NOVA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

Fundamentos de Sistemas de Operação

LEI - 2023/2024

Vitor Duarte
Mª. Cecília Gomes

1

Aula 17

- Ficheiros: aspetos da implementação do open, read e fork.
- Referências para blocos nos inodes
- Cache e buffer de blocos

- OSTEP: cap. 39, 40

2

Acesso a um ficheiro

- 1º passo - pedir ao SO acesso ao ficheiro: open

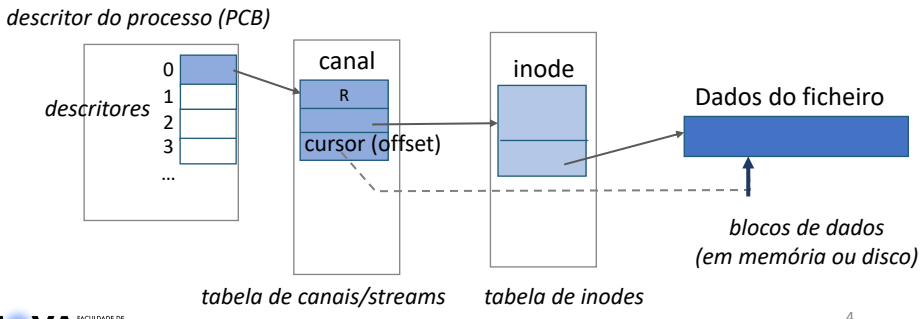
```
int open(char *filename, int flags)
```

- Permite que o SO:
  - Verificar se o ficheiro existe, e obter inode
  - Verificar se o processo pode usar o ficheiro
  - Iniciar um novo canal (um cursor/*offset* de posição no ficheiro, inode em memória, *buffers, etc*)
  - Fixar se é para leitura ou/e escrita
- Depois - as restantes operações serão mais fáceis (usam descritor que referencia o ficheiro aberto)

3

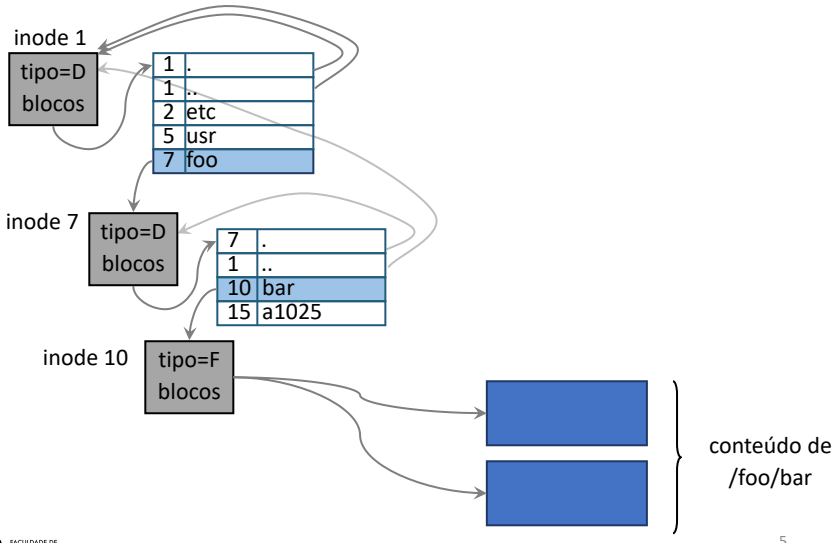
Open e canais/descritores

- Após cada open com sucesso, é usada uma entrada numa tabela de ficheiros abertos pelo processo
- *File Descriptor*: o número dessa entrada na Tabela, usado nas restantes operações
- O SO faz cache do que lê dos discos (inodes e dados)



4

Caminho /foo/bar



5

Open + read de /foo/bar

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data[0]	bar data[1]	bar data[2]
open(bar)			read			read				
				read			read			
					read					
read()					read			read		
					write					
read()					read				read	
					write					
read()					read					read
					write					

t

Figure 40.3: File Read Timeline (Time Increasing Downward)

