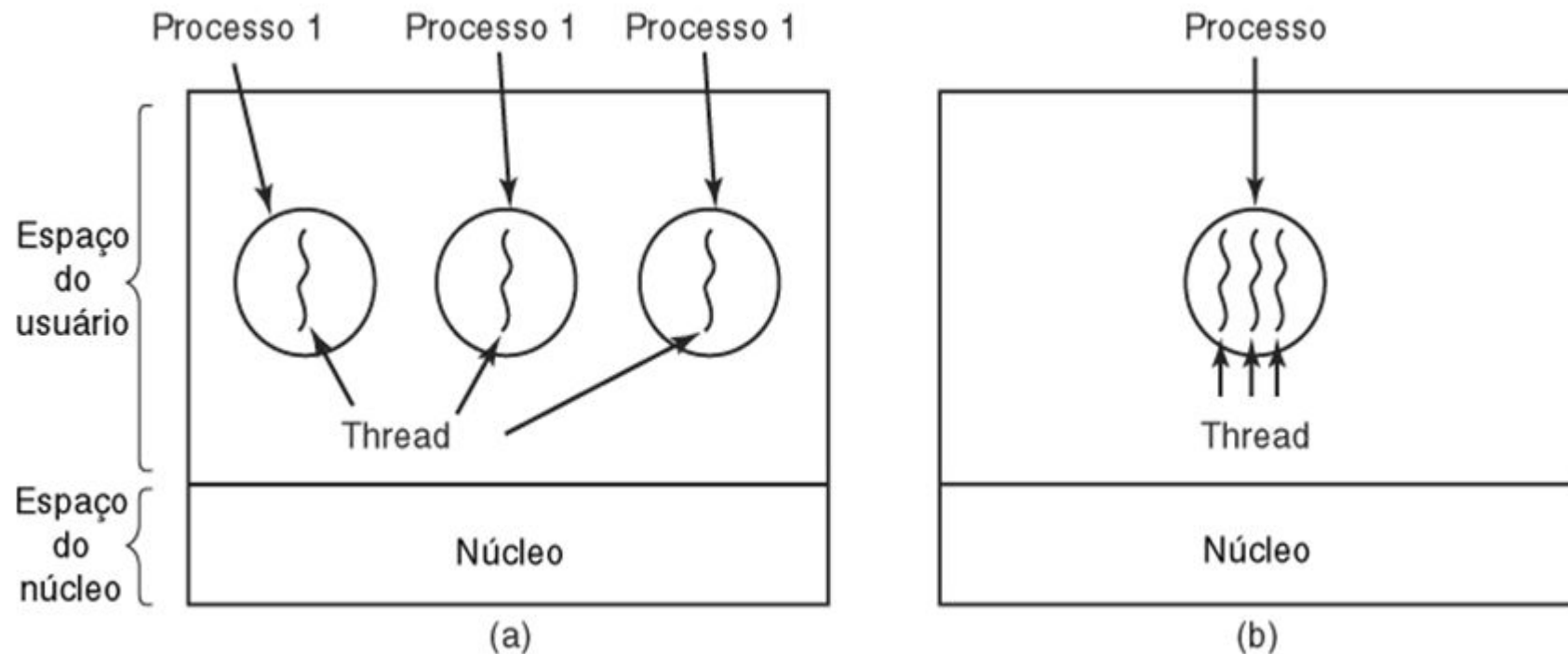


O conceito de Thread

- Vale ressaltar que, quanto ao desempenho, o uso de threads não resulta em ganho de desempenho quando todos eles são CPU-bound.
- No entanto, quando há grande quantidade de computação e de E/S, os threads permitem que essas atividades se sobreponham e, deste modo, acelerem a aplicação

O modelo de thread clássico: multithread

O termo multithread é usado para descrever a situação em que se permite a existência de múltiplos threads



