



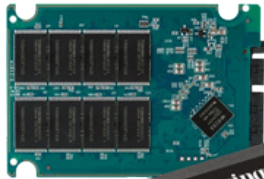
Sistemas de ficheiros

Patrício R. Domingues

Departamento de Eng
Informática

ESTG/Politécnico de Leiria

- Conteúdo adaptado de:
 - CS 5600 - Computer Systems - Professor Christo Wilson, Files and Directories
<https://cbw.sh/5600/index.html>



UNIDADES DE ARMAZENAMENTO SI VS. IEC



Unidade de armazenamento(#1)

- ✓ Unidade de armazenamento – octeto (byte)
 - Um octeto tem 8 bits
 - Historicamente, tamanho do *byte* esteve associado ao hardware
 - Existiram bytes de 6 bits e de 9 bits (~1960)
 - Hoje é aceite *byte* = 8 bits
 - O termo **octeto** evita ambiguidade
 - Unidade byte: **B** (maiúsculo)
 - Unidade bit: **b** (minúsculo)

✓ Múltiplos do byte. Por exemplo, o que é um *kilobyte*?

- Sistema Internacional (SI):
 - Sufixo kilo = 1000 = 10^3
 - Exemplos: 1 km = 1000 mts; 1 KV = 1000 volts, 1 KG = 1000 grs, ...
- Mas...os computadores são dispositivos binários
 - Armazenamento é medido em potências de 2
 - $2^{10} = 1024$
 - 1 KB = 1024 bytes

Qual é a unidade correta? >>

✓ KiloByte (prefixo Kilo)

- Sistema Internacional (SI)
- 1 KB = 1000 bytes

✓ Potências de 2

- Definido pelo International Electrotechnical Commission (IEC) – IEC60027-2
- $2^{10} = 1024$ bytes
- Nome: Ki**i**Byte
- 1 KiB = $2^{10} = 1024$ bytes

Multiples of bytes						V · T · E
Decimal			Binary			
Value		Metric	Value	JEDEC	IEC	
1000	kB	kilobyte	1024	KB kilobyte	KiB kibibyte	
1000 ²	MB	megabyte	1024 ²	MB megabyte	MiB mebibyte	
1000 ³	GB	gigabyte	1024 ³	GB gigabyte	GiB gibibyte	
1000 ⁴	TB	terabyte	1024 ⁴	- -	TiB tebibyte	
1000 ⁵	PB	petabyte	1024 ⁵	- -	PiB pebibyte	
1000 ⁶	EB	exabyte	1024 ⁶	- -	EiB exbibyte	
1000 ⁷	ZB	zettabyte	1024 ⁷	- -	ZiB zebibyte	
1000 ⁸	YB	yottabyte	1024 ⁸	- -	YiB yobibyte	
Orders of magnitude of data						

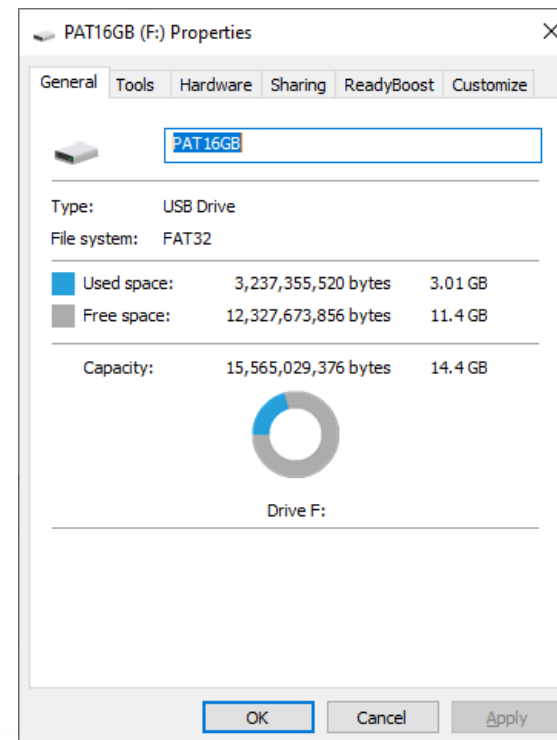
<http://en.wikipedia.org/>

- ✓ Vendedores de dispositivos de armazenamento usam unidades SI
- ✓ SO e outro software usam unidades BI (KiB, MiB, GiB, etc.) definidos pelo IEC
- ✓ Exemplo
 - Pen USB vendida com 16 GB
 - Tem 14.4 GiB



When we say	but mean	we're this far off
1 kilobyte	2^{10} bytes	2.4%
1 megabyte	2^{20} bytes	4.9%
1 gigabyte	2^{30} bytes	7.4%
1 terabyte	2^{40} bytes	10.0%
1 petabyte	2^{50} bytes	12.6%
1 exabyte	2^{60} bytes	15.3%

<http://cnx.org/contents/q4PmFDpc@1/Prefixes-for-binary-multiples>



- ✓ 1 KiB = 2^{10} bytes
- ✓ 1 MiB = 2^{20} bytes
- ✓ 1 GiB = 2^{30} bytes
- ✓ 1 TiB = 2^{40} bytes
- ✓ 1 PiB = 2^{50} bytes
- ✓ Exemplos
 - ✓ 2^{21} bytes? $2^{21} = 2^1 \times 2^{20} \Leftrightarrow 2 \text{ MiB}$
 - ✓ 2^{45} bytes? $2^{45} = 2^5 \times 2^{40} \Leftrightarrow 32 \text{ TiB}$
 - ✓ 2^{59} bytes? $2^{59} = 2^9 \times 2^{50} \Leftrightarrow 512 \text{ PiB}$

SISTEMAS DE FICHEIROS (SF)

Sistema de ficheiros

- ✓ Um dispositivo de armazenamento (e.g., HDD/SSD/Stick USB, etc.) é um conjunto de blocos vazios
 - Como são armazenados ficheiros nesses dispositivos?
 - Como é mantida a informação referente à localização de cada ficheiro?
 - Como se consegue o melhor desempenho possível?
 - Como se mantém a consistência e coerência dos dados em face de situações de falha?
 - SO “crasha”, falha de energia elétrica, etc.
 - **Resposta**
 - Organização do disco e Sistema de ficheiros

- ✓ Particionamento e *mounting*
- ✓ Sistema **FAT**
- ✓ Sistema **ext**: *inodes* e blocos
- ✓ Sistema **ext2**: grupos de blocos
- ✓ Sistema **ext3**: *journaling*
- ✓ Sistema **ext4**: extensões e *b-trees*

PARTICIONAMENTO DO DISCO

Arranque do SO

- ✓ Uma das primeiras tarefas do SO durante a inicialização (*bootup*) é ativar o sistema de ficheiros *root*. Passos:
 1. Localizar os dispositivos de boot
 - Discos internos e externos (HDD, SSD,...)
 - Disquete, CD, DVD, pen USB
 2. Localizar as partições em cada um dos dispositivos de armazenamento
 - Leitura do MBR, tabela de partições estendidas, etc.
 - MBR: Master Boot Record
 3. Efetuar o *mount* de uma ou mais partições
 - O sistema de ficheiros fica disponível
 - Ficheiros podem ser lidos e escritos (se acesso escrita)

- Antes de poder ser usado, um disco deve ser particionado
- Organização do disco - particionamento
 - Quantas partições; Onde começam/acabam; Tipo de cada partição; Se há uma partição de arranque
- Duas normas distintas para o particionamento
 - **Master Boot Record (MBR)**: primeiros 512 octetos do disco. Abordagem mais antiga (1983, mas ainda em uso).
 - **GUID Partition Table (GPT)**: abordagem mais recente, associadas ao firmware UEFI (substituto da BIOS)
- Disco com capacidade para arranque do sistema
 - Disco *bootable*

- **MBR**

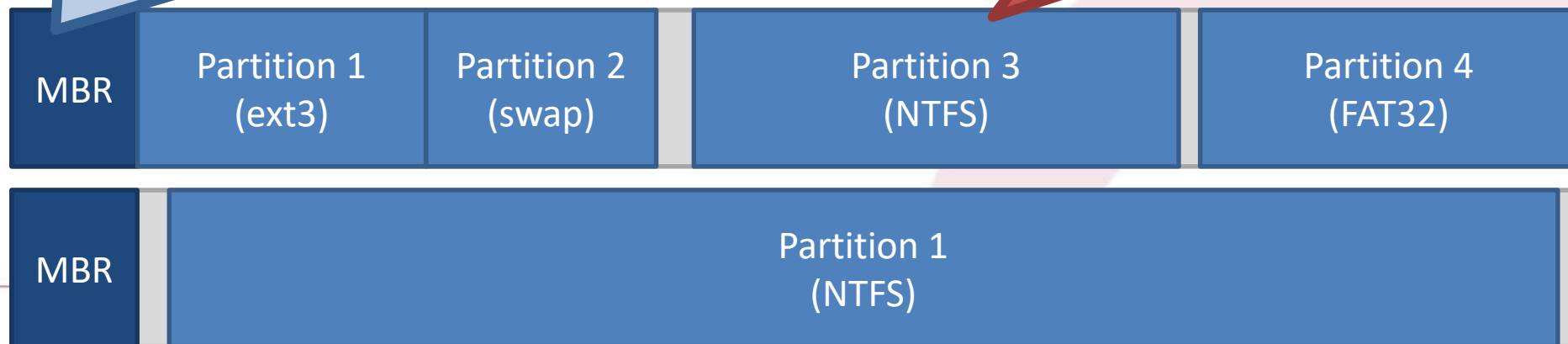
- Lançado em 1983 com o IBM PC DOS 2.0
- Localizado no 1º setor do disco (setor 0)
- Contém um *boot loader* para o sistema operativo instalado e a tabela de partições
 - Boot loader
 - Código que inicia o carregamento do sistema operativo, transferindo o controlo para um determinada partição do disco
- Suporta um máximo de 4 partições
- A MBR usa 32 bits para endereçamento dos setores
 - Para discos com setor=512 bytes, MBR suporta discos até 2 TiB
 - $2^{32} \times 512 \text{ bytes} = 2^{32} \times 2^9 = 2^{41} \text{ bytes}$
 - Para discos com setor= 4 KiB, MBR suporta discos até 16 TiB

Master Boot Record (#2)

Address		Description	Size (Bytes)
Hex	Dec.		
0x000	0	Bootstrap code area	446
0x1BE	446	Partition Entry #1	16
0x1CE	462	Partition Entry #2	16
0x1DE	478	Partition Entry #3	16
0x1EE	494	Partition Entry #4	16
0x1FE	510	<i>Boot signature</i>	2
Total:			512

Exemplo
Disco 1: 1 MBR e 4 partições
Cada partição tem um sistema de ficheiros distintos

Disco 1
Disco 2





MASTER BOOT RECORD

≥ INVOKE-IR

BY: JARED ATKINSON

TEMPLATE BY: ANGE ALBERTINI



BOOT CODE

FIELDS

VALUES

```
000: 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010: 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020: BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030: E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040: B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050: F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060: 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070: 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080: 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090: 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0: 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0: 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0: AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0: E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0: 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0: 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100: 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110: 53 66 55 66 68 00 00 00 66 68 00 7C 00 00 66
120: 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130: 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140: 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150: 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160: 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170: 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180: 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190: 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0: 65 6D 00 00 00 63 7B 9A 82 D4 BA 7D 00 00 00 20
1C0: 21 00 07 FE FF FF 00 08 00 00 00 90 36 06 80 FE
1D0: FF FF 07 FE FF FF 00 A0 36 06 00 60 09 00 00 00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

CHS ADDRESSING

00100000 00100001 00000000

00100000 100001 0000000000
Head - 1st byte
Sector - 2nd byte (0-5 bits)
Cylinder - 2nd byte (6-7 bits)
3rd byte

PARTITION TABLE

jump to boot program
disk parameters
boot program code
disk signature

82D4BA7D

status
starting head
starting sector
starting cylinder
partition type
ending head
ending sector
ending cylinder
relative start sector
total sectors

0x00 - Non-Bootable
0x20
0x21
0x00
0x07 - NTFS
0xFE
0x3F
0x3FF
0x800
0x6369000

status
starting head
starting sector
starting cylinder
partition type
ending head
ending sector
ending cylinder
relative start sector
total sectors

0x80 - Bootable
0xFE
0x3F
0x3FF
0x07 - NTFS
0xFE
0x3F
0x3FF
0x636A000
0x96000

partition type
0x00 - EMPTY

partition type
0x00 - EMPTY

END OF MBR

marker

0x55AA

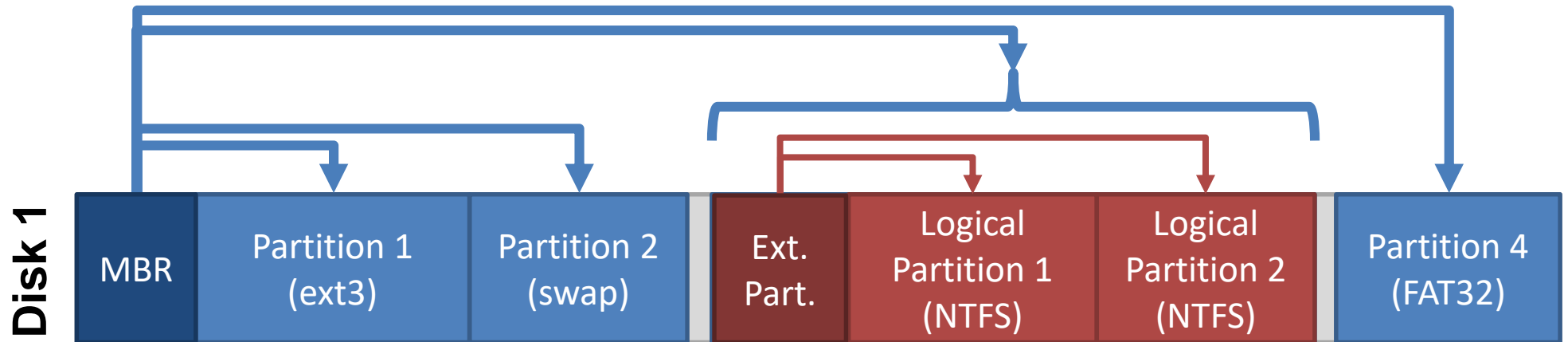
PARTITION TYPES

0x00 - EMPTY	0x83 - LINUX
0x01 - FAT12	0x84 - HIBERNATION
0x04 - FAT16	0x85 - LINUX_EXTENDED
0x05 - MS_EXTENDED	0x86 - NTFS_VOLUME_SET
0x06 - FAT16	0x87 - NTFS_VOLUME_SET_1
0x07 - NTFS	0xa0 - HIBERNATION_1
0x0b - FAT32	0xa1 - HIBERNATION_2
0x0c - FAT32	0xa5 - FREEBSD
0x0e - FAT16	0xa6 - OPENBSD
0x0f - MS_EXTENDED	0xa8 - MACOSX
0x11 - HIDDEN_FAT12	0xa9 - NETBSD
0x14 - HIDDEN_FAT16	0xab - MAC_OSX_BOOT
0x16 - HIDDEN_FAT16	0xb7 - BSDI
0x1b - HIDDEN_FAT32	0xb8 - BSDI_SWAP
0x1c - HIDDEN_FAT32	0xee - EFI_GPT_DISK
0x1e - HIDDEN_FAT16	0xef - EFI_SYSTEM_PARTITION
0x42 - MS_MBR_DYNAMIC	0xfb - VMWARE_FILE_SYSTEM
0x82 - SOLARIS_X86	0xfc - VMWARE_SWAP
0x82 - LINUX_SWAP	

Arranque do sistema: 1) Power-on self-test (POST); 2) Pesquisa de dispositivo de arranque; 3) Leitura do MBR para memória

Partições estendidas

- ✓ MBR
 - limite de 4 partições do sistema MBR
- ✓ SO modernos suportam partições estendidas para mitigar esse limite



- Partições estendidas podem ser específicas de um SO
 - Outros SO podem não conseguirem aceder/interpretar as partições estendidas

Norma GPT (#1)

- GPT é uma secção da norma UEFI
 - G = GUID: Identificador universal
 - Normal usa 64 bits para identificar os setores
 - Suporta discos até 8 ZiB (2^{64} setores x 512 B/setor= 2^{73} bytes)
 - Disco com setores de 512 bytes
 - Suporta N partições ($N \gg 4$)
- Organização
 - MBR “protegida” no setor 0
 - Protegida para evitar que software não-GPT escreva onde não deve
 - Cabeçalho GPT no setor 1
 - Existe uma cópia de salvaguarda no último setor do disco
 - O cabeçalho GPT tem um ponteiro para a tabela de partições (Partition Entry Array)

PROTECTIVE MBR

FIRST SECTOR OF DRIVE
FOR BREAKDOWN SEE MBR POSTER

```

000 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020 BD BE 07 80 7E 00 00 7C 08 0F 85 0E 01 83 C5 10
030 E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0 AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0 E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66
120 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0 65 6D 00 00 00 63 7B 9A 00 00 00 00 00 00 00 00
1C0 02 00 EE FF FF FF 01 00 00 00 FF FF FF FF 00 00
1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
  
```

IMPORTANT PROTECTIVE MBR VALUES

system id EE - EFI GPT partition
GPT header sector offset 1

Norma GPT (#2)

- Organização
 - MBR “protegida” no setor 0
 - Protegida para evitar que software não-GPT escreva onde não deve
 - Cabeçalho GPT no setor 1
 - Existe uma cópia de salvaguarda no último setor do disco

PROTECTIVE MBR

FIRST SECTOR OF DRIVE
FOR BREAKDOWN SEE MBR POSTER

```

000 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020 BD BE 07 80 7E 00 00 7C 08 0F 85 0E 01 83 C5 10
030 E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0 AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0 E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66
120 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0 65 6D 00 00 63 7B 9A 00 00 00 00 00 00 00 00
1C0 02 00 EE FF FF FF 01 00 00 00 FF FF FF FF 00 00
1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
  
```

GPT HEADER

