

INFRA-ESTRUTURA DE SOFTWARE

Introdução

Carlos Ferraz
<cagf@cin.ufpe.br>

Para que serve?

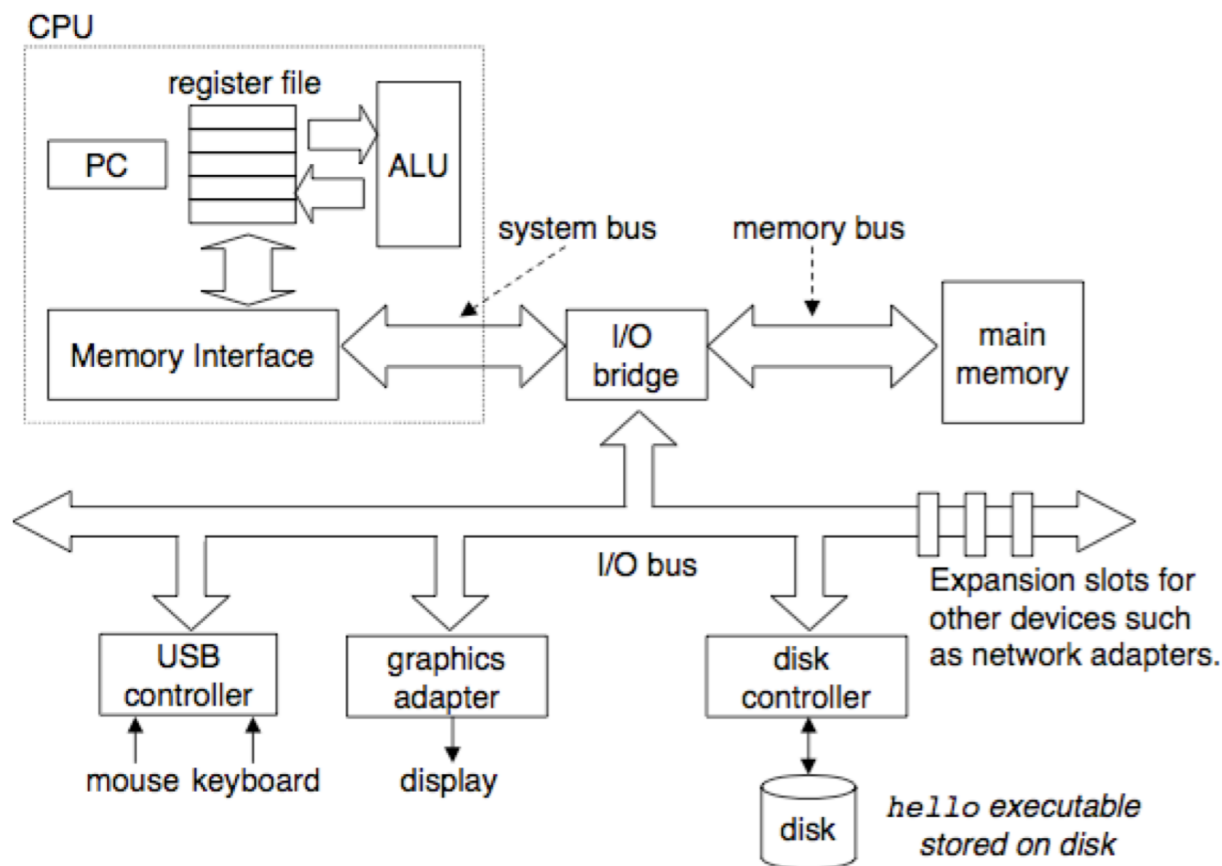
Abstrair do **hardware**



Duro, **difícil!**....

Abstração - guarde esta palavra!

Conteúdo do Curso

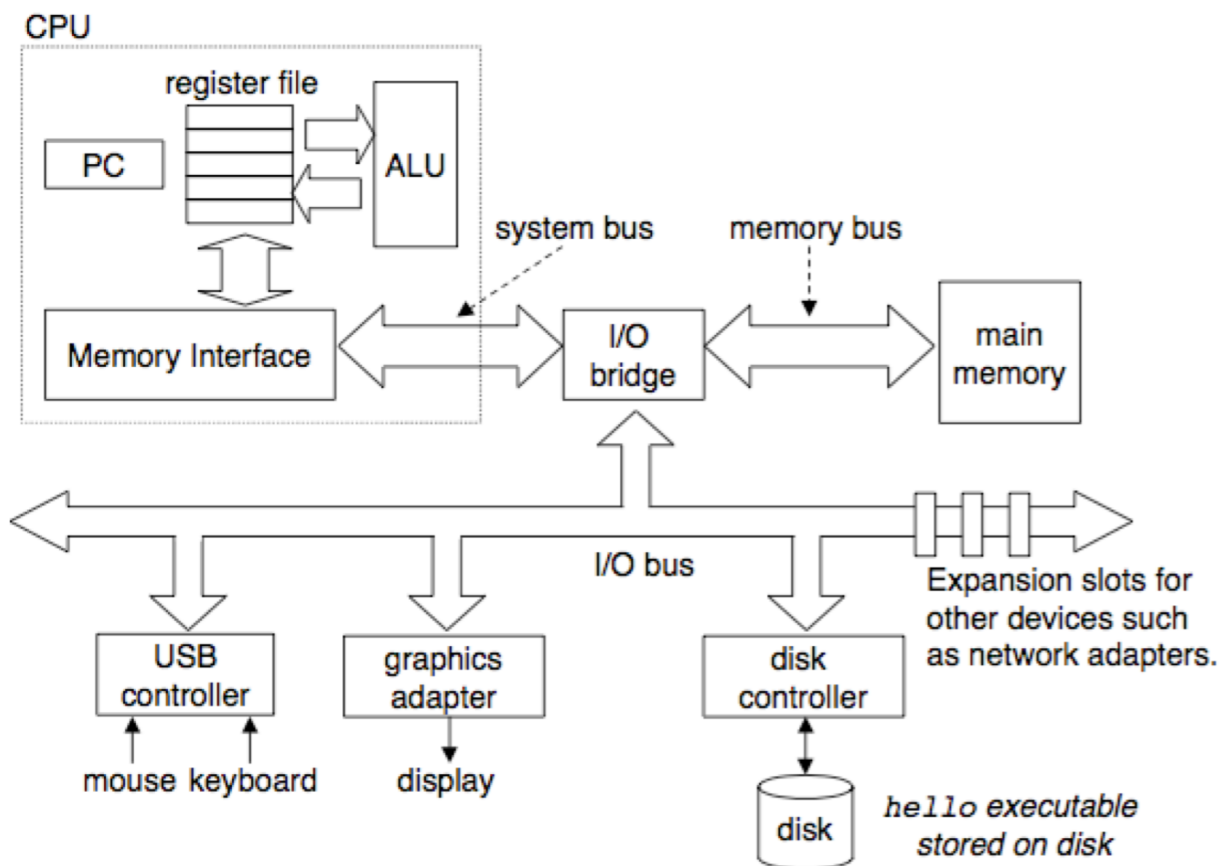


- Gerência de Processos
- Gerência de Memória
- Sistema de Arquivos
- Gerência de Entrada/Saída
- Sistemas Distribuídos

Perguntas

1. Quem decide qual programa vai rodar na CPU?

2. Quem gerencia informação em disco?



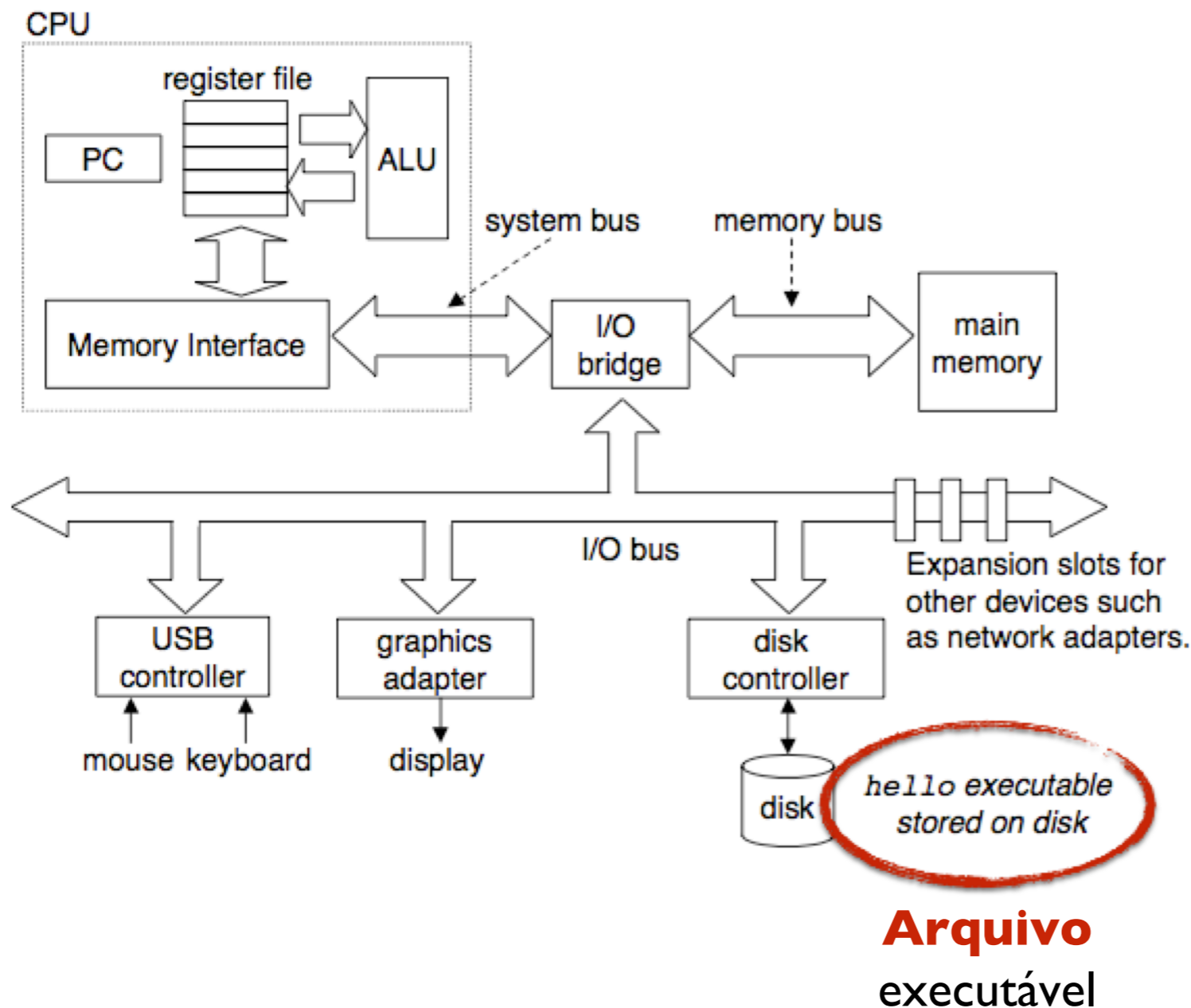
Aplicação

Sistema Operacional

Computador

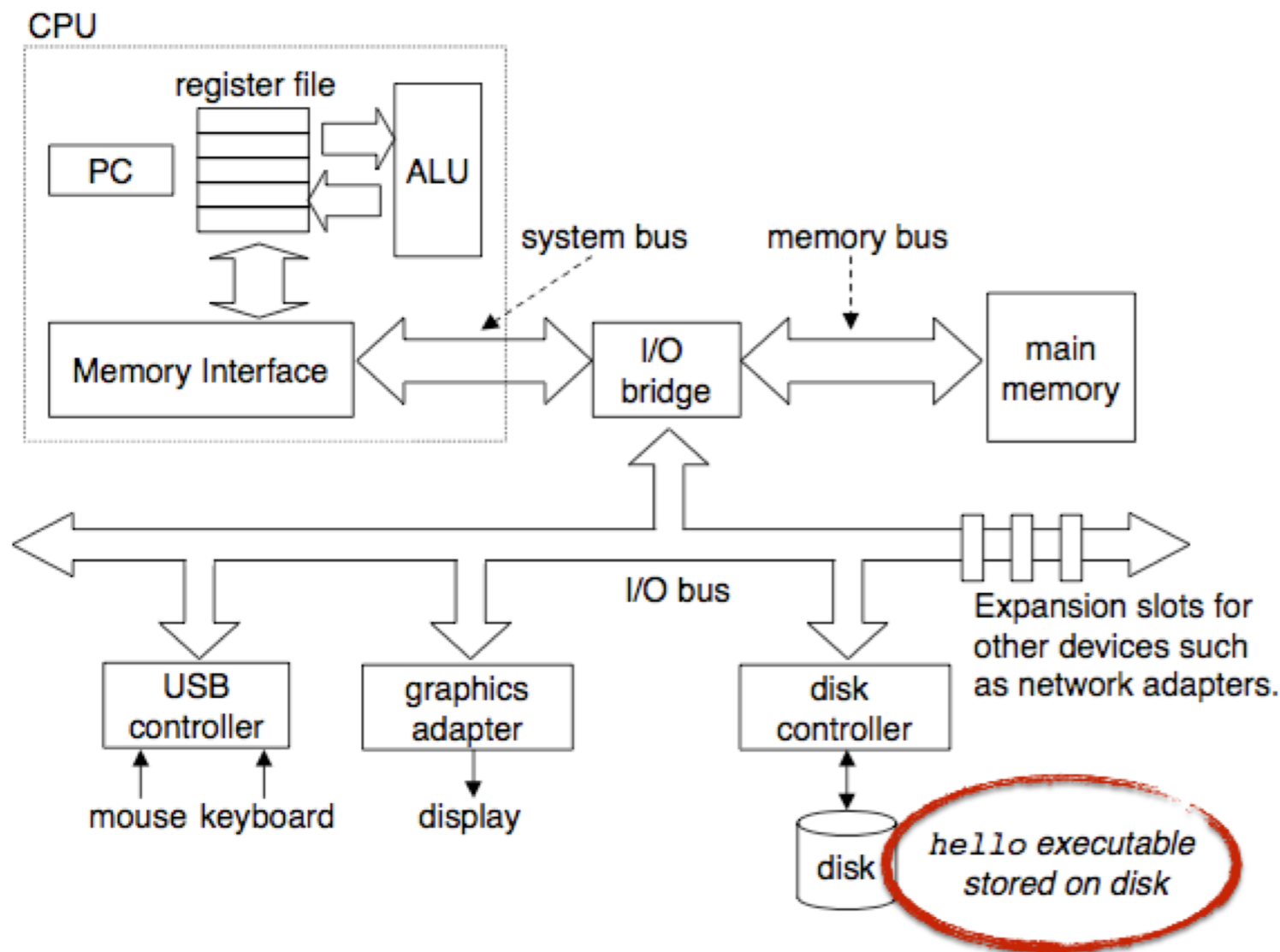
Rede

Conteúdo



- Gerência de Processos
- Gerência de Memória
- Sistema de Arquivos
- Gerência de Entrada/Saída
- Sistemas Distribuídos

Conteúdo

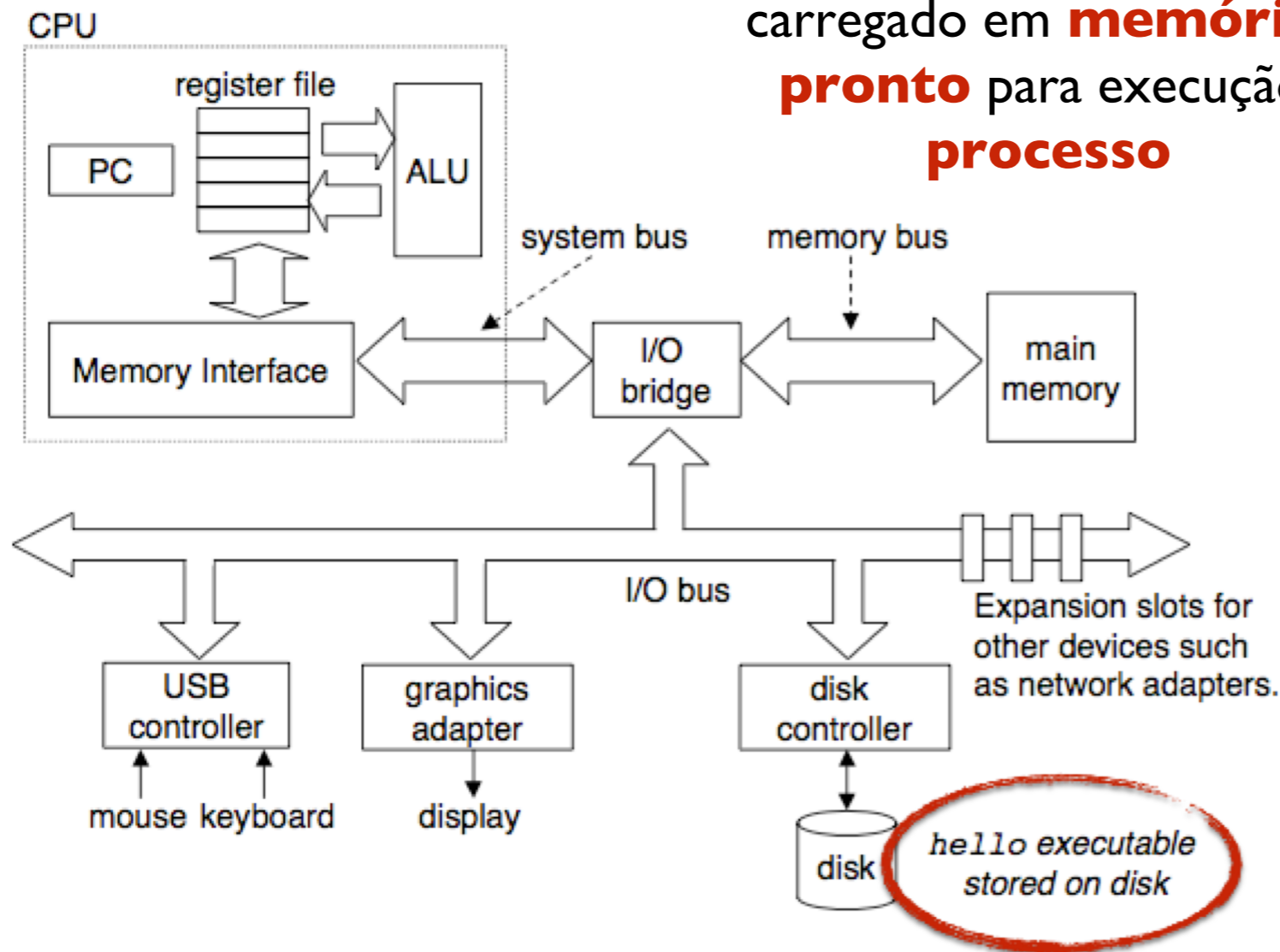


Programa
executável
gravado em disco

- Gerência de Processos
- Gerência de Memória
- Sistema de Arquivos
- Gerência de Entrada/Saída
- Sistemas Distribuídos