6

Create + write de `/foo/bar`

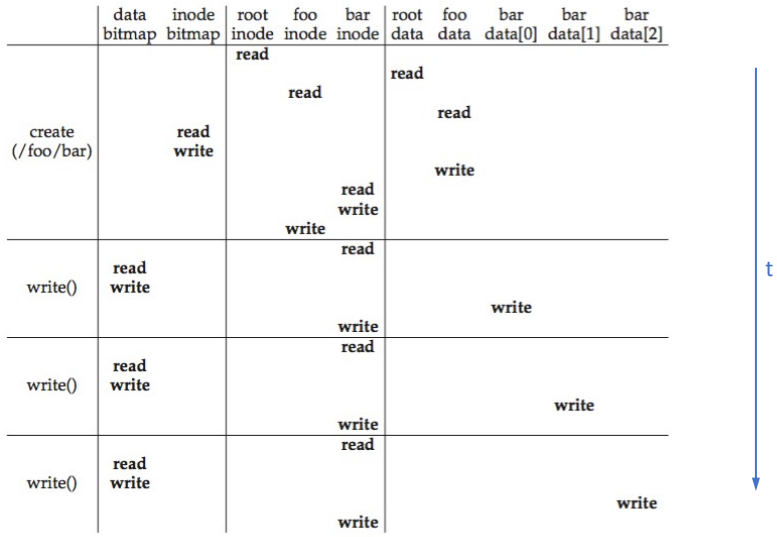
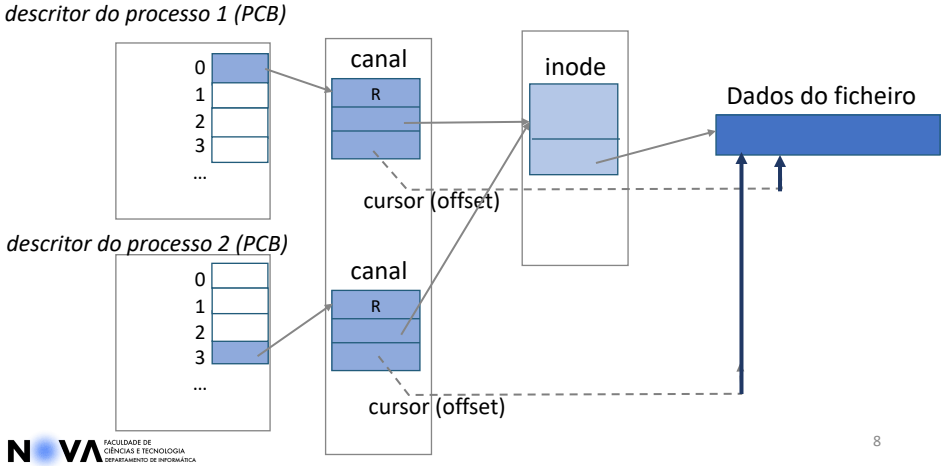


Figure 40.4: File Creation Timeline (Time Increasing Downward)