```

200 45 46 49 20 50 41 52 54 00 00 01 00 5C 00 00 00
210 F3 73 9F 97 01 00 00 00 00 00 00 00 00 00 00
220 FF FF 3F 01 00 00 00 00 22 00 00 00 00 00 00
230 DE FF 3F 01 00 00 00 00 10 E1 13 F9 35 08 F1 4C
240 96 C7 38 0B 5D B4 A4 2D 02 00 00 00 00 00 00
250 80 00 00 00 80 00 00 00 3B 04 A4 F8
  
```

signature	EFI PART
revision	1.0
header size	92
header CRC32	979F73F3
my LBA	1
alternate LBA	20971519
first usable LBA	34
last usable LBA	20971486
disk guid	f913e110-0835-4cf1-96c7-380b5db4a42d
partition entry LBA	2 (sector containing of partition table)
# of partition entries	128
size of partition entry	128
partition entry array CRC32	F8A4043B

IMPORTANT PROTECTIVE MBR VALUES

system id	EE - EFI GPT partition
GPT header sector offset	1

GUID PARTITION TABLE

≥ INVOKE-IR

BY: JARED ATKINSON
TEMPLATE BY: ANGE ALBERTINI

PROTECTIVE MBR

FIRST SECTOR OF DRIVE
FOR BREAKDOWN SEE MBR POSTER

```

000 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020 BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030 E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070 7C 68 01 00 68 10 00 B4 42 8A 56 00 88 F4 CD 13
080 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0 AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0 E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110 53 66 55 66 68 00 00 00 66 68 00 7C 00 00 66
120 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0 65 6D 00 00 00 63 7B 9A 00 00 00 00 00 00 00
1C0 02 00 EE FF FF FF 01 00 00 00 FF FF FF FF 00 00
1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
  
```

IMPORTANT PROTECTIVE MBR VALUES

system id EE - EFI GPT partition
GPT header sector offset 1

GPT HEADER

```

200 45 46 49 20 50 41 52 54 00 00 01 00 5C 00 00 00
210 F3 73 9F 97 01 00 00 00 00 00 00 00 00 00 00
220 FF FF 3F 01 00 00 00 00 22 00 00 00 00 00 00 00
230 DE FF 3F 01 00 00 00 00 10 E1 13 F9 35 08 F1 4C
240 96 C7 38 0B 5D B4 A4 2D 02 00 00 00 00 00 00 00
250 80 00 00 00 80 00 00 00 3B 04 A4 F8
  
```

signature
revision
header size
header CRC32
my LBA
alternate LBA
first usable LBA
last usable LBA
disk guid
partition entry LBA
of partition entries
size of partition entry
partition entry array CRC32

EFI PART
1.0
92
979F73F3
1
20971519
34
20971486
f913e110-0835-4cf1-96c7-380b5db4a42d
2 (sector containing of partition table)
128
128
F8A4043B

PARITION ARRAY

```

400 16 E3 C9 E3 5C 0B B8 4D 81 7D F9 2D F0 02 15 AE
410 47 8A 1A FF F8 08 AB 43 B4 10 53 69 7F 0B 23 23
420 22 00 00 00 00 00 00 00 21 00 01 00 00 00 00
430 00 00 00 00 00 00 00 00 4D 00 69 00 63 00 72 00
440 6F 00 73 00 6F 00 66 00 74 00 20 00 72 00 65 00
450 73 00 65 00 72 00 76 00 65 00 64 00 20 00 70 00
460 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

e3c9e316-0b5c-4db8-817d-f92df00215ae
ff1a8a47-08f8-43ab-b410-53697f0b2323
34
65569
0
Microsoft reserved partition

```

480 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7
490 42 AE 76 6D C1 B6 BE 4F 8D 42 20 CD 36 60 26 B4
4A0 00 08 01 00 00 00 00 00 FF 07 00 00 00 00 00 00
4B0 00 00 00 00 00 00 00 00 42 00 61 00 73 00 69 00
4C0 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00
4D0 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
4E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
6d76ae42-b6c1-4fbe-8d42-20cd366026b4
67584
2164735
0
Basic data partition

```

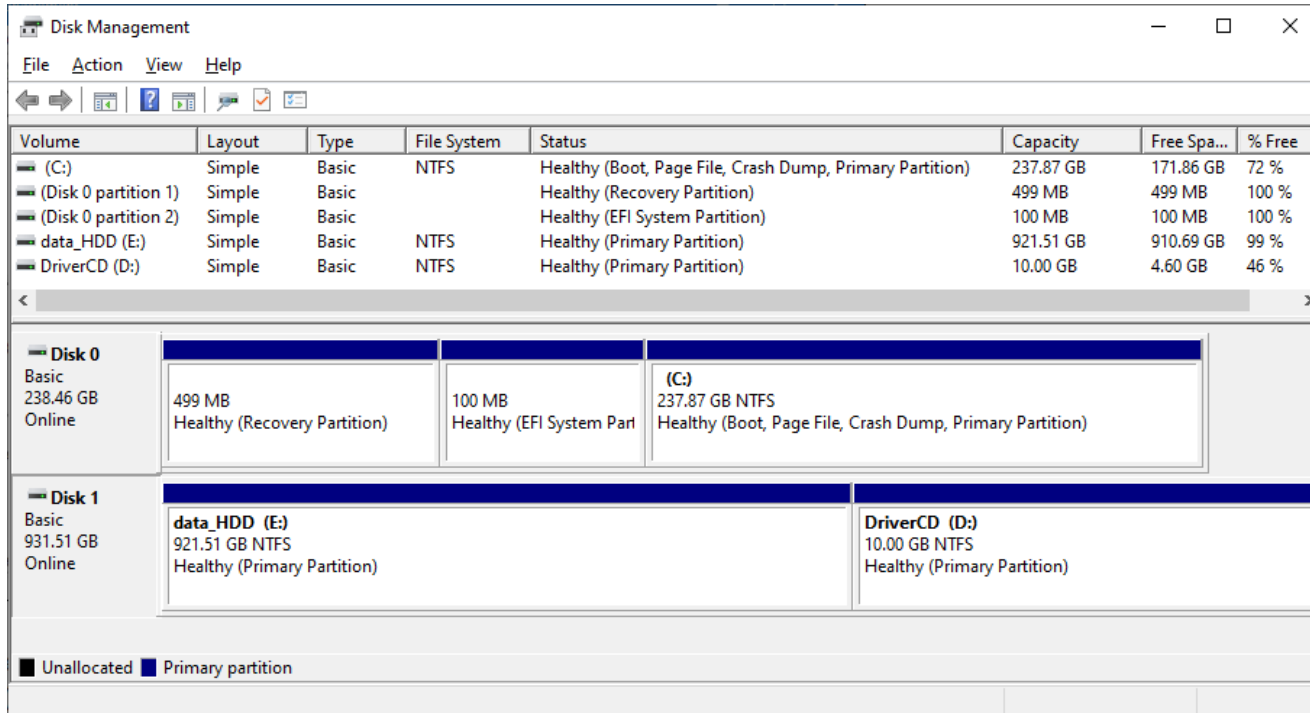
500 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7
510 3A 5C 79 D6 4D 8A B4 4F 91 A0 48 88 12 CC E0 27
520 00 08 00 00 00 00 00 00 FF 07 41 00 00 00 00 00
530 00 00 00 00 00 00 00 00 42 00 61 00 73 00 69 00
540 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00
550 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
d6795c3a-8a4d-4fb4-91a0-488812cce027
2164736
4261887
0
Basic data partition

<http://www.invoke-ir.com/search/label/Guid%20Partition%20Table>

- Windows: *disk management*

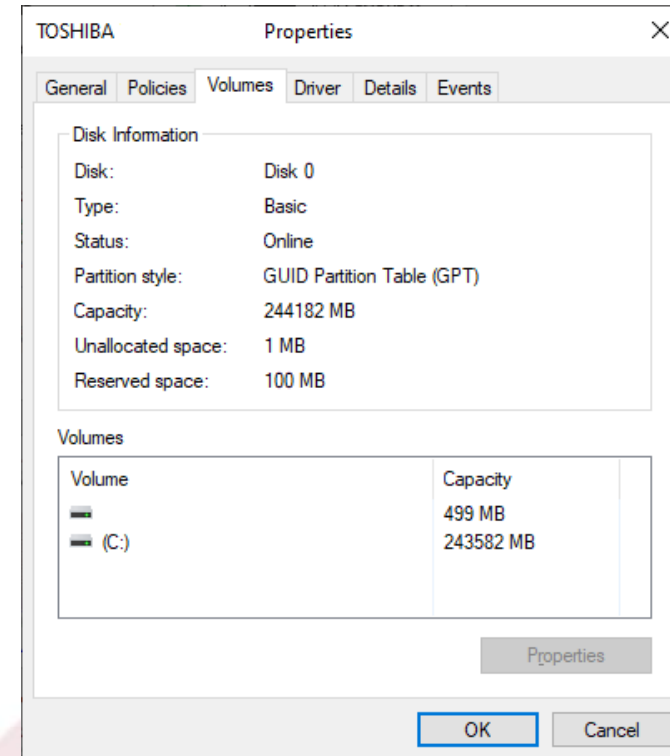


Disk Management

Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (Boot, Page File, Crash Dump, Primary Partition)	237.87 GB	171.86 GB	72 %
(Disk 0 partition 1)	Simple	Basic		Healthy (Recovery Partition)	499 MB	499 MB	100 %
(Disk 0 partition 2)	Simple	Basic		Healthy (EFI System Partition)	100 MB	100 MB	100 %
data_HDD (E:)	Simple	Basic	NTFS	Healthy (Primary Partition)	921.51 GB	910.69 GB	99 %
DriverCD (D:)	Simple	Basic	NTFS	Healthy (Primary Partition)	10.00 GB	4.60 GB	46 %

Disk	Layout	Type	File System	Status	Capacity	Free Space	% Free
Disk 0	Basic	Basic		Online	238.46 GB		
Disk 1	Basic	Basic		Online	931.51 GB		

Legend: ■ Unallocated ■ Primary partition



TOSHIBA Properties

General Policies Volumes Driver Details Events

Disk Information

Disk: Disk 0
Type: Basic
Status: Online
Partition style: GUID Partition Table (GPT)
Capacity: 244182 MB
Unallocated space: 1 MB
Reserved space: 100 MB

Volumes

Volume	Capacity
(C:)	243582 MB

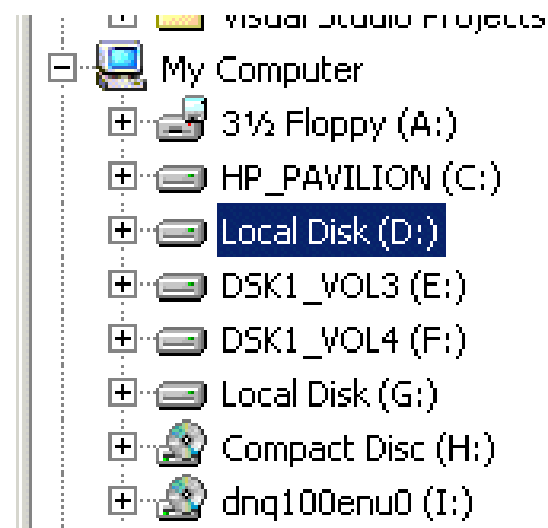
Properties

OK Cancel

SISTEMA DE FICHEIROS *ROOT*

Root File Systems (#1)

- ✓ O SO Windows disponibiliza sistema de ficheiros com possibilidade de várias raízes
 - Cada para dispositivo/partição é identificada por uma letra + “:”
 - C:, D:, E:, ...
 - E o A: e B:? Reservado para drives de disquetes
 - Internamente, o SO mantém somente uma raiz



Root File Systems (#2)