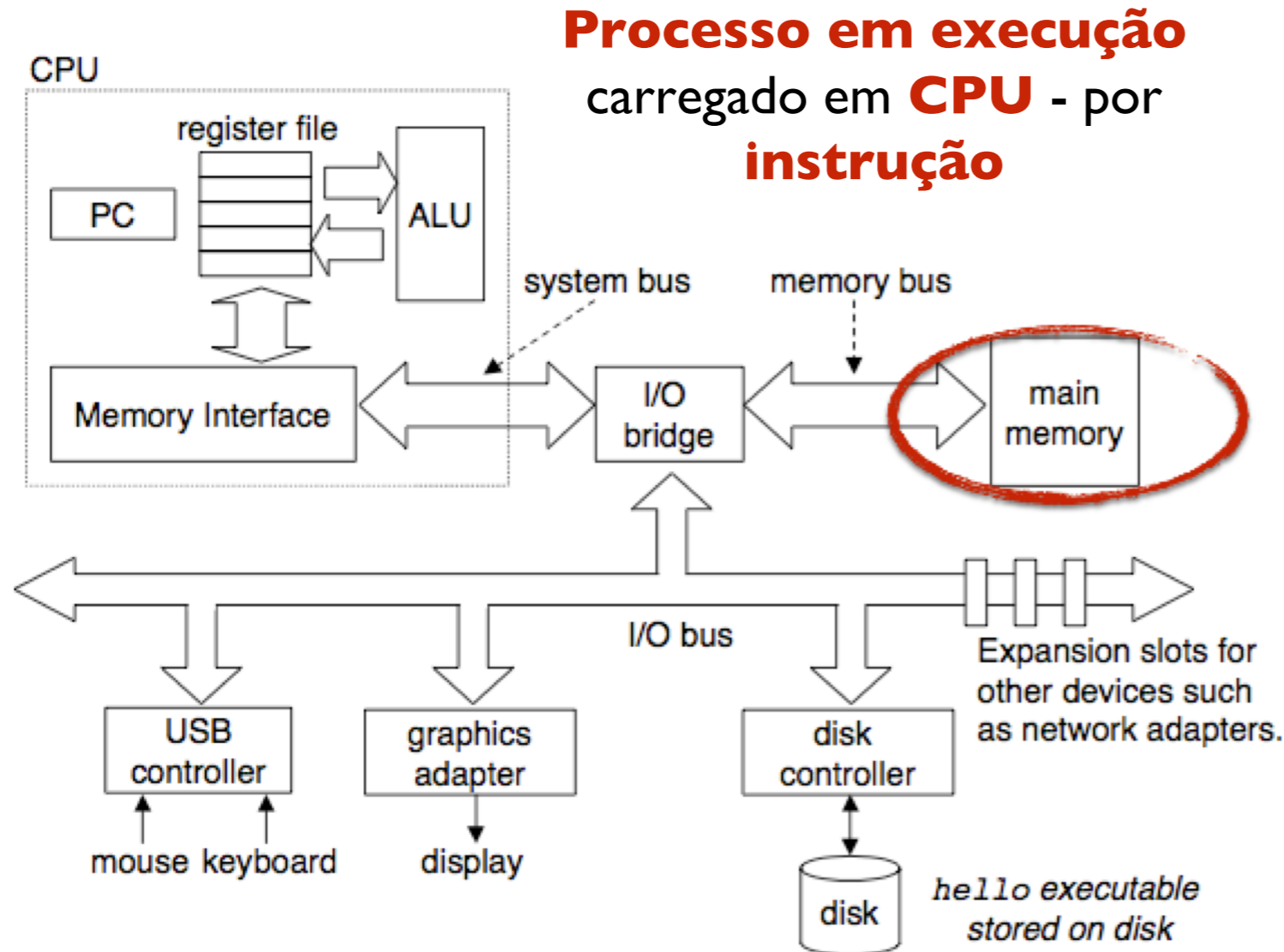
Conteúdo

Programa executável
carregado em **memória** -
pronto para execução:
processo



- Gerência de Processos
- Gerência de Memória
- Sistema de Arquivos
- Gerência de Entrada/Saída
- Sistemas Distribuídos

Conteúdo



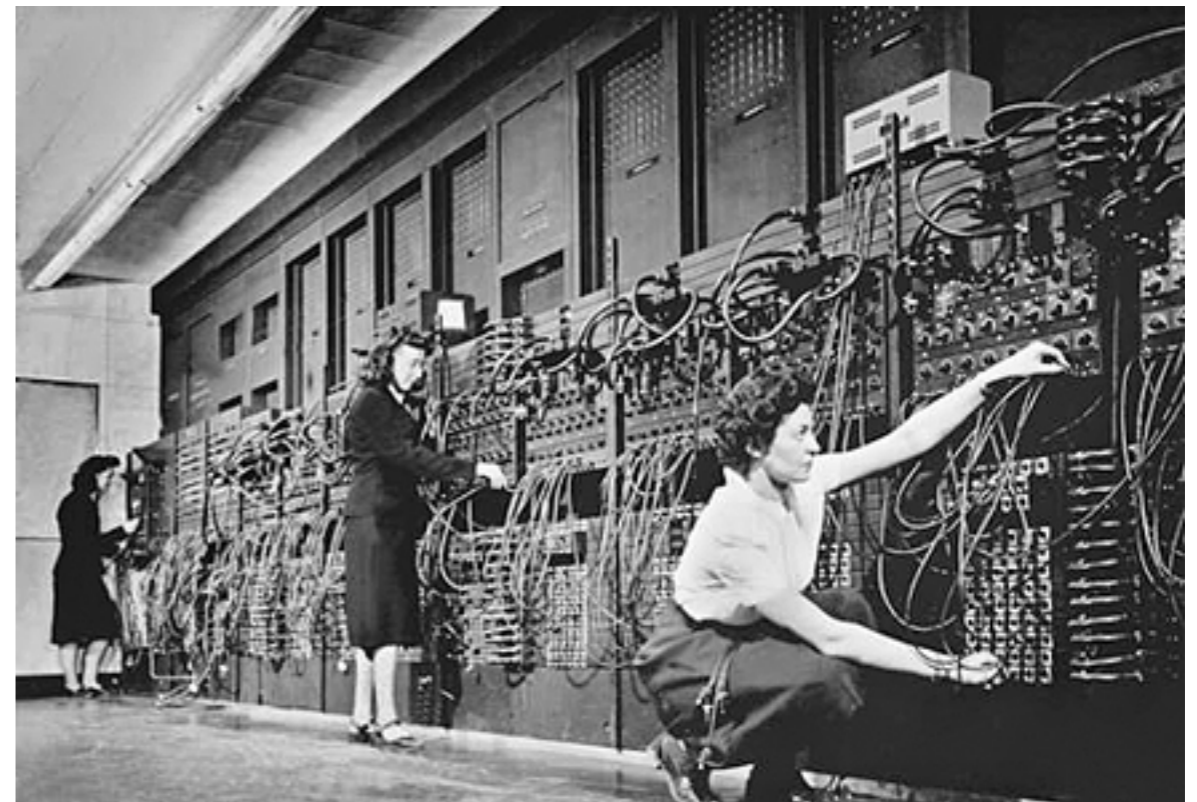
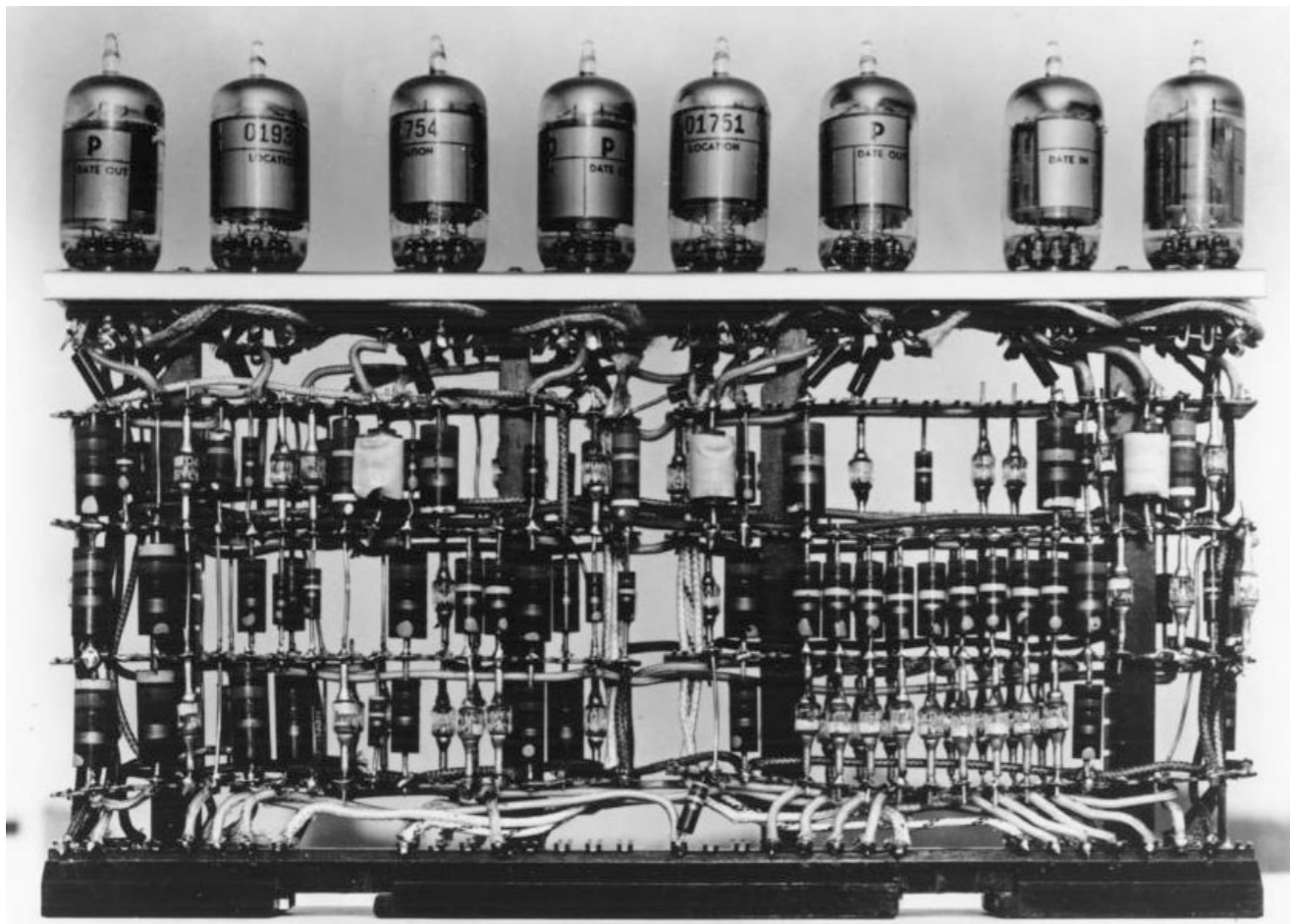
- Gerência de Processos
- Gerência de Memória
- Sistema de Arquivos
- Gerência de Entrada/Saída
- Sistemas Distribuídos

Introdução

UM POUCO DE HISTÓRIA...

História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação



História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - **transistores**, sistemas em lote

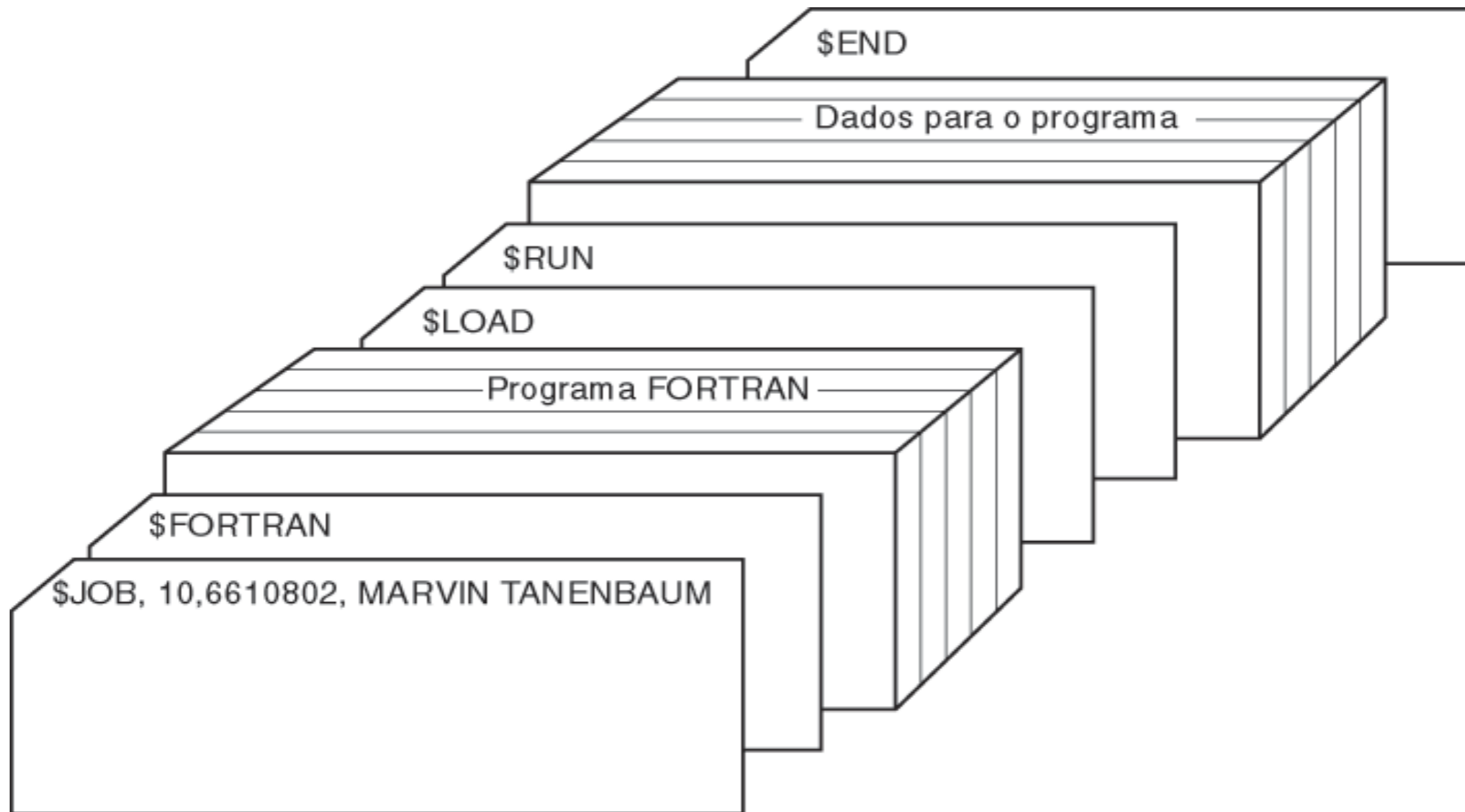
Second Generation - 1956-1963: Transistors

Transistors replaced vacuum tubes and ushered in the second generation of computers. The transistor was invented in 1947 but did not see widespread use in computers until the late 50s.

*smaller, faster and cheaper.



Sistemas em lote



- Estrutura de um job típico (**lote** de cartões) – 2a. geração

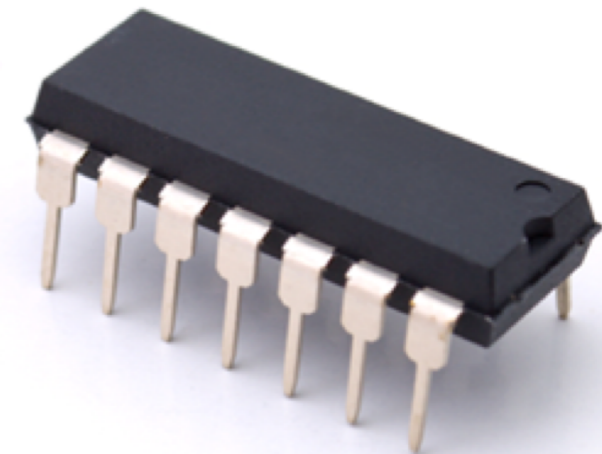
História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - transistores, sistemas em lote
- Terceira geração: 1965 – 1980
 - **ICs (circuitos integrados)** e multiprogramação



Vacuum tubes: slow, expensive, fragile

Transistors: much simpler, much smaller, much cheaper, more reliable, no warm up, much faster.

