Arquitetura de Computadores

LEI – 2022/23 - DI-FCT/NOVA

Aula T3 – 10 de março de 2023

Sumário

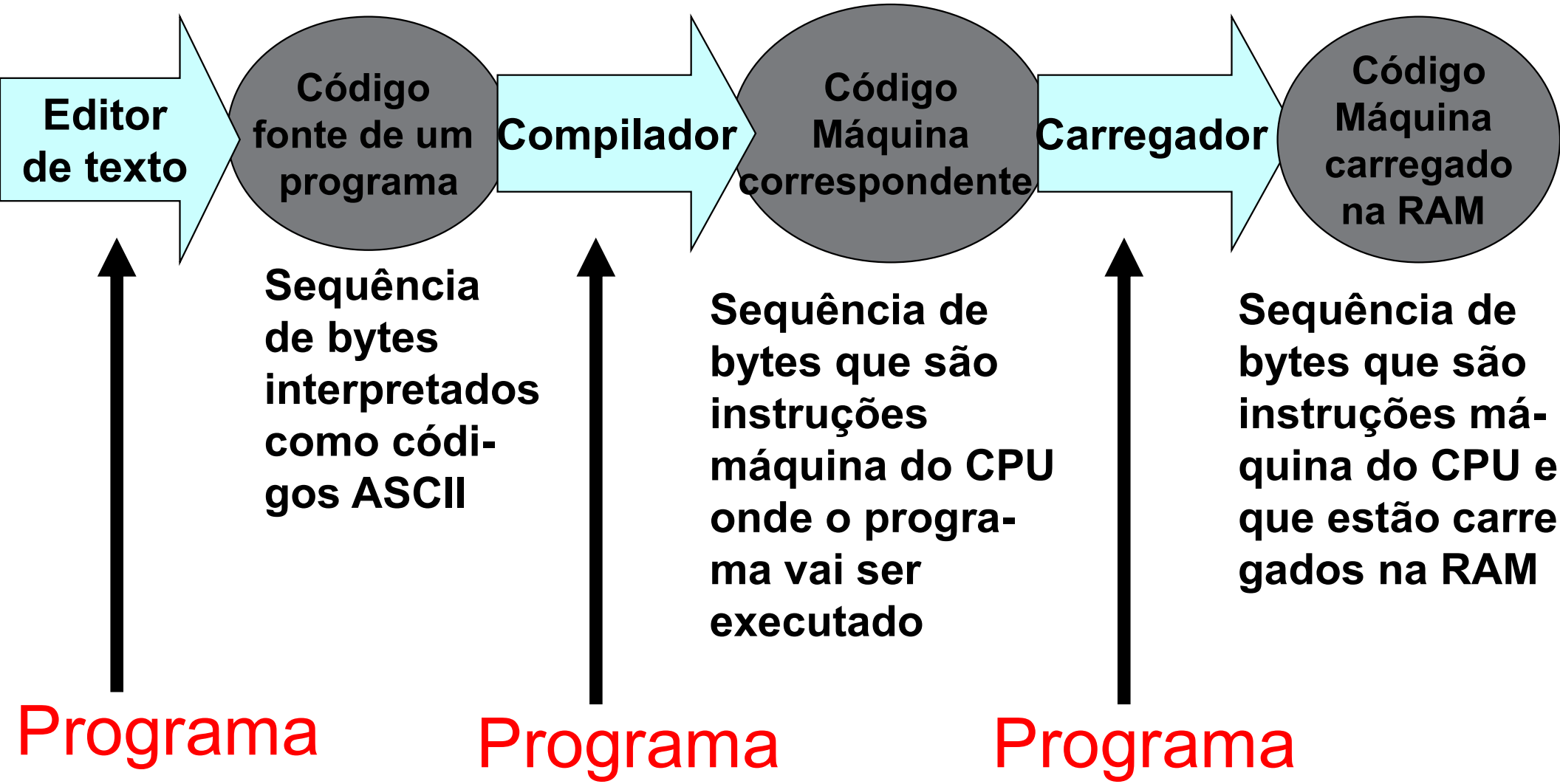
Funções do sistema operative:ficheiros e diretorias

Bibliografia: Operating Systems: Three EasyPieces Cap.2 e cap.39 (secção 1)

Funções do SO

- O sistema de operação (SO) é o componente do software de sistema que serve de suporte a todo software de aplicação e de sistema
- Os SOs controlam a execução de programas, a gestão de recursos, protecção e segurança
- O SO cria uma máquina virtual com instruções de alto nível que os programas podem usar fazendo chamadas ao sistema
- O SO impede o acesso directo aos periféricos e ficheiros pelos programas

Ciclo de vida de um programa

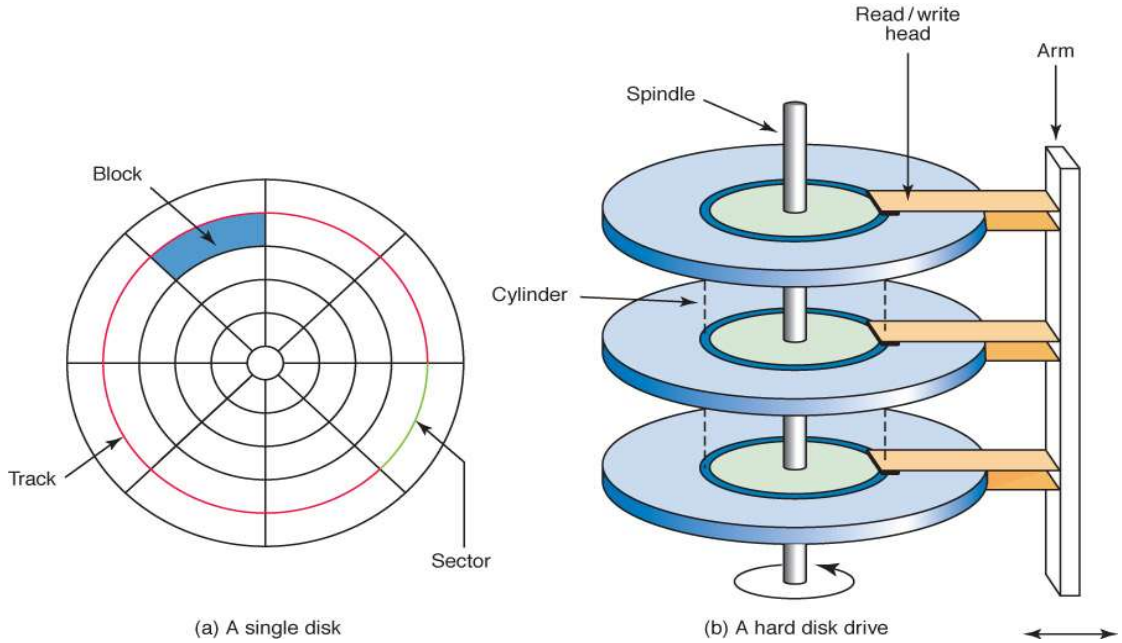


Ficheiro

- A informação é guardada de forma permanente.
- Um ficheiro é um espaço de endereçamento logicamente contíguo, contendo um conjunto de dados “inter-relacionados” e é acessível através de um identificador único
- Um ficheiro pode conter dados (texto, imagem, som, etc.) ou programas.
- O SO encarrega-se de gerir os ficheiros e os discos em que estes residem. A organização do disco é escondida aos programadores e utilizadores.

