UNIVERSIDADE
VILA VELHA
ESPÍRITO SANTO

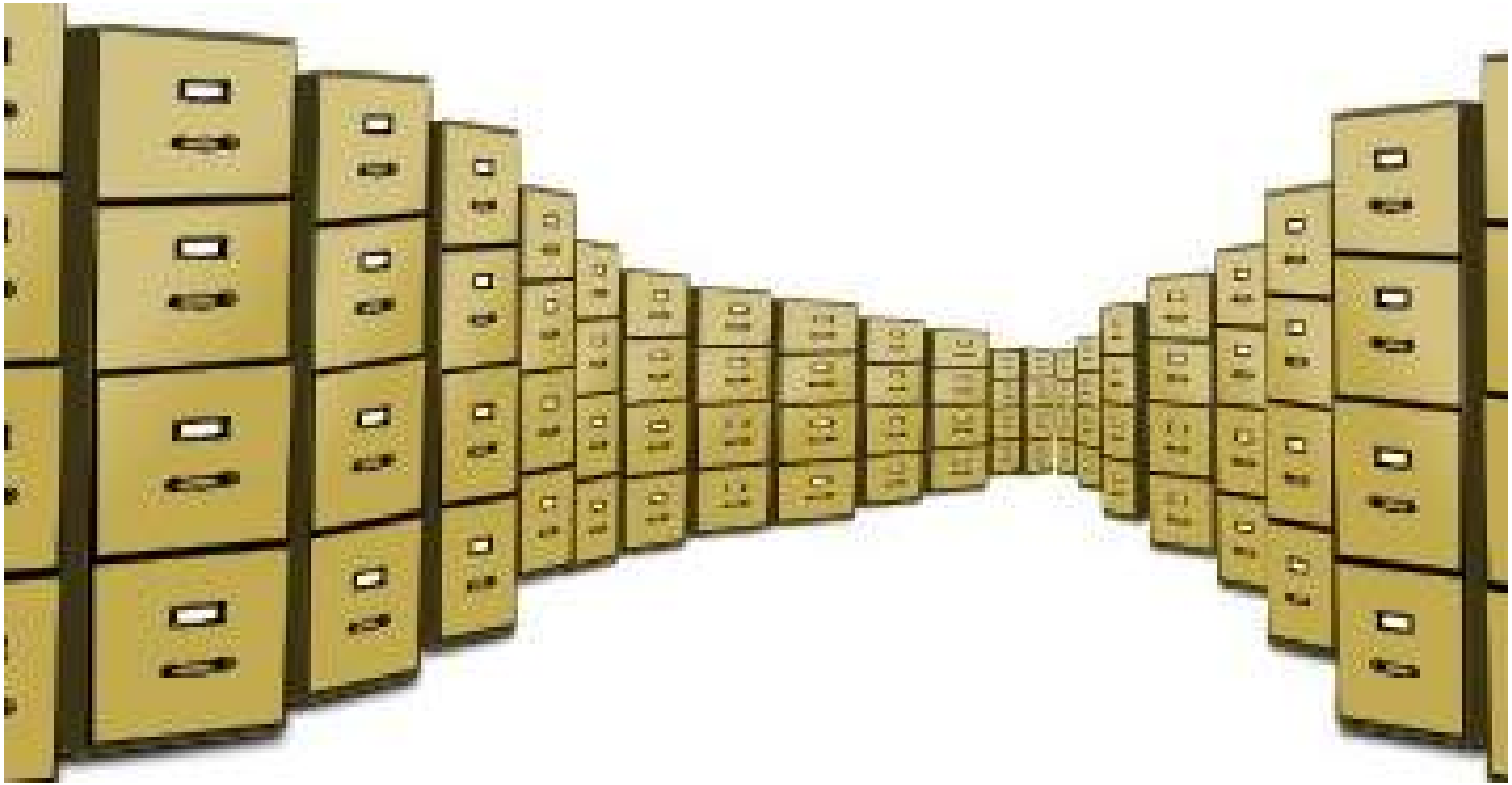
Filesystems

Prof. Jean-Rémi Bourguet

Sistemas Operacionais

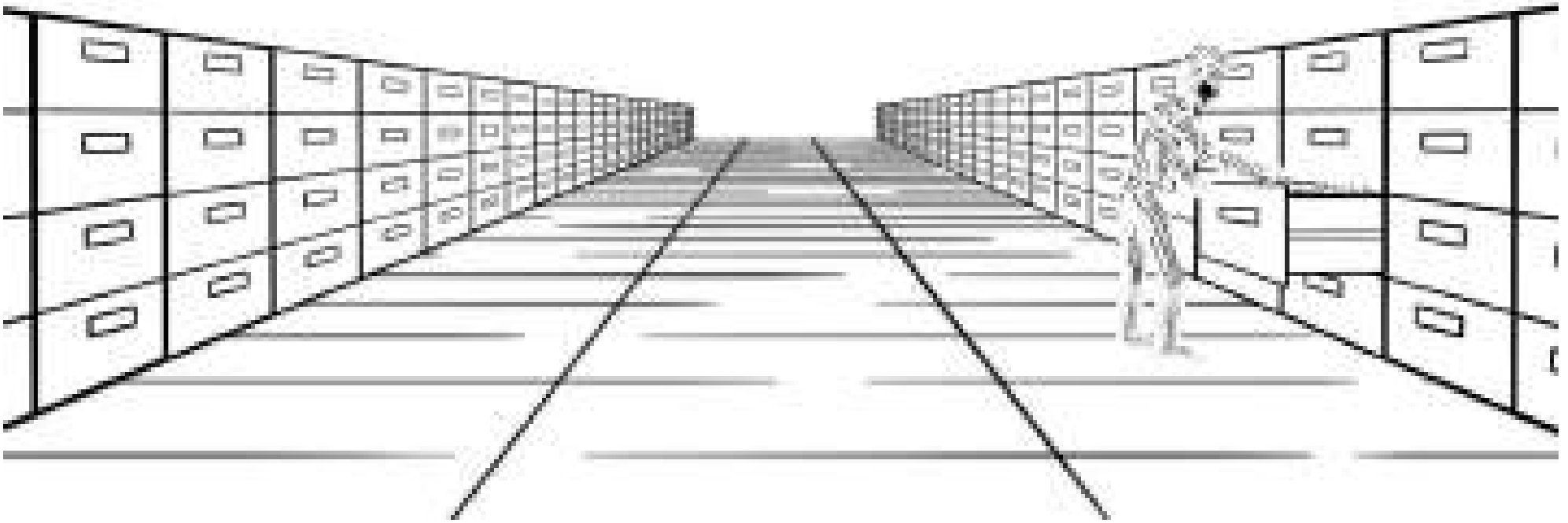
Filesystem (Sistema de arquivos)

- Em cada **partição** para **armazenar** e **organizar** o **acesso dos dados**.



VFS (Virtual Filesystem)

- **Interface** entre **kernel** e **FS** para **reconhecer diversos** tipos de **FS**.