7

Novo open do mesmo ficheiro

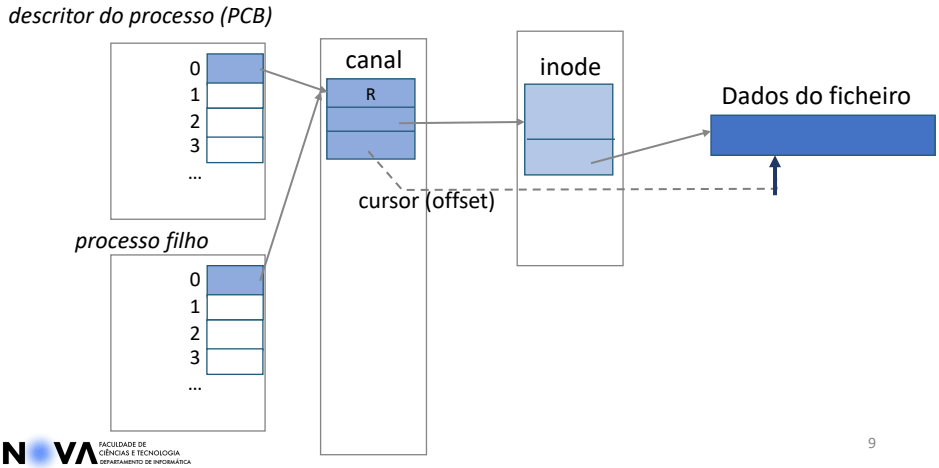
- Outro processo abre o mesmo ficheiro
- Cada um lê a sua sequência de bytes dentro da sua sessão



8

Caso do fork

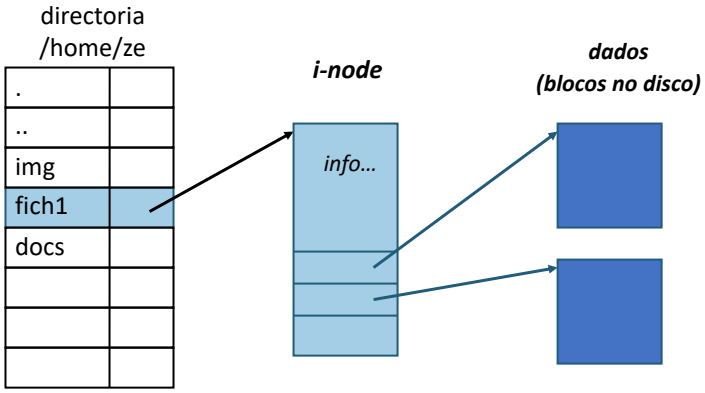
- O processo faz fork
  - Os processos partilham o canal, logo, a posição no ficheiro



9

Indicar os blocos nos inodes

- Referências diretas
  - O inode tem uma tabela com todos os nº dos blocos



10

Capacidades - exemplo

Size	Name	What is this inode field for?
2	mode	can this file be read/written/executed?
2	uid	who owns this file?
4	size	how many bytes are in this file?
4	time	what time was this file last accessed?
4	ctime	what time was this file created?
4	mtime	what time was this file last modified?
4	dtime	what time was this inode deleted?
2	gid	which group does this file belong to?
2	links_count	how many hard links are there to this file?
4	blocks	how many blocks have been allocated to this file?
4	flags	how should ext2 use this inode?
4	osd1	an OS-dependent field
60	block	a set of disk pointers (15 total)
4	generation	file version (used by NFS)
4	file_acl	a new permissions model beyond mode bits
4	dir_acl	called access control lists

Figure 40.1: **Simplified Ext2 Inode**

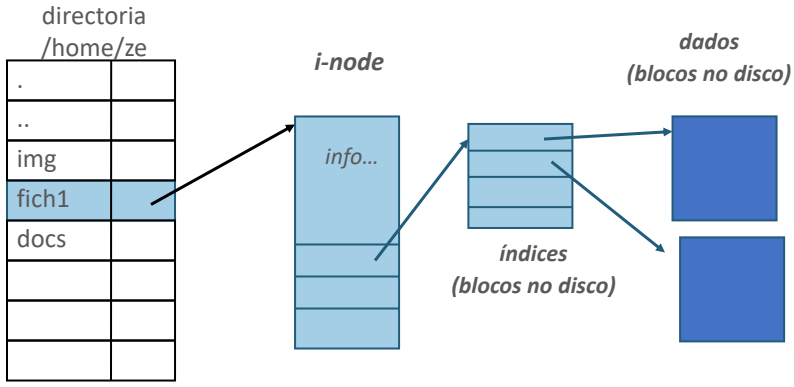
soma: 112 bytes

- Se blocos de 4Kbytes,
- se cada inode ocupar 256 bytes, temos 16 inodes por bloco e cada inode tem mais 144 bytes para indexar blocos,
- cada inode pode guardar 36 endereços de 32bits
- tamanho máximo possível para os ficheiros 144Kbytes e o volume pode ter um máximo de