- Linux tem somente uma raiz
 - Partição associado ao diretório root (“/”)
 - As restantes partições são associadas a um diretório abaixo da raiz (“/”)
 - Utilitário
 - `df -h`

```
[user@linux~] df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda7	39G	14G	23G	38%	/
/dev/sda2	296M	48M	249M	16%	/boot/efi
/dev/sda5	127G	86G	42G	68%	/media/cbw/Data
/dev/sda4	61G	34G	27G	57%	/media/cbw/Windows
/dev/sdb1	1.9G	352K	1.9G	1%	/media/cbw/NDSS-2013

1 drive, 4
partições

1 drive, 1
partição



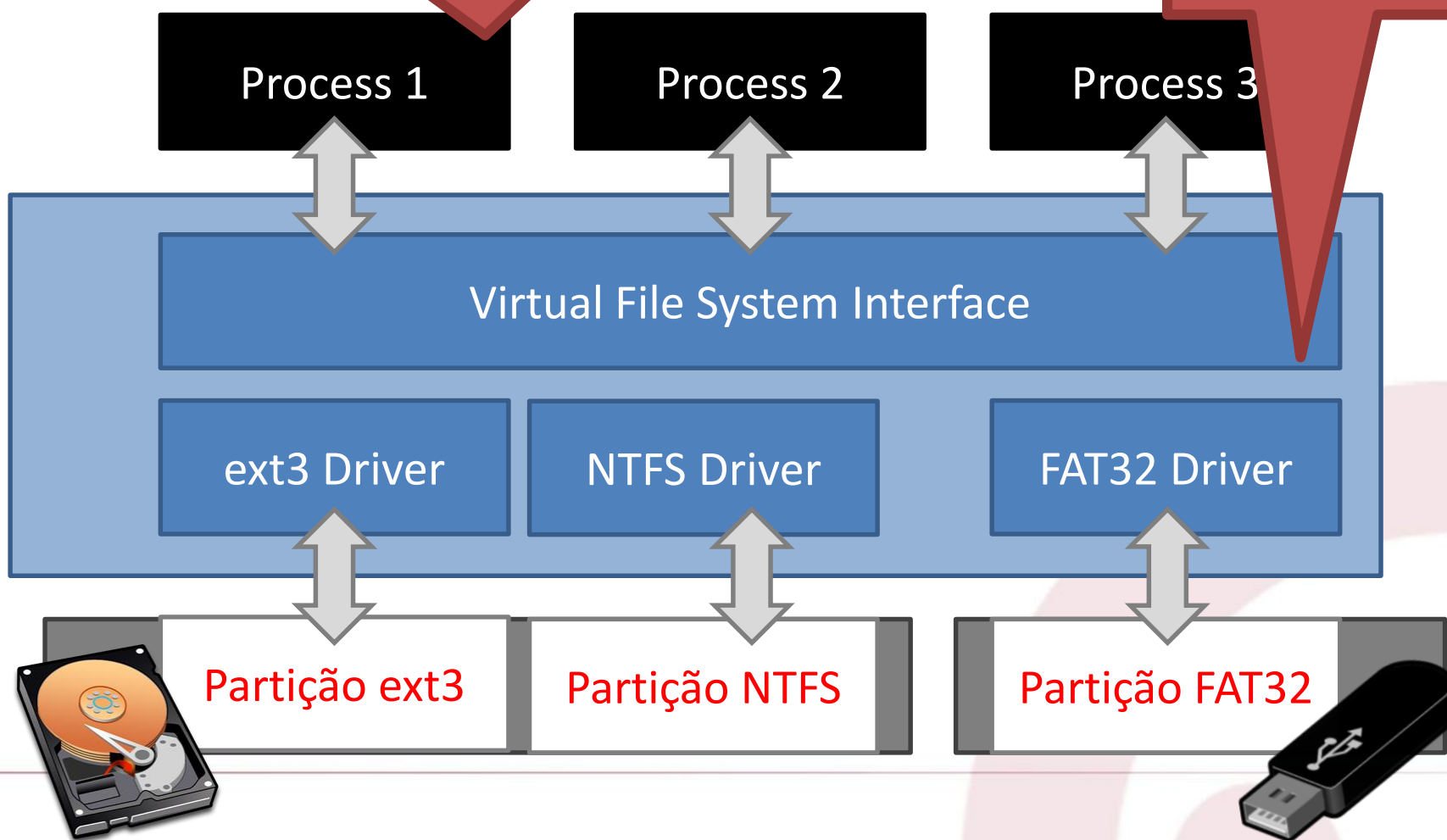
✓ Situação

- ✓ O SO pode ter partições com sistemas de ficheiros distintos
 - FAT32, NTFS, ext2, ext3, ext4, zfs, ...
 - Seria muito pouco prático que os processos tivessem que recorrer a API distintas consoante o sistema de ficheiros
 - Uso de camada de virtualização
 - Linux: Virtual File System interface (VFS). Os processos usam todos a mesma API.

Virtual File System (VFS)

Processos desconhecem os pormenores de baixo nível do sistema de ficheiros

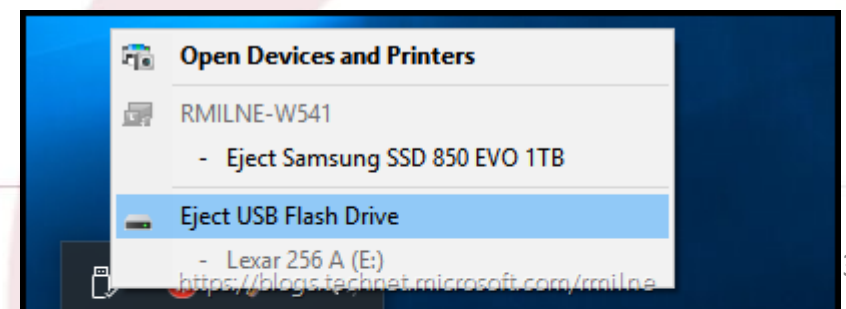
É relativamente simples adicionar um sistema de ficheiros





mount/unmount dinâmico

- ✓ Quando se insere dispositivo de armazenamento, o SO liga-se ao dispositivo
 - ✓ *mount*
 - ✓ Exemplo: USB stick
- ✓ O que significa ejetar de forma segura um dispositivo?
 - ✓ Caches são escritas (*flushed*) para o dispositivos
 - Remoção do dispositivo ao nível do SO
 - Windows 10 19.09 já não usa cache de write para evitar problemas de *unmount* forçado



SISTEMA DE FICHEIROS: FAT

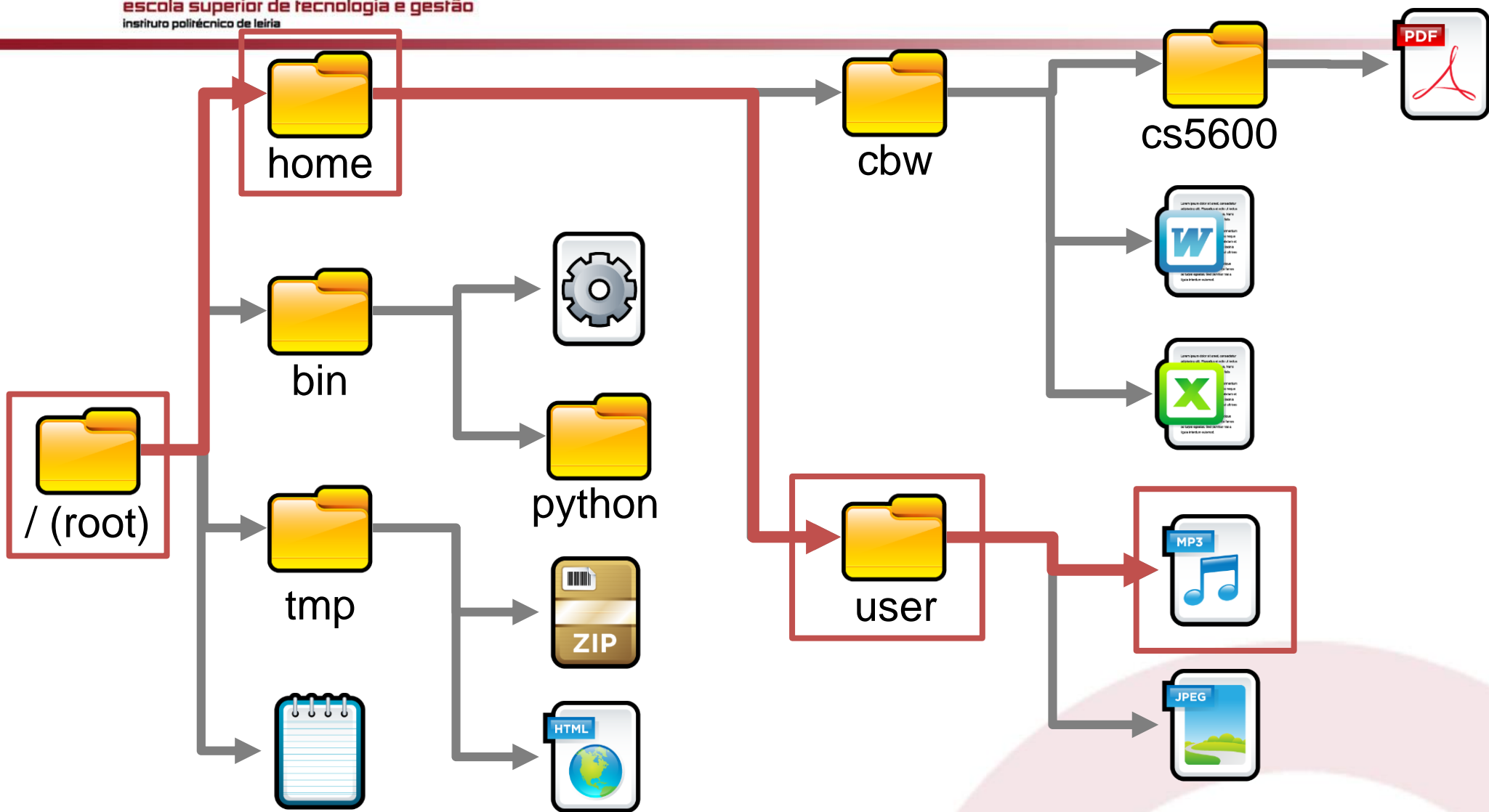
- ✓ O SO pode localizar e associar partições
 - ✓ Windows: c:, d:, etc.
 - ✓ Linux/Unix: /, ...
- ✓ Mas, como está organizado o sistema de ficheiros?
 - Nome de ficheiros
 - Hierarquia de diretórios
 - Metadados como a data/hora de criação, acesso e modificação; permissões; etc.
- Estruturas para suporte a essas funcionalidades



IPL

escola superior de tecnologia e gestão
instituto politécnico de leiria

Árvore de diretório



- ✓ Diretório absoluto
 - E.g. `/home/user/info.txt`

Ficheiros (lembrete)

✓ Ficheiro: duas partes

– Ficheiro propriamente dito

- Um ou mais blocos de dados (0 e 1)
- Um ficheiro pode conter qualquer coisa
 - Deve ser devidamente interpretado

– Metadados associados ao ficheiro

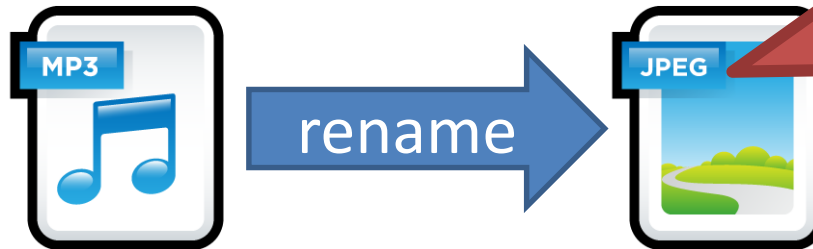
- Nome, tamanho total
- Em que diretório está?
- Data/hora de modificação, acesso, criação
- Dono e grupo
- Permissões: read/write/execute





Nome ficheiro – extensão (#1)

- ✓ Uso do “.” para criar “extensão”
 - e.g. program.exe, image.jpg, music.mp3
- ✓ A extensão de um ficheiro não tem significado absoluto: pode ser dada qualquer extensão, independentemente do conteúdo



O conteúdo passou de MP3 para JPG, só pela mudança de nome?
NÃO!

- Windows explorer (e outros similares) fazem uso da extensão para identificar o tipo de ficheiro
 - Associação sujeito a falhas

- ✓ O utilitário **file** deteta o tipo de ficheiro, independentemente da extensão

Exemplos

file screenshot.png

screenshot.png: PNG image data, 816 x 573, 8-bit/color RGB, non-interlaced

file memtest86+.bin

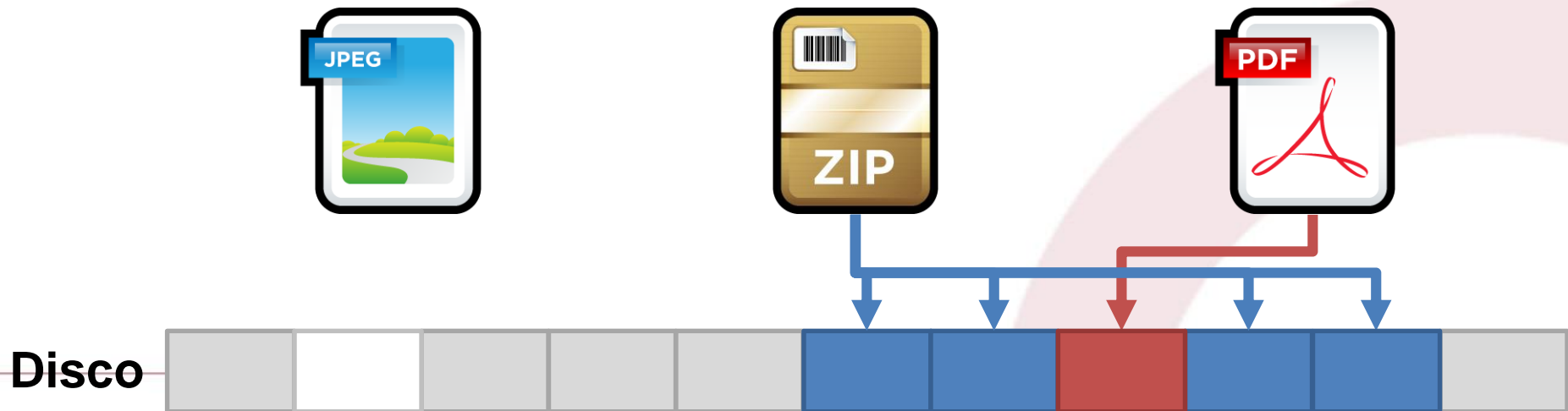
memtest86+.bin: DOS/MBR boot sector

file EmptyProject-Templatev3.03.zip

EmptyProject-Templatev3.03.zip: Zip archive data, at least v2.0 to extract

Metadados específicos

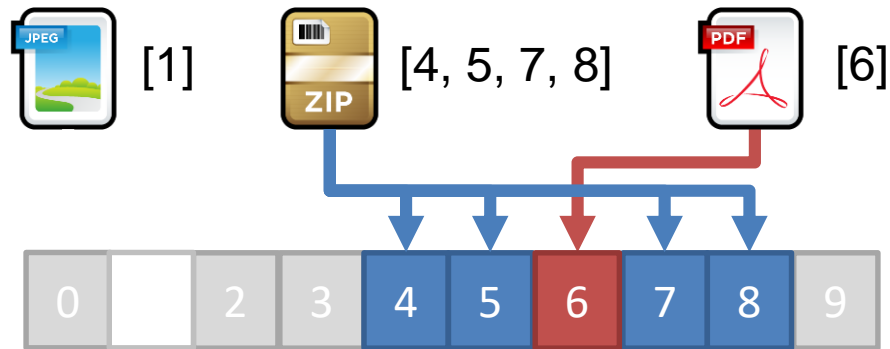
- ✓ Metadados para gestão do ficheiro
 - Identificador univoco
 - Nome do ficheiro pode não ser único
 - Estrutura que localiza os blocos do ficheiros no disco
- ✓ A associação ficheiro/blocos do disco é uma das funções principais do sistema de ficheiros



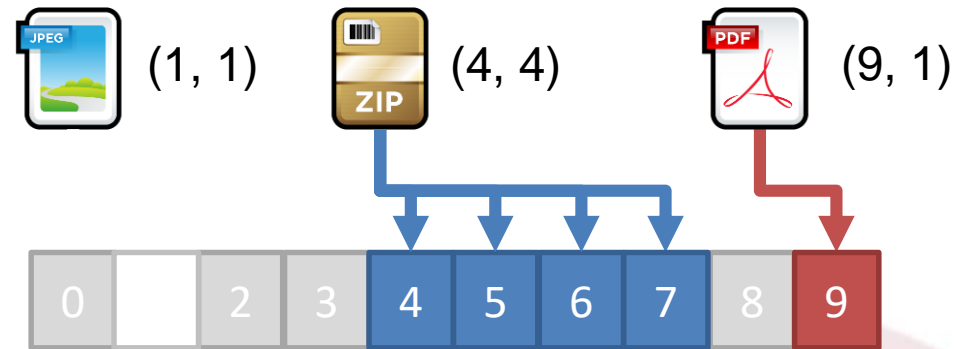
Mapeamento ficheiros para blocos

- ✓ Cada ficheiro é composto por ≥ 1 blocos
- ✓ Como mapear um ficheiro para blocos?
- ✓ Depende do sistema de ficheiros

Lista de blocos



Par (inicio, fim)




- Problema?
 - Ficheiros de grandes dimensões



- Problema?
 - Fragmentação
 - E.g.: adicionar novo ficheiro que ocupa três blocos

Sistema FAT

- FAT – File Allocation Table (1977)
 - Sistema de ficheiros
 - Muitas variantes: FAT12, FAT16, VFAT, FAT32, ExFAT, etc.
 - Muitos dispositivos usam FAT32 (1996)
 - Cameras de video usam ExFAT

- ✓ Origem do nome
 - Tabela empregue para registar localização de diretórios e ficheiros
- ✓ Sistema muito popular
 - Empregue por partições de arranque EFI
 - Formato por omissão de pen USB e cartões de memória



Type:	USB Drive	
File system:	FAT32	
 Used space:	3,237,355,520 bytes	3.01 GB
 Free space:	12,327,673,856 bytes	11.4 GB
Capacity:	15,565,029,376 bytes	14.4 GB

FAT

- Dados básicos do sistema de ficheiros
 - Versão FAT, localização ficheiros *boot*
 - Total de blocos
- Índice do diretório root do sistema FAT

File allocation table (FAT)
Identifica blocos em uso/livres
Lista ligada para gestão de ficheiros

- Zona de blocos de dados
 - Dados ficheiros e diretórios
 - Blocos de tamanho uniforme (4 KiB a 64 KiB)
- Ficheiros podem ocupar vários blocos

Disco

Super
Block



Entradas da FAT (#1)

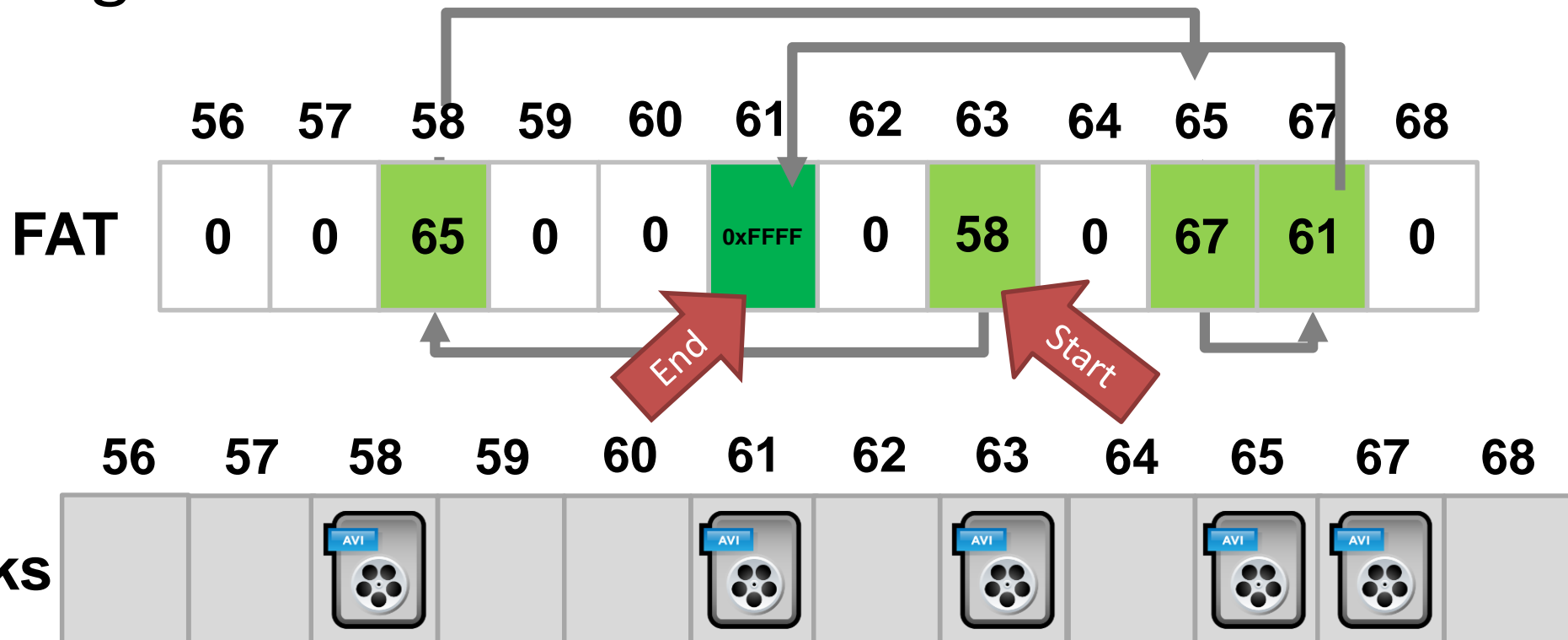
- ✓ tamanho da (FAT) é igual ao número de *clusters* do disco
 - Cluster: agregado com N blocos
 - N é constante para um dado disco formatado em FAT
 - Cluster é a unidade mais pequena de armazenamento considerada pela FAT
 - Um cluster só pode guardar dados de um só ficheiro
 - **Exemplo:** cluster de 8 KiB significa que um ficheiro de 1 byte irá necessitar de 8 KiB no disco

Entradas da FAT (#2)

- ✓ Versão da FAT associada ao número de bits de cada entrada
- ✓ **Exemplos**
 - FAT16: cada entrada FAT tem 16 bits
 - Partição só poderá ter 2^{16} (65536) entradas, i.e. clusters
 - Determina o tamanho máximo da partição que pode ser suportada
 - FAT32: cada entrada FAT tem...28 bits
 - Partição pode ter até 2^{28} entradas
 - Com cluster de 64 KiB (máximo), tem-se:
 - $2^{28} \times 64\text{KiB} = 2^{28} \times 64 \times 2^{10} = 2^{28} \times 2^6 \times 2^{10} = 2^{44} \text{ B} = 16\text{TiB}$
 - FAT32 suporta partição até 16 TiB (limite teórico) com uso de cluster=64KiB

Fragmentação

✓ Os blocos de um ficheiro não precisam de ser contíguos



Valores para entradas FAT:

- 0 – entrada vazia (não ocupada)
- $1 < N < 0xFFFF$ – Próximo bloco da cadeia
- 0xFFFF – fim da cadeia

FAT: o bom e o mau...

- ✓ O bom. A FAT disponibiliza:
 - Hierarquia de diretórios e ficheiros
 - Ficheiros de tamanho variável
 - Metadados para ficheiros e diretórios
- ✓ O mau.
 - Muitos SO limitam tamanho da FAT32 a 2 TiB
 - Localizar blocos livres requer iterar a lista de blocos da FAT
 - Operação potencialmente demorada
 - Sujeita a **fragmentação**
 - **Interna**: espaço não aproveitado do cluster.
 - Cluster de 8 KiB, 1 ficheiro de 1 byte irá ocupar 8 KiB no disco
 - **Externa**: os vários clusters que armazenam um ficheiro podem estar distribuídos pelo disco
 - Ineficiente para os discos HDDs



SISTEMAS DE FICHEIROS EXT

- ✓ FAT permite
 - Organização em hierarquia de diretórios
 - Armazenar ficheiros de tamanho variável
- ✓ Eficiência da FAT é baixa
 - Muita pesquisa e iteração nas listas ligada de *clusters*
 - Encontrar blocos livres requer iterar os blocos
- ✓ Sistemas de ficheiros do Linux usam estruturas mais eficientes
 - Sistema *Extended File System* (ext) usa **index nodes** (**inodes**) para diretórios e ficheiros

- ✓ FAT assente em lista ligada
 - Mecanismo simples e uniforme
 - ... mas não otimizado para ficheiros grandes