Onde guardar o código fonte e os programas?

- Em sectores do disco, porque é um dispositivo de armazenamento permanente
- Os discos são complicados!
- Disco
  - S superfícies, cada com P pistas, cada com S sectores
  - Pode ser visto como contendo N blocos todos do mesmo tamanho (por ex: 1024 bytes)



- Ficheiro: abstracção fornecida pelo sistema operativo que permite esconder a complexidade do disco
  - Nome que é uma cadeira de caracteres (ASCII)
  - Operações: abrir, fechar, ler, escrever
  - Atributos (tamanho, data de criação, …)
  - Blocos do disco onde está guardado (o último não está normalmente todo preenchido)

Ciclo de vida de um programa agora com ficheiros

Programador

Editor de texto

Ficheiro fonte do programa

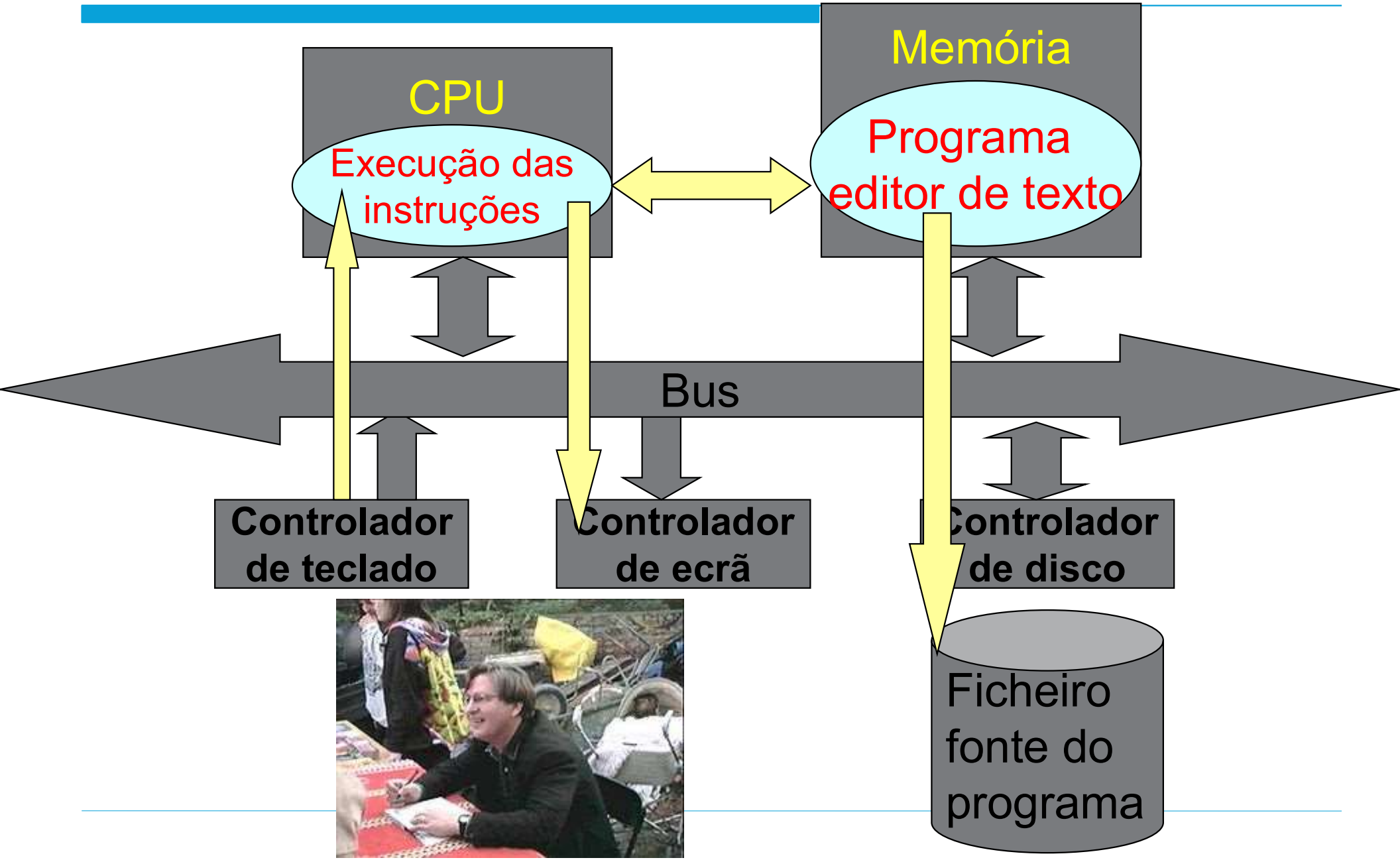
Compilador

Ficheiro executável