O modelo de thread clássico

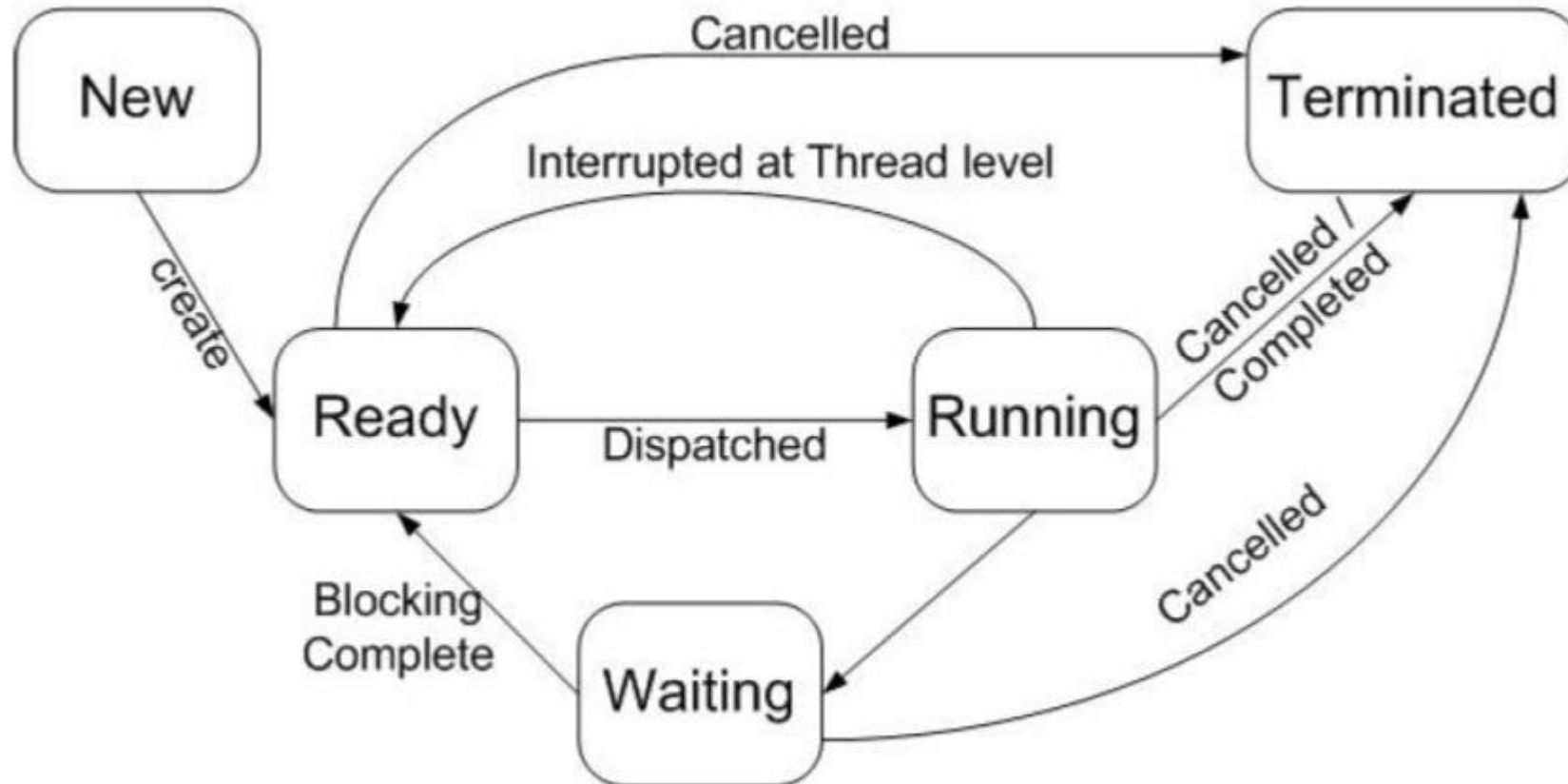
- Se uma thread abre um arquivo, este fica visível para quem?
 - Para todos os outros threads no processo
 - Eles podem ler e escrever nele
- Se cada thread tivesse seu próprio espaço de endereçamento, arquivos abertos, alarmes, etc?
 - Eles seriam processos separado

O modelo de thread clássico

Então, por que usar thread?

- O que deseja-se conseguir com o conceito de thread é a capacidade de compartilhar um conjunto de recursos de forma que eles possam cooperar na realização de uma tarefa

Máquina de estados dos threads



Programação com threads: Exemplo

Em Java, é possível criar thread de duas formas:

- NomeClasse **extends** Thread
- NomeClasse **implements** Runnable
- Ambas as abordagens irão implementar o método run()
 - Sobrescrita do run() usando Runnable é obrigatório
 - Sobrescrita do run() usando Extends é opcional, mas o run() da superclasse Thread não faz nada!