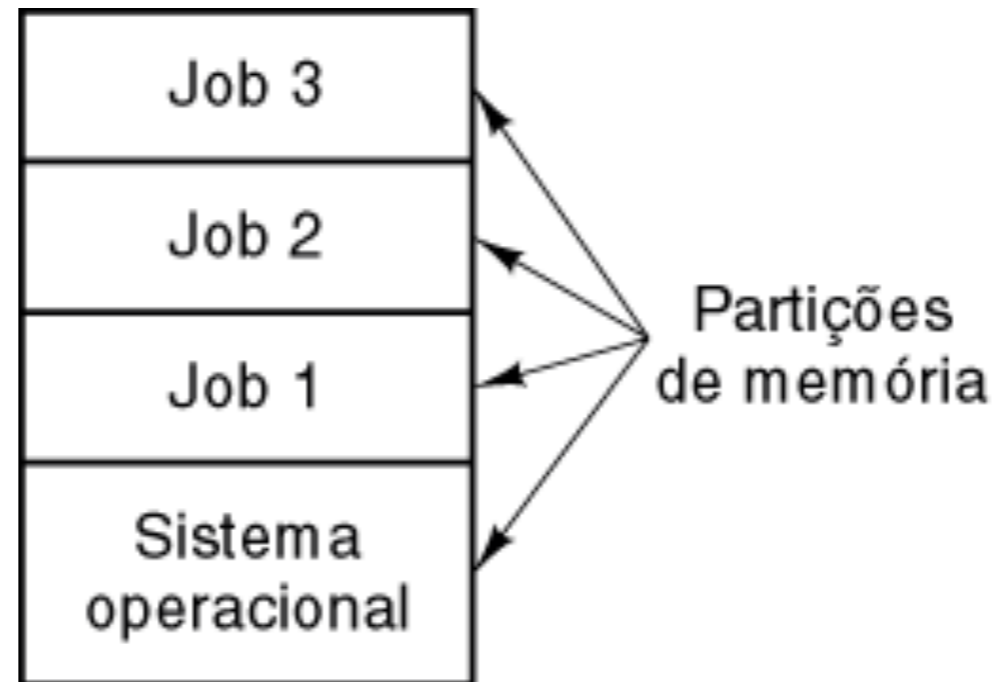


Integrated circuits: miniaturization added to all the existing benefits, enabled unthought-of possibilities

História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - transistores, sistemas em lote
- Terceira geração: 1965 – 1980
 - CIs (circuitos integrados) e **multiprogramação**

Multiprogramação



Três jobs na memória – 3a. geração

História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - transistores, sistemas em lote
- Terceira geração: 1965 – 1980
 - CIs (circuitos integrados) e multiprogramação
- Quarta geração: 1980 – presente
 - **Computadores pessoais**
- Hoje: onipresença – **computação ubíqua**

Introdução

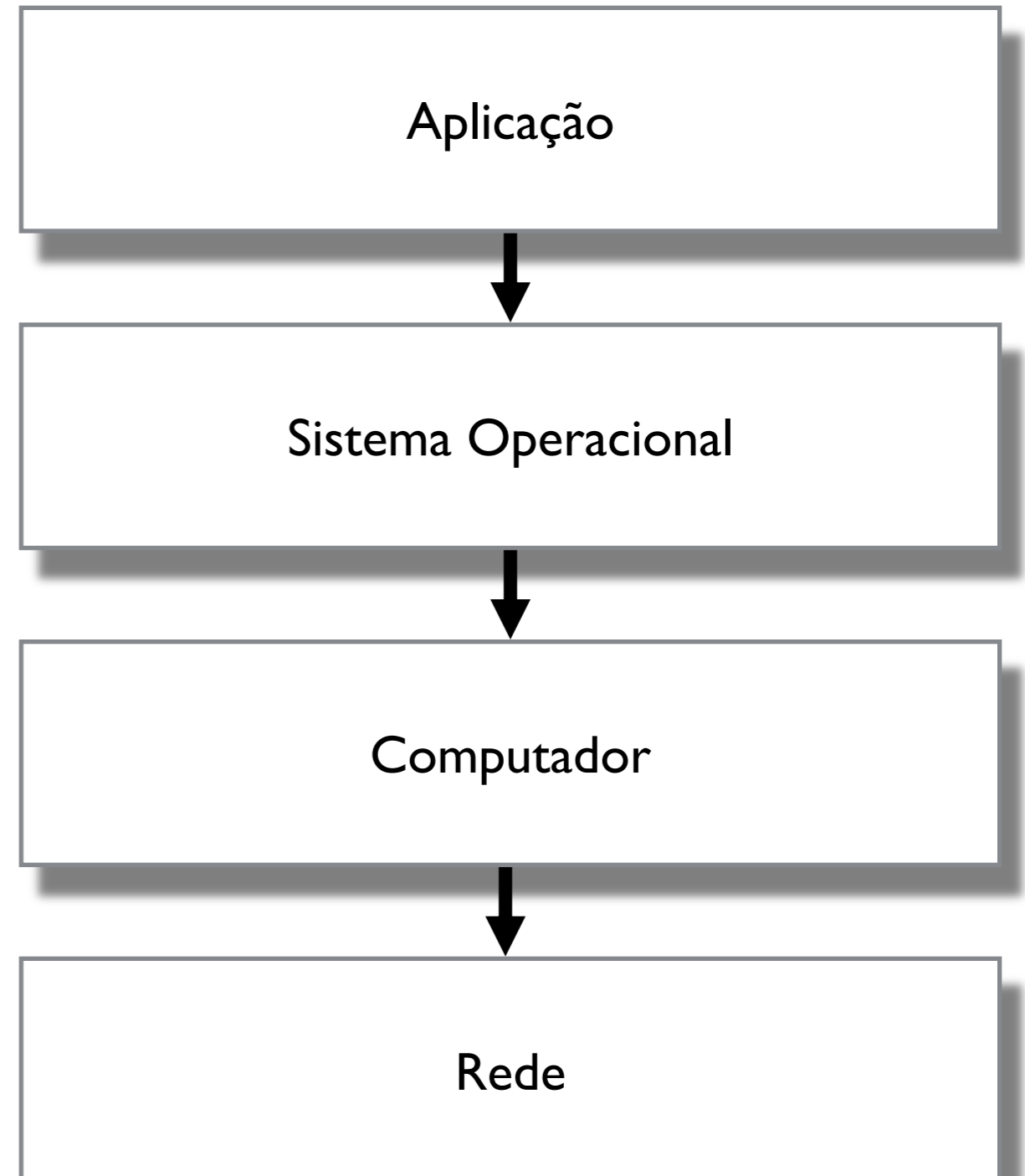
VISÃO GERAL

Tópicos

- Sistema Operacional
- Modos de Operação
- Um pouco de hardware (CPU/registadores, Memória/hierarquia ...)
- Software Básico: gerando um executável
- Estruturas de Sistemas Operacionais

Sistema Operacional

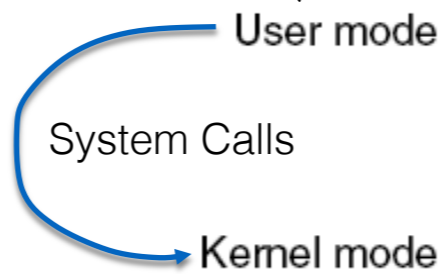
- Máquina abstrata
- Gerenciador de recursos



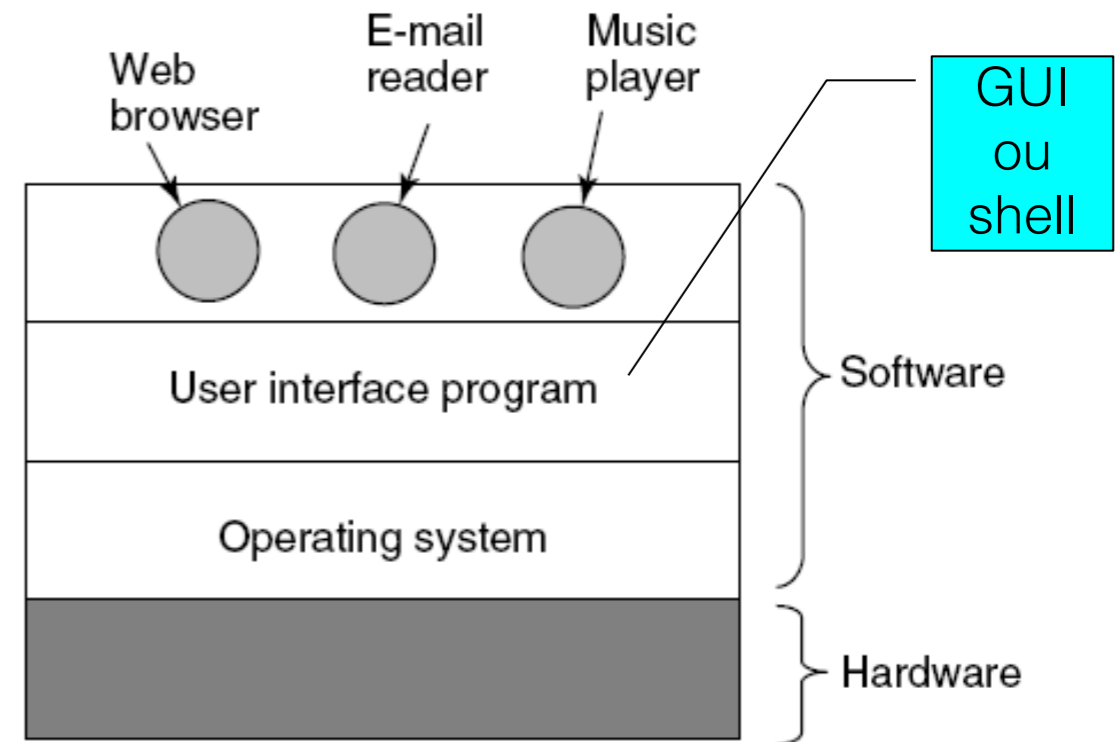
Modos de Operação

- Modo Usuário
- Modo Núcleo

Não pode executar instruções que afetam o controle da máquina ou fazem E/S



Acesso completo a todo o hardware e pode executar qualquer instrução que a máquina seja capaz de executar



Diversidade de Sistemas Operacionais

- Sistemas operacionais de **computadores de grande porte** (*mainframe*)
- Sistemas operacionais de servidores / **redes**
- Sistemas operacionais de **multiprocessadores** (paralelismo)
- Sistemas operacionais de computadores pessoais
- Sistemas operacionais de dispositivos portáteis/ **móveis** (ex. celulares)
- Sistemas operacionais de **tempo-real**
- Sistemas operacionais **embarcados**
- Sistemas operacionais de cartões inteligentes
- Sistemas operacionais de sensores

Tópicos

- Sistema Operacional
 - diversidade
- Modos de Operação
- Um pouco de hardware (CPU/registadores, Memória/hierarquia ...)
- Software Básico: gerando um executável
- Estruturas de Sistemas Operacionais

CPU

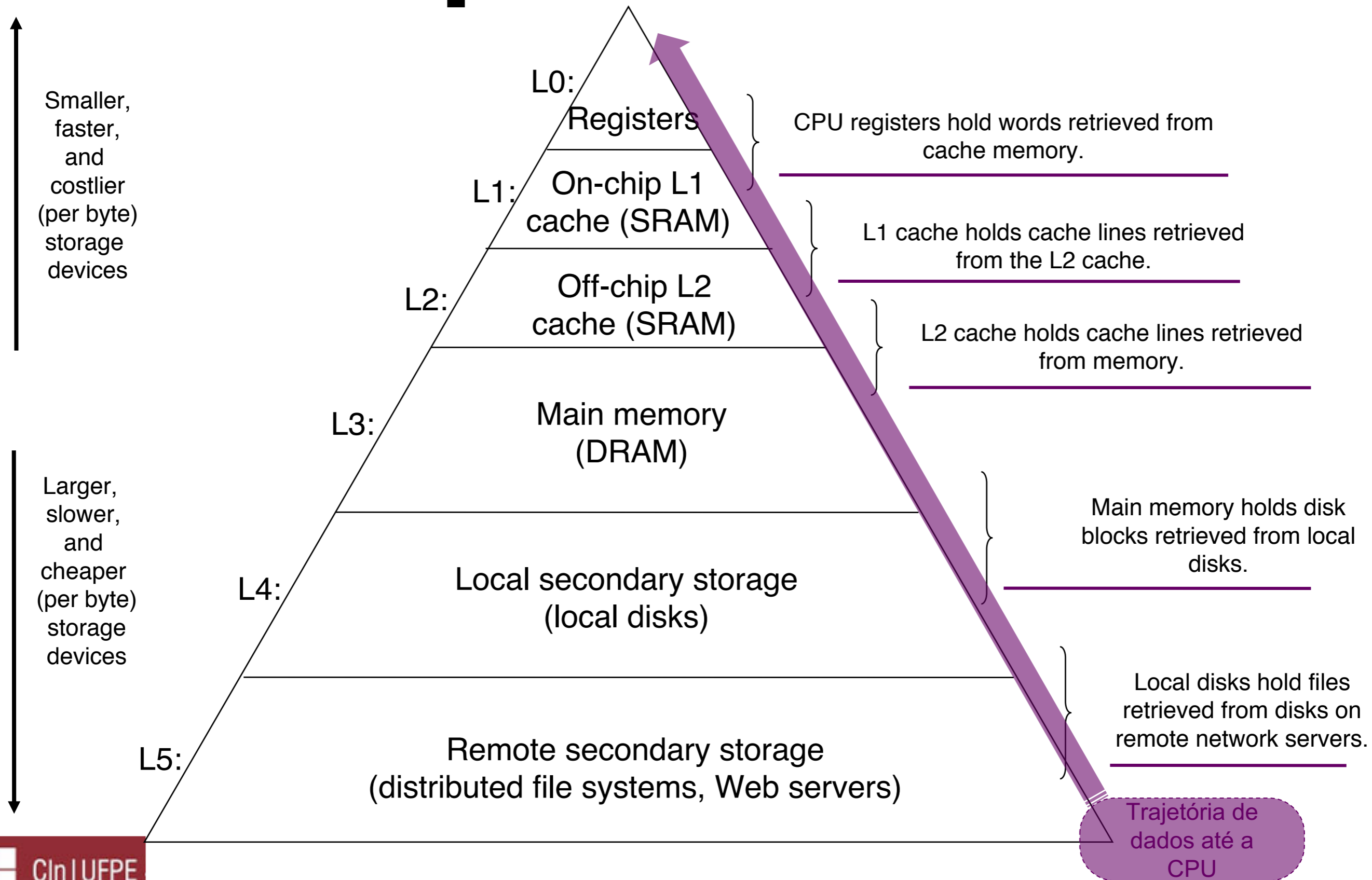
- ALU: Unidade Aritmética e Lógica
- **Registradores**
 - Funcionam como **memória** de acesso **extremamente rápido**
 - **Pouca capacidade** de armazenamento
 - Funções específicas, ex:
 - Program Counter (PC): contém o endereço da próxima instrução a ser executada
 - Instruction Register (IR): onde é copiada cada instrução a ser executada

- Ciclo ***fetch-decode-execute***:
 - busca instrução na memória
 - atualiza PC
 - decodifica instrução
 - executa instrução

Memória

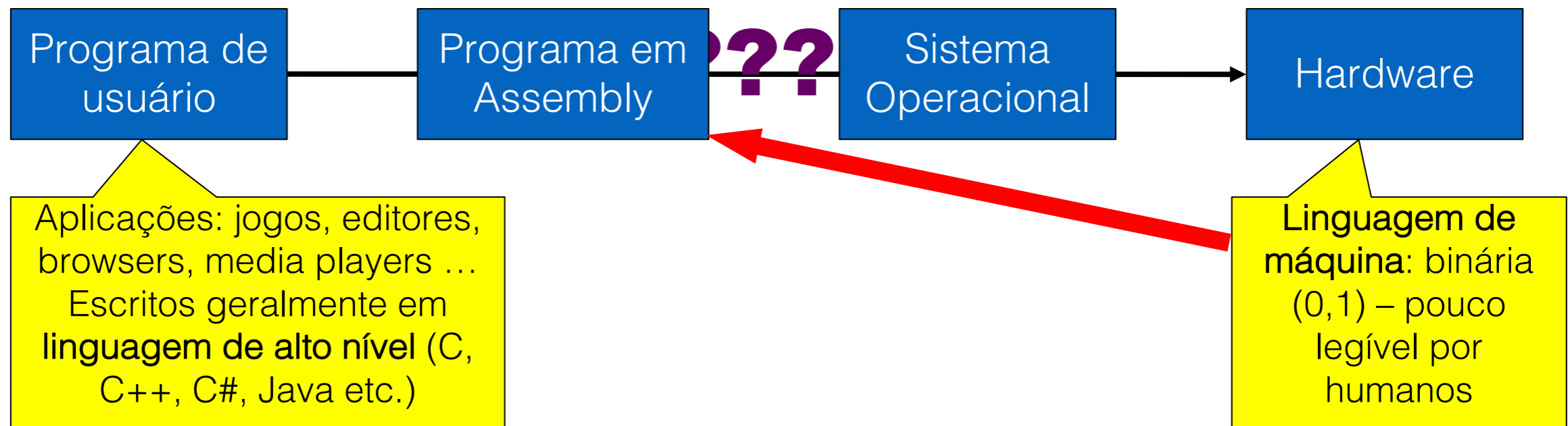
- Logicamente, a memória principal corresponde a um enorme vetor (array) de bytes
 - cada posição tem um endereço único (índice do vetor)
- Os registradores da CPU muitas **vezes** são usados para armazenar endereços de memória
 - Assim, o número de bits em cada registrador **limita** o número de **posições de memória endereçáveis**
 - Ex.: 8 bits → 256 posições...

Hierarquia de Memória



Software Básico

[A. Raposo e M. Endler, PUC-Rio, 2008]

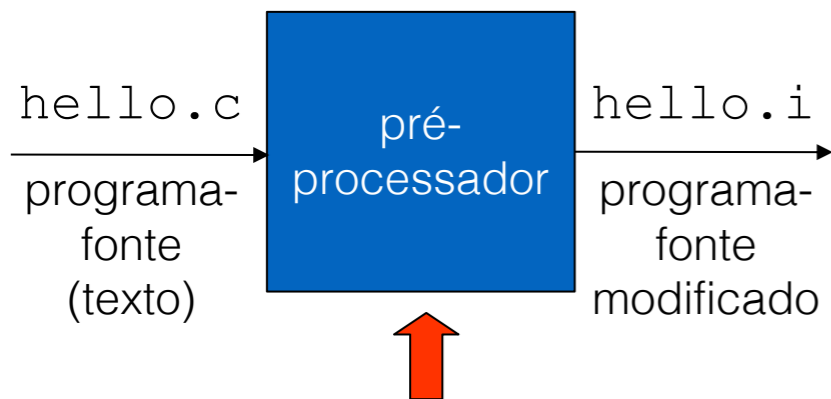


“Conhecendo mais sobre o que está ‘por baixo’ do programa, você pode escrever programas mais eficientes e confiáveis”

Gerando um executável

```
unix> gcc -o hello hello.c
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```

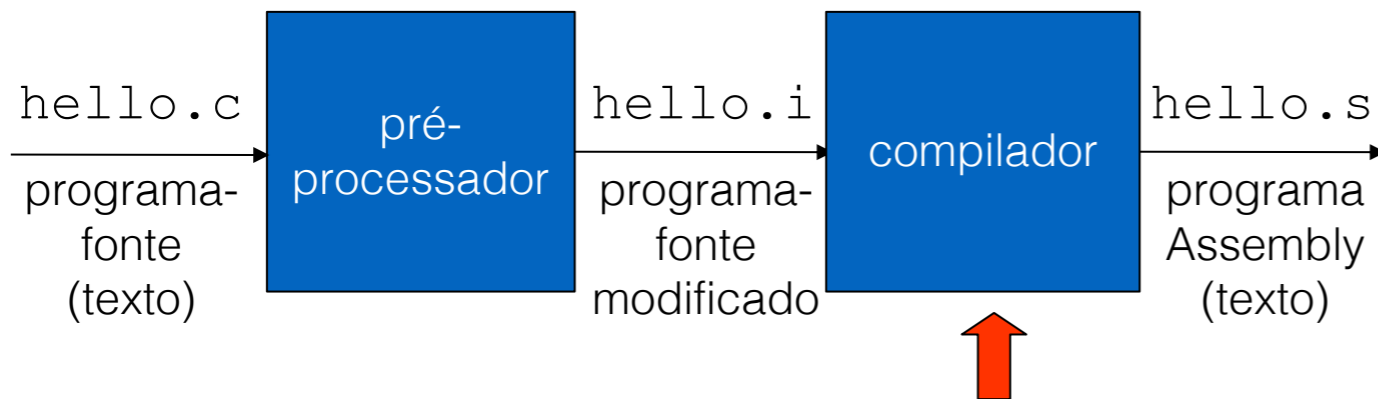


- Modifica o programa em C de acordo com diretivas começadas com #
 - Ex.: #include <stdio.h> diz ao pré-processador para ler o arquivo stdio.h e inseri-lo no programa fonte
- O resultado é um programa expandido em C, normalmente com extensão .i, em Unix

Gerando um executável