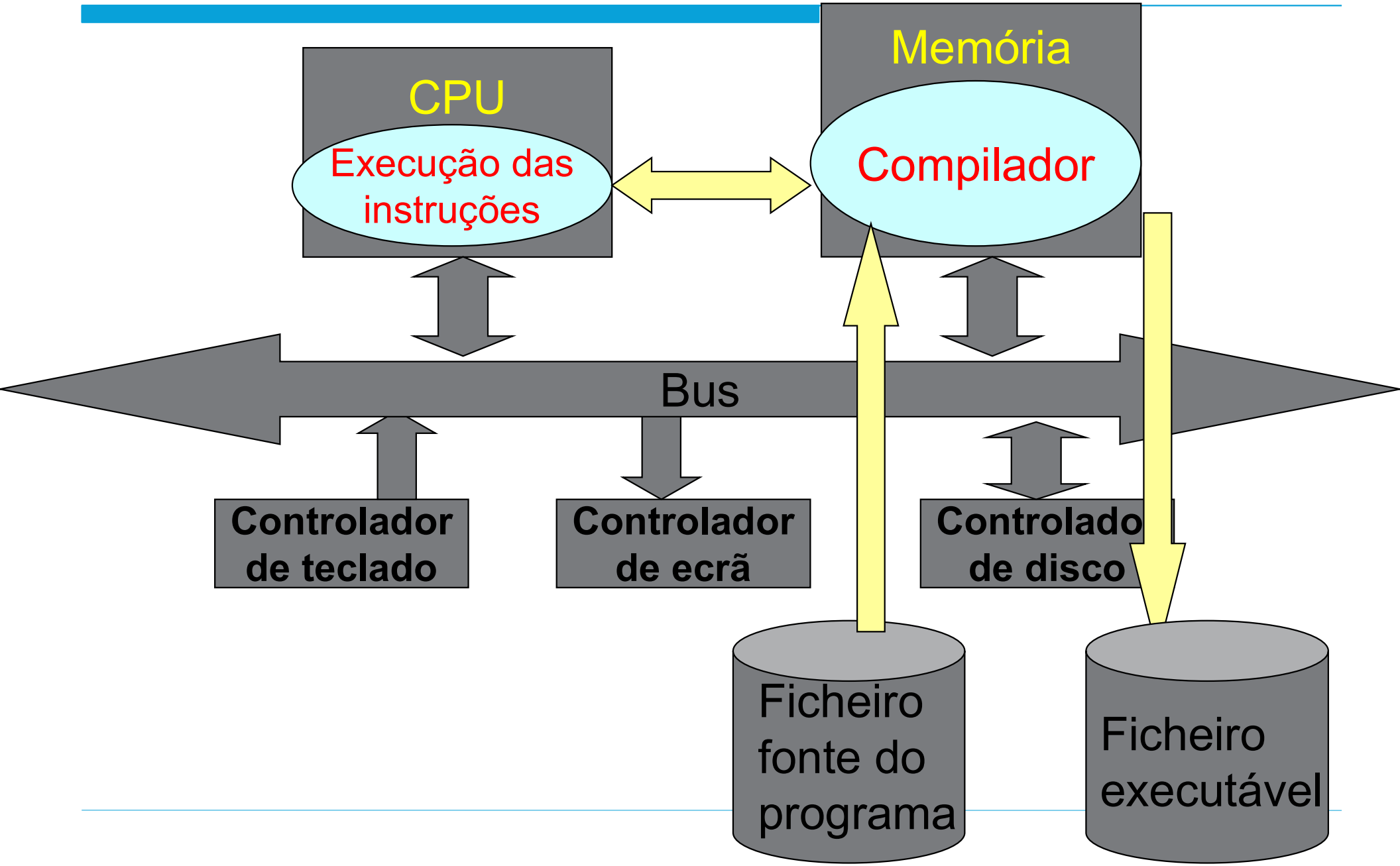
Carre-gador

Código carregado na RAM

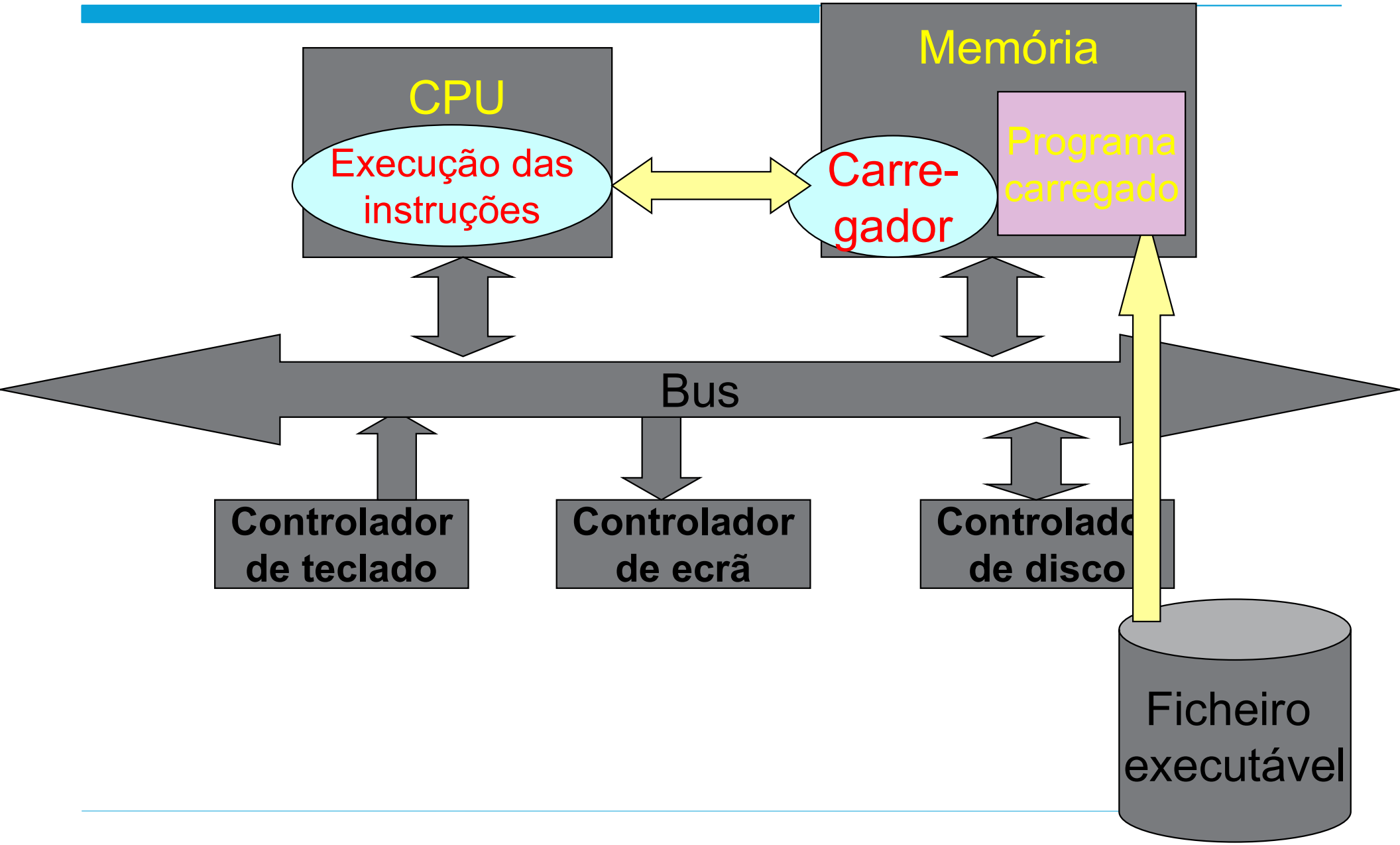
Edição do programa



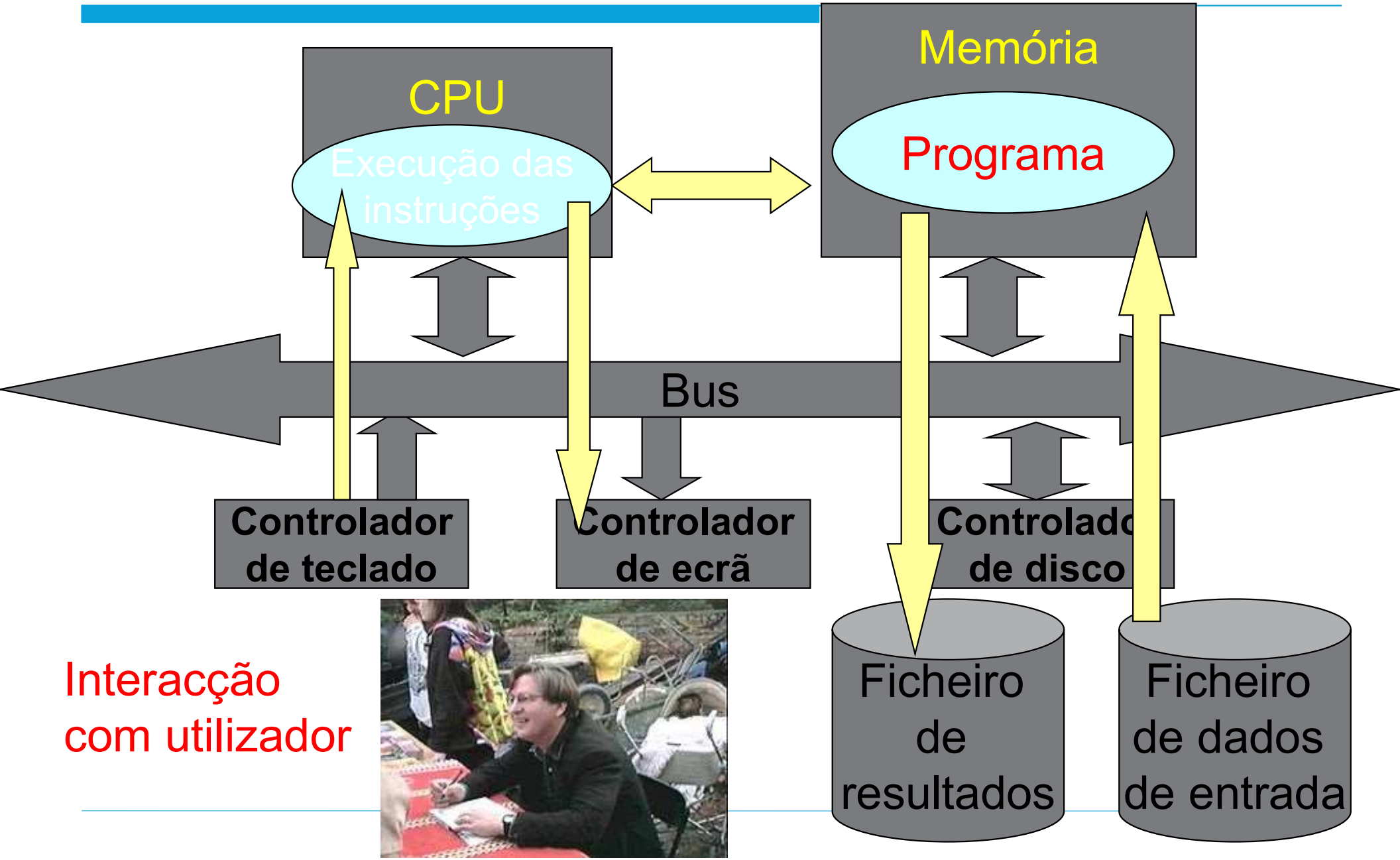
Geração do código



Carregamento



Execução



Interacção com utilizador

Executar Programas, como ?

É preciso oferecer a cada programa uma máquina com

- *CPU: para executar instruções*
- *RAM: para guardar código dados e pilha*
- *Canais de entrada e saída: Formas de comunicar com o exterior*

Executar Programas, como ?

- Virtualização: O sistema operativo cria as máquinas virtuais para cada programa executar a partir dos recursos físicos
  - CPUs
  - RAM
  - Dispositivos de entrada saída

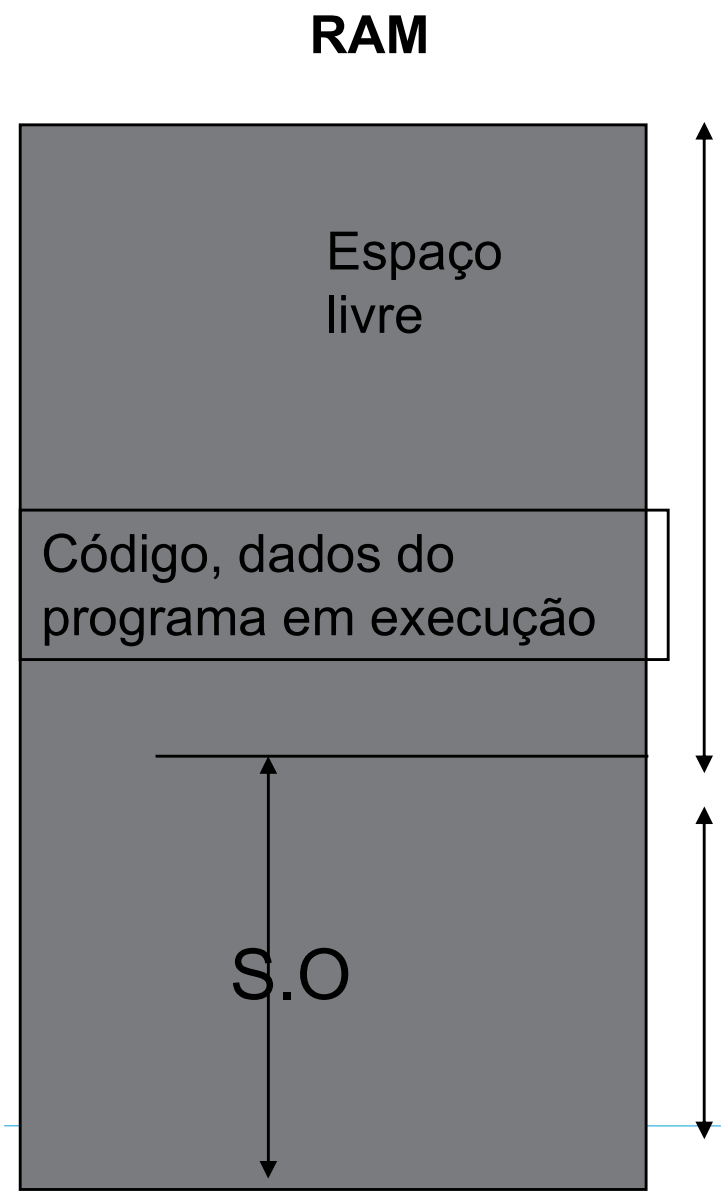
Duas Abstracções Suportadas peloSO

- Ficheiro
  - Disponibilizado ao programador como forma de esconder os detalhes de manipulação da memória persistente (discos)