Programação com threads: Exemplo

```
public class SimpleThreadUsingExtends extends Thread{

    public void run()
    {
        for(int i=0; i<5; i++) {
            System.out.println("Run method executed
by child Thread");
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Programação com threads: Exemplo

```
public class SimpleThreadUsingImplements extends  
ConsolePrinter implements Runnable{  
  
    @Override  
    public void run() {  
        System.out.println("Run method executed by child Thread");  
    }  
}
```


Programação com threads: Exemplo completo

```
public class SimpleThreadUsingExtends extends Thread{

    public void run()
    {
        for(int i=0; i<5; i++) {
            System.out.println("Run method executed by child
Thread");
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace(); } } }

    public static void main(String[] args)
    {
        SimpleThreadUsingExtends t = new
SimpleThreadUsingExtends();
        t.start();
        for(int i=0; i<5; i++) {
            System.out.println("Main method executed by main
thread");
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            } } } }
```

Programação com threads: Exemplo

```
public class SimpleThreadUsingImplements extends ConsolePrinter implements Runnable{

    @Override
    public void run() {
        System.out.println("Run method executed by child Thread");
    }

    public void m1()
    {
        System.out.println("Hello Visitors");
    }

    public static void main(String[] args) {
        SimpleThreadUsingImplements t = new SimpleThreadUsingImplements();
        t.m1();
        Thread t1 = new Thread(t);
        t1.start();

        System.out.println("Main method executed by main thread");
        t.writeConsole("Escreva na tela...");
    } }
```

Programação com threads: Exemplo

```
package threads;