```
unix> gcc -o hello hello.c
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```

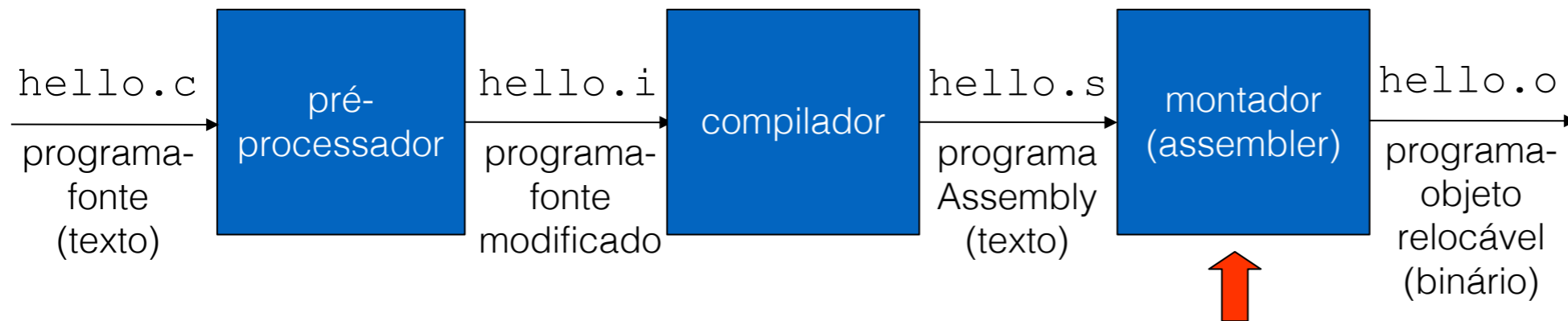


- **Compilador** traduz o programa .i em um programa em Assembly
 - É o formato de saída comum para os compiladores nas várias linguagens de programação de alto nível
 - i.e., programas em C, Java, Fortran, etc vão ser traduzidos para a mesma linguagem Assembly

Gerando um executável

```
unix> gcc -o hello hello.c
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```

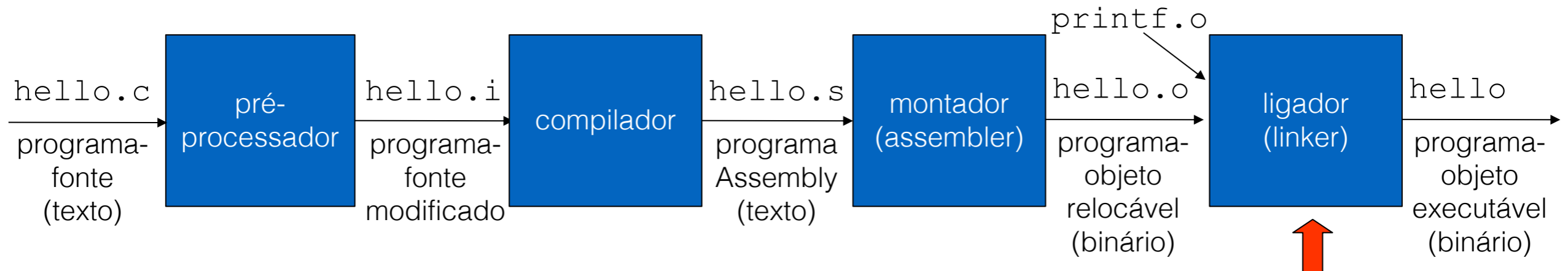


- **Montador** (Assembler) transforma o programa em Assembly em um programa binário em linguagem de máquina (chamado programa-objeto)
 - Os módulos de programas, compilados ou montados, são armazenados em um formato intermediário (“*Programa-Objeto Relocável*” – extensão `.o`)
- Endereços de acesso e a posição do programa na memória ficam **indefinidos**

Gerando um executável

```
unix> gcc -o hello hello.c
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```

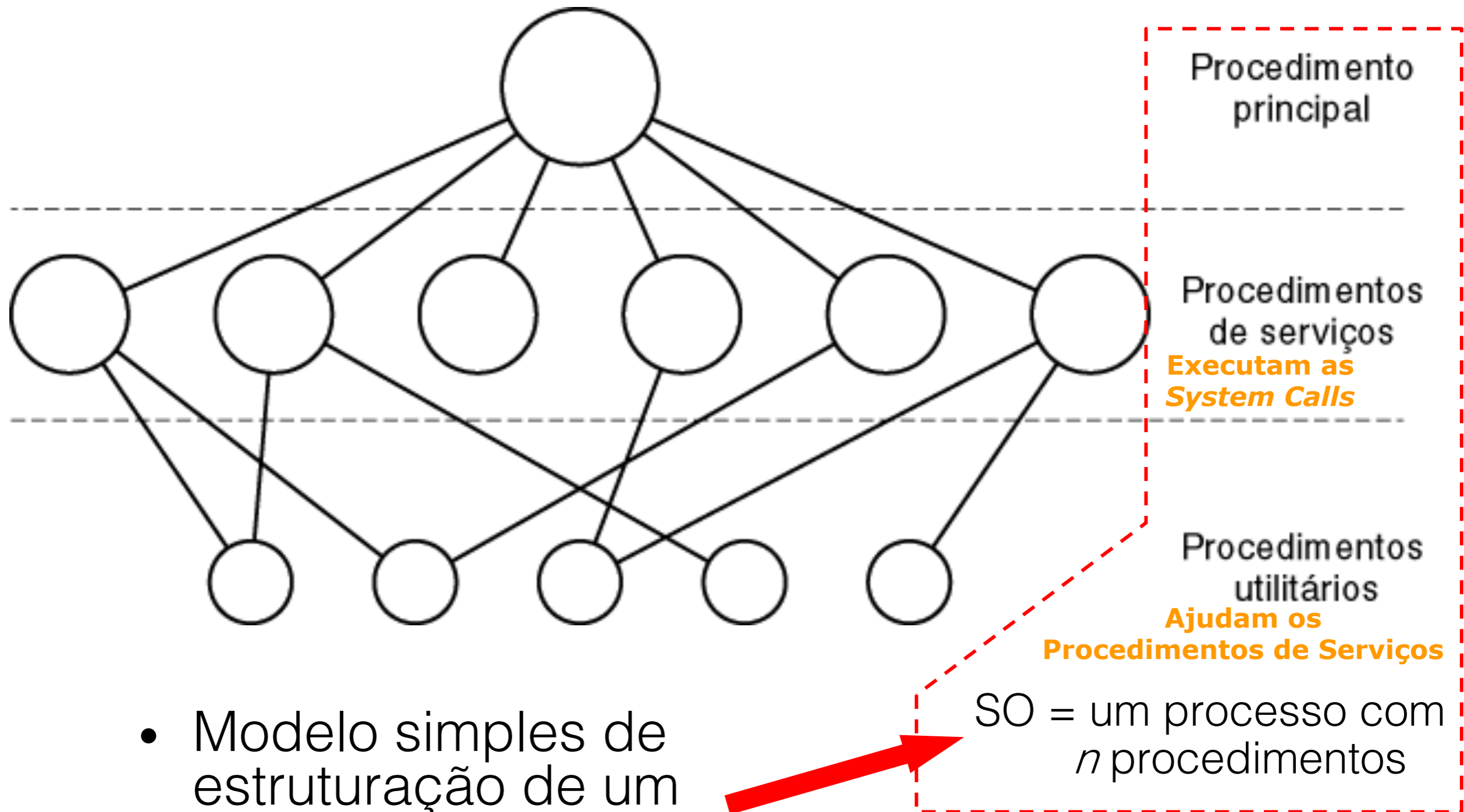


- O **ligador** (linker) gera o programa executável a partir do .o gerado pelo assembler
 - No entanto, pode haver funções-padrão da linguagem (ex., `printf`) que não estão definidas no programa, mas em outro arquivo .o pré-compilado (`printf.o`)
 - O ligador faz a junção dos programas-objeto (.o) necessários para gerar o executável

Estruturas de Sistemas Operacionais

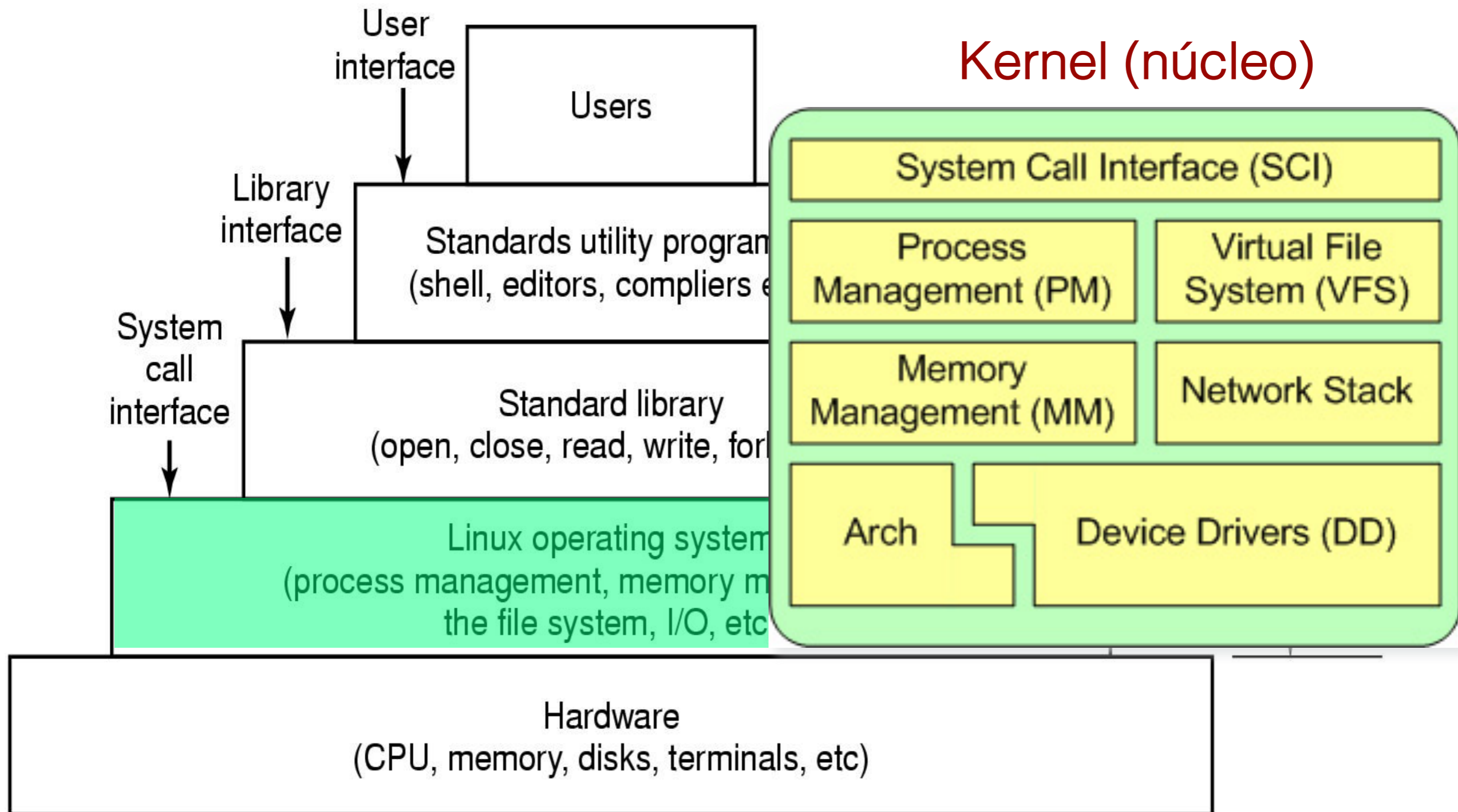
- Monolítica
- Em camadas
- Cliente-Servidor
- Virtualização