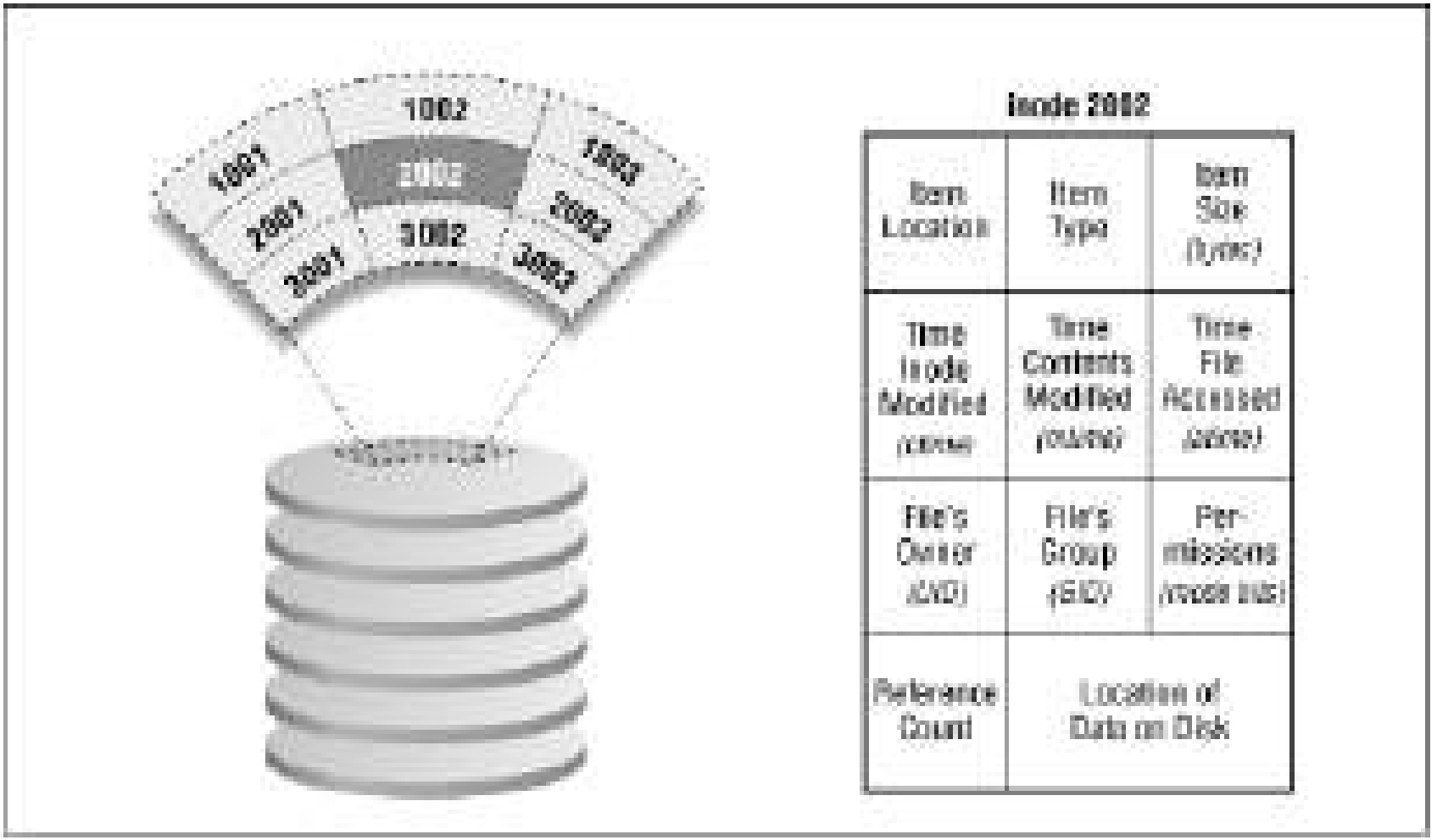


Área de controle e a Área de dados

- **A área de controle**: as **metadados** sobre os arquivos da partição.
- **A área de dados**: o verdadeiro **conteúdo dos arquivos**.