- ✓ Qual a distribuição de ficheiros segundo o tamanho?
 - Ficheiros pequenos são muito mais frequentes
- ✓ Ideia chave:
 - otimizar o sistema de ficheiros para ficheiros de pequenas dimensões

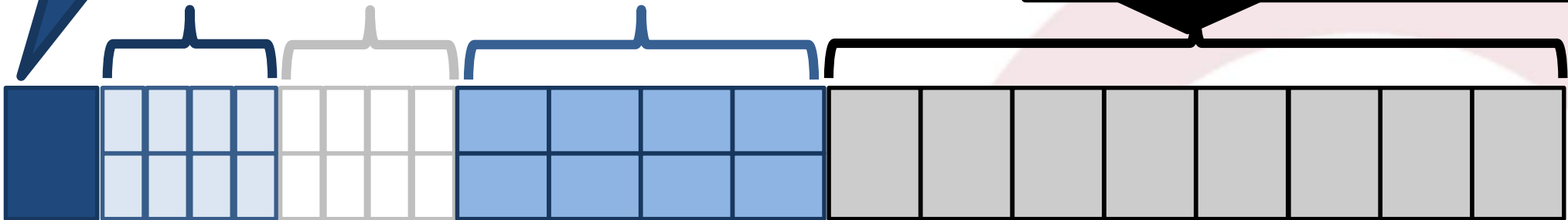
- Super block, armazena:
 - Tamanho e localização dos *bitmaps*
 - Número e localização de *inodes*
 - Número e localização de blocos de dados
 - Índice de inodes raiz

Bitmap de blocos de dados (livres/usados)

Bitmap de inodes (livres/usados)

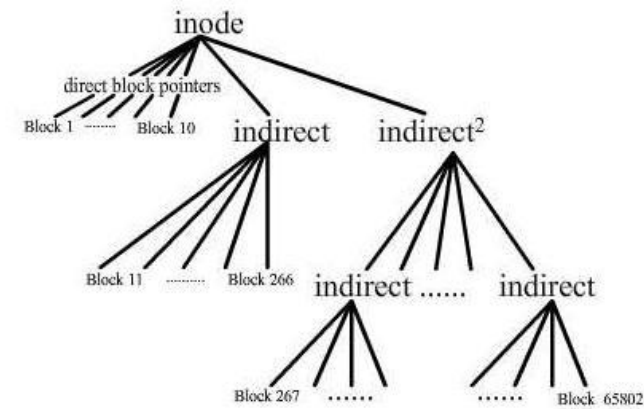
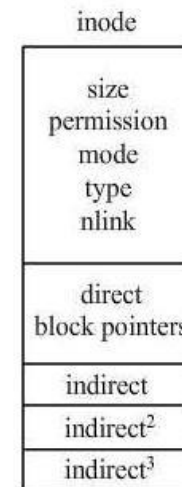
- Tabela de inode
- Cada inode representa ficheiro/diretório

Blocos dados (4KB cada)



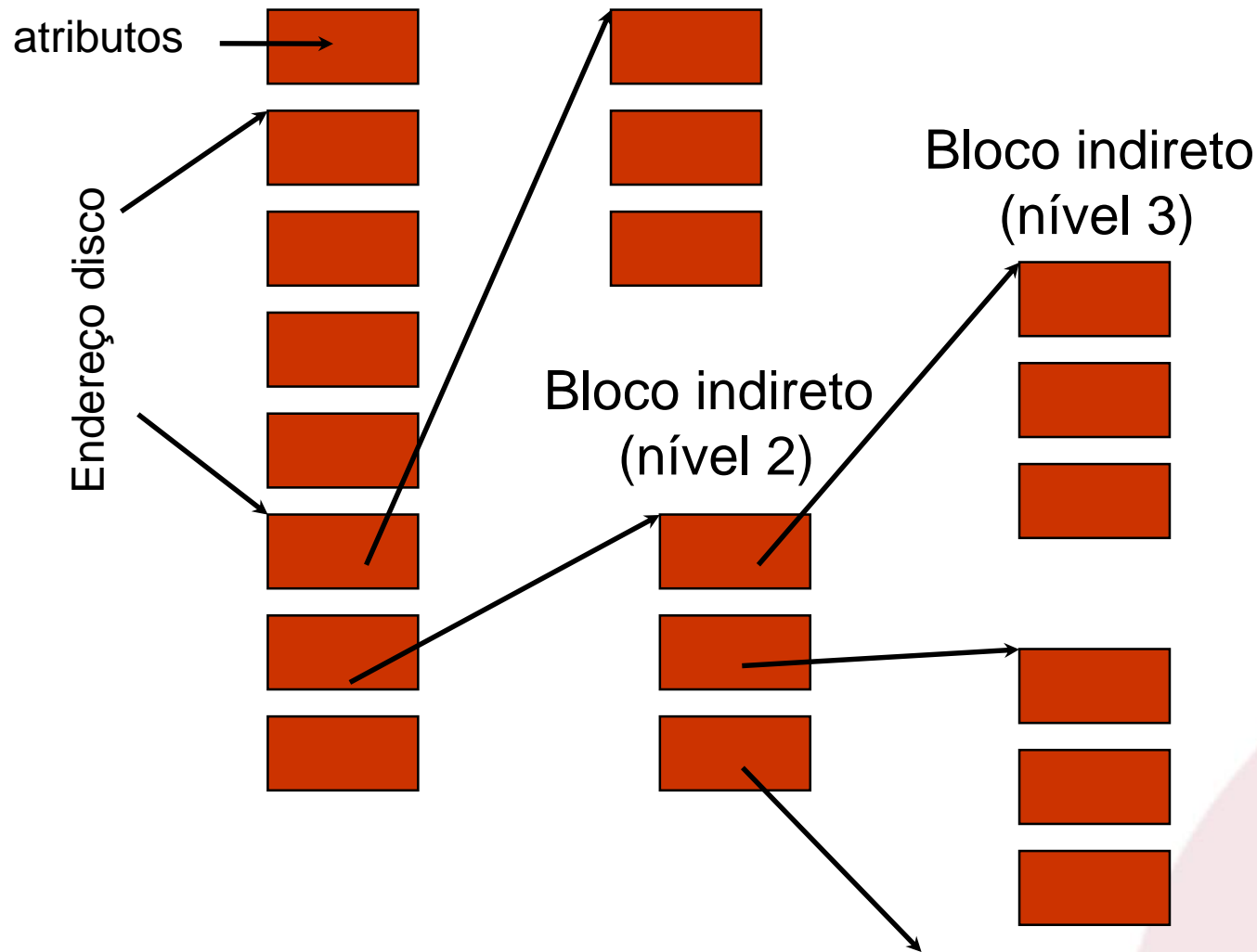
inodes (1)

- *inode*: estrutura de dados empregue por muitos sistemas de ficheiros Un*x
- Cada *inode* guarda a localização no disco de um ficheiro/diretório
- Um *inode* organiza-se em níveis
 - Um *inode* tem 12 ponteiros para blocos de dados de nível 1 (um ponteiro por bloco)
 - Um *inode* tem ainda um ponteiro para bloco de nível 2. Esse ponteiro aponta para um bloco que aponta para outros blocos de dados
 - Um *inode* tem também um ponteiro um bloco de nível 3: aponta para um bloco que aponta para um bloco que aponta para bloco de dados (dois níveis de indireção)
 - Um *inode* tem Também um ponteiro para um bloco de nível 4. (três níveis de indireção)



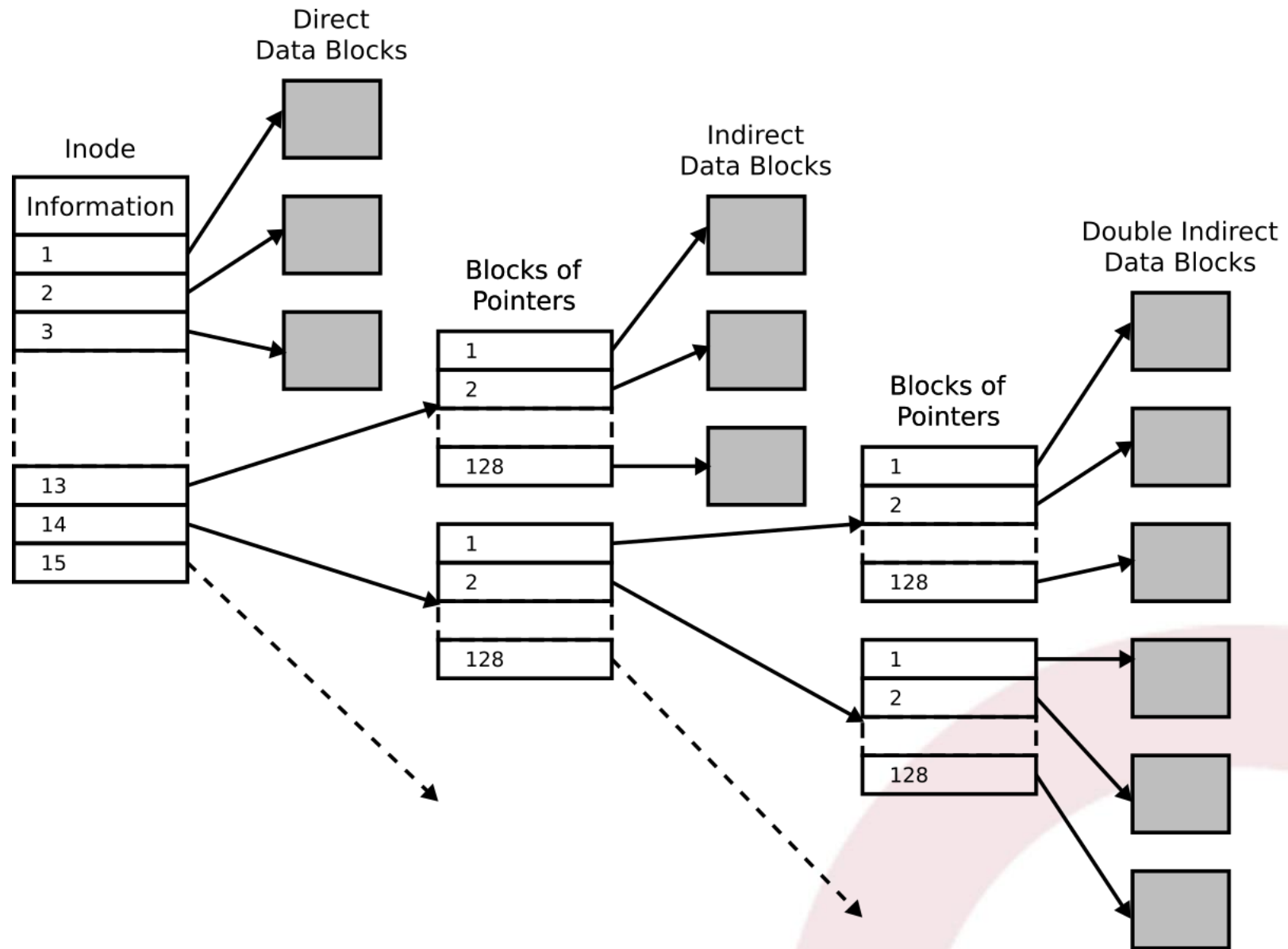
inodes (2)

“i-node” Acesso direto (nível 1)



- Otimizado para ficheiros de pequenas dimensões
 - Acesso direto
- Suporta ficheiros de grandes dimensões
 - Acesso mais lento, dado os níveis de indireção
- No Unix, grande parte dos ficheiros são pequenos

inodes (3)

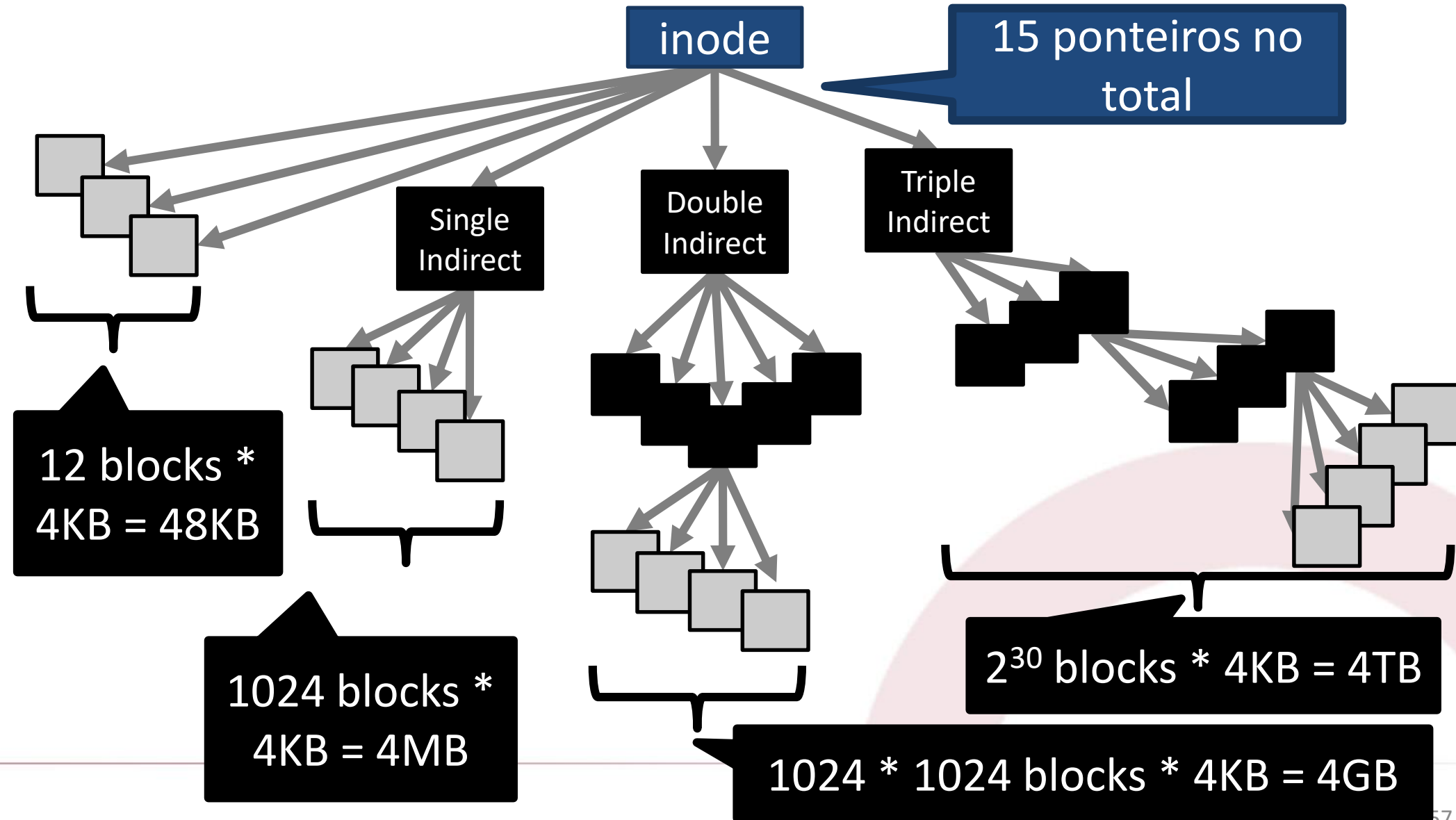


Ext2: conteúdo de um *inode*

tamanho (bytes)	Nome	Propósito do campo
2	mode	Read/write/execute?
2	uid	User ID of the file owner
4	size	Size of the file in bytes
4	time	Last access time
4	ctime	Creation time
4	mtime	Last modification time
4	dtime	Deletion time
2	gid	Group ID of the file
2	links_count	How many hard links point to this file?
4	blocks	How many data blocks are allocated to this file?
4	flags	File or directory? Plus, other simple flags
60	block	15 direct and indirect pointers to data blocks

inode Block Pointers

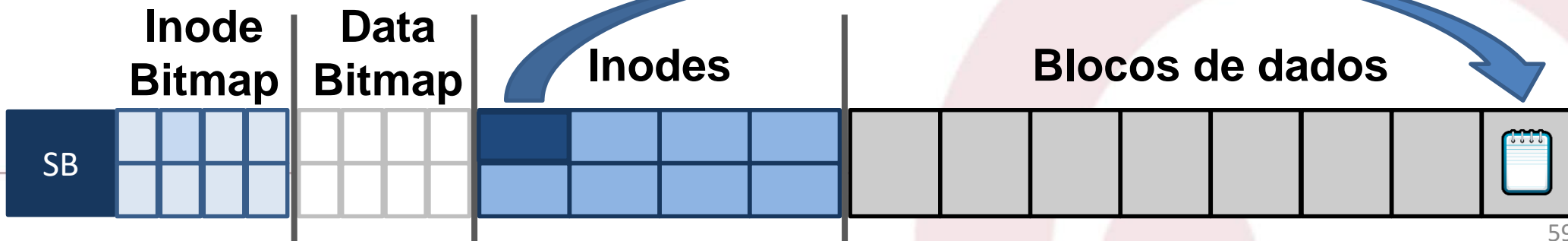
- ✓ Cada *inode* é a raiz de uma árvore não balanceada de blocos de dados



Vantagens dos *inodes*

- ✓ Otimizado para ficheiros pequenos
 - Cada inode aponta diretamente até 48 KiB de dados
 - Um nível de indireção para ficheiros de 4MB
- ✓ Acesso mais rápida aos ficheiros
 - Localidade de dados
 - Menos operações de *seek* (em HDD)
 - Não precisa de iterar lista ligada (ao contrário da FAT)
- ✓ Gestão mais fácil de blocos livres
 - Bitmaps pode ser guardado na memória
 - *inode* e blocos de dados geridos separadamente

- ✓ O bom *ext* (inodes) suporta:
 - Ficheiros/diretórios e metadados
 - Melhor desempenho do que a FAT
- ✓ O mau: baixa localidade
 - Não está otimizado para disco rígidos (HDD)
 - Deslocamento do braço do disco são onerosos
 - Não é problema para SSD
 - inodes e os respetivos blocos de dados estão afastados no disco

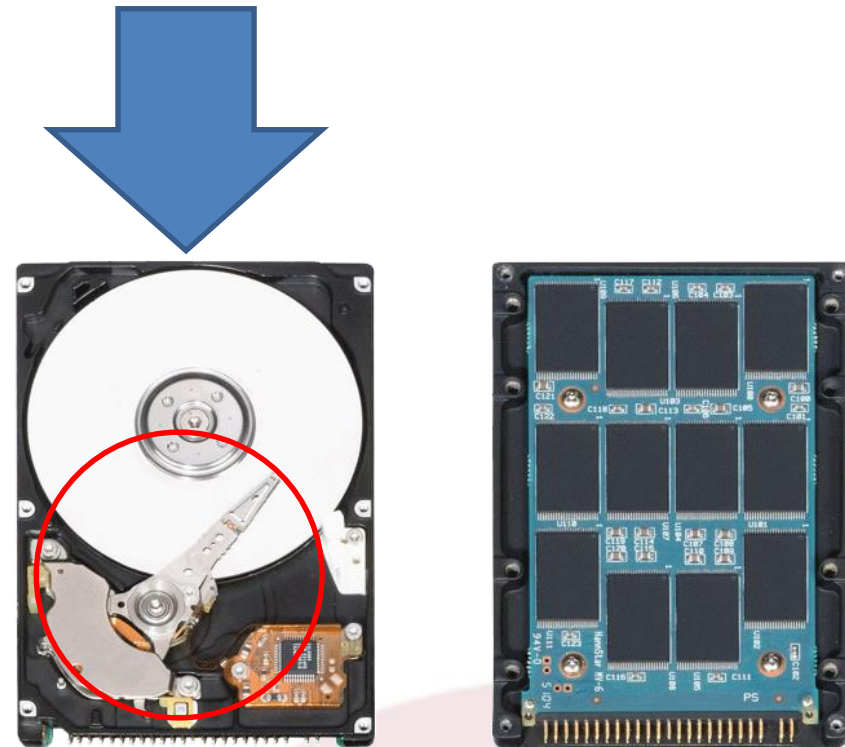


SISTEMA DE FICHEIROS EXT2

- ✓ ext (“extended file system”)
 - *inodes* são árvores não equilibradas de blocos de dados
 - Otimizado para o caso comum: ficheiros pequenos
- ✓ **Problema:** *ext* tem reduzida localidade
 - inodes afastados dos respetivos dados
 - Potencia operações onerosas dos HDD
- ✓ **Problema:** *ext* está sujeito a fragmentação
 - *ext* selecciona o 1º bloco livre que encontrar para escreve
 - Não procura manter os blocos contíguos

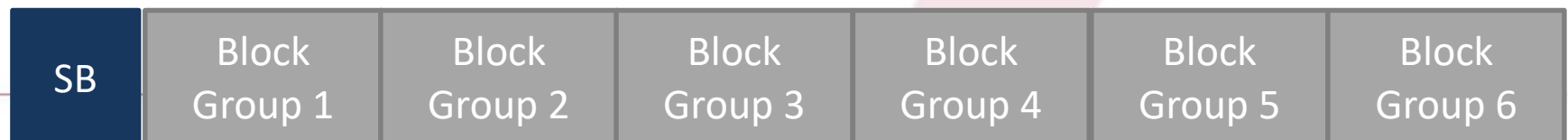
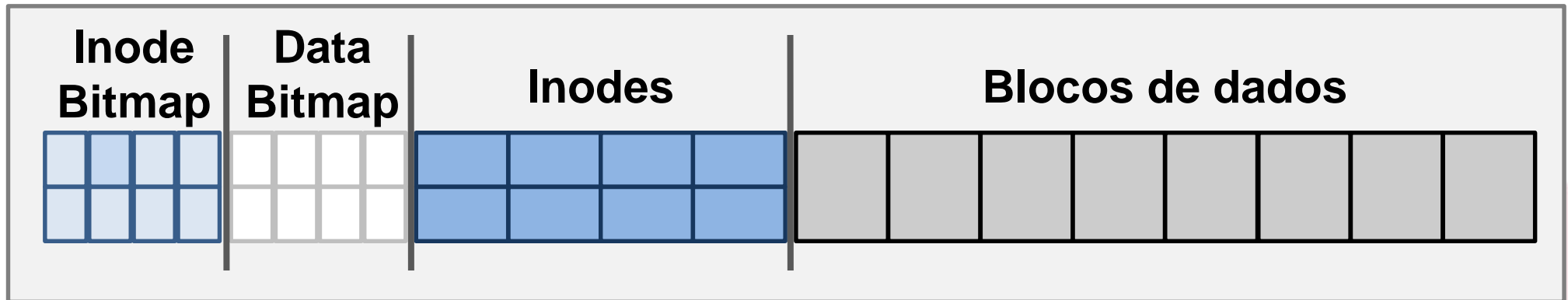
Fast File System (FFS)

- ✓ FFS criado em Berkeley/1984
 - Tentativa de sistema de ficheiro otimizado para o dispositivo persistente
 - Em 1984: disco rígidos
- ✓ **Observação:** processos tendem a aceder a ficheiros que estão no mesmo ou em diretórios próximos do diretório corrente:
 - localidade espacial
- ✓ Ideia chave: juntar grupos de diretórios e os respetivos ficheiros em grupo
 - Empregue no **ext2**: [bloco de grupo](#)



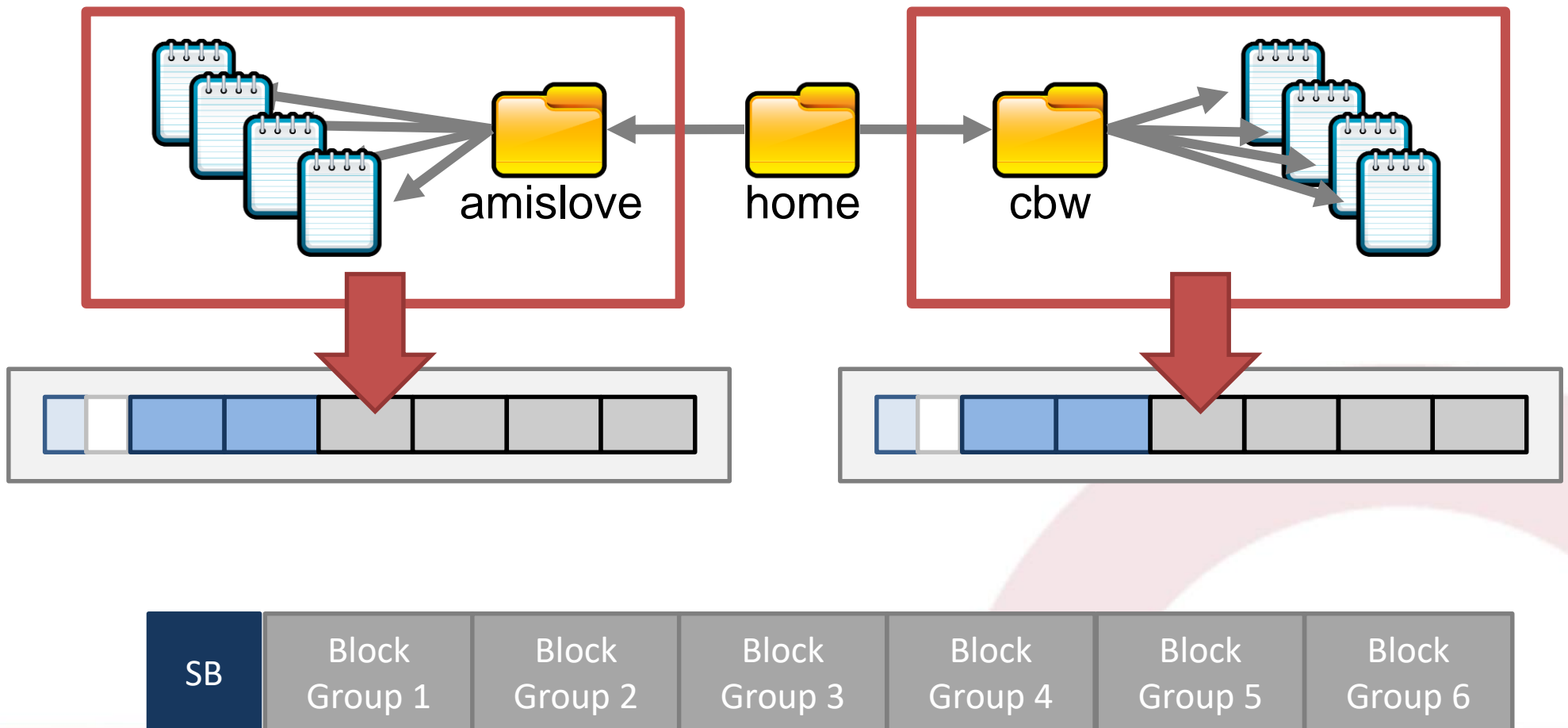
Bloco de grupo

- ✓ No **ext** existe somente um conjunto de estrutura de dados
 - Um *data bitmap*, um *inode bitmap*
 - Uma tabela de *inodes*, um vetor de blocos de dados
- ✓ No **ext2**, cada bloco de grupos tem as suas infraestruturas contains its own key data structures



Política de alocação

- ✓ ext2 tenta guardar ficheiros e diretórios relacionados no mesmo bloco de grupo



ext2: o bom e o mau...

- ✓ O bom - **ext2** disponibiliza:
 - Todas as funcionalidades do **ext...**
 - ... com melhor desempenho, dado o melhor uso da localidade espacial
- ✓ O mau
 - Ficheiros de grandes dimensões podem estar em vários grupos de blocos.
 - Sistema de ficheiros mais complexo, aumenta a possibilidade de corromper os dados
 - E.g. *inodes* não válidos, entradas de diretórios incorretas, etc.

EXT3 - JOURNALING

- Sistema de ficheiros ext3
- Compatível com ext2
 - Acrescentar consistência e confiabilidade ao sistema de ficheiros ext2
 - *Journaling*

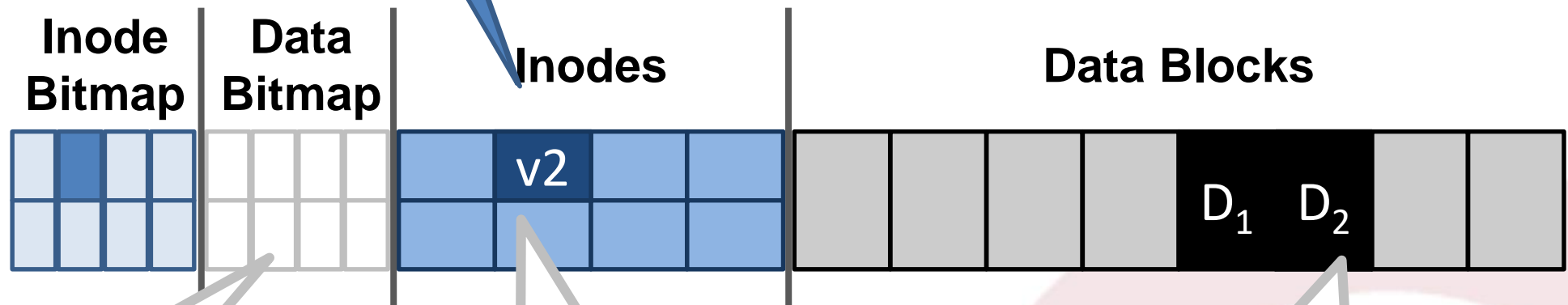
- ✓ Uma operação de escrita num ficheiro pode levar a várias escritas no sistema de ficheiros
 - Exemplo: acrescentar um bloco a um ficheiro existente
 1. Atualizar o mapa binário de blocos livres
 2. Atualizar o *inode* associado ao ficheiro
 3. Escrever os dados do utilizador
- ✓ E se ocorre um *crash* do computador no meio do processo? (e.g., entre o passo 1 e o passo 2)
 - Possibilidade de inconsistência do SF



Exemplo: acréscimo a um ficheiro

owner: user
permissions: rw
size: 2
pointer: 4
pointer: 5
pointer: null
pointer: null

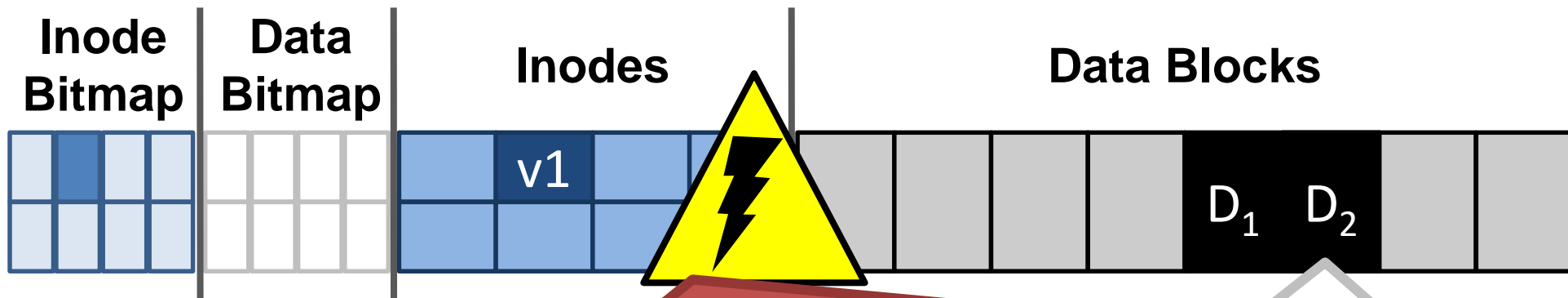
- ✓ As três operações são independentes
 - Podem ser efetuadas por qualquer ordem
- ✓ ...mas o computador pode *crashar* a qualquer momento...



Atualizar
mapa
binário

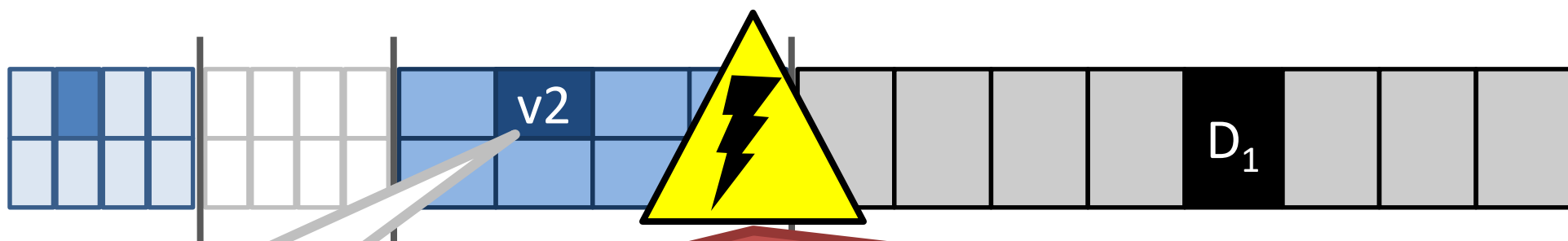
Atualizar o
inode

Escrita dos
dados



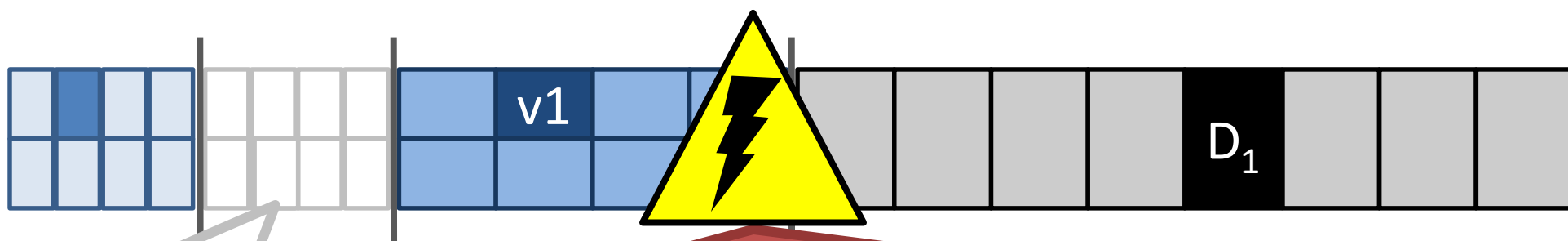
Resultado: SF consistente, mas há perda de dados

Escrita dados



Atualiza *inode*

Resultado: *inode* aponta para lixo; SF incoerente (mapa binário vs. inode)



Atualiza mapa binário

Resultado: perda de espaço, e SF fica inconsistente (mapa binário vs. inode)

Consistência e *crash*

- ✓ O disco assegura que a escrita num setor é atômica
 - O mesmo NÃO sucede para a escrita em múltiplos setores/blocos

- ✓ Como assegurar a consistência após um *crash*?

- ✓ Duas alternativas

1. Não se preocupar com a consistência

- SF poderá estar inconsistente após o crash
- Execução de uma aplicação “reparadora” do SF no arranque do SO
- [File system checker](#) (*fsck*)

OU

2. Usar um *log de transações* para tornar múltiplas escritas atômicas

- O log regista o histórico das escritas para disco
- Após o crash, podem ser aplicadas as operações registadas no log para concluir as atualizações
- [Journaling file system](#)

Opção 1: verificador SF

- ✓ Ideia principal:
 - Reparar as inconsistências do SF no arranque
 - Unix: utilitário *fsck*
 - *Windows*: utilitário *chkdsk*
 - Analisa todo o SF várias vezes, procurando detetar e corrigir inconsistências
- ✓ Porquê durante o arranque?
 - Não pode existir atividade no SF
 - Após *fsck* ter terminado, o arranque prossegue normalmente