Estrutura de Sistemas Operacionais: Sistema Monolítico

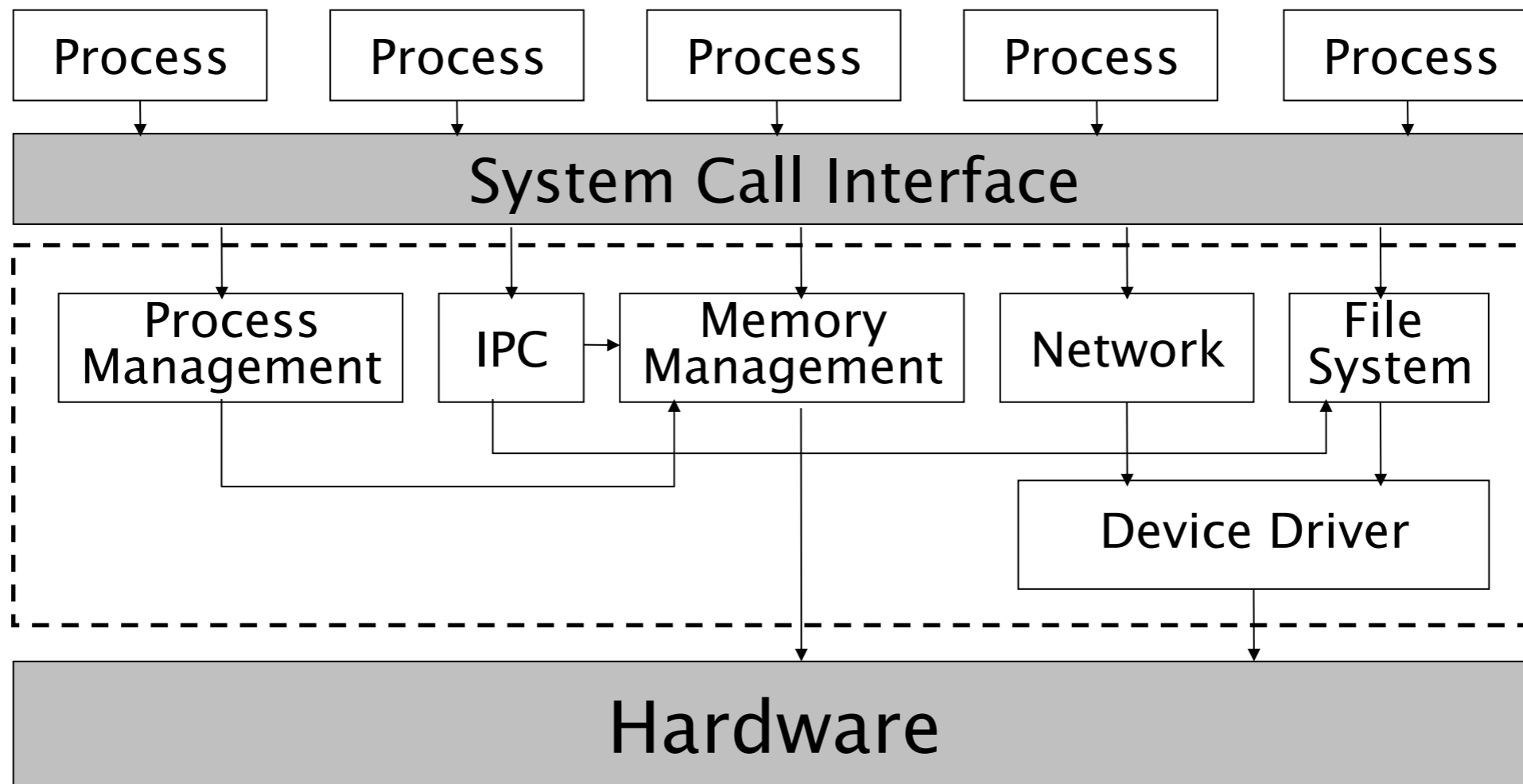


- Modelo simples de estruturação de um sistema **monolítico**

Camadas em Linux



Linux Kernel: Relacionamentos



APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content

View

Notification Manager

Package Manager

Telephony Manager

GTalk Service

LIBRARIES

Surface Manager

Media Framework

OpenGL |

ANDROID

SGL

SSL

ANDROID RUNTIME

LINUX KERNEL

Display Driver

Camera Driver

Bluetooth Driver

Flash Memory Driver

Binder (IPC) Driver

USB Driver

Keypad Driver

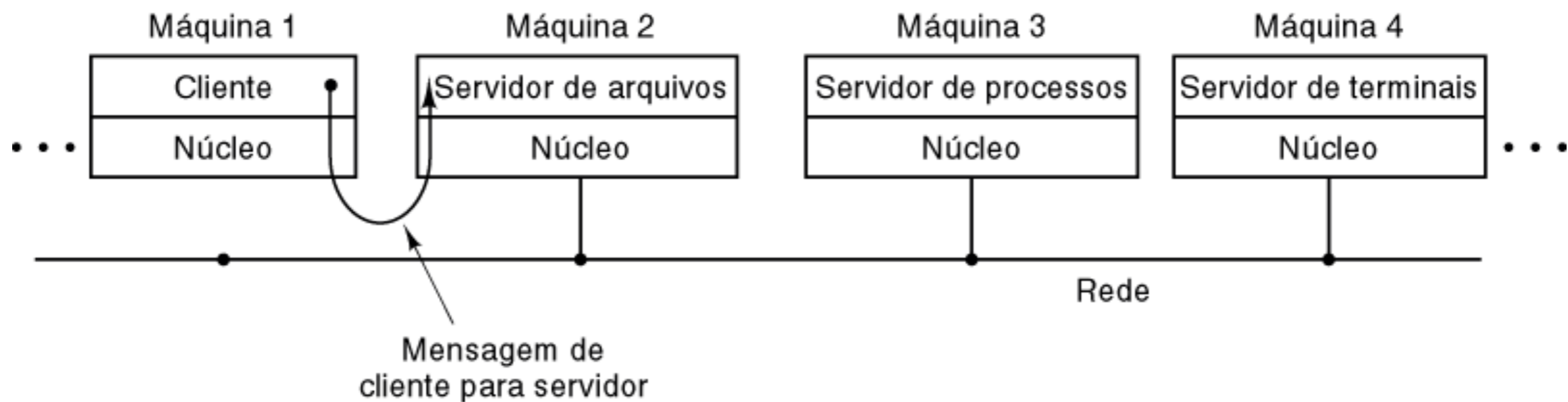
WiFi Driver

Audio Drivers

Power Management

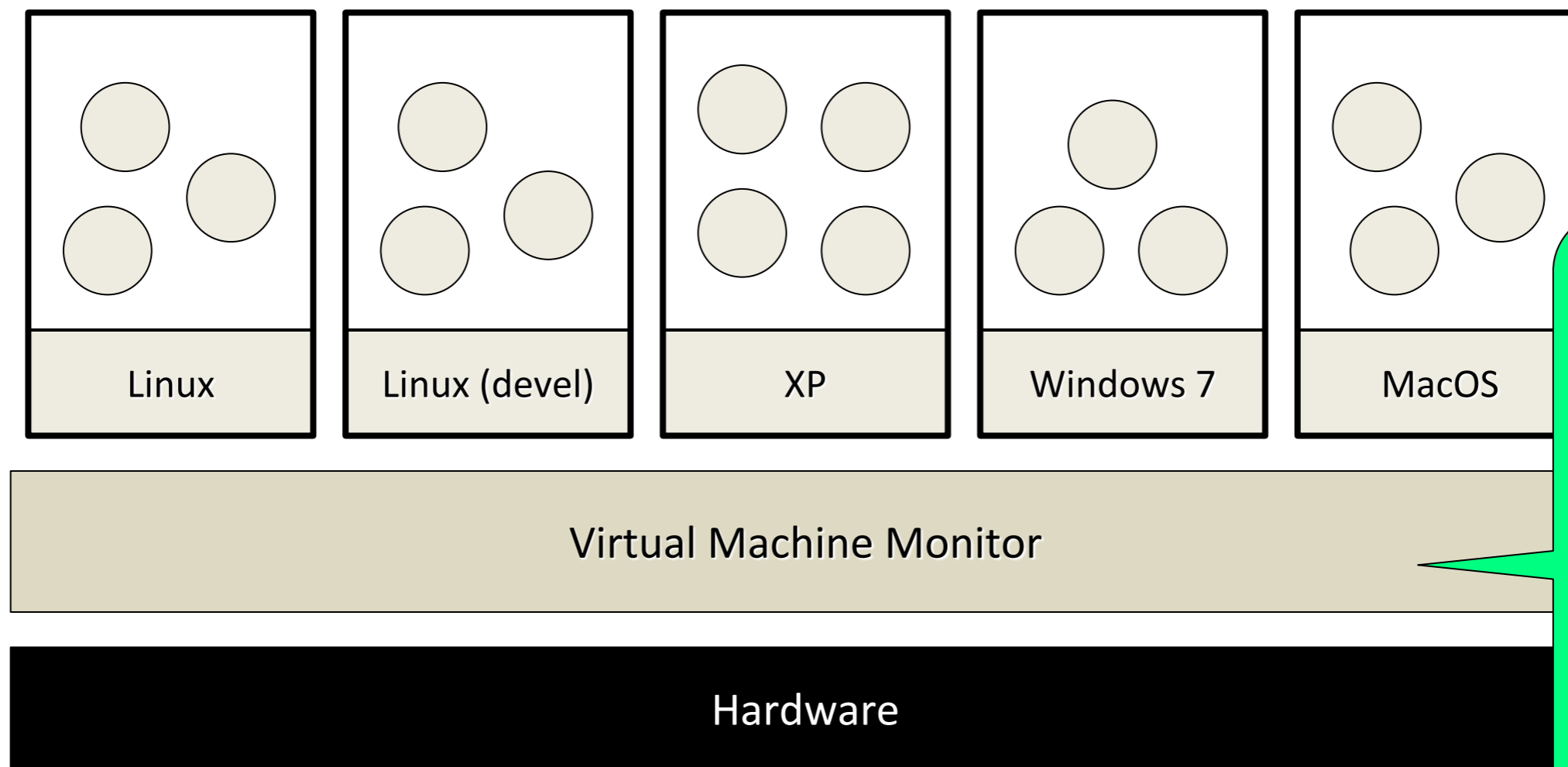
Que estratégias você imagina para economizar energia?

Estrutura de Sistemas Operacionais: Cliente-Servidor



O modelo **cliente-servidor** em um **sistema (operacional) distribuído**

Estrutura de Sistemas Operacionais: Máquina Virtual (Virtualização)

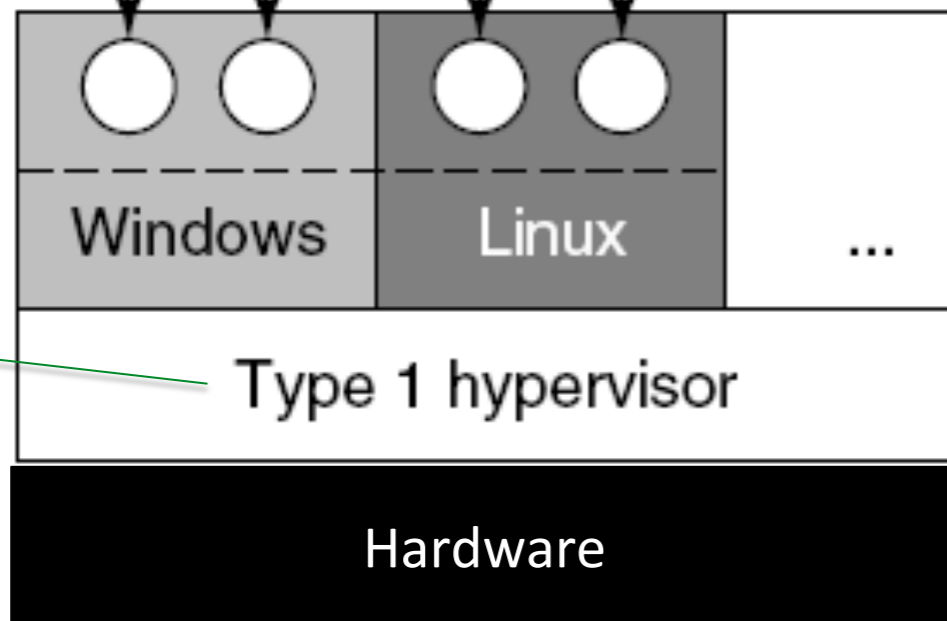


VMM opera na interface de hardware, fornecendo uma interface idêntica para os SOs acima

Virtual Machines: Tipos (Arquiteturas)

native, bare-metal
hypervisor
Ex. Xen

Excel Word Mplayer Apollon

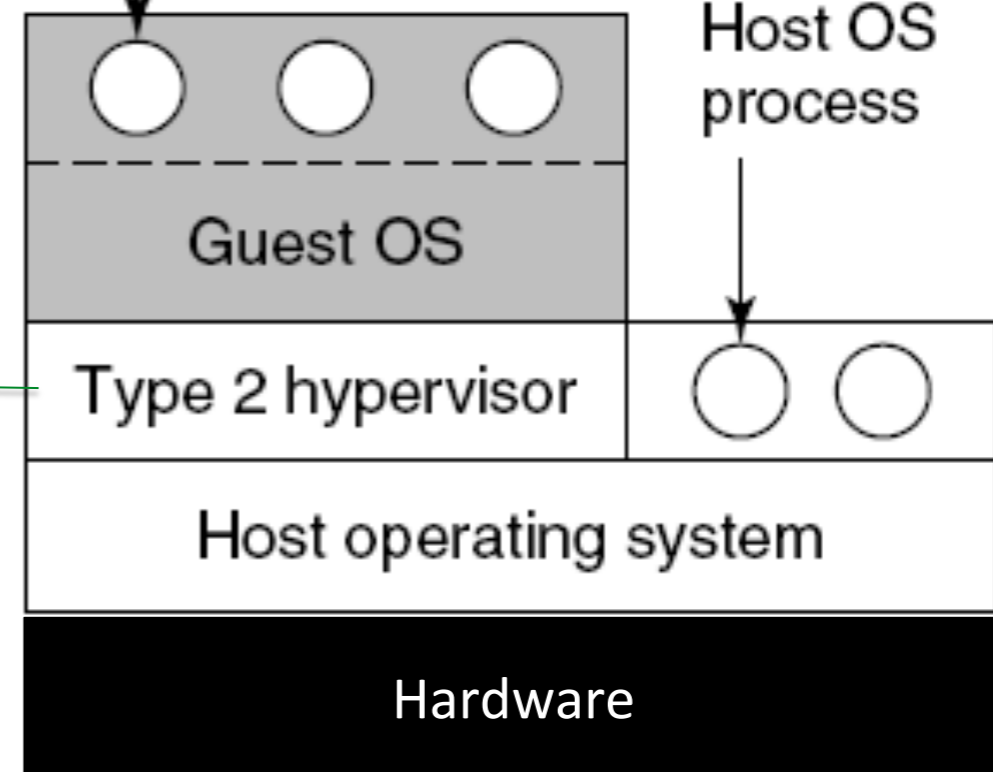


Hipervisor Tipo 1

hosted hypervisor
Ex. VirtualBox

Guest OS process

Host OS
process



Hipervisor Tipo 2

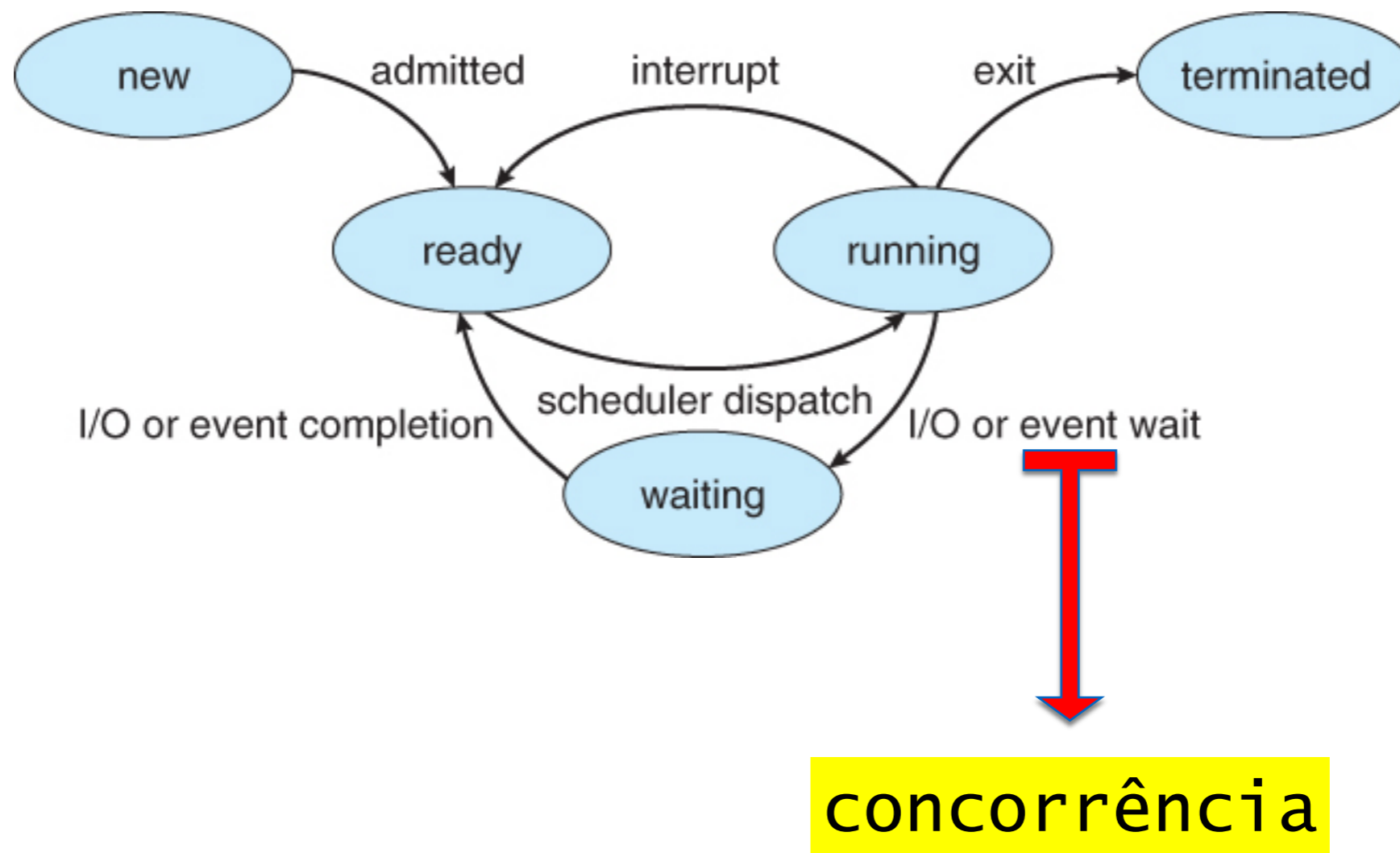
Introdução

GERÊNCIA DE PROCESSOS

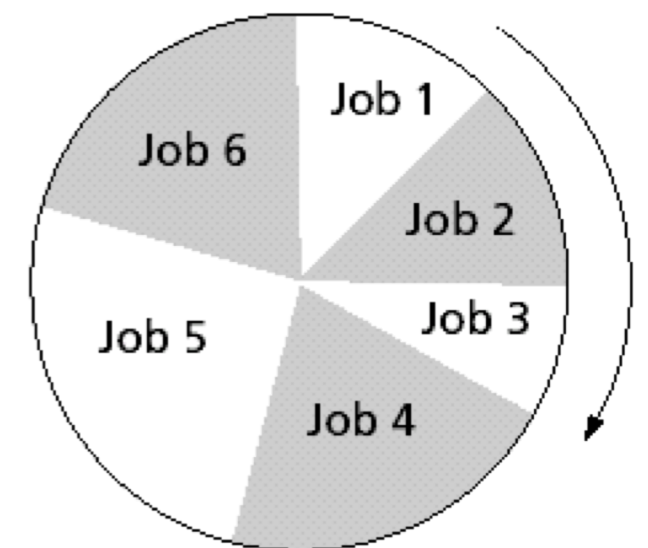
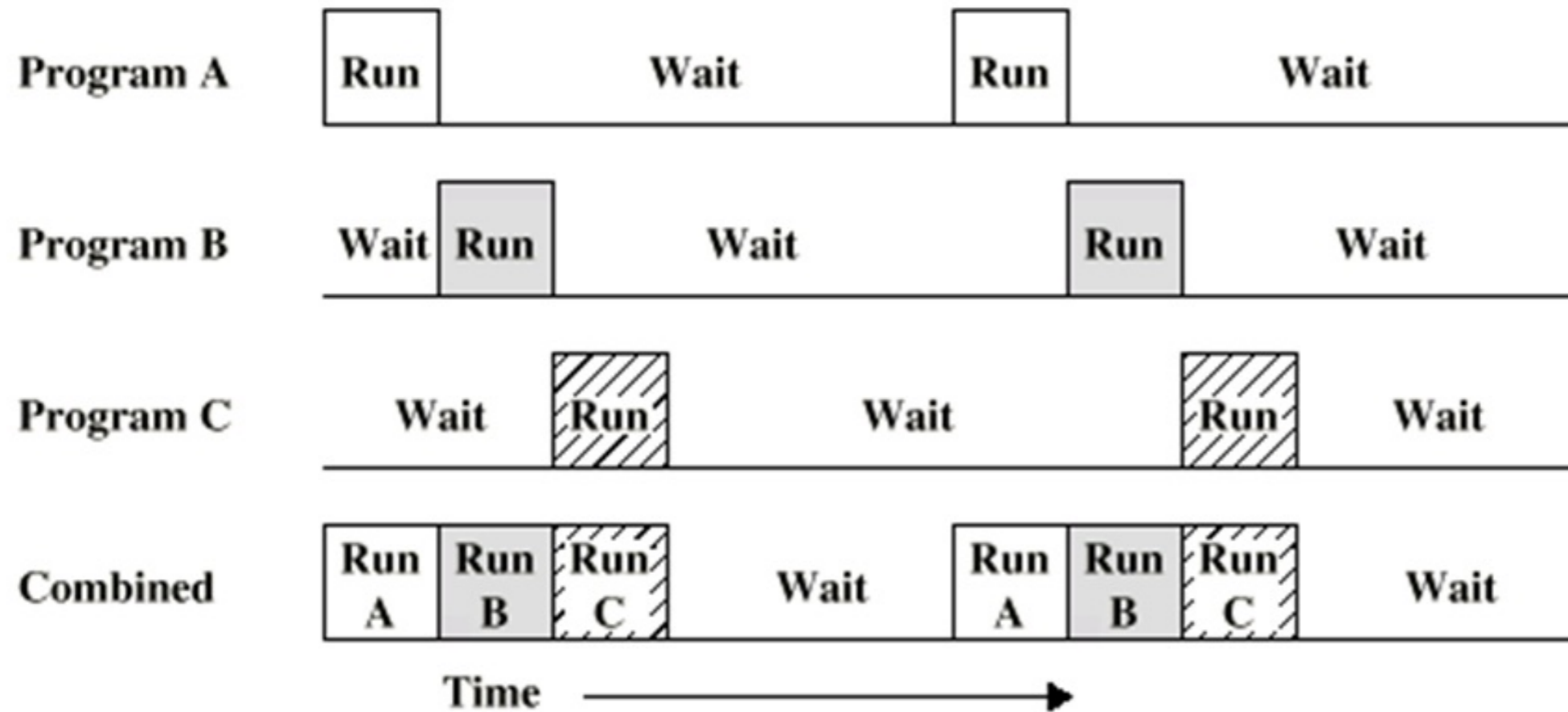
Tópicos

- Processo: máquina de estados
- Multiprogramação e Compartilhamento de Tempo
- Escalonamento
- Concorrência