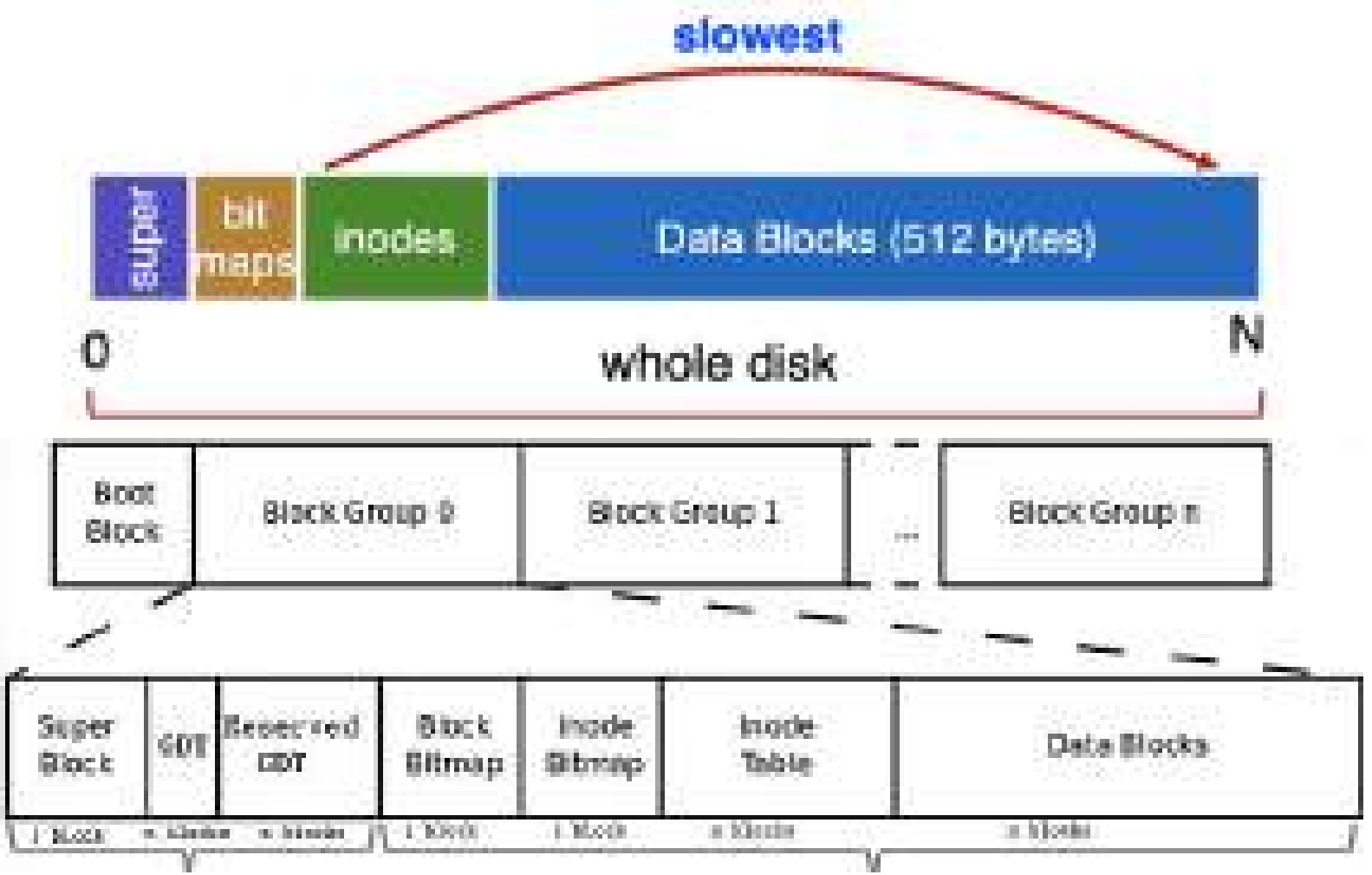
inode (index node)

- Tanto **os dados** quanto os inodes são **armazenados dentro de blocos**.