```
swapon on /dev/hda3
Adding 803240k swap on /dev/hda3. Priority:-1 extents:1 across:803240k
Done activating swap.
Will now check root file system.
fsck 1.39 (29-May-2006)
[/sbin/fsck.ext3 (1) -- /] fsck.ext3 -a -CB /dev/hda1
/dev/hda1 has gone 283 days without being checked, check forced.
/dev/hda1:
Inode 2897156 has illegal block(s).

/dev/hda1: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
(i.e., without -a or -p options)
fsck died with exit status 4
* Root file system check failed with error code 4.
A log is being saved in /var/log/fsck/checkroot if that location is writable.
* An automatic file system check (fsck) of the root filesystem failed.
A manual fsck must be performed, then the system restarted.
The fsck should be performed in maintenance mode with the
root filesystem mounted in read-only mode.
* The root filesystem is currently mounted in read-only mode.
A maintenance shell will now be started.
After performing system maintenance, press CONTROL-D
to terminate the maintenance shell and restart the system.
Give root password for maintenance
(or type Control-D to continue):
compaudit:91: command not found: getent
# fsck /dev/hda1
fsck 1.39 (29-May-2006)
w2fsck 1.39 (29-May-2006)
/dev/hda1 contains a file system with errors, check forced.
Pass 1: Checking inodes, blocks, and sizes
Inode 2897156 has illegal block(s). Clear(y)? yes