11

Indicar os blocos nos inodes

- Referências indiretas (blocos de referências)
  - O inode tem uma tabela com nº dos blocos que contém os nº dos blocos de dados (referências indiretas)



12

Capacidades - exemplo

Size	Name	What is this inode field for?
2	mode	can this file be read/written/executed?
2	uid	who owns this file?
4	size	how many bytes are in this file?
4	time	what time was this file last accessed?
4	ctime	what time was this file created?
4	mtime	what time was this file last modified?
4	dtime	what time was this inode deleted?
2	gid	which group does this file belong to?
2	links_count	how many hard links are there to this file?
4	blocks	how many blocks have been allocated to this file?
4	flags	how should ext2 use this inode?
4	osd1	an OS-dependent field
60	block	a set of disk pointers (15 total)
4	generation	file version (used by NFS)
4	file_acl	a new permissions model beyond mode bits
4	dir_acl	called access control lists

Figure 40.1: **Simplified Ext2 Inode**

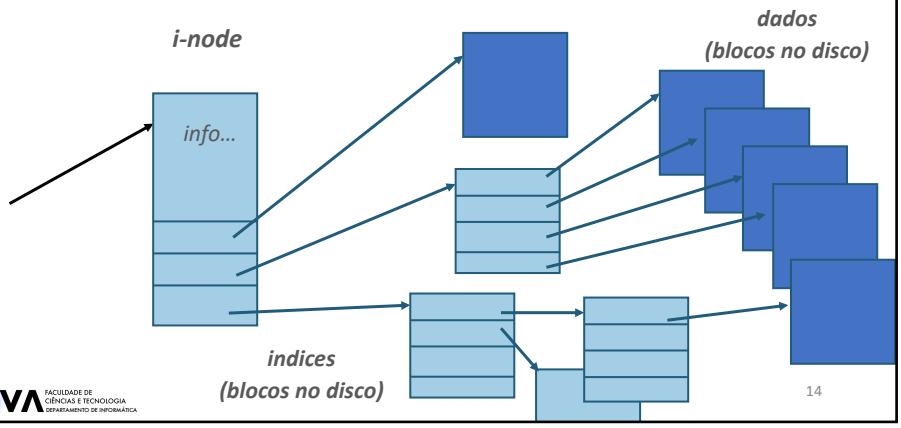
soma: 112 bytes

- Se blocos de 4Kbytes,
- se cada inode ocupar 256 bytes, temos 16 inodes por bloco e cada inode tem mais 144 bytes para indexar blocos,
- cada inode pode guardar 36 endereços de 32bits
- tamanho máximo possível para os ficheiros 36*4K/4 blocos = 144 Mbytes

13

Indicar os blocos nos inodes

- Referências diretas e indiretas
  - O inode tem uma tabela com nº blocos dos dados (diretos) e as duas últimas entradas para um bloco indireto e um bloco duplamente indireto, ...