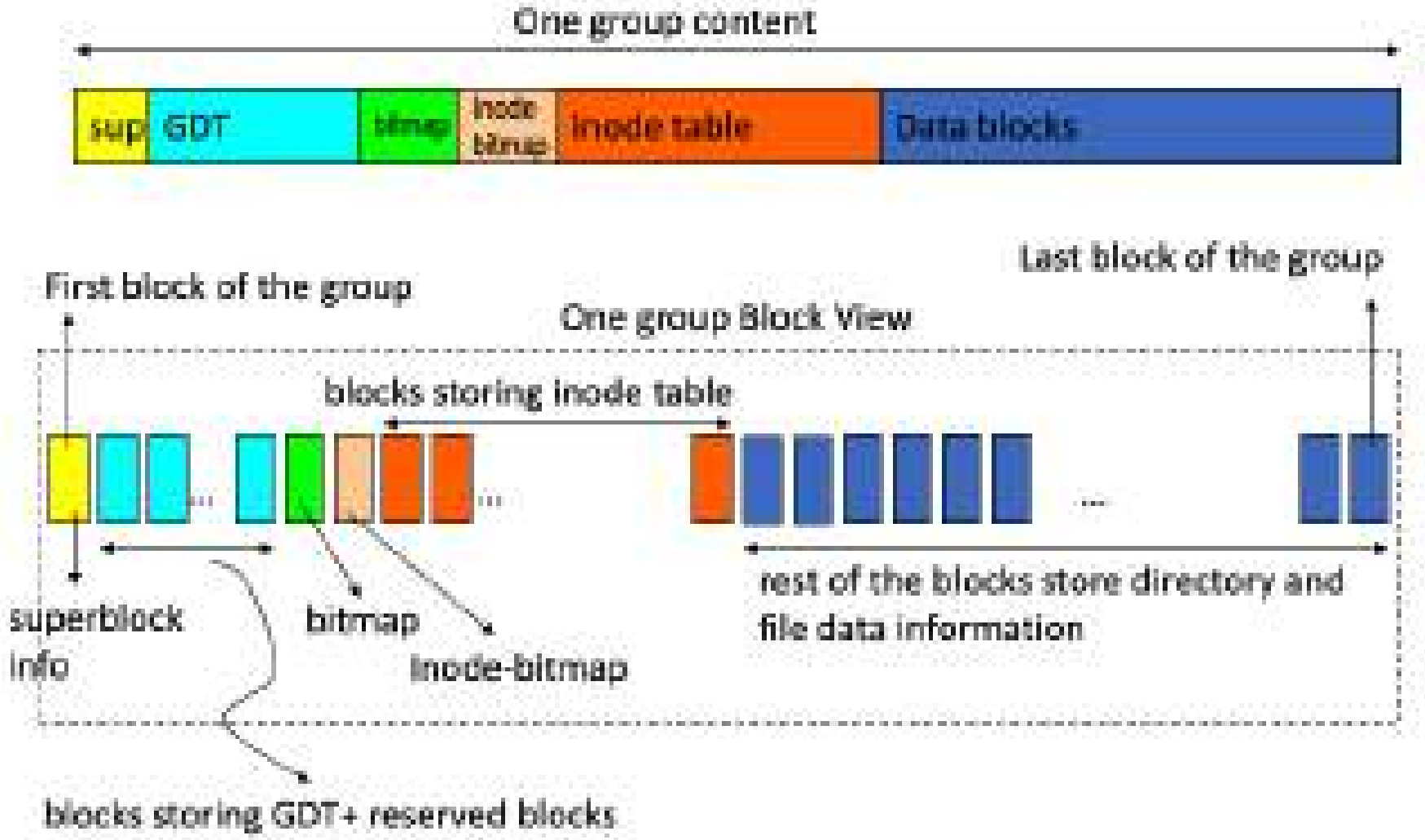
Estrutura dos Exts

- **Boot Block**: dados críticos para encontrar o restante o resto e dar boot.
- **Block Groups**: cada grupo contém sua próprias áreas de controle.



Estrutura dos Exts

- **Super Bl**: núm de blocos/inodes, estado FS (replicado em cada grupo).
- **G**roup **D**escriptor **T**able contagem e aponta para bitmaps/inode table.



Bitmap de Inodes e de Blocos

- Indica quais inodes/**blocos de dados** estão **ocupados** (1) ou **livres** (0).
- Quando um **novo arquivo/diretório** é criado, o **bitmap é consultado**...