Illegal block #3 (541865746) in inode 2897156. CLEARED.
Inode 2897156, i_blocks is 48, should be 32. Fix(y)? yes

Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Block bitmap differences: -4194834
Fix(y)? yes

Free blocks count wrong for group #128 (8, counted=1).
Fix(y)? yes

Free blocks count wrong (3483689, counted=3483618).
Fix(y)? yes

/dev/hda1: ***** FILE SYSTEM WAS MODIFIED *****
/dev/hda1: ***** REBOOT LINUX *****
/dev/hda1: 877268/14893856 files (9.1% non-contiguous), 26388892/29784582 blocks
# reboot_
Err 3 #2
```

https://upload.wikimedia.org/wikipedia/commons/7/74/Fsck_output.jpg

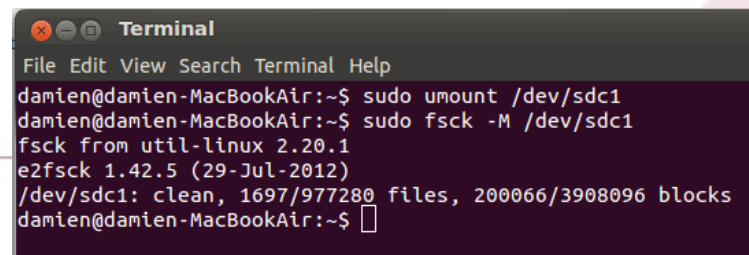
fsck: o bom e o mau...

✓ O bom do *fsck*

- Liberta o SF de se preocupar com a consistência
- O SF fica mais simples

✓ O mau do *fsck*

- Aplicação difícil de implementar
 - Podem existir muitos tipos de inconsistências num SF
 - Casos raros são difíceis de tratar (*corner cases*)
- *fsck* é lento
 - Analisa o SF várias vezes
 - Aplicação não escala com o tamanho dos discos (e.g., 10 TiB...)