- Processo
  - Disponibilizado ao programador como forma de esconder os detalhes da gestão de recursos necessários à execução de programas

Programa em execução ou Processo

- SO suporta para cada programa em execução
  - Um CPU virtual suportado na partilha do tempo do CPU (ou CPUs) reais
  - Um conjunto de memória (imagem do processo) obtida por partilha espacial da memória
  - Um conjunto de canais de entrada-saída que permitem enviar e receber bytes
    - De/Para Periféricos
    - De/Para Ficheiros

VIRTUALIZAÇÃO DA RAM



A repartição da RAM é feita espacialmente, sendo atribuídos aos processos faixas de endereços contíguos

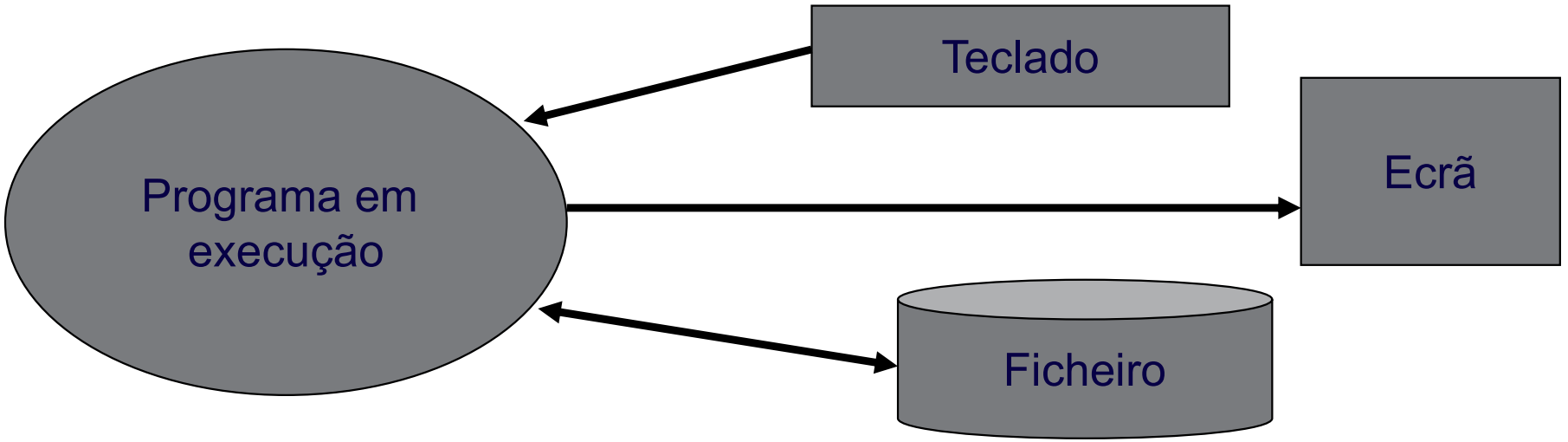
RAM disponível para o processo. Quando um processo é lançado ocupa memória; quando termina a memória é libertada