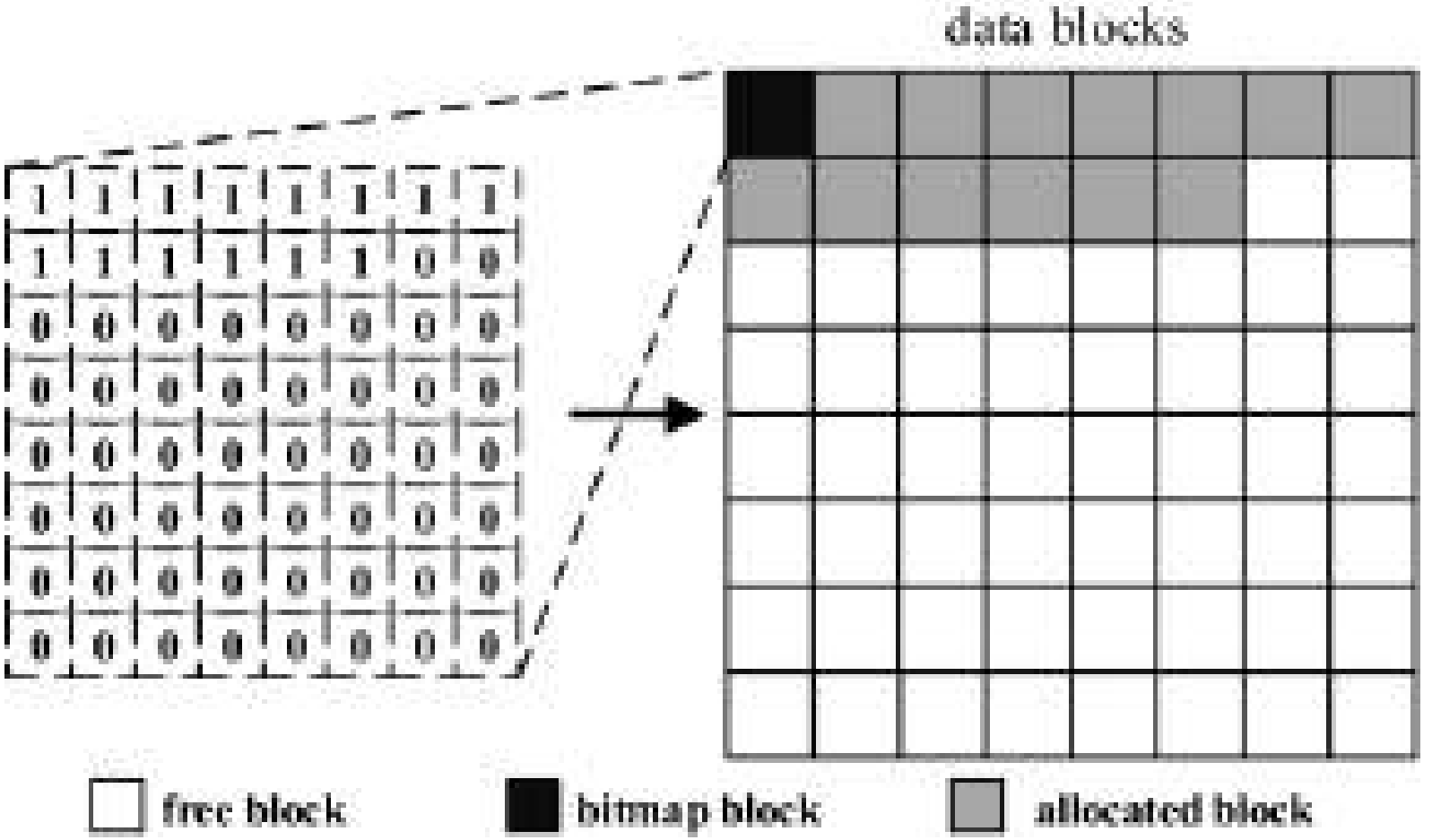
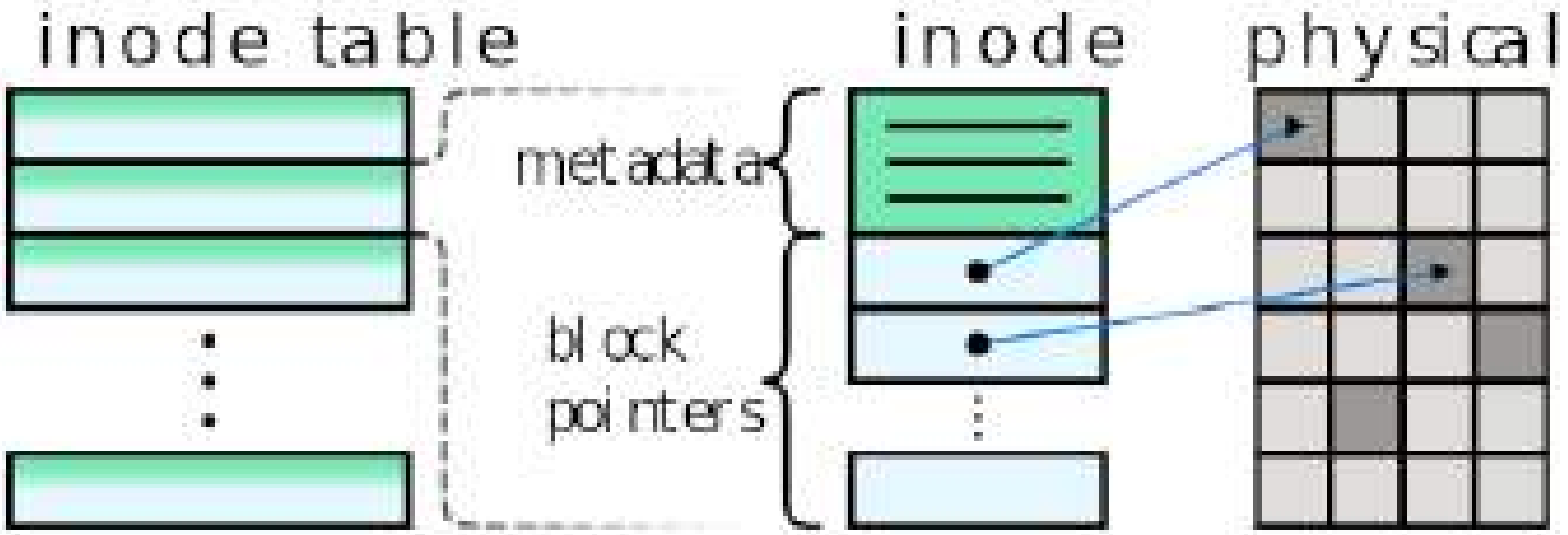