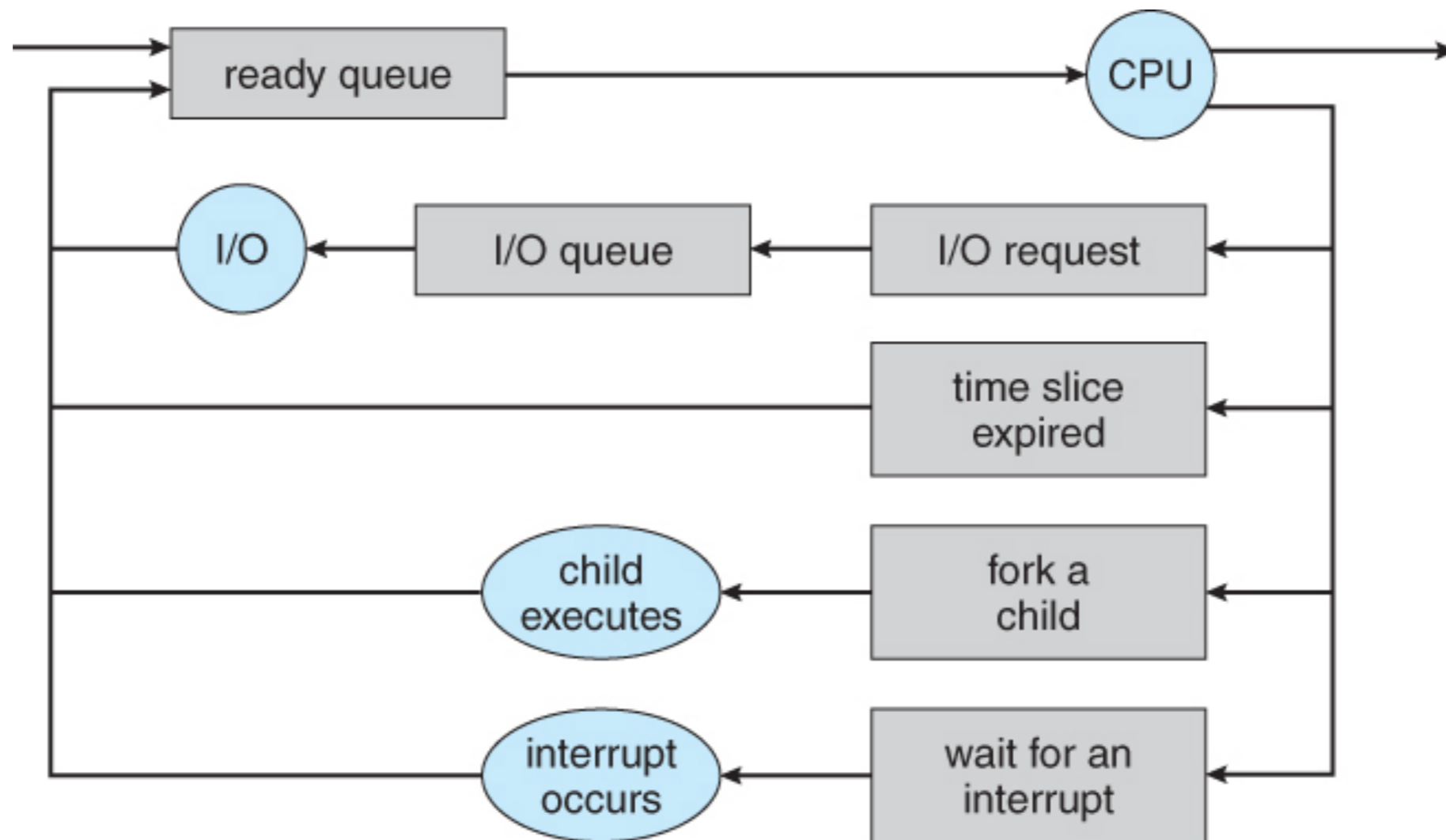
Máquina de Estados de um Processo



Multiprogramação e Compartilhamento de Tempo

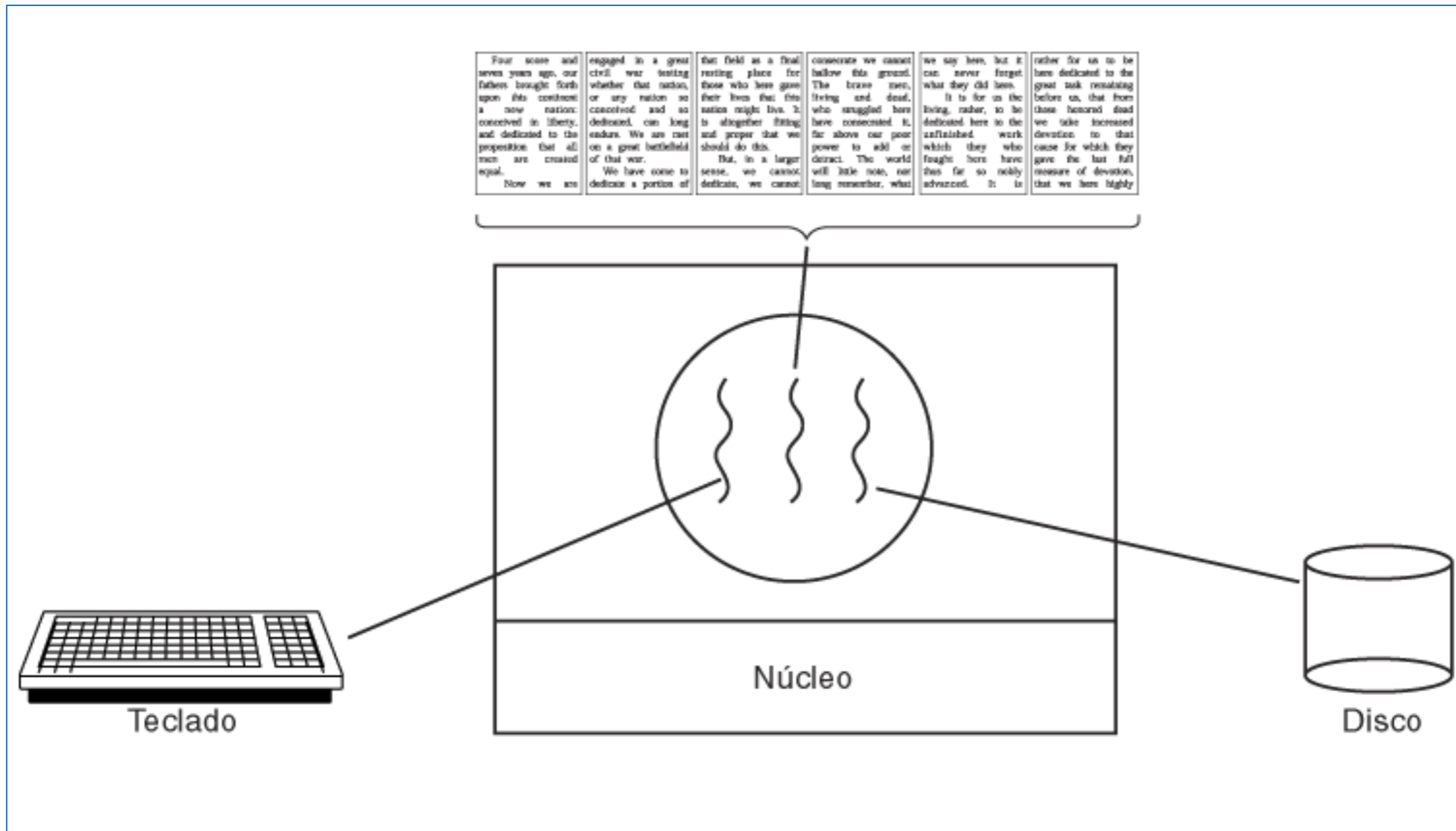


Escalonamento de Processos



Algoritmos diversos

Concorrência

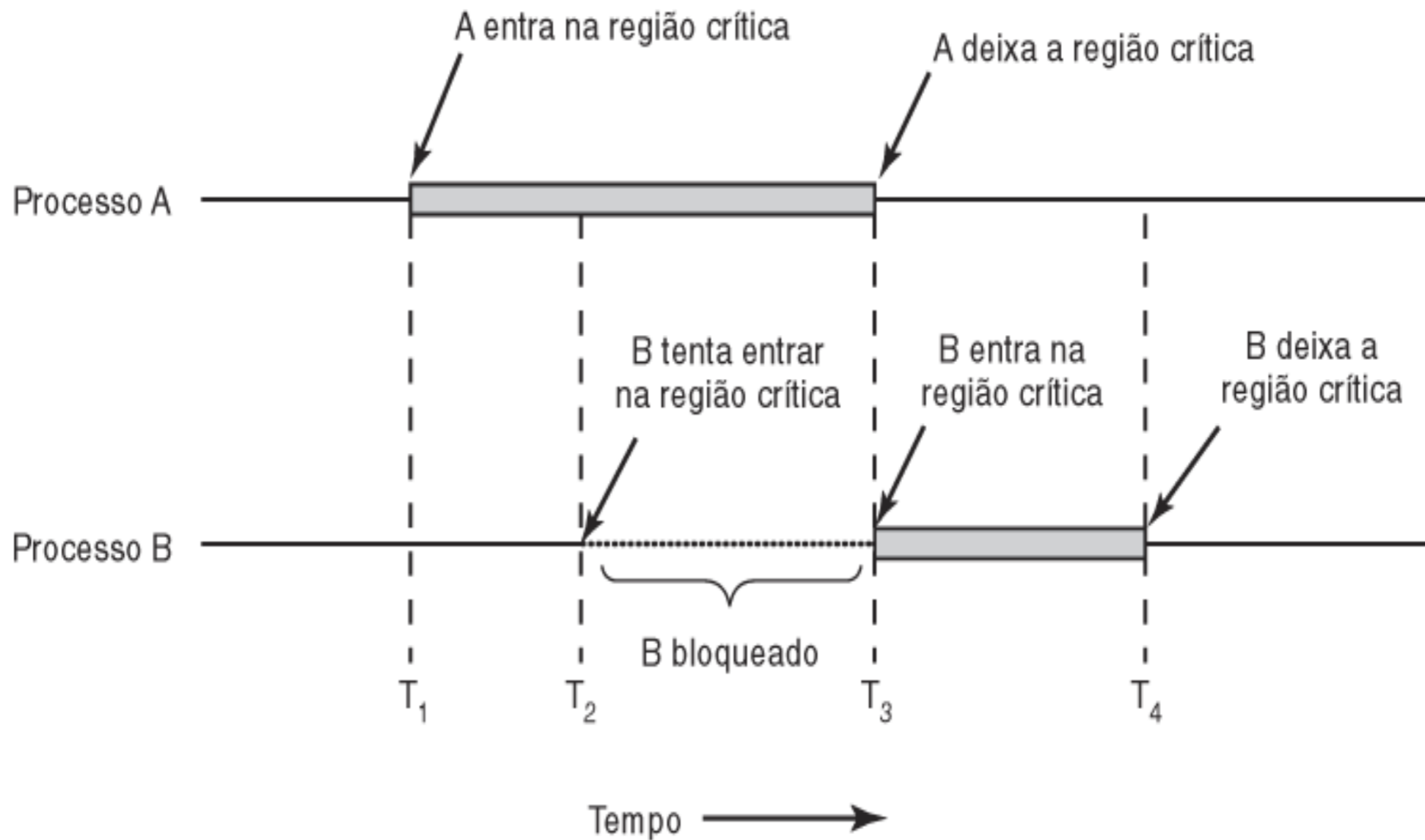


Um processador de texto com 3 *threads*

Problemas com Concorrência

- Não-determinismo
 - $x = 1 \parallel x = 2$
 - Qual o valor de “x” após a sua execução?
- Dependência de Velocidade
 - $[[f(); x = 1]] \parallel [[g(); x = 2]]$
 - O valor final de x depende de qual das funções, f() e g(), terminar primeiro
- *Starvation*
 - Processo de baixa prioridade precisa de um recurso que nunca é fornecido a ele...
- *Deadlock*
 - dois processos bloqueiam a sua execução, pois um precisa de um recurso bloqueado pelo outro processo

Conceitos Fundamentais



Região crítica e Exclusão mútua

Introdução

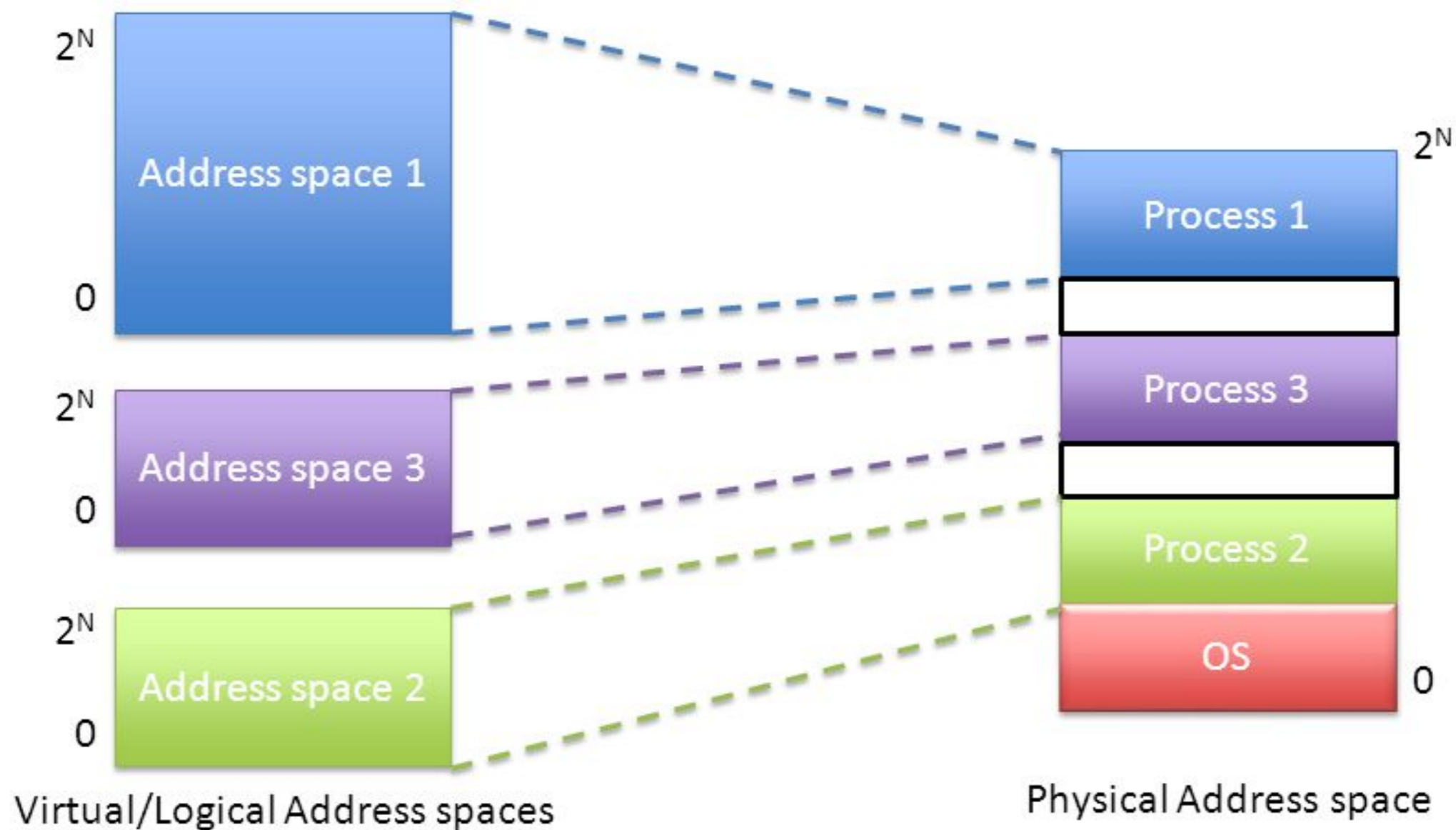
GERÊNCIA DE MEMÓRIA

Tópicos

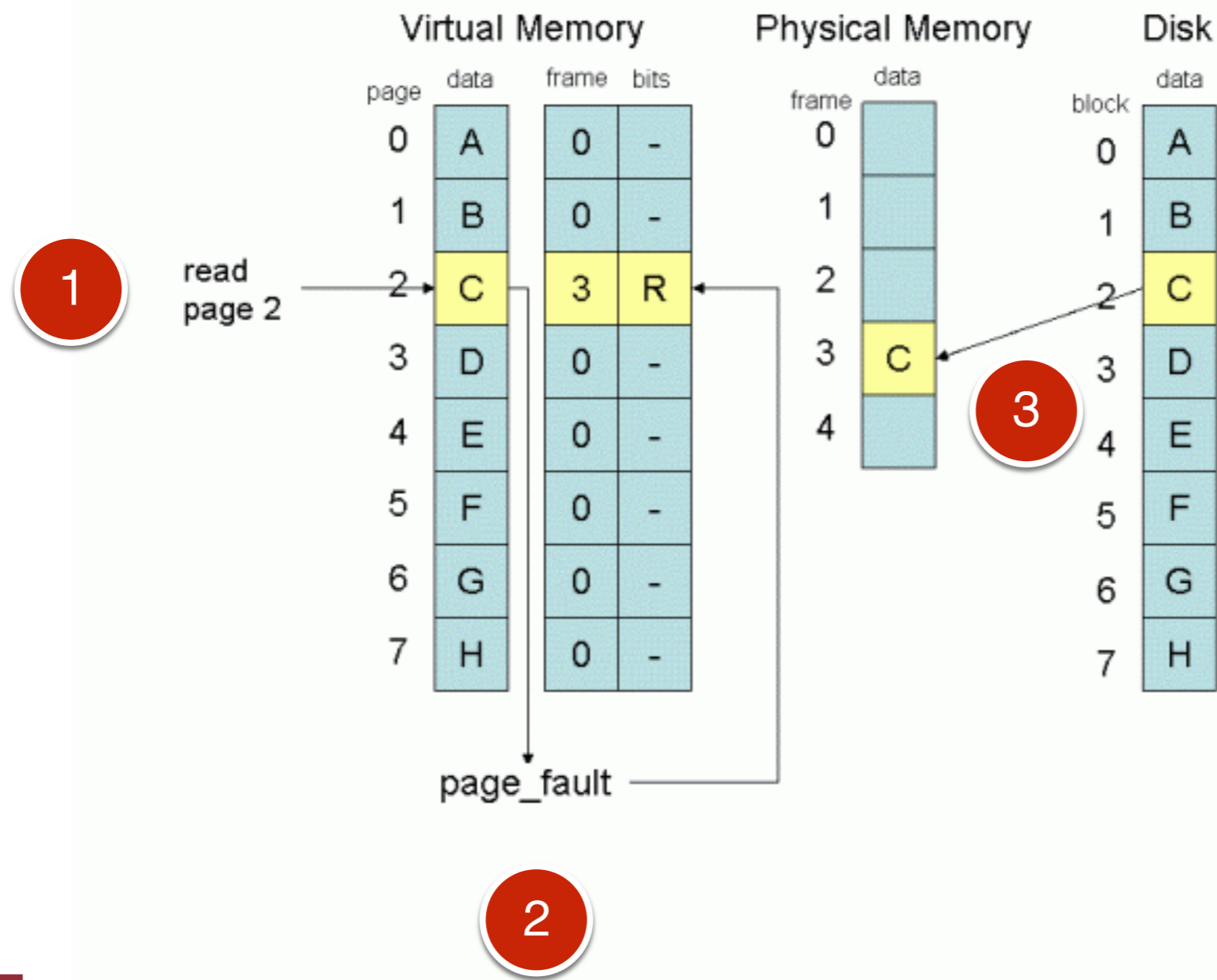
- Espaço de Endereçamento
- Memória Virtual
- Paginação