```
Terminal
File Edit View Search Terminal Help
damien@damien-MacBookAir:~$ sudo umount /dev/sdc1
damien@damien-MacBookAir:~$ sudo fsck -M /dev/sdc1
fsck from util-linux 2.20.1
e2fsck 1.42.5 (29-Jul-2012)
/dev/sdc1: clean, 1697/977280 files, 200066/3908096 blocks
damien@damien-MacBookAir:~$
```

Opção 2: *Journaling*

✓ *Journaling*

- Registrar em log as últimas escritas através de um mecanismo confiável
- Quando ocorre crash, uso do log para efetuar as operações interrompidas pelo crash
- Uso de **write-ahead log (WAL)**
 - Designação tradicional: **journal**

✓ *Journaling*

- Empregue pelo ext3, pelo NTFS (Windows), pelo APFS (iOS e macOS) e por bases de dados SQLite

Write-Ahead Log (WAL)

- ✓ **Ideia base:** escritas para disco são primeiramente efetuadas para um *log* (registo)
 - Após a escrita do log, a escrita efetua-se normalmente
 - O log serve para registar a “transação”
- ✓ O que sucede quando ocorre um *crash*?
 - Se a escrita no log é interrompida?
 - Transação incompleta
 - Dados do utilizador são perdidos, mas SF fica consistente
 - Escrita no log bem sucedida, mas escrita no SF falha
 - SF fica inconsistente, mas os registos do *log* permite solucionar o problema

Exemplo

- ✓ Escrita de dados num ficheiro
 - **Três escritas:** inode v2, mapa binário v2, dados D_2
- ✓ Antes de efetuar a escrita, escreve para o *log*

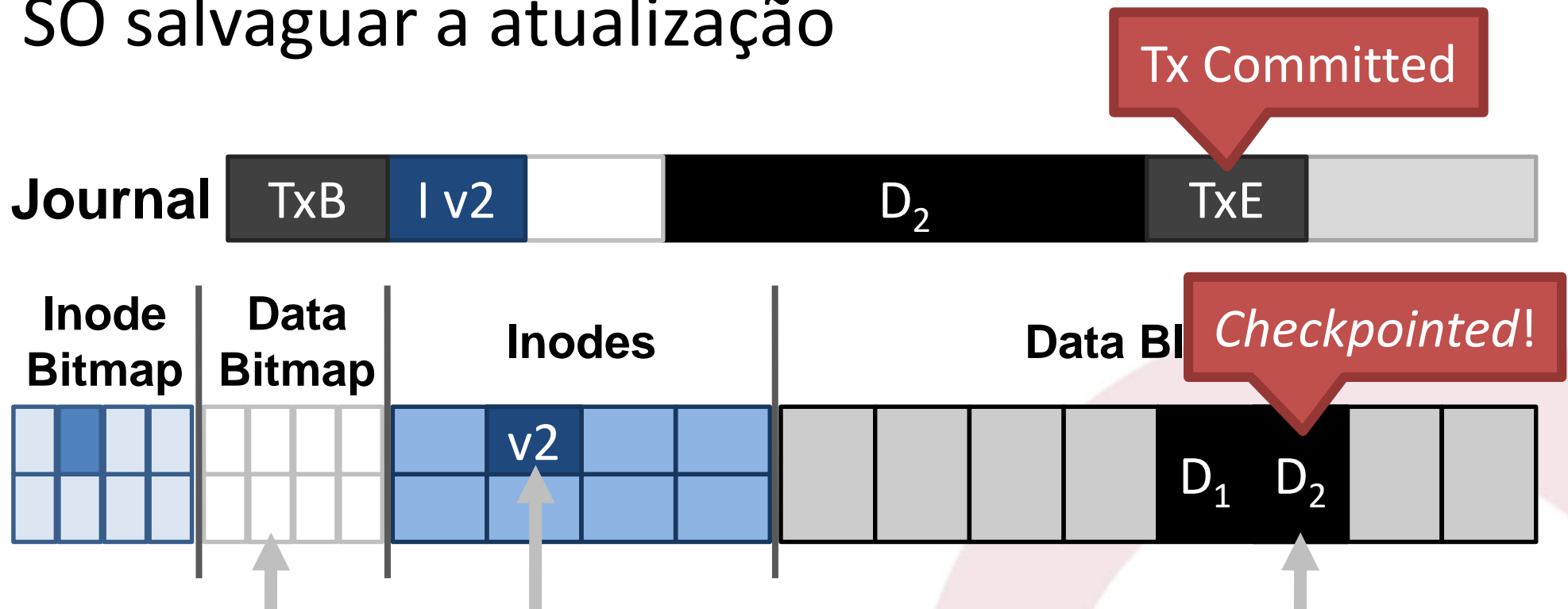
Journal



1. Inicia transação com identificador $ID=k$
2. Escreve os metadados
3. Escreve os dados
4. Encerra transação $ID=k$

Commits e Checkpoints

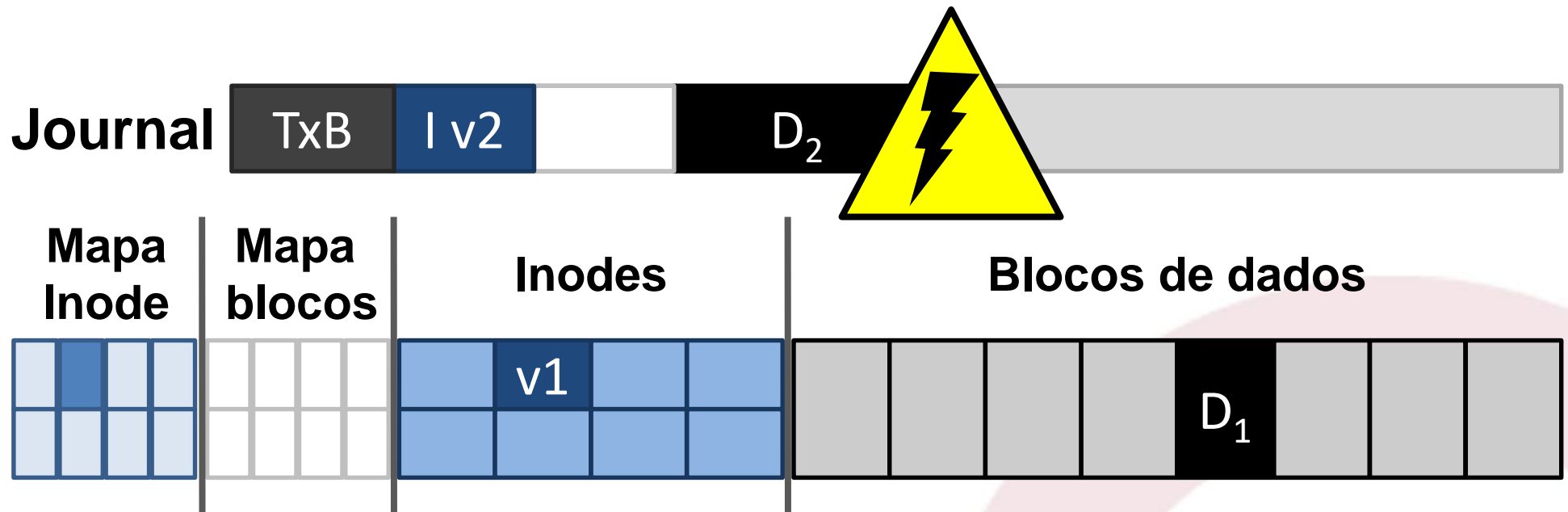
- ✓ Transação em *commit* quando todas as escritas no *log* estão finalizadas
- ✓ Logo que a transação esteja no estado *commit*, o SO salvaguardar a atualização



- Último passo: libertar a transação

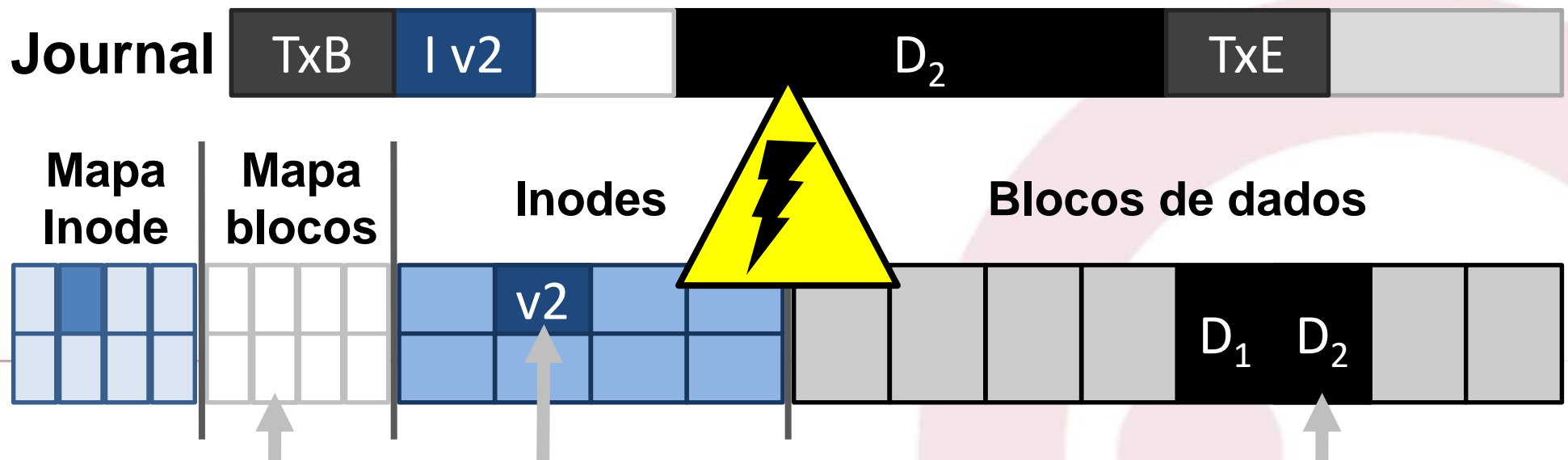
Recuperação *crash* (1)

- ✓ E se o sistema *crasha* durante a operação de log?
 - Se a transação não tiver *committed*, há perda de dados
 - Mas...o SF permanece consistente



Recuperação *crash* (2)

- ✓ E se o sistema crasha durante a operação de checkpoint?
 - No iniciar do sistema, as transações registadas mas não terminadas são novamente efetuadas, conforme a ordem original (fase de recuperação)
 - Não existe perda de dados, nem de consistência do SF



✓ O bom do *journaling*

- Robusto, recuperação rápida do SF
 - Não é necessário análise a todo o SF
- Implementação relativamente simples

✓ O mau do *journaling*

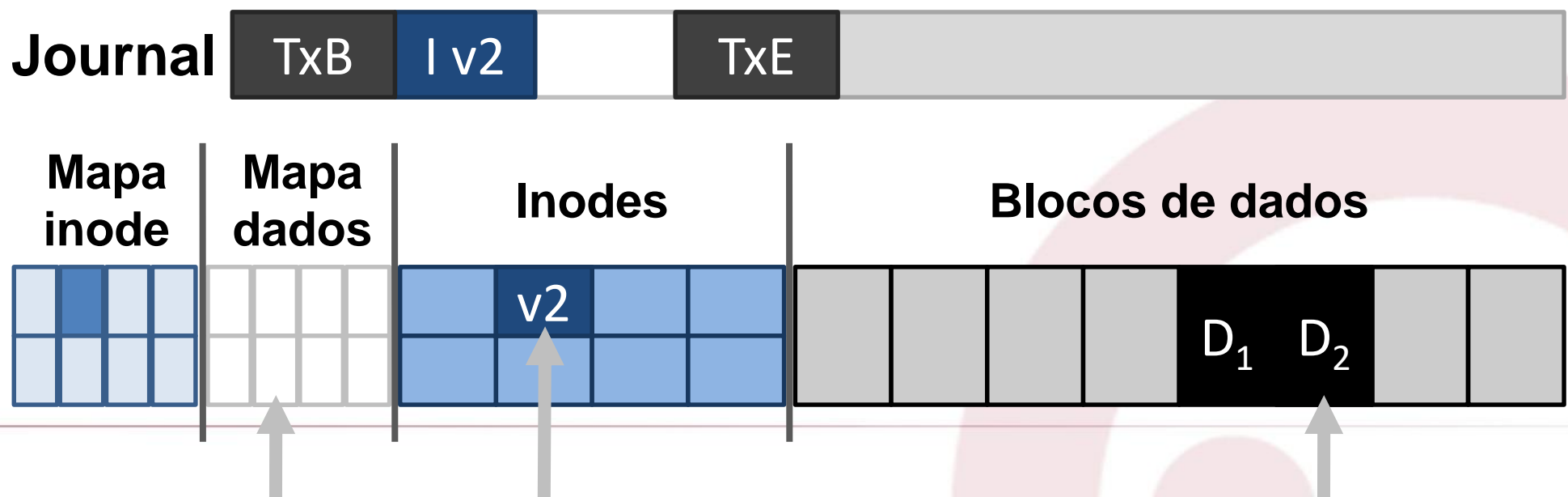
- Aumento das escritas para disco
 - 2x mais escritas
- Operações de apagar não são fáceis de registar em *journal*: perigo de um bloco (apagado) ser reutilizado

Acelerar o *journaling*

- ✓ Journaling aumenta consideravelmente as escritas
 - Até 2x mais
- ✓ SO tipicamente “*bufferizam*” escritas para ficheiros
 - Escritas recolhidas na memória, e periodicamente, efetivação dos *buffers* para disco
 - **Exemplo:** ext3 efetua atualizações de 5 em 5 segundos
 - Minimizar as perdas que possam ocorrer
- ✓ Equilíbrio entre desempenho e persistência
 - Intervalo de escrita maior = menos escritas, com maior volume de dados
 - Se existir um crash, ocorrerá perda de dados

Journaling dos metadados

- ✓ Journaling de dados do utilizador é oneroso
 - Necessidade de escrever duas vezes os dados no disco
 - Mas os metadados de ficheiro são pequenos
- ✓ ext3 implementa **journaling** dos metadados
 - *Journaling lógico*



- ✓ Os SO modernos disponibilizam SF com journaling
 - ext3/ext4 / Linux
 - NTFS / Windows
 - APFS / macOS
- ✓ SF tolerantes a crashes, mantendo bom desempenho
- ✓ Problemas ao nível do volume de escritas
 - Compromisso volume de escritas vs. consistência do SF

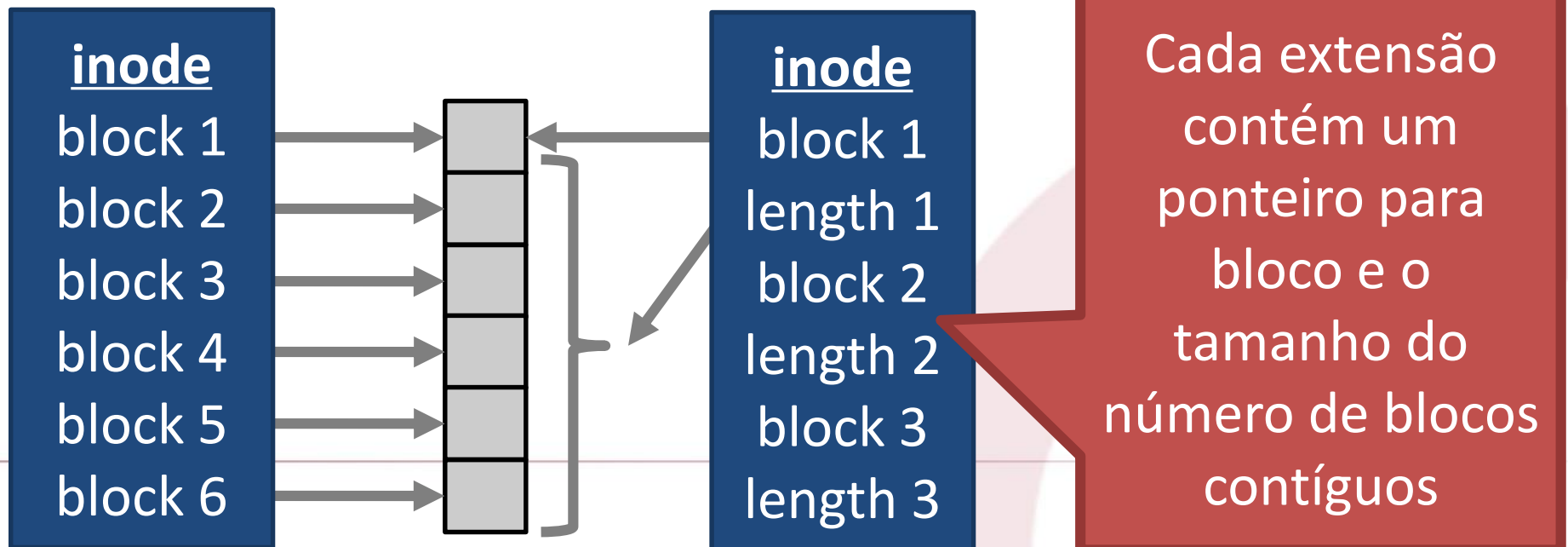
EXT4: MELHORAR DESEMPENHO

Otimizar *inodes*

- ✓ *inodes* usam **indireção** quando o tamanho do ficheiro ultrapassa a capacidade do nível 1
- ✓ Problema: *inodes* não são eficientes para ficheiros de grandes dimensões
 - **Exemplo:**
 - Ficheiro 100MiB requer 25600 blocos de nível i ($i > 1$), considerando blocos de dados de KiB
 - Dá origem a fragmentação externa do ficheiro
 - Fragmentos do ficheiros distribuídos pelo disco
 - Solução
 - Uso de grupos de blocos contíguos

Uso de extensões

- ✓ SF modernos procuram minimizar fragmentação
 - Em HDD, fragmentação origina muitos movimentos de posicionamento da cabeça leitura/escrita
 - LENTO!
- ✓ **Extensões** mais apropriadas para ficheiros de grandes dimensões



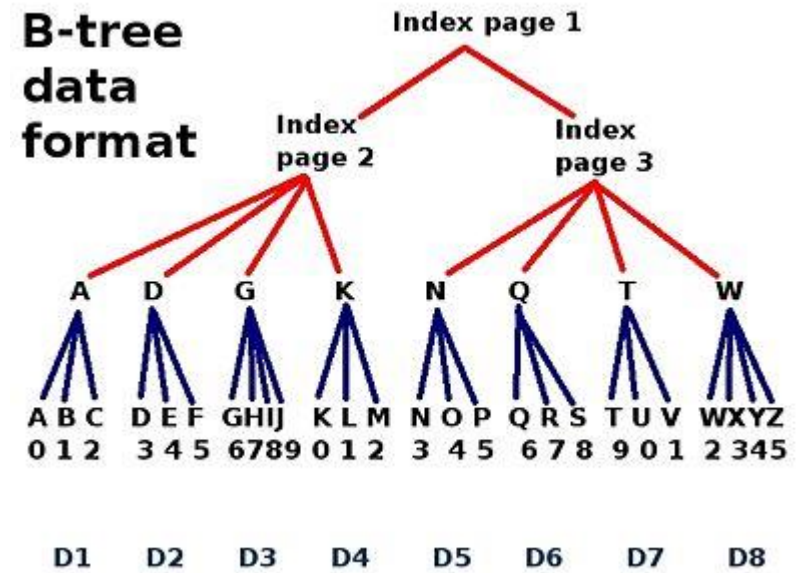
- ✓ ext4 e NTFS usam extensões
- ✓ Um inode do ext4 usa 4 extensões em vez de ponteiros para blocos
 - Cada extensão pode endereçar até 128 MiB de espaço consecutivo (*contíguo*)
 - Se for necessário mais espaço, é empregue mais uma extensão
- NOTA: designação anglo-saxónica - **Extent**

Otimizar diretórios

- ✓ No SF ext, ext2, ext3, cada diretório é um ficheiro com uma lista de entradas
 - Entries não estão ordenadas
 - Algumas entradas podem estar vazias, se os respetivos ficheiros foram apagados
- ✓ Problema: pesquisa por ficheiros em diretórios com muitas entradas cresce lineamente com n (diz-se que é **$O(n)$**)
 - Efeito prático: não guardar mais do que 10000 ficheiros num diretório

SF modernos: B-Trees

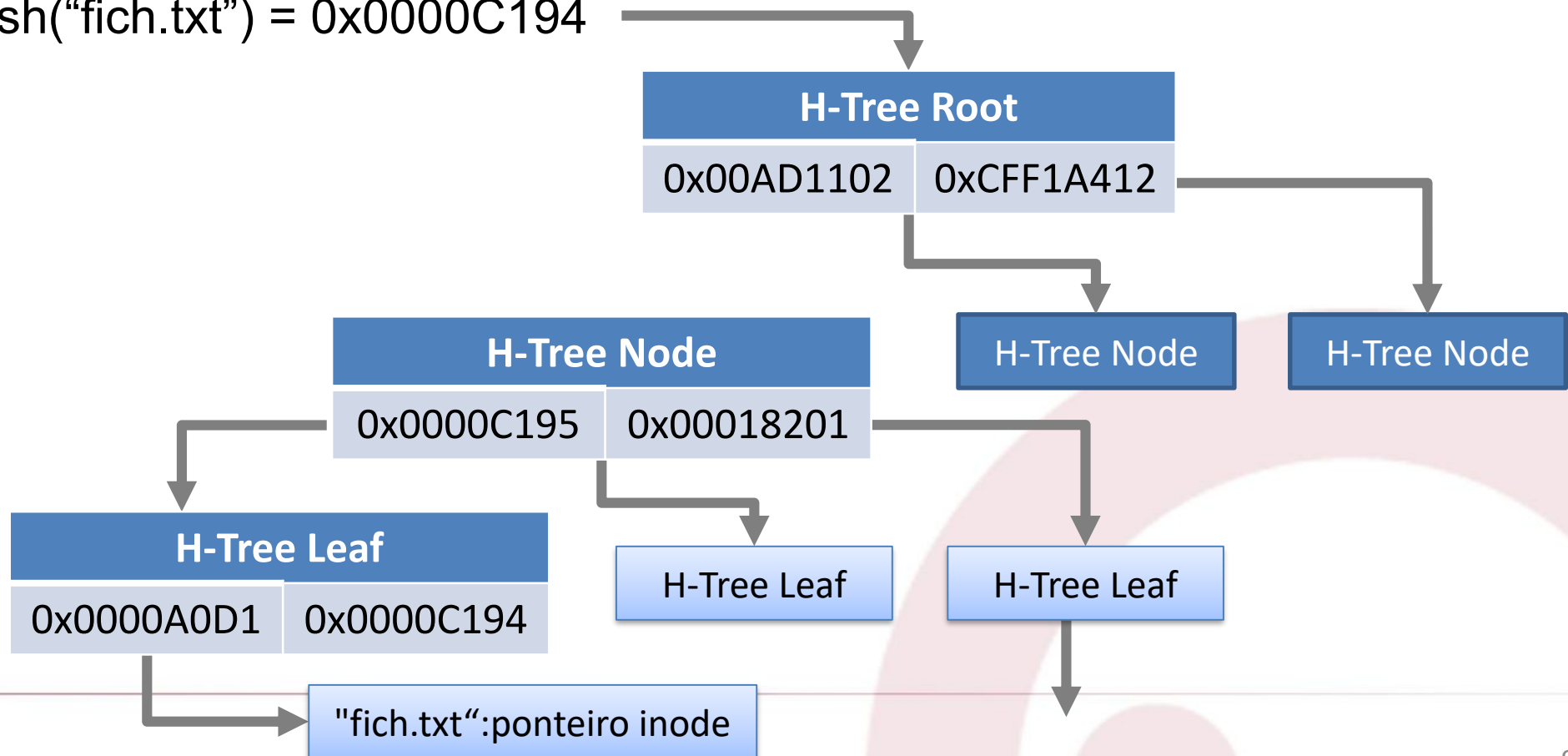
- ✓ Ext4/NTFS/APFS usam **B-Trees** para melhorar desempenho de diretórios
 - Itens guardados de forma ordenada nos blocos
 - Tempo de pesquisa cresce com $\log N$, i.e., $O(\log N)$
 - B-Tree
 - Arvore equilibrada
 - Otimizada para armazenamento no disco
 - Cada bloco guarda entre m e $2m$ itens



Exemplo *B-Tree*

- ✓ ext4 usa variante de B-Tree: H-Tree
 - *H* de *hash* (Também designada de B+Tree)
- ✓ Abrir ficheiro `open("fich.txt", "r")`

`hash("fich.txt") = 0x0000C194`



ext4: o bom e o mau...

- ✓ O bom – ext4/NTFS/APFS disponibilizam:
 - Funcionalidades típicas de SF
 - Desempenho melhorado relativamente ao ext3
 - Suporte de escala para diretórios
- ✓ O mau
 - ext4 melhoria incremental do ext3
 - SF mais avançados assentam em COW
 - Copy-on-write (btrfs e ZFS)

Ext4 – limites



- ✓ Linux disponibiliza SF diferentes:
 - ✓ ext2, ext3 e ext4, XFS, JFS, ReiserFS, btrfs
 - ext4: 4th *extended filesystem*

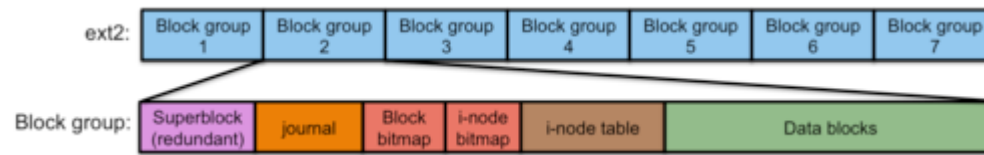
- Também empregue pelo Android

– Limites

- Partição: to 1 ExaByte (EiB)
 - Tamanho máximo ficheiro: 16 TiB
 - Máximo 64000 entradas por diretório
 - ext3 limitado a 32000 entradas por diretório

Limit	ext3	ext4	XFS
max file system size	16 TiB	16 TiB	16 EiB
max file size	2 TiB	16 TiB	8 EiB
max extent size	4 kiB	128 MiB	8 GiB
max extended attribute size	4 kiB	4 kiB	64 kiB
max inode number	2 ³²	2 ³²	2 ⁶⁴

<http://linuxmantra.com/2013/09/xfs-in-rhel6.html>



<http://www.cs.rutgers.edu/~pxk/416/notes/13-fs-studies.html>



MARS SPIRIT ROVER (2004)

Mars Rover Spirit (2004) #1



- ✓ Problema com a memória flash
 - Dois veículos autónomos (“rovers”) da NASA
 - Spirit & Opportunity
 - Spirit com problemas em 2004.01.21
 - Comportamento anormal...
 - *debugging* à distância: milhões de kms...
 - Origem do problema: software do SF na memória *flash*
 - Duas configurações amplificaram a situação de crash/reboot despoletada pelo erro principal
 - Cenário típico da cadeia de erros (“*chain of errors*”)

Mars Rover Spirit (2004) #2

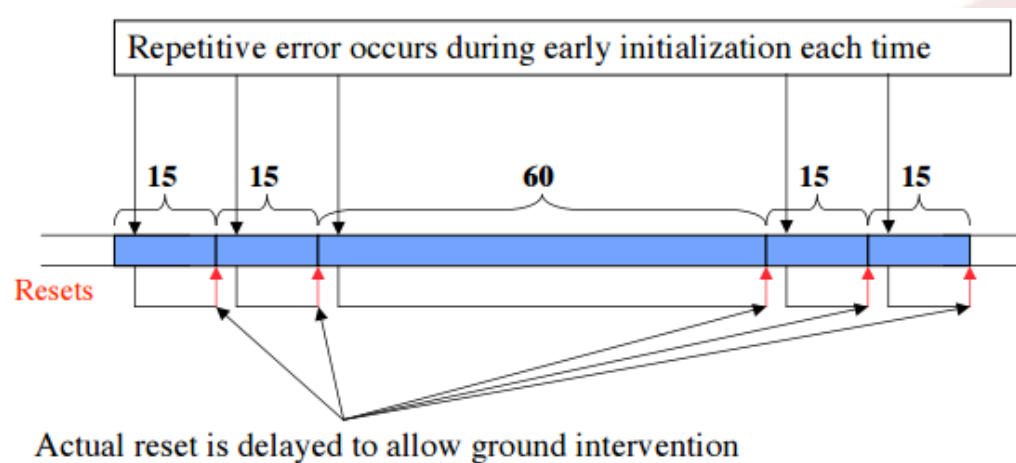


- Problemas no contacto com o Spirit
 - Muitos ciclos de crash/reboot que levavam a “low power” (não desligava durante a “noite”)
- Inexistência de dados na memória flash indicava:
 - Avaria na memória flash
- OU
- Problema no SF que gere a memória flash
- OU
- Problema no software que lê os dados e os transmite para a Terra

Mars Rover Spirit (2004) #3

✓ O problema

- Biblioteca empregue para SF (tipo FAT) guarda na memória toda a estrutura do SF, i.e.:
 - i) diretórios e ficheiros
 - ii) ficheiros apagados (entradas marcadas com o octeto E5)
- A estrutura é recriada sempre que o sistema inicia
 - A estrutura estava demasiadamente grande, ocupando toda a memória
 - A falta de espaço na memória principal ativava o cão de guarda (*watchdog*) que ordenava operação de reboot
 - O sistema estava em ciclos de *crash/reboot*...

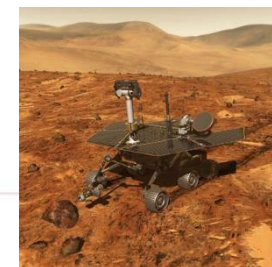


Mars Rover Spirit (2004) #4

- Após análises, estudos e muitos dias...
- Partição foi formatada, reconstruindo-se o SF
 - Problema resolvido
 - Fonte: “The Mars Rover Spirit FLASH Anomaly”, Glenn Reeves, Tracy Neilson, 2004

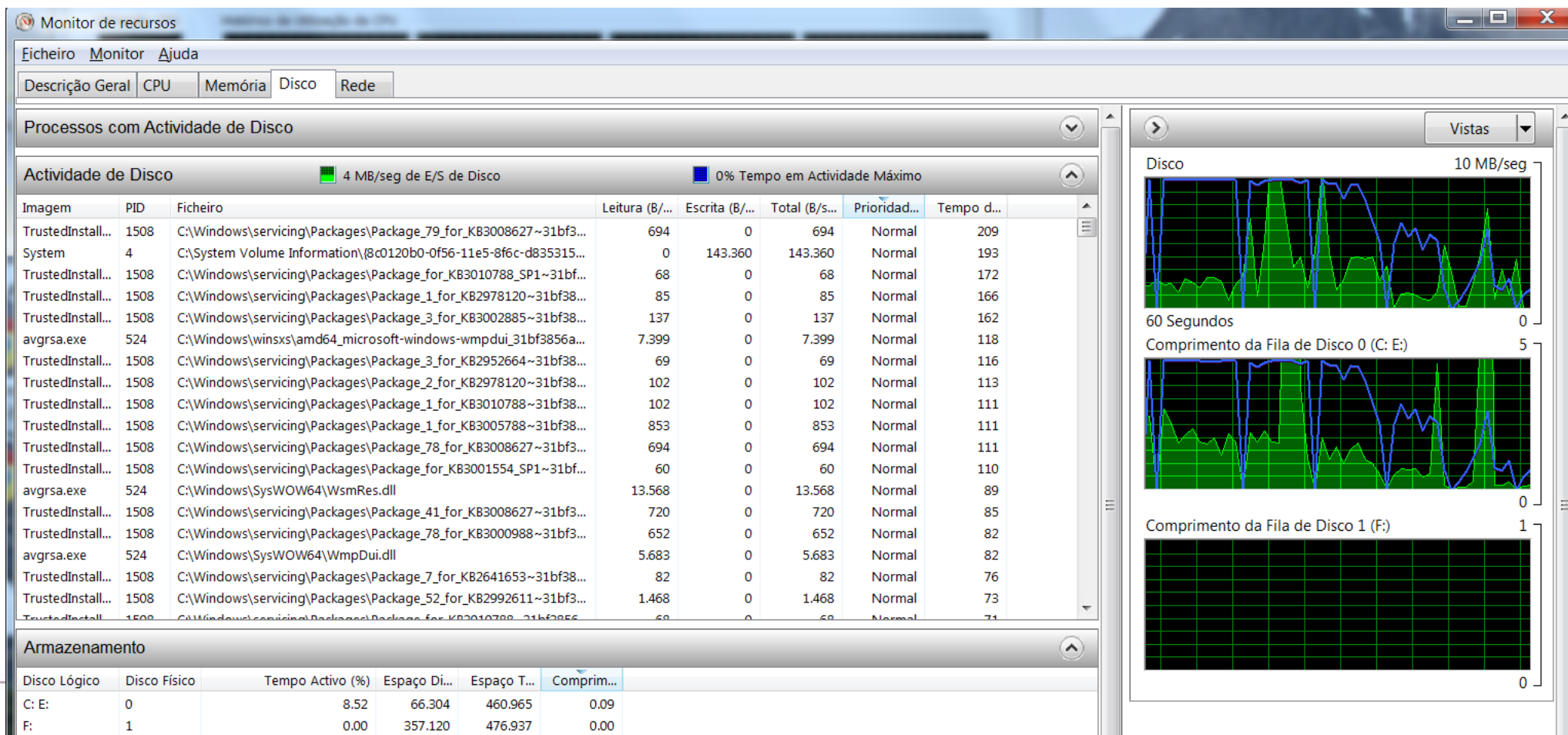
<https://www.cs.princeton.edu/courses/archive/fall11/cos109/mars.rover.pdf>

- Spirit – momentos marcantes
 - 2003.06.10: Lançado dos EUA.
 - 2004.01.04: pouso em Marte. Missão prevista 90 dias.
 - 2006: Uma roda da frente gripou. Passa a andar de marcha atrás.
 - 2009.05.01: Spirit fica atolado. A NASA tenta durante 8 meses libertá-lo. A missão prossegue, mesmo atolado.
 - 2010.03.22: Perdida a comunicação com o Spirit.
 - 2011.05.11: Sem comunicação do Spirit, missão é terminada. Percorreu 7.730 metros e enviou mais e 124000 fotos.



Atividade SF@Windows

✓ Resource monitor (\geq windows 7)



✓ Unix – utilitário df

- Opção -T mostra o tipo de sistema de ficheiros