RAM ocupada pelo SO

Canais de entrada/saída

Um programa em execução vê periféricos e ficheiros através de canais de entrada saída

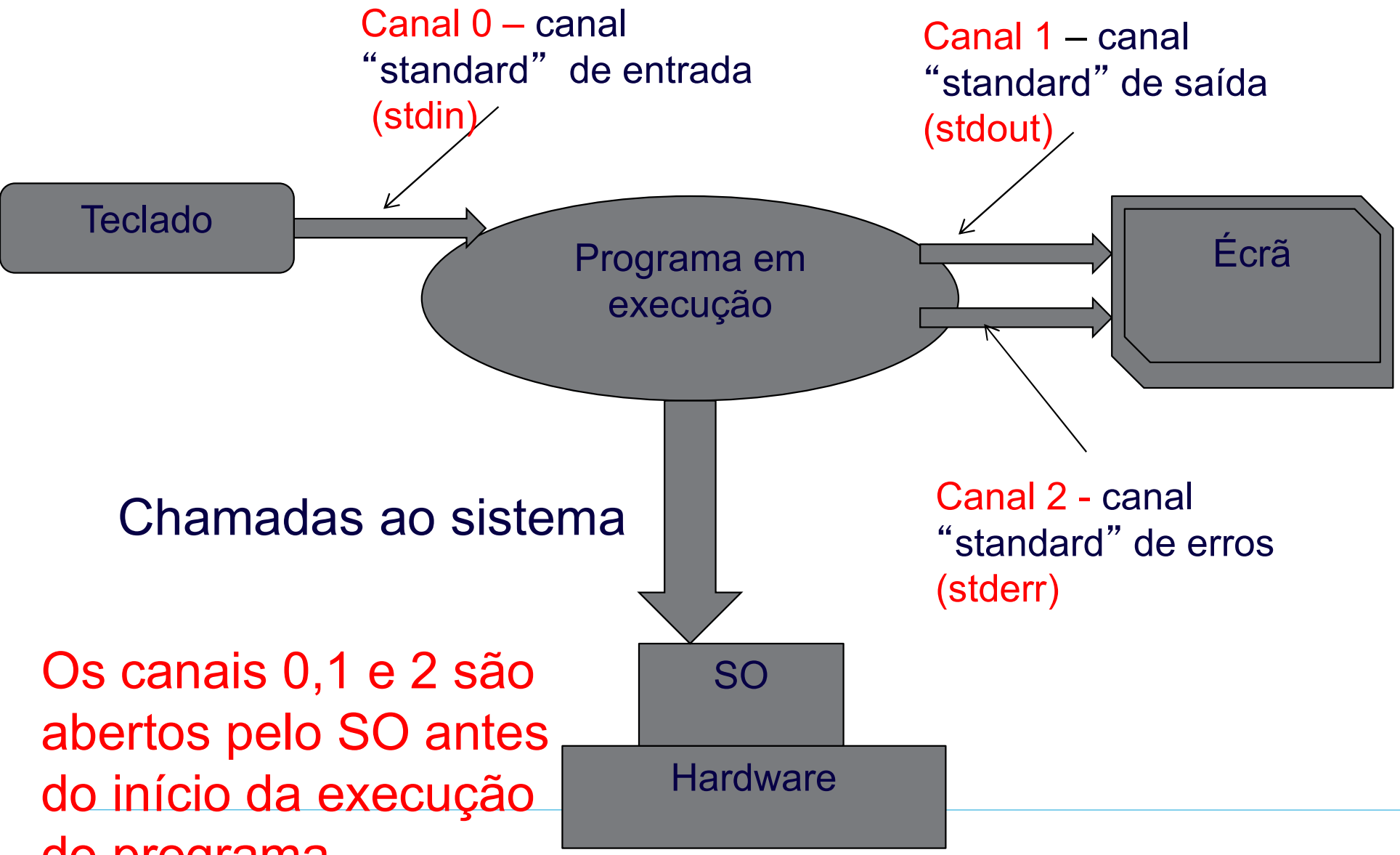
Esses canais permitem ler e escrever bytes de forma fiável e mantendo a ordem



Os canais são identificados por números: começam em 0 e vão crescendo à medida das necessidades

- Os canais 0 (stdin), 1 (stdout) e 2 (stderr) são abertos pelo SO quando o programa começa a executar
- o programa pode abrir mais canais usando a chamada open

Programa em execução



Os canais 0,1 e 2 são abertos pelo SO antes do início da execução do programa

Ficheiro

- A informação é guardada de forma permanente.
- Um ficheiro é um espaço de endereçamento logicamente contíguo, contendo um conjunto de dados “inter-relacionados” e é acessível através de um identificador único
- Um ficheiro pode conter dados (texto, imagem, som, etc.) ou programas.
- O SO encarrega-se de gerir os ficheiros e os discos em que estes residem. A organização do disco é escondida aos programadores e utilizadores.

Discos e afins

- Armazenamento permanente
- Discos rígidos magnéticos
  - Mecânico: rotação do disco e deslocamento da cabeça para ler e escrever informação.
- “Discos” de estado sólido
  - Eletrónico: acesso imediato a qualquer posição para ler e escrever.
  - Mais rápidos que os discos rígidos magnéticos.
  - Exemplos: cartão SD, pen, disco SSD.

Ficheiro

- Um ficheiro é constituído por uma sequência de bytes que representam dados (texto, imagem, som, etc.) ou programas.
- A informação é guardada de forma permanente e é acessível através de um identificador único (o nome).
- O nome de um ficheiro é uma sequência de caracteres (uma string em linguagens de programação como o C, Java, Python).
- O SO encarrega-se de gerir os ficheiros e os discos em que estes residem. A organização do disco é escondida aos programadores e utilizadores.

Tipos de ficheiros

- Ficheiros de texto
  - Contêm apenas bytes que podem ser interpretados como caracteres imprimíveis (letras, algarismos e sinais de pontuação) ou de formatação (por exemplo, fim-de-linha).
  - A codificação usada é a ASCII ou outra (ISO-8859, UNICODE/UTF).
- Ficheiros binários
  - A interpretação do seu conteúdo está a cargo do programa que o utiliza; os bytes representam qualquer valor.
  - Podem ser dados (inteiros, reais), programas (códigos máquina de um dado CPU), sons (segundo uma codificação como o MP3), imagens (codificadas num formato como o JPEG), filmes (no formato MPEG, por exemplo), etc.