Address Spaces

- Translation from logical to physical addresses



Paginação



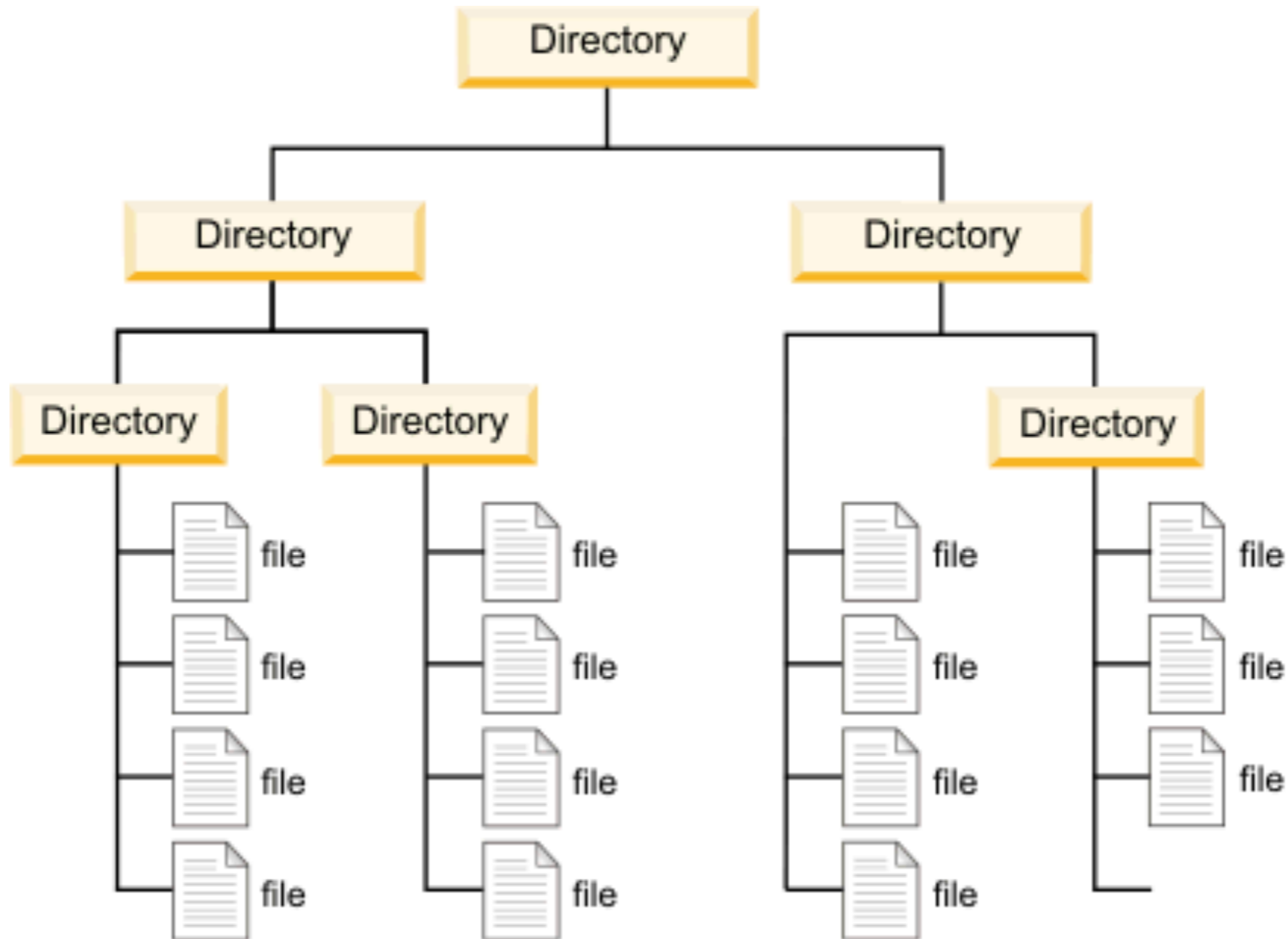
Introdução

SISTEMA DE ARQUIVOS

Tópicos

- Arquivos
- Diretórios
- Alocação de Espaço em Disco

Arquivos e Diretórios



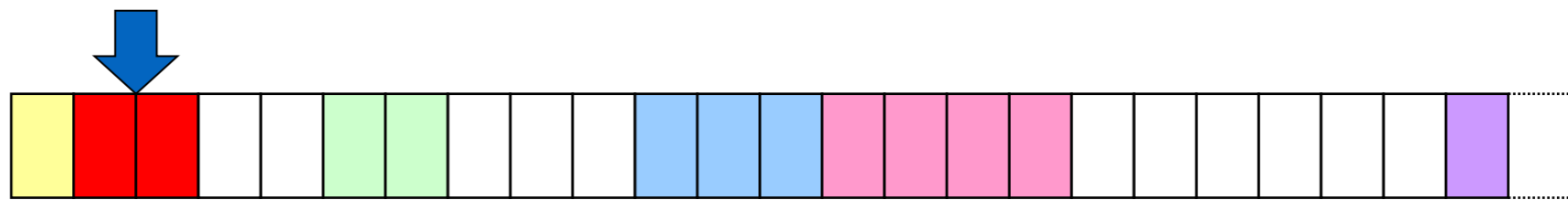
ZOSB025

Alocando um arquivo com 2 blocos

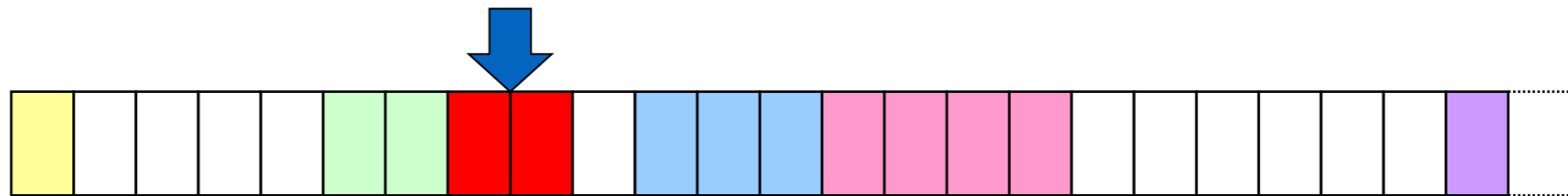
Situação inicial



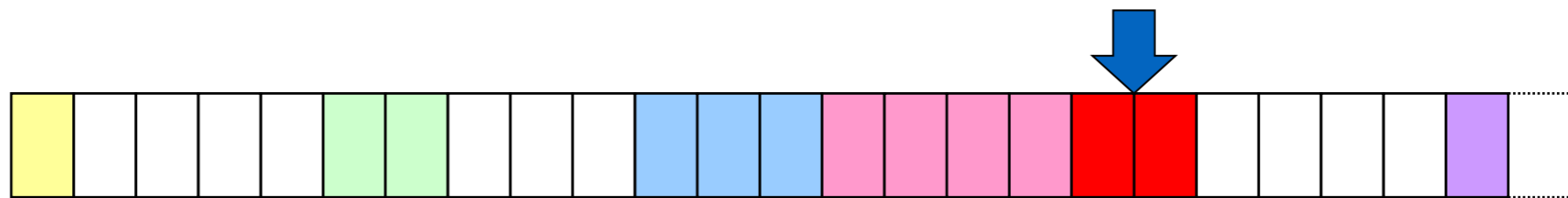
First-fit



Best-fit



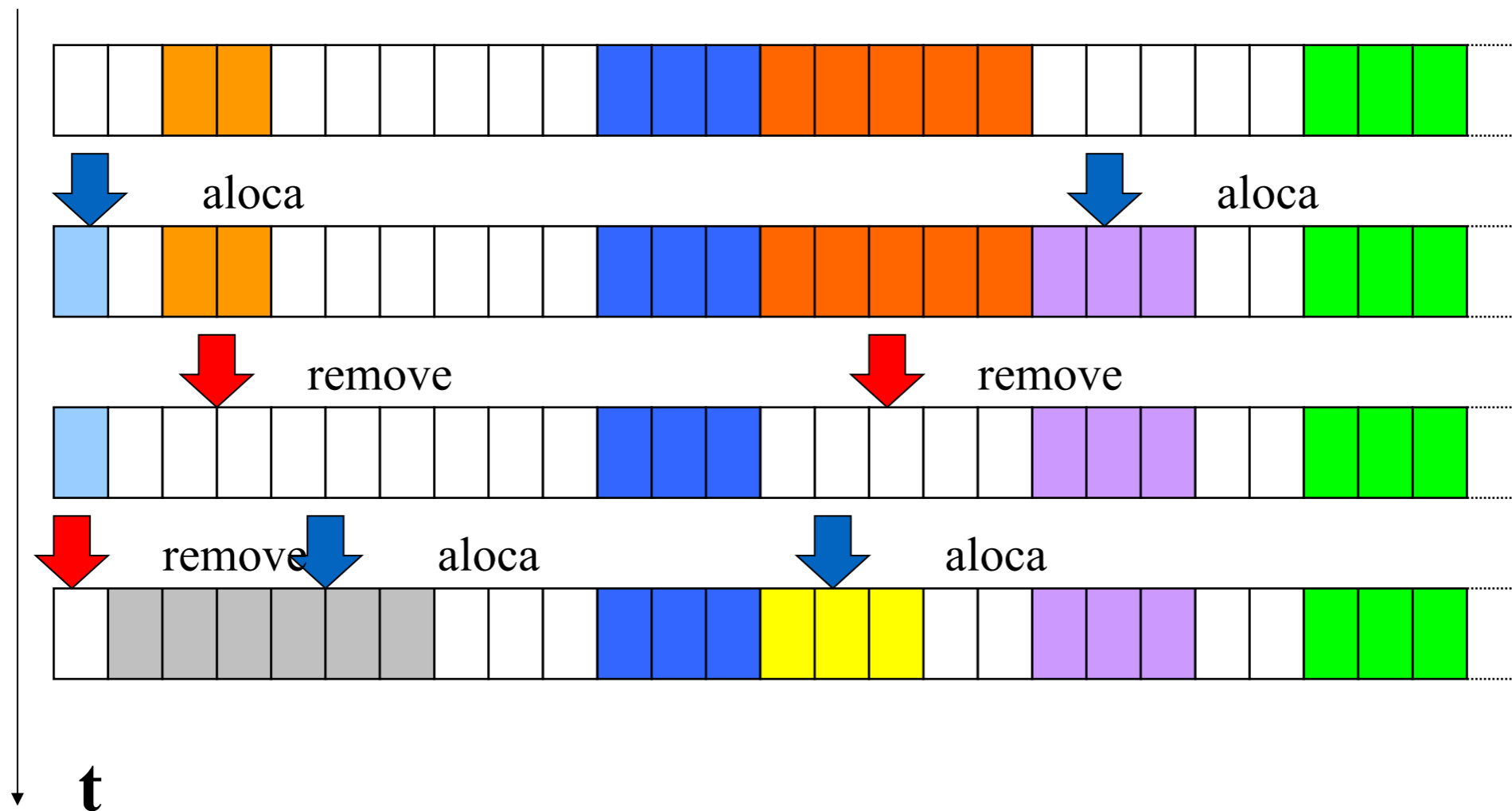
Worst-fit



Alocação contígua

Mais rápido
Menos desperdício
Mais expansível

Evolução da fragmentação



Agora, como alocar um arquivo com 4 blocos ?

Estratégias de defragmentação

Situação inicial



Moveu 6 blocos



Moveu 4 blocos



Moveu 2 blocos



✓ Alocação contígua

Alocação por lista encadeada

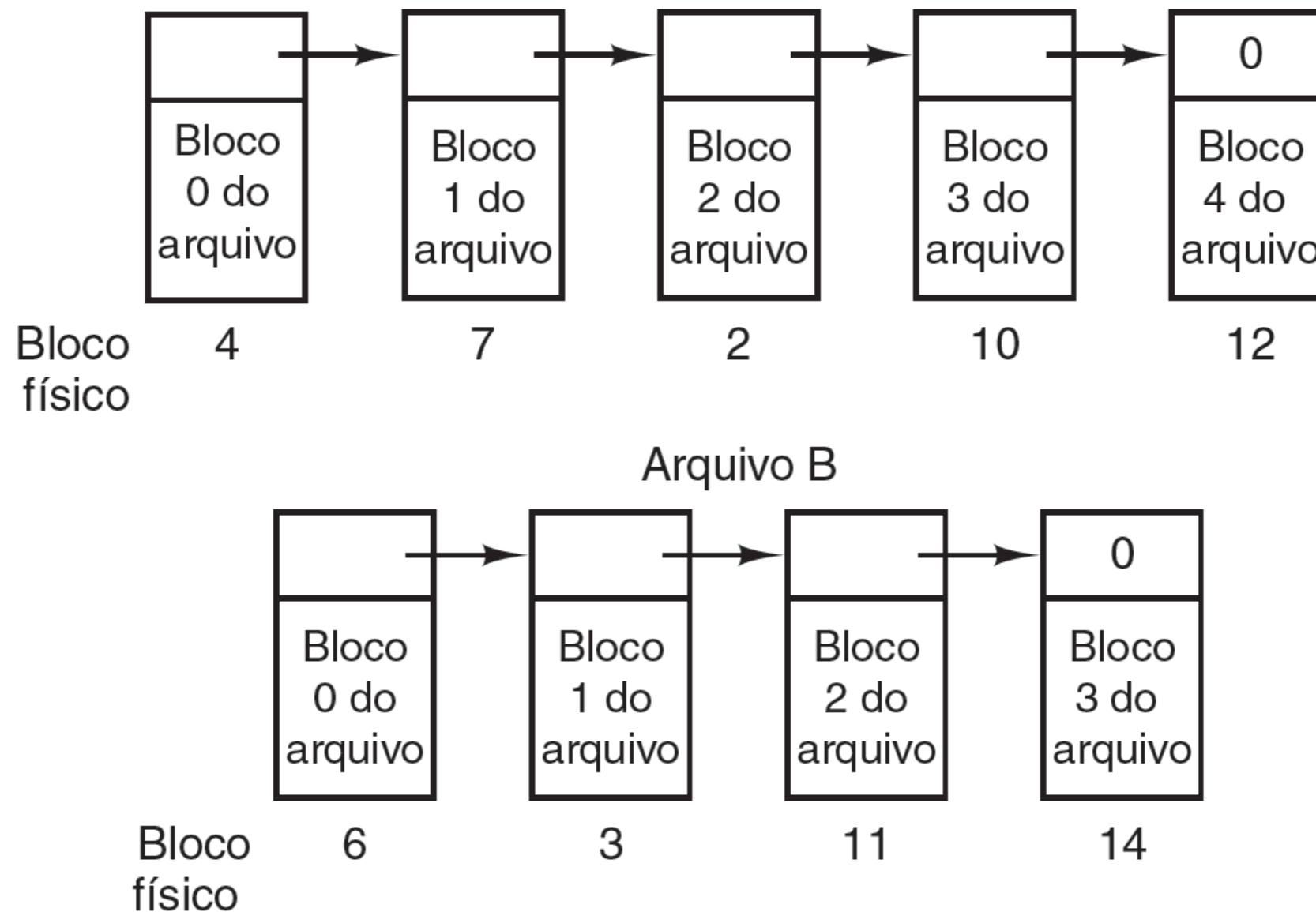
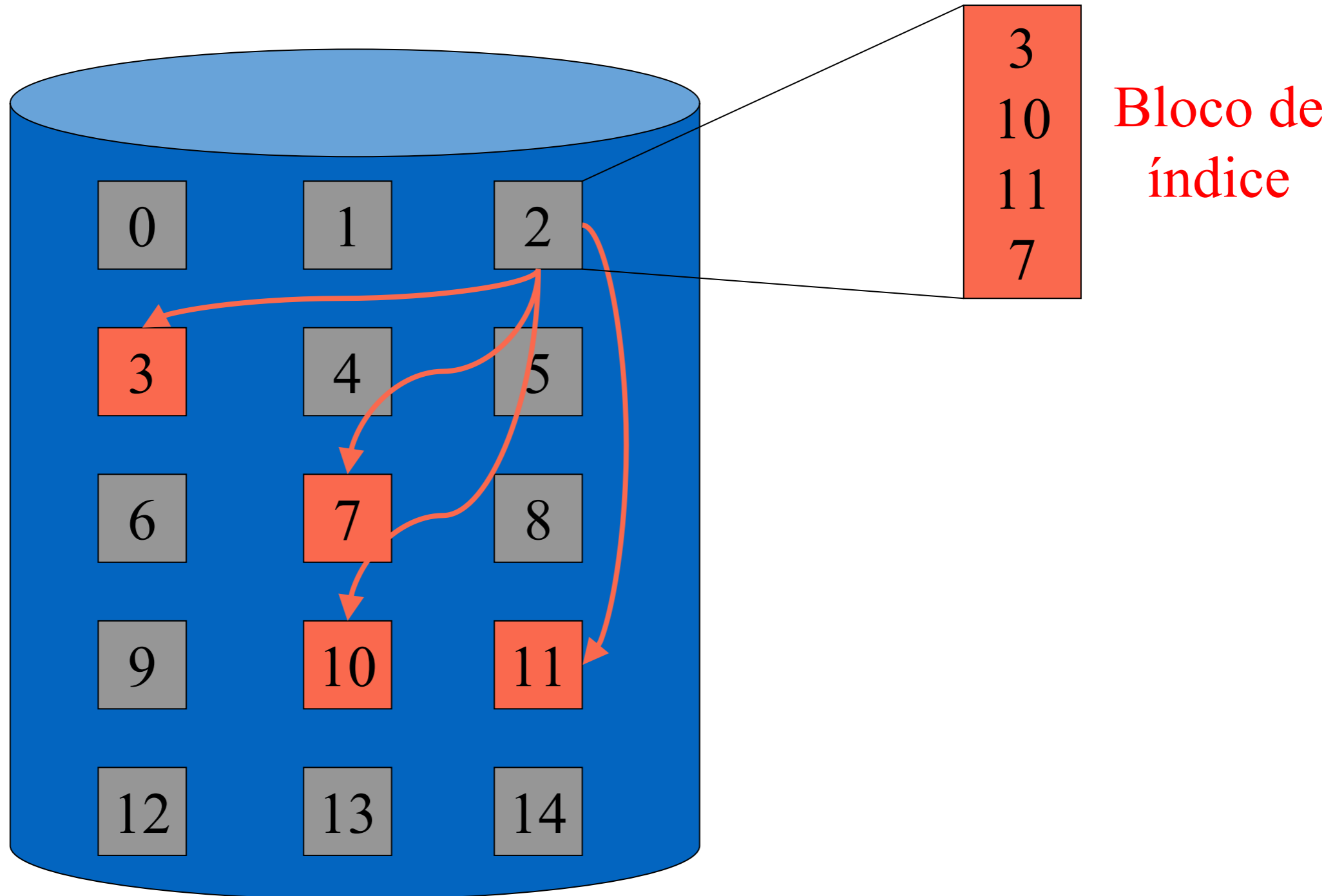


Figura 4.9 Armazenamento de um arquivo como uma lista encadeada de blocos de disco.

- ✓ Alocação contígua
- ✓ Alocação por lista encadeada

Alocação Indexada



Introdução

GERÊNCIA DE ENTRADA/SAÍDA

Tópicos

- Diversidade de Dispositivos de E/S
- Uniformidade de Acesso
- Como a CPU sabe que um dispositivo de E/S terminou sua execução?