public class ConsolePrinter {

    public void writeConsole(String string)
    {
        System.out.println("Console Printer: " + string);
    }
}
```

THREADS – PARTE 2



www.shutterstock.com · 659858677



www.shutterstock.com · 794782192

Implementação de Threads

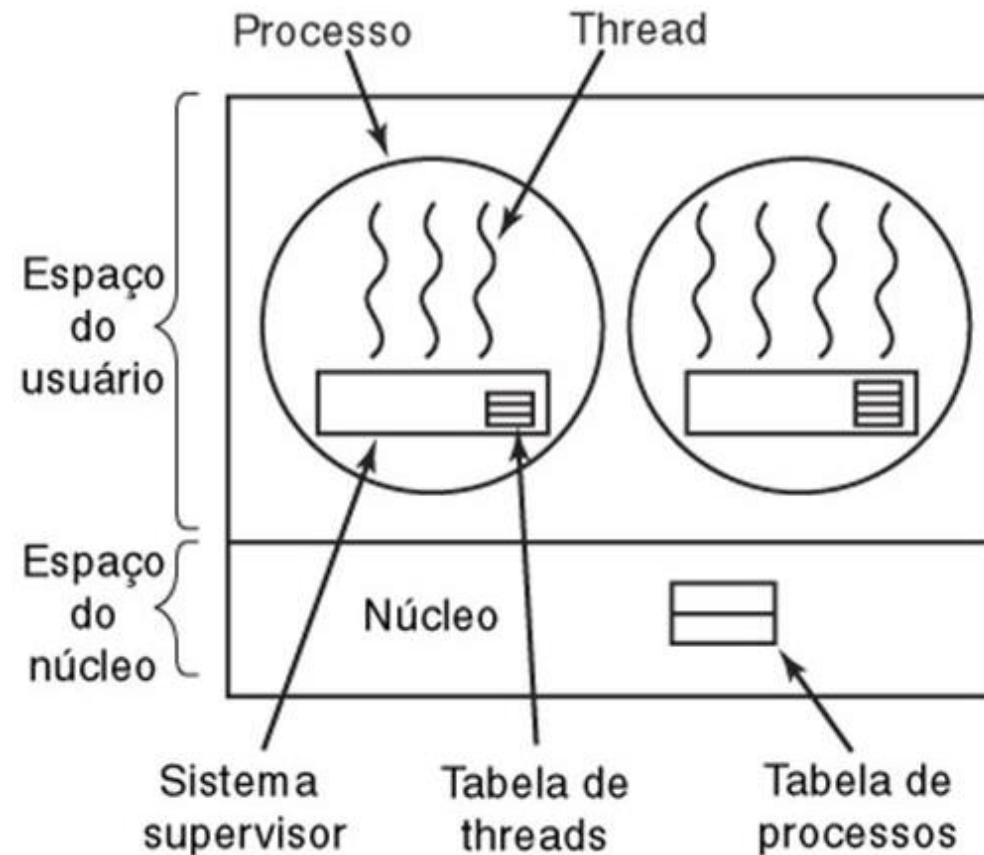
Há dois modos principais de implementar um pacote de threads:

- No **espaço do usuário**
- No modo **núcleo**

Implementação de Threads no Espaço do Usuário

- Insere-se todo o pacote de threads dentro do espaço do usuário
 - O núcleo não é informado sobre esses pacotes
 - O que compete ao núcleo é o gerenciamento comum dos processos monothreads
- Vantagem:
 - Um pacote de thread de usuários pode ser implementado em um sistema operacional que não suporta thread
 - Com essa abordagem, os threads são implementados por uma biblioteca dentro do programa do usuário

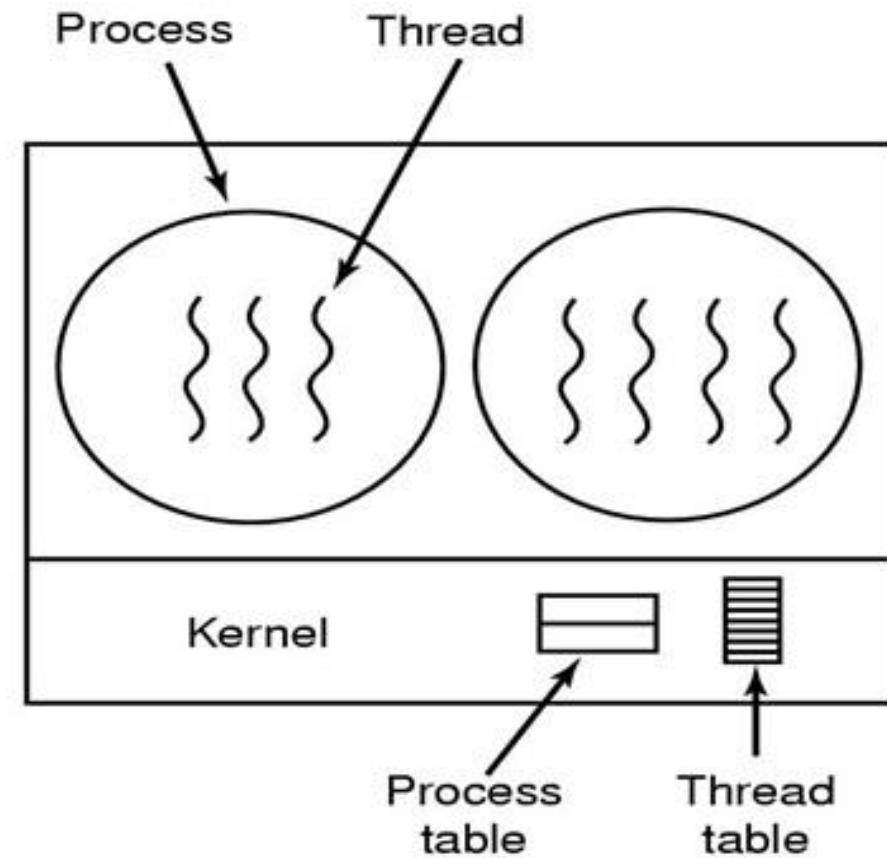
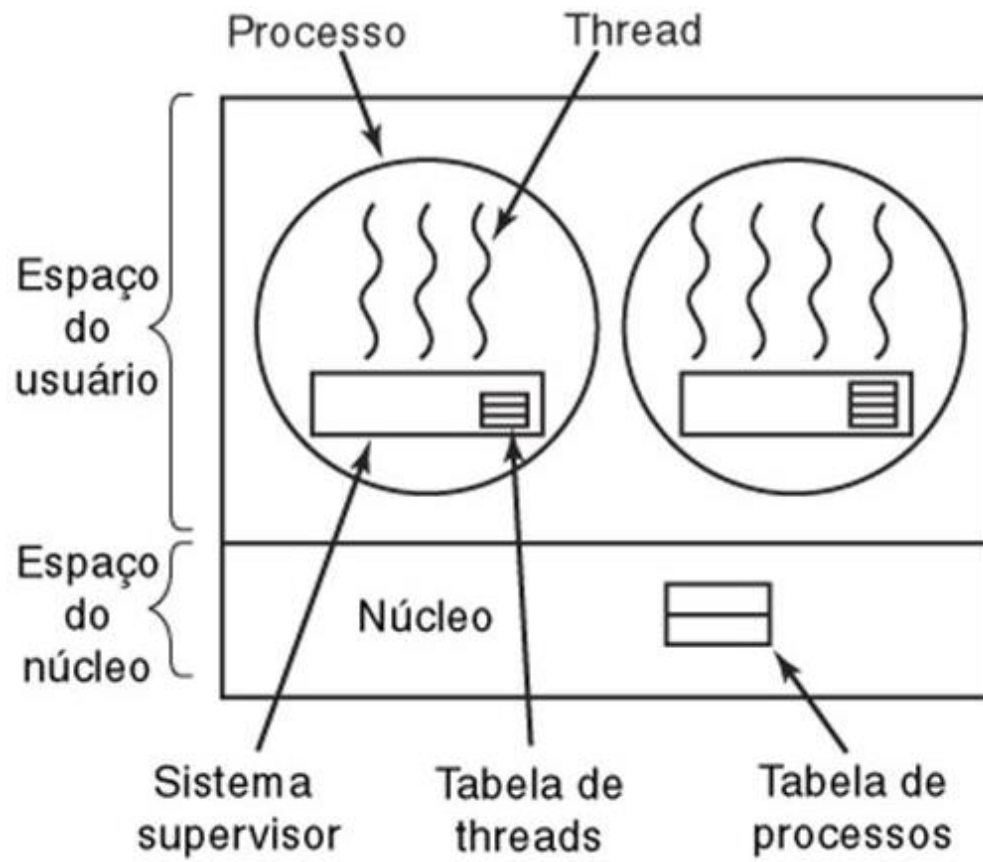