Ficheiro imprime.py

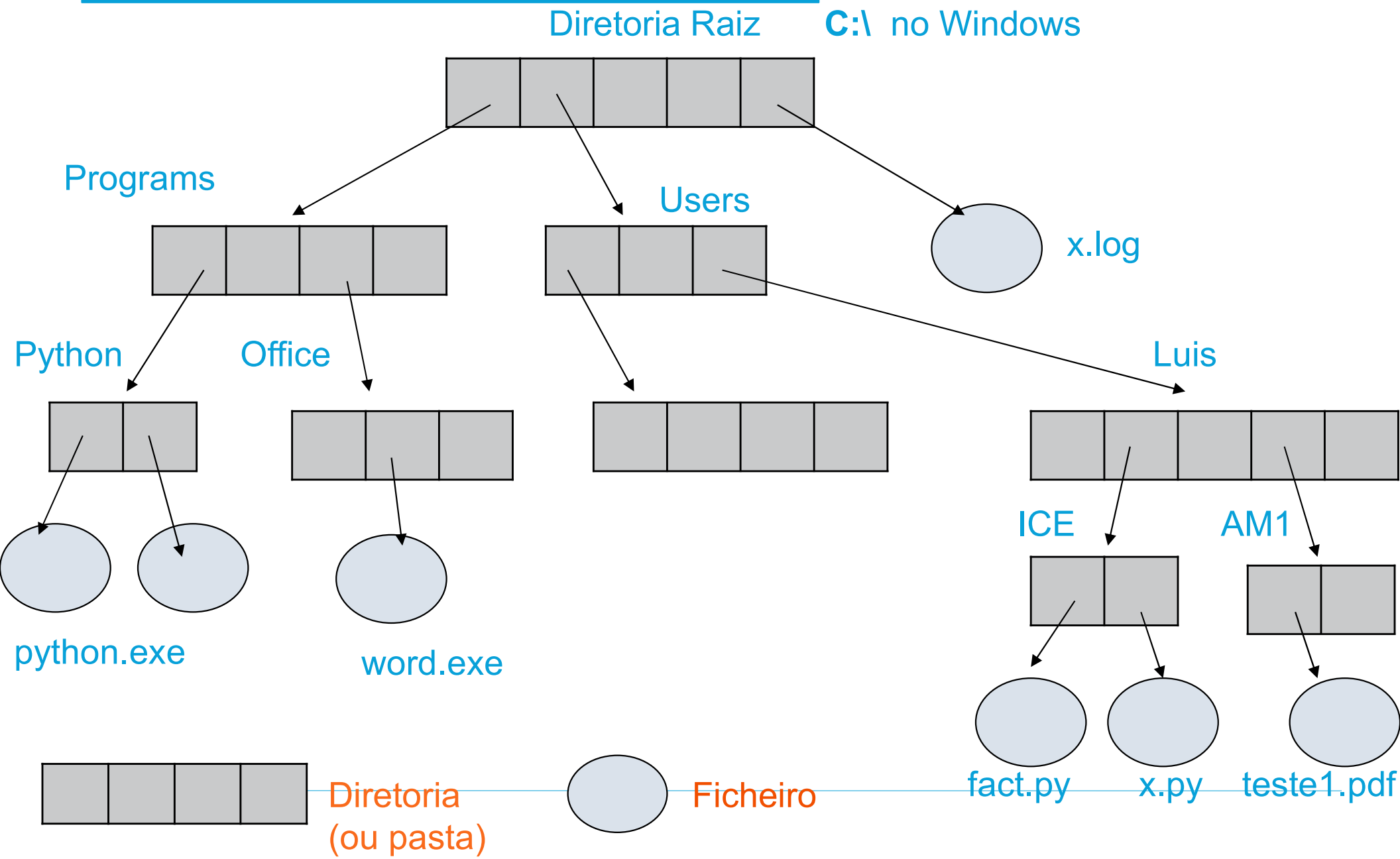
- Considere o ficheiro imprime.py com um programa na linguagem Python (que escreve mensagens no ecrã).

```
def imprime( n ):
    for i in range( n ):
        print( 'hello' )
```

imprime.py codificado em ASCII

d 100	e 101	f 102	 32	i 105	m 109	p 112	r 114	i 105	m 109
e 101	(40	<sp> 32	n 110	<sp> 32	) 41	: 58	<nl> 10	<tab> 9	f 102
o 111	r 114	<sp> 32	i 105	<sp> 32	i 105	n 110	<sp> 32	r 114	a 97
n 110	g 103	e 101	(40	<sp> 32	n 110	<sp> 32	) 41	<sp> 32	: 58
<nl> 10	<tab> 9	<tab> 9	p 112	r 114	i 105	n 110	t 116	(40	<sp> 32
' 39	h 104	e 101	l 108	l 108	o 111	' 39	<sp> 32	) 41	<nl> 10

Sistema de ficheiros [1]



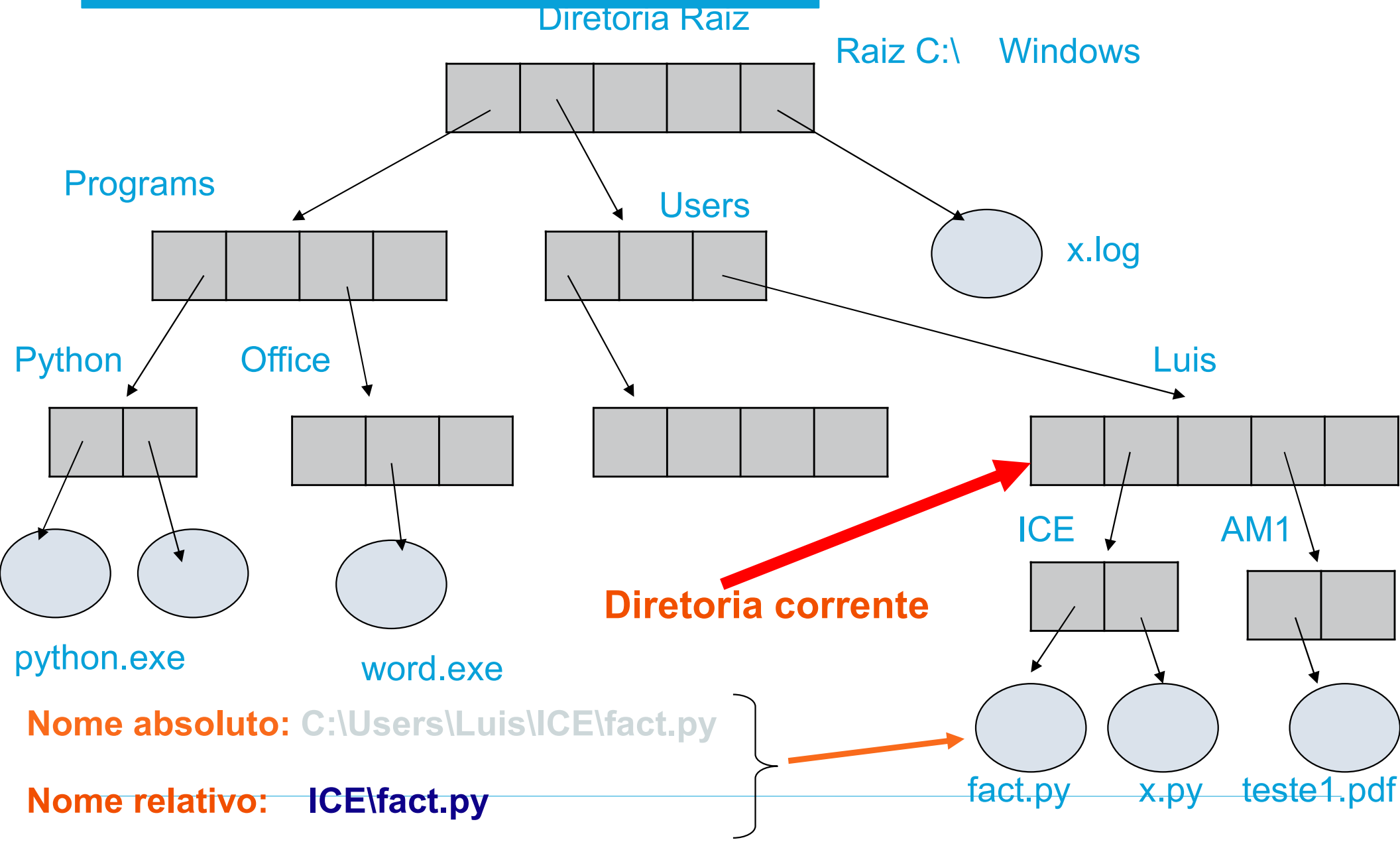
Sistema de ficheiros [2]