Diversidade de dispositivos

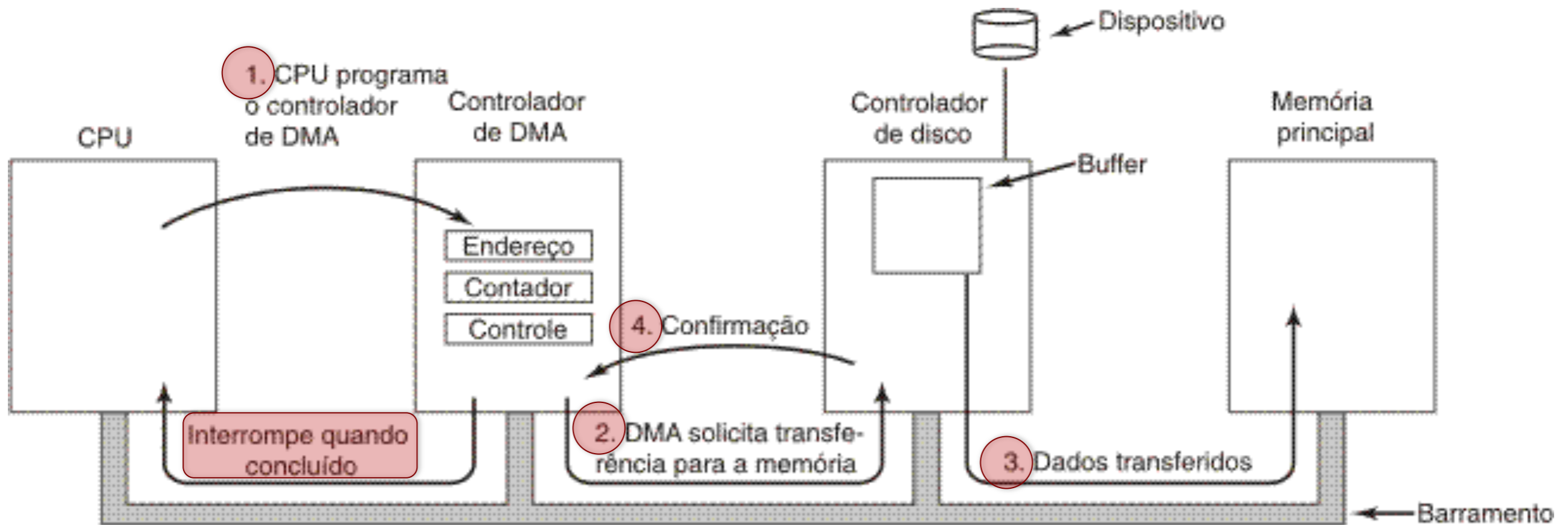
Hardware de E/S

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner at 300 dpi	1 MB/sec
Digital camcorder	3.5 MB/sec
4x Blu-ray disc	18 MB/sec
802.11n Wireless	37.5 MB/sec
USB 2.0	60 MB/sec
FireWire 800	100 MB/sec
Gigabit Ethernet	125 MB/sec
SATA 3 disk drive	600 MB/sec
USB 3.0	625 MB/sec
SCSI Ultra 5 bus	640 MB/sec
Single-lane PCIe 3.0 bus	985 MB/sec
Thunderbolt 2 bus	2.5 GB/sec
SONET OC-768 network	5 GB/sec

Como a CPU sabe que o dispositivo já executou o comando?

- E/S Programada
 - CPU lê constantemente o status do controlador e verifica se já acabou (*Polling* ou *Busy-waiting*)
 - Desvantagem: Espera até o fim da operação
- E/S por Interrupção
 - CPU é interrompida pelo módulo de E/S e ocorre transferência de dados
 - CPU continua a executar outras operações
 - Desvantagem: toda palavra lida do (ou escrita no) periférico passa pela CPU
- E/S por DMA - Acesso Direto à Memória
 - Quando necessário, o controlador de E/S solicita ao controlador de DMA a transferência de dados de/para a memória
 - Nesta fase de transferência não há envolvimento da CPU
 - Ao fim da transferência, a CPU é interrompida e informada da transação

Acesso Direto à Memória (DMA)



Operação de uma transferência com DMA

Objetivos da gerência de E/S

- Eficiência
- **Uniformidade** (desejável, diante da alta **diversidade**, associada à **heterogeneidade**):
 - Todos dispositivos enxergados da forma mais uniforme possível
- Esconder os detalhes (estes são tratados pelas camadas de mais baixo nível)
- Fornecer **abstrações genéricas**: read, write, open, close etc.

Introdução

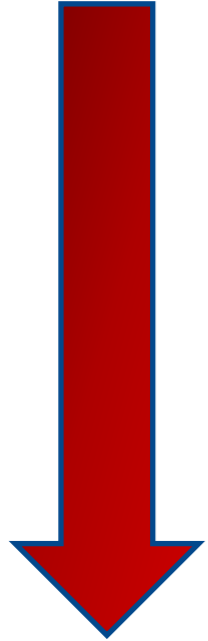
SISTEMAS DISTRIBUÍDOS

Tópicos

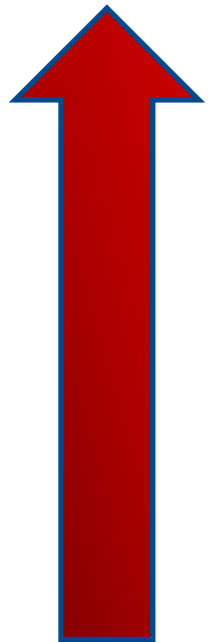
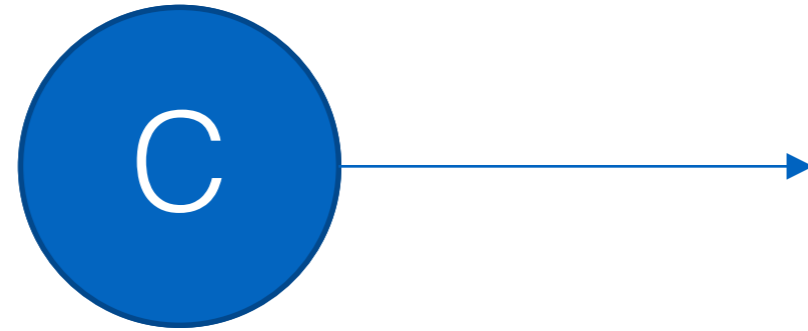
- Modelo Cliente-Servidor
- Paralelismo
- Abertura
- Falha Parcial
- Middleware



Modelo **Cliente-Servidor**

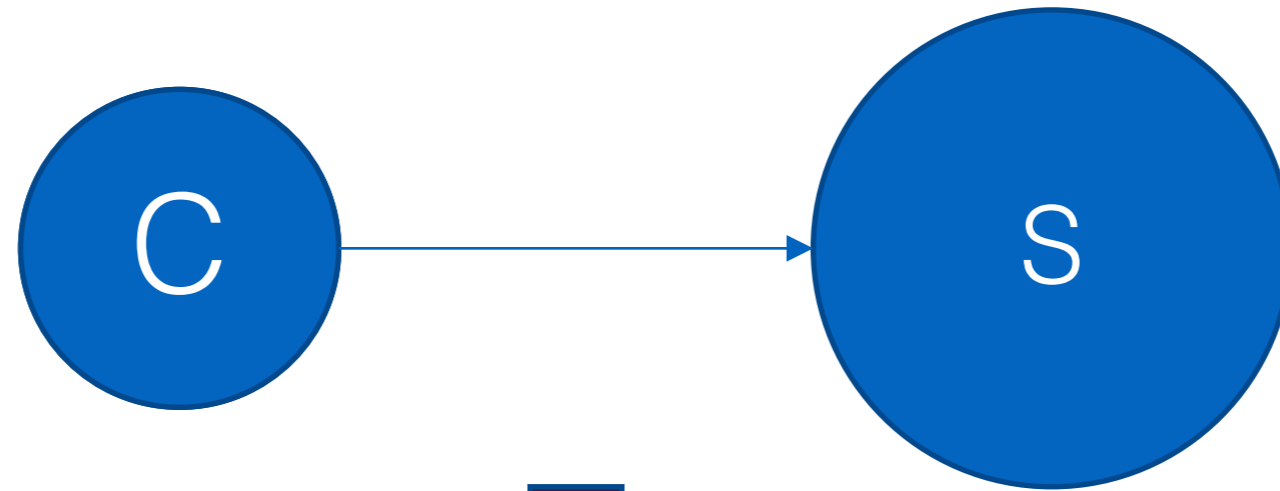


Desempenho

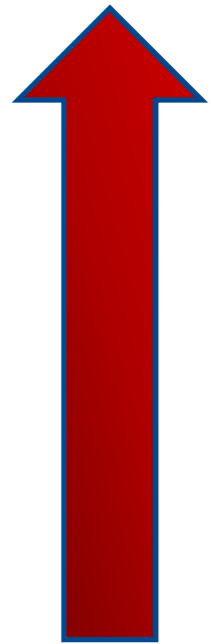
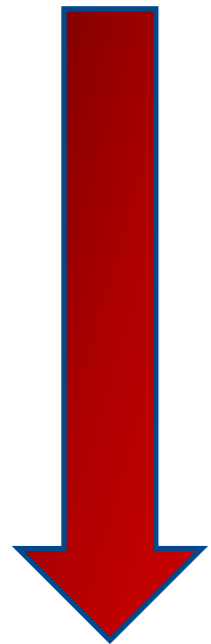


Paralelismo

Modelo **Cliente-Servidor**



Desempenho



Cloud



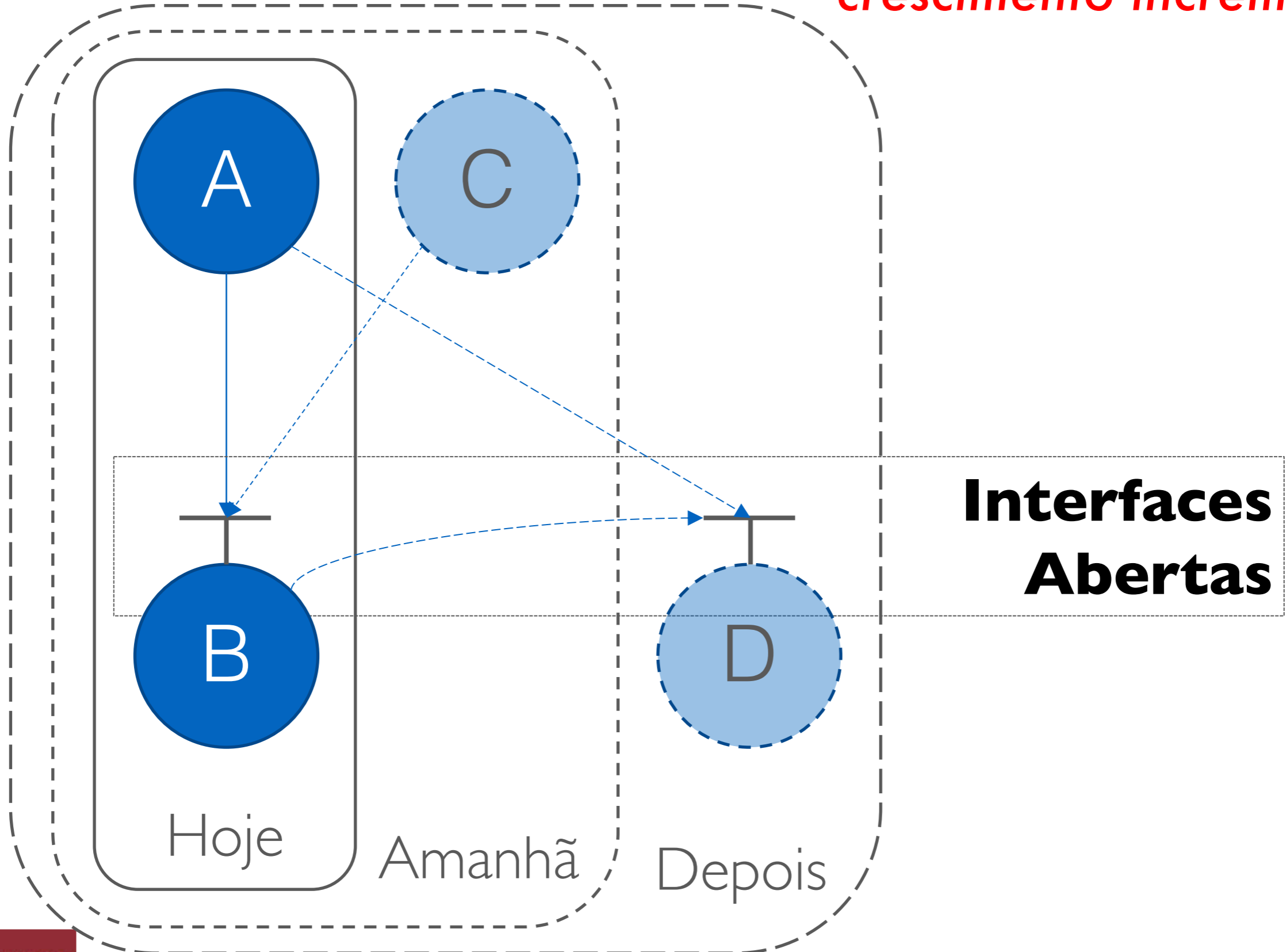
Datacenter

Paralelismo

Abertura



*Expansibilidade;
crescimento incremental*



Falha Parcial

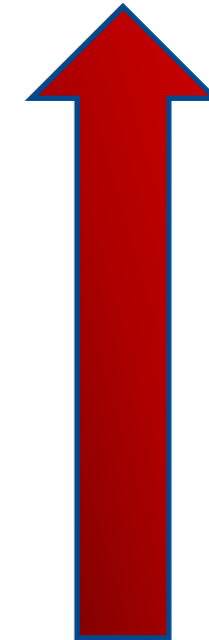


Disponibilidade

Um sistema...

Disponível: **não** está em estado de falha em um determinado instante de tempo

Confiável: **não** falha por um período de tempo



**Replicação (sw)
&
Redundância (hw)**

Paralelismo

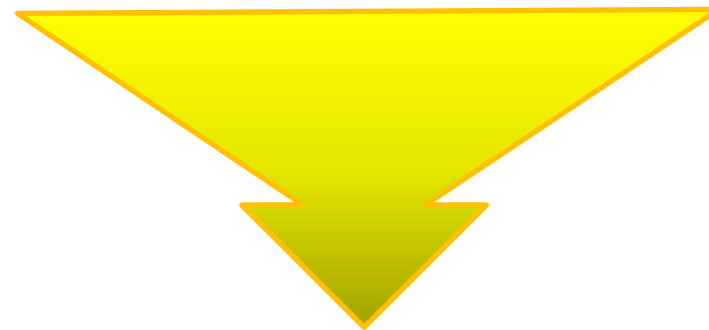


desempenho

Abertura



expansibilidade

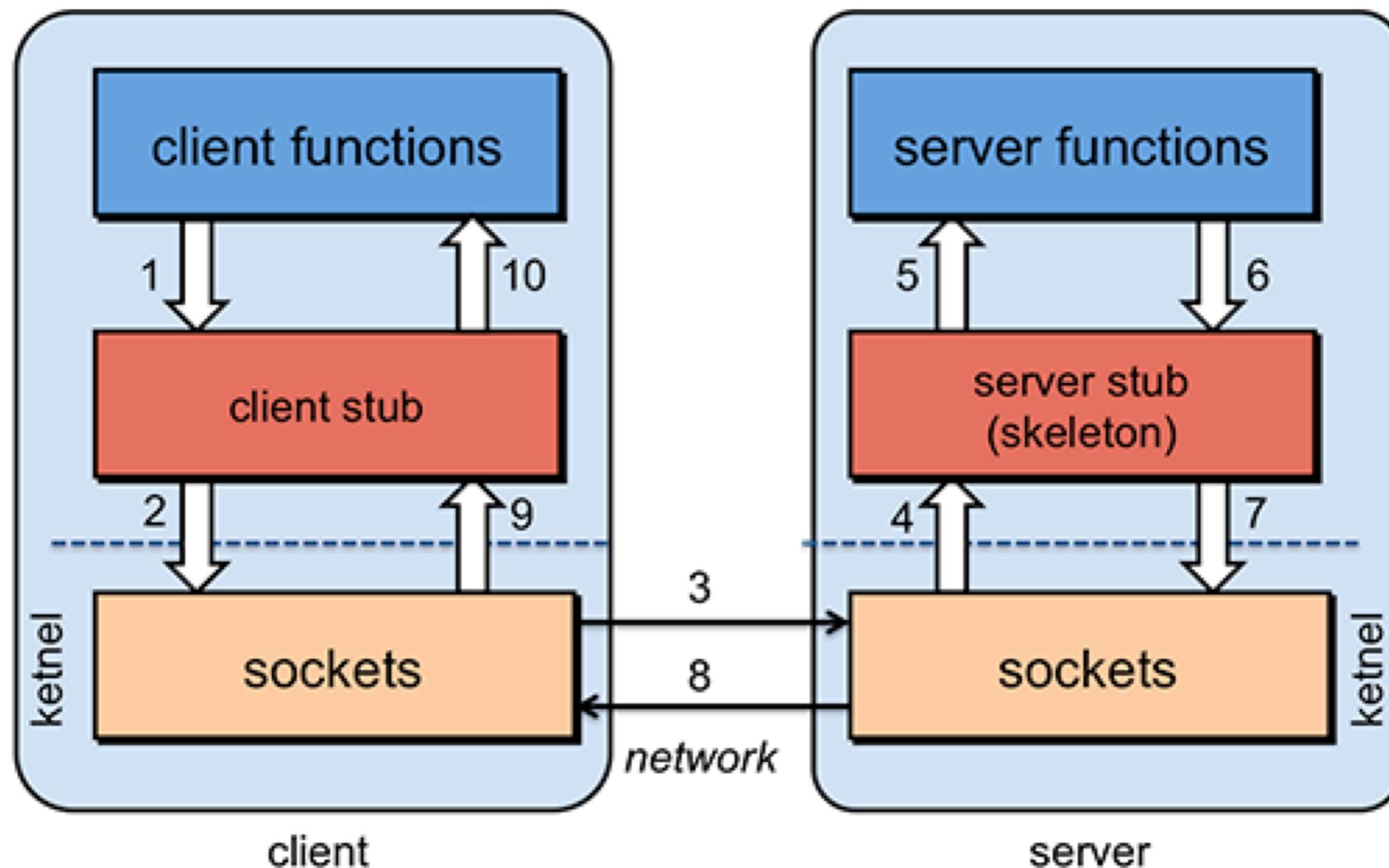


Desempenho “semelhante” para qualquer tamanho (escala)



Escalabilidade

Remote Procedure Call



Transparência



Middleware

Abstrações!!!