```
user@ubuntu: ~  
File Edit Tabs Help  
user@ubuntu:~$ df -T
```

Filesystem	Type	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda2	ext3	20414332	2952088	16418584	16%	/
none	tmpfs	4	0	4	0%	/sys/fs/cgroup
udev	devtmpfs	503420	12	503408	1%	/dev
tmpfs	tmpfs	102604	844	101760	1%	/run
none	tmpfs	5120	0	5120	0%	/run/lock
none	tmpfs	513004	0	513004	0%	/run/shm
none	tmpfs	102400	16	102384	1%	/run/user
/dev/sdb1	ext3	103079200	519600	97316824	1%	/home

- Formatação
 - Cria as estruturas do sistema de ficheiros
 - *super block, inodes, mapa binário inodes, FAT, etc.*
 - Comandos
 - `format` (Win32), `mke2fs` (Linux)
 - Exemplo: `mke2fs -t ext4 /dev/sda3`
- Setores impróprios para uso
 - Praticamente todos os discos têm setores/blocos inoperacionais
 - Gasto do material com as elevadas densidade de dados nos discos modernos
 - `scandisk` (Win32) **OU** `badblocks` (Linux)
 - Blocos inoperacionais colocados na lista de “*bad-blocks*”
 - Disco modernos gerem eles próprios os setores inoperacionais (transparente para o SO)

Manutenção do SF (#2)

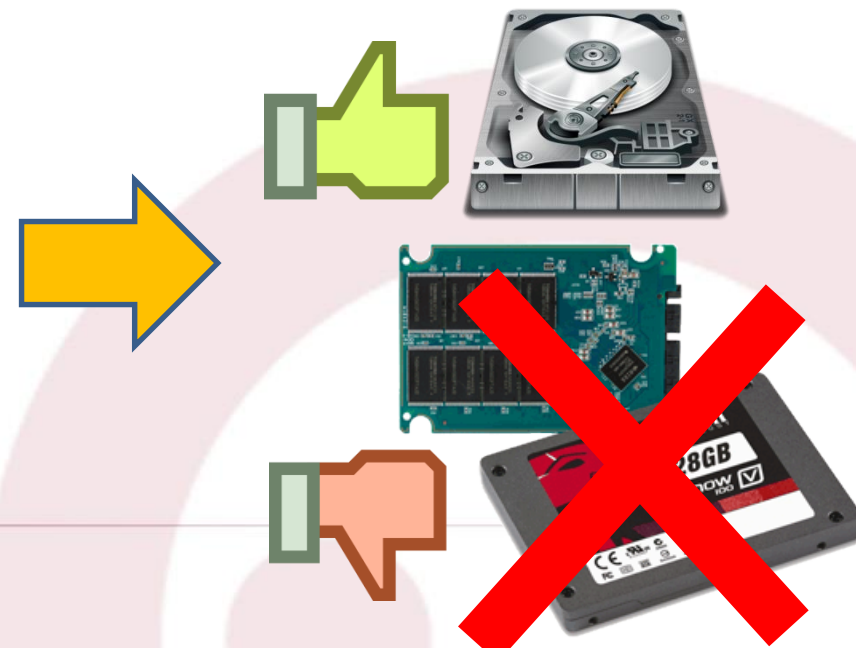
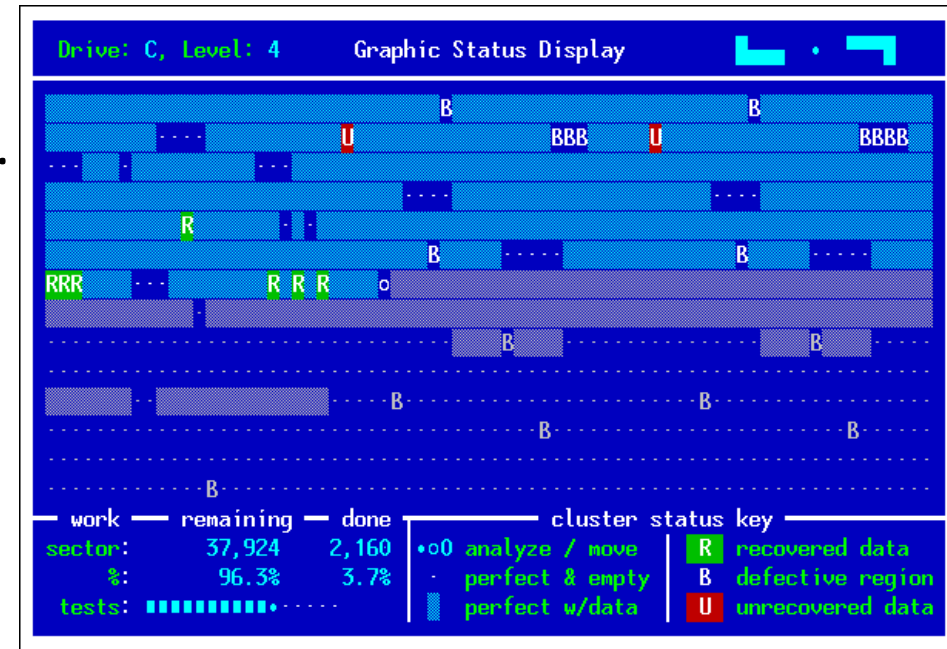
✓ Recuperação

- “lost+found” (unix), correção do SF, etc.
- Ferramentas externas
 - Spinrite

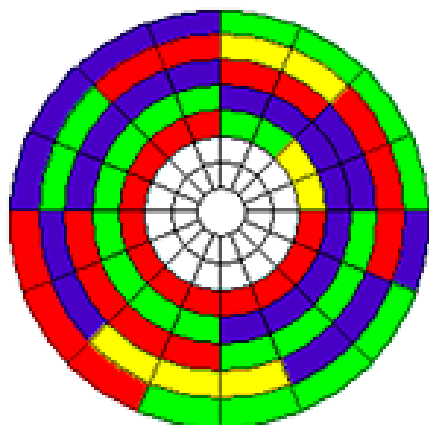


✓ Defragmentação

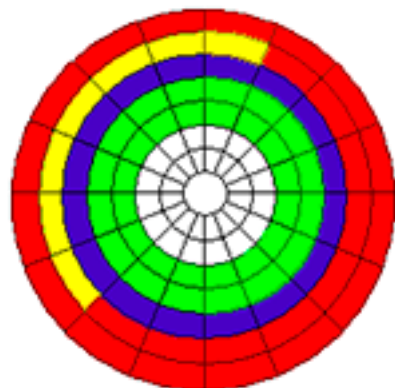
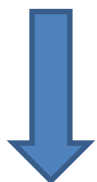
- Reorganizar os blocos do disco para maximizar a sequencialidade dos ficheiros
- Importante para HDD, nada recomendado para SSD
 - SSD não apresentam fragmentação passível de ser solucionada com a desfragmentação tradicional



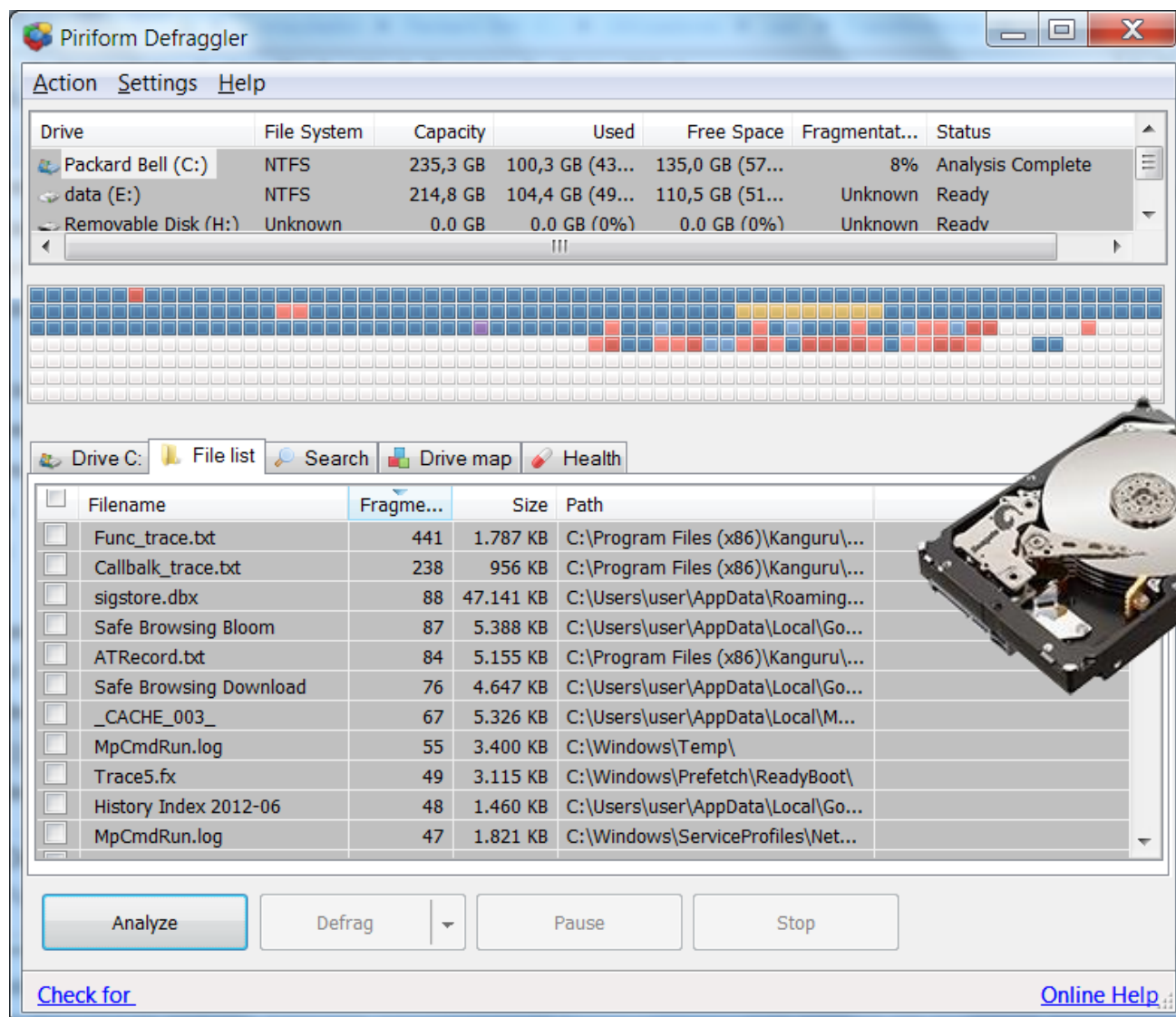
Defragmentação (windows)



■ File A ■ File D
■ File B ■ File C



■ File A ■ File D
■ File B ■ File C



Drive	File System	Capacity	Used	Free Space	Fragmentat...	Status
Packard Bell (C:)	NTFS	235,3 GB	100,3 GB (43...)	135,0 GB (57...)	8%	Analysis Complete
data (E:)	NTFS	214,8 GB	104,4 GB (49...)	110,5 GB (51...)	Unknown	Ready
Removable Disk (H:)	Unknown	0.0 GB	0.0 GB (0%)	0.0 GB (0%)	Unknown	Ready

Filename	Frage...	Size	Path
Func_trace.txt	441	1.787 KB	C:\Program Files (x86)\Kanguru\...
Callbalk_trace.txt	238	956 KB	C:\Program Files (x86)\Kanguru\...
sigstore.dbx	88	47.141 KB	C:\Users\user\AppData\Roaming...
Safe Browsing Bloom	87	5.388 KB	C:\Users\user\AppData\Local\Go...
ATRecord.txt	84	5.155 KB	C:\Program Files (x86)\Kanguru\...
Safe Browsing Download	76	4.647 KB	C:\Users\user\AppData\Local\Go...
_CACHE_003_	67	5.326 KB	C:\Users\user\AppData\Local\M...
MpCmdRun.log	55	3.400 KB	C:\Windows\Temp\
Trace5.fx	49	3.115 KB	C:\Windows\Prefetch\ReadyBoot\
History Index 2012-06	48	1.460 KB	C:\Users\user\AppData\Local\Go...
MpCmdRun.log	47	1.821 KB	C:\Windows\ServiceProfiles\Net...

[Check for...](#)
[Online Help](#)



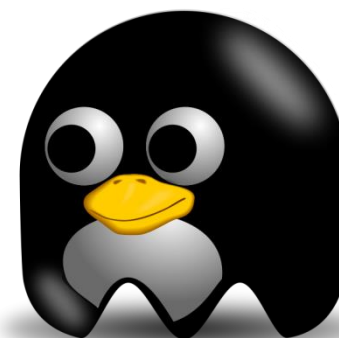
<http://learn.caconnects.org/mod/resource/view.php?id=138>

✓ SD “**procfs**”

- Pseudo sistema de ficheiros
- Ativado em **/proc**
- O conteúdo do SF procfs apenas existe na memória
- É uma interface para acesso às estatísticas do kernel e para configurar parâmetros do kernel
- Os dados são disponibilizados através de ficheiros e diretórios
- Podem ser empregues as ferramentas e software de acesso a ficheiros e diretórios
 - find, cut, cat, tr, grep, etc.

– Exemplos

- cat /proc/interrupts
- cat /proc/cpuinfo



✓ cat /proc/cpuinfo

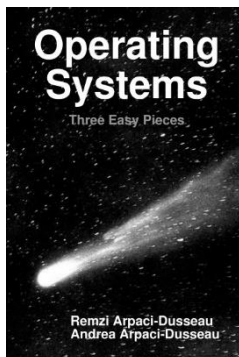
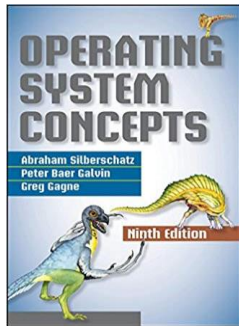
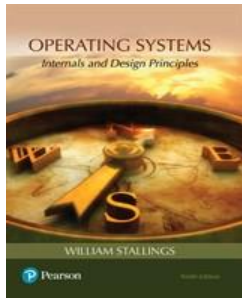
```
processor           : 0
vendor_id           : GenuineIntel
cpu family          : 6
model               : 8
model name          : Pentium III (Coppermine)
stepping            : 10
cpu MHz             : 930.335
cache size          : 256 KB
fdiv_bug            : no
hlt_bug             : no
f00f_bug            : no
coma_bug            : no
fpu                 : yes
fpu_exception       : yes
cpuid level         : 2
wp                  : yes
flags               : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge
                    mca cmov pat pse36 mmx fxsr sse
bogomips            : 1862.26
```

✓ cat /proc/cpuinfo

```
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz
stepping      : 7
microcode     : 0x29
cpu MHz       : 2301.000
cache size    : 3072 KB
physical id    : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid: 0
fdiv_bug      : no
f00f_bug      : no
coma_bug      : no
fpu           : yes
fpu_exception : yes
cpuid level    : 13
wp            : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss nx
rdtscp lm constant_tsc arch_perfmon pebs bts tsc_reliable nonstop_tsc aperfmperf pni pclmulqdq ssse3 cx16 sse4_1
sse4_2 popcnt aes lahf_lm epb dtherm ida arat pln pts
bugs          :
bogomips      : 4602.00
clflush size   : 64
cache_alignment: 64
address sizes: 36 bits physical, 48 bits virtual
power management:
```

- ✓ Associados ao DOS/Windows
 - FAT12, FAT16, FAT32, FAT64/exFAT
 - NTFS
- ✓ Associados ao Unix
 - ext2, ext3, ext4, reiserfs, btrfs, jfs, zfs
- ✓ Associados aos sistemas Apple
 - MFS (Macintosh File System)
 - HFS (Hierarchical File System) e HFS+
 - HFSX (dispositivos móveis)
 - APFS (Apple File System) (2016)

Bibliografia



- CS 5600 - Computer Systems - Professor Christo Wilson, Files and Directories
<https://cbw.sh/5600/index.html>
- Capítulos 11, 12: “Operating Systems – Internals and Design Principles”, William Stallings, 9th edition, 2018
- Capítulo 12: “Operating Systems Concepts”, A. Silberschatz, 9th edition, 2016
- Capítulos 35 a 46: Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Arpaci-Dusseau Books, August 2018.
 - <http://pages.cs.wisc.edu/~remzi/OSTEP/>



Sistemas Operativos

Patricio Domingues