- O sistema de ficheiros está organizado hierarquicamente em forma de árvore (“invertida”).
  - Raiz (a diretoria do topo, da qual “descendem” as outras diretorias e os outros ficheiros).
    - **C:** no Windows
  - Ramos (as outras diretorias).
  - Folhas (ficheiros).
- Cada diretoria (ou diretório, ou pasta) pode conter ficheiros e outras (sub-)diretorias.
- Nesta árvore, a pesquisa de um ficheiro é eficiente.

Sistema de ficheiros [3]

- O nome completo ou caminho para o ficheiro (*pathname*) pode ser:
  - absoluto – o caminho completo a partir da diretoria raiz;
  - relativo – o caminho a partir da **diretoria corrente**.
- Nos caminhos, os nomes de subdiretorias e ficheiros são separados por um carácter especial (**** em Windows, */* em macOS e Linux).
- Notações especiais (que podem ser usadas nos caminhos):
  - "**.**" representa a diretoria corrente;
  - "**..**" representa a diretoria pai da diretoria corrente.

Sistema de ficheiros [4]



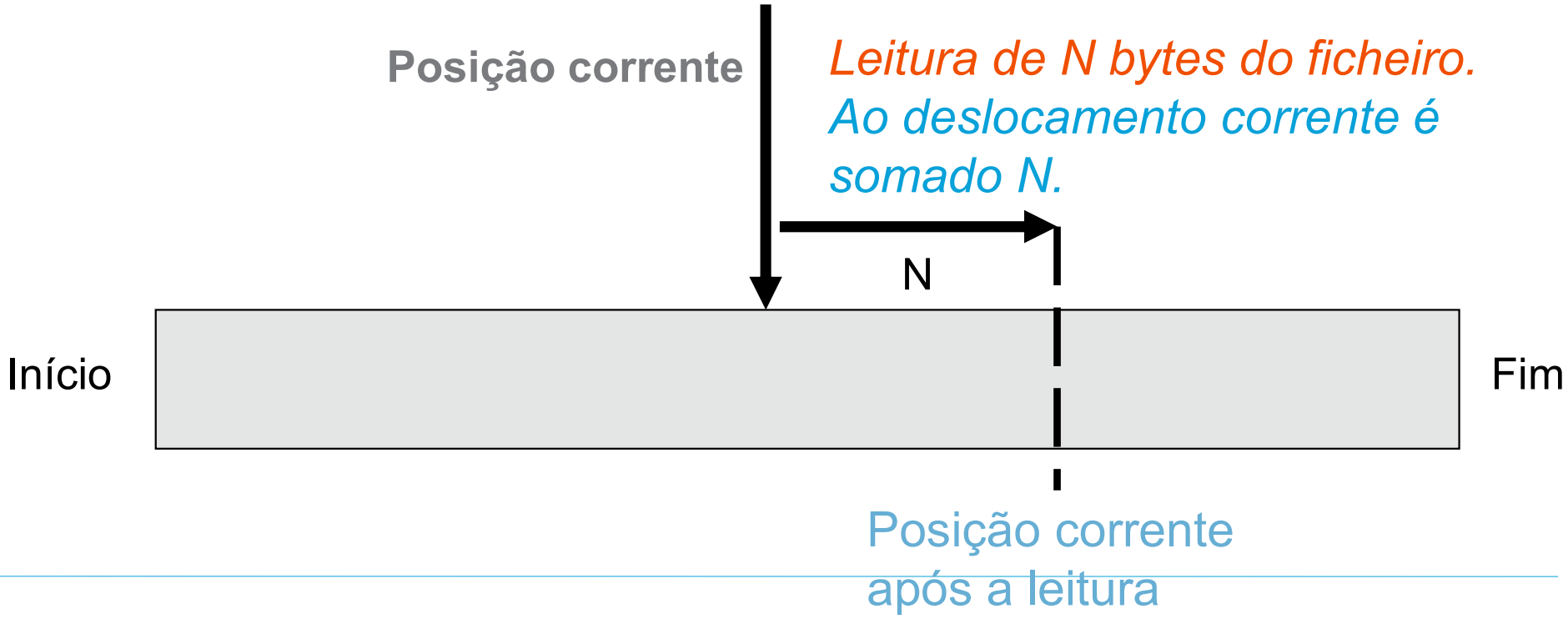
Nome absoluto: C:\Users\Luis\ICE\fact.py

Nome relativo: **ICE\fact.py**

Ficheiros – Abertura … fecho

- Para ler ou escrever num ficheiro, é necessário começar por "abrir" o ficheiro. No fim das leituras/escritas, é necessário "fechar" o ficheiro. À medida que se lê ou se escreve, a posição corrente avança.

  Não se lê o que já foi lido; escreve-se a seguir ao que já foi escrito.



Abertura de um ficheiro

canal = **open**(nomeFicheiro, modo')

- Abre o ficheiro de texto nomeFicheiro para:
  - leitura
  - para escrita
- A função open devolve um índice na tabela de canais abertos que permite o acesso ao ficheiro.
- Se o ficheiro não existir, a execução lança um erro.