Implementação de Threads no Espaço do Usuário



Implementação de Threads no Espaço do Usuário

- Problemas com pacotes de threads de usuário:
 - O controle de execução das threads é feito pelo programa, se ele não realizar o chaveamento de forma devida, algumas threads podem nunca executar!
 - O thread deve ceder voluntariamente a vez para o outro, gerando problemas de gerenciamento e sincronismo se feita de forma indevida.
 - Aumento da complexidade de gerenciamento pelo programador do aplicativo do usuário.

Implementação de Threads no Núcleo



Implementação de Threads no Núcleo

- Problemas com pacotes de threads no núcleo:
 - O controle de execução das threads é feito pelo sistema operacional, ele pode interromper a execução de qualquer thread ou processo, impedindo que o programa execute.
 - Aumento da complexidade de gerenciamento pelo sistema operacional.
 - Grande tabela com todos os processos e threads que estão sendo executados, pode tornar o sistema operacional lento.

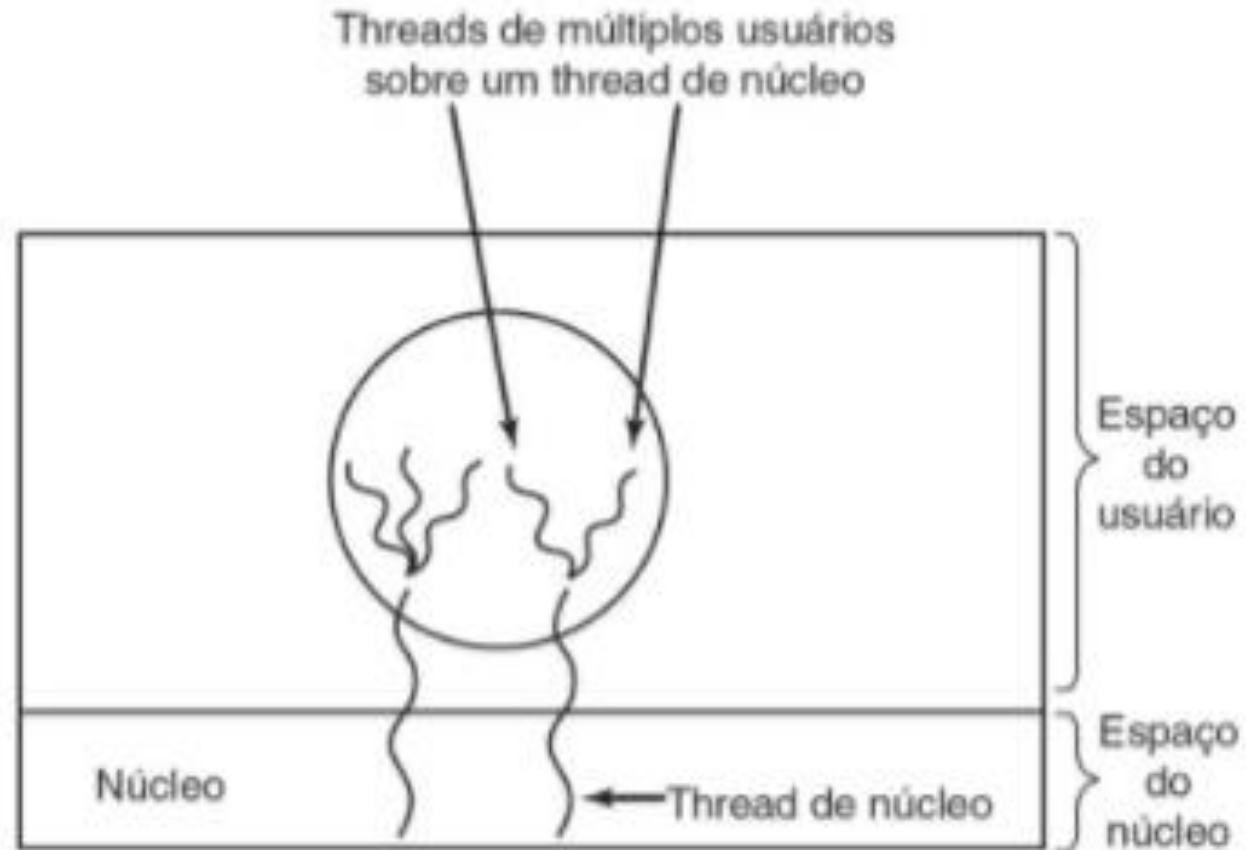
Implementação Híbrida

- Permite extrair alguns **benefícios** da abordagem de threads de usuário e threads de núcleo.
- É possível reduzir a complexidade no sistema operacional ao usar threads de núcleo (um dos principais problemas)
 - Solução: usar menos threads de núcleo e multiplexar (relacionar 1 pra muitos) threads de usuários sobre algum ou todos os threads

Implementação Híbrida

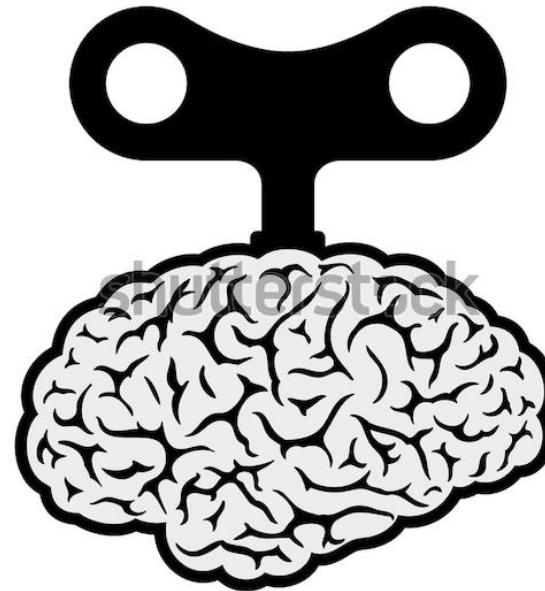
- O sistema operacional conhece apenas as threads de núcleo, reduzindo a quantidade de informação e complexidade do sistema operacional.
- Um thread de núcleo pode referenciar diversos threads de usuário
- Os threads são então gerenciados pelo sistema operacional e também mantém a independência dos threads de usuário!

Implementação híbridas



Obrigado pessoal!

- Se tiver dúvidas, veja o vídeo novamente ou pergunte!
- Até a próxima!



www.shutterstock.com · 160202087



www.shutterstock.com · 114717280