14

Capacidades - exemplo

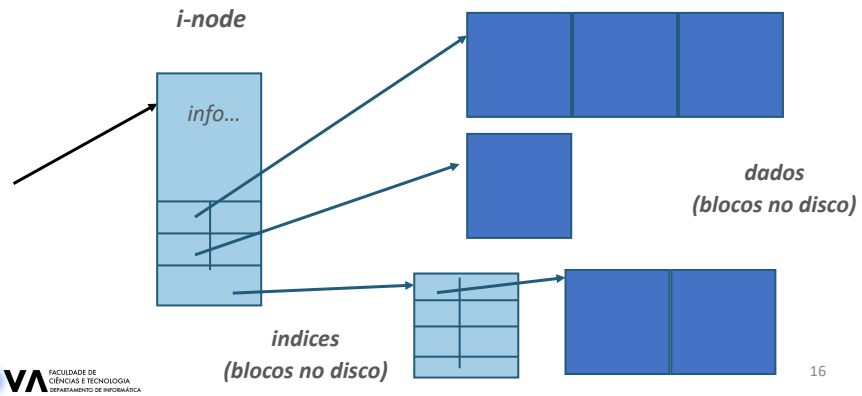
- Se blocos de 4Kbytes,
- se cada inode ocupar 256 bytes, temos 16 inodes por bloco e cada inode tem mais 144 bytes para indexar blocos,
- cada inode pode guardar 36 endereços de 32bits
- 34 blocos diretos
- + 1K blocos indiretos
- + 1K*1K blocos duplamente indiretos
- tamanho máximo possível para os ficheiros:

(34+1K+1M)*4K ≈ 4Gbytes

15

Indicar os blocos nos inodes

- *Extents*:
  - O inode tem uma tabela com ínicio e fim de sequências contíguas de blocos
  - pode ainda ser combinado com blocos indiretos com *extents*.



16

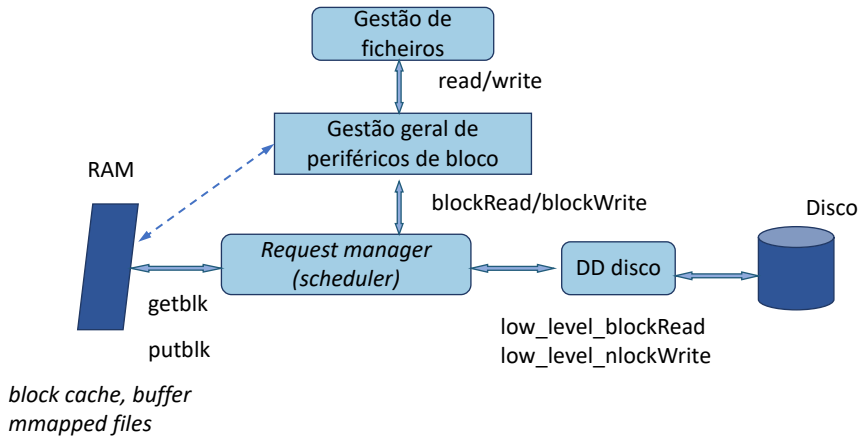
Eficiência do sistema de IO

- Dispositivos independentes uns dos outros e autónomos em relação ao CPU
- Eficiência obtida à custa da sobreposição (overlapping) da execução pelo CPU e pelos dispositivos
  - Necessidade de o CPU responder rapidamente aos pedidos dos periféricos (interrupções)
  - Por outro lado, o tratamento desses pedidos não deve ocupar muito tempo de CPU (DMA, rotinas pequenas)
  - SO deve suportar o assincronismo entre os processos e os dispositivos
    - Utilização e partilha de "buffers/cache", ...
    - Escalonamento de operações de entrada/saída dobre os volumes

Periféricos tipo bloco

- Tipicamente discos rígidos – latência elevada, taxa de transferência melhor se sequencial
- Muitas vezes repetem-se acessos ao mesmo bloco
  - Por exemplo, o inode e o bloco da directoria raiz
- Guardar os blocos em memória reduz o número de acessos ao disco
- Atrasar as escritas permite juntar pedidos e escalonar estes para melhor desempenho
- A cache/buffer de blocos (*block buffer cache*) tem duas funções:
  - Reservatório (pool) de **buffers** para E/S em curso
  - **cache** para operações de E/S já terminadas
- O *request manager* gere a leitura e escrita de conteúdos de blocos do disco de/para cache/buffers.

Cache/buffer de blocos do disco



block cache, buffer
mmapped files