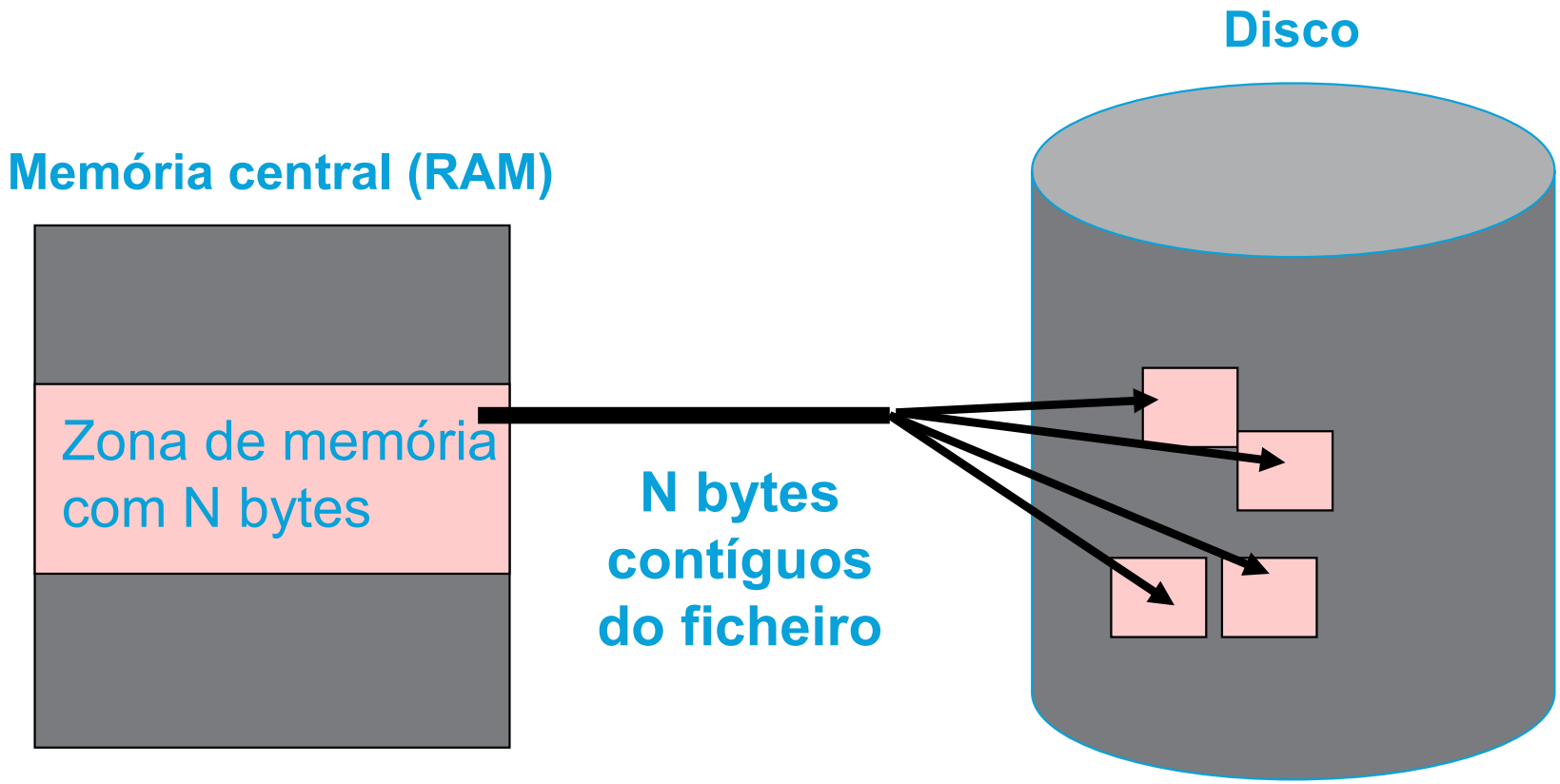
Fecho de um ficheiro

close(canal)

- Usando o canal devolvido pelo open, após usar o ficheiro deve-se fechá-lo, usando a função

Escrita num ficheiro

- A informação guardada na memória central é transferida para o disco.



Escrita num ficheiro

nbytes = **write**(canal, bytes a transferir)

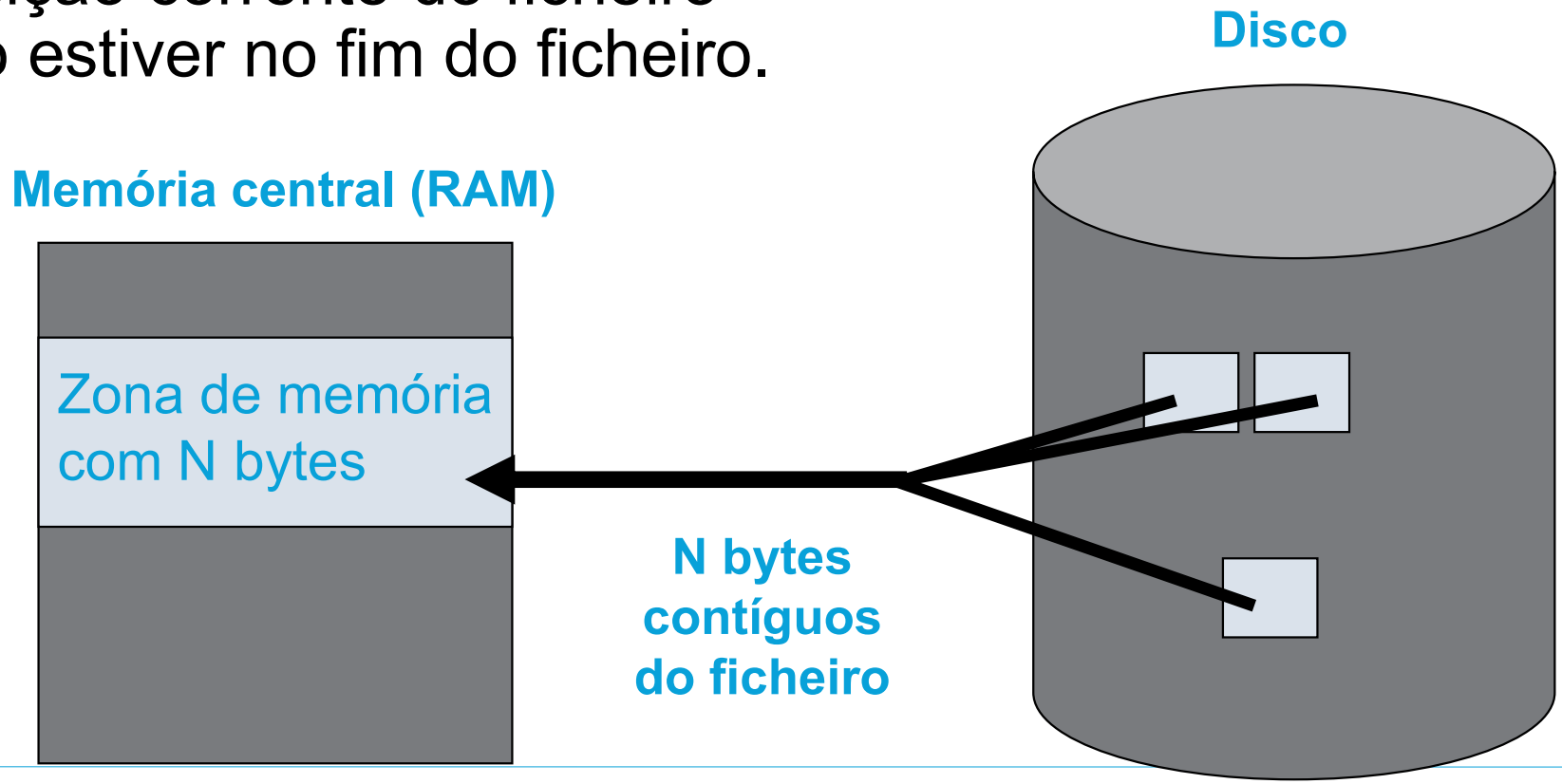
- A função **write** escreve uma sequência de bytes no ficheiro

Caracteres especias nas strings

- Alguns caracteres especiais de formatação:
  - `\n` mudança de linha
  - `\t` tab

Leitura de um ficheiro

- A informação guardada no disco é transferida para a memória central.
- Só se podem ler dados se a posição corrente do ficheiro não estiver no fim do ficheiro.



Leitura de um ficheiro (2)

read(canal, espaço para transferir os bytes)