Tabela de Inodes

- **Inodes** (256 B) alocados em sequência e **localizáveis por numero**.
- Inodes **alocados dentro** de blocos chamamos de **blocos de controle**.



dump com dd e xxd

- Pode **observar terceiro sector** da partição do meu sistema em Ext4.

$ sudo dd if=/dev/sda2 bs=512 count=1 skip=2 2> /dev/null | xxd -g 1

```
00000000: 00 00 ee 00 00 e5 b7 03 40 98 2f 00 5e 37 82 01  ........@./.^7..
00000010: 09 1f df 00 00 00 00 00 02 00 00 00 02 00 00 00  ................
00000020: 00 80 00 00 00 80 00 00 00 20 00 00 48 d9 1e 67  ......... ..H..g
00000030: 47 d9 1e 67 3c 0a ff ff 53 ef 01 00 01 00 00 00  G..g<...S.......
00000040: 6e 2c f6 60 00 00 00 00 00 00 00 00 01 00 00 00  n,.`............
00000050: 00 00 00 00 0b 00 00 00 00 01 00 00 3c 00 00 00  ............<...
00000060: c6 02 00 00 6b 04 00 00 7f 66 9e 0f 9b 54 4d b4  ....k....f...TM.
00000070: 97 d9 28 ac 89 67 9e e3 00 00 00 00 00 00 00 00  ..(..g..........
00000080: 00 00 00 00 00 00 00 00 2f 00 00 00 00 00 00 00  ......../.......
00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04  ................
000000d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000e0: 08 00 00 00 00 00 00 00 9b 00 0e 00 f4 13 bc ee  ................
000000f0: e7 65 45 71 8a bd 73 cb 26 c2 ee f9 01 01 40 00  .eEq..s.&.....@.
00000100: 0c 00 00 00 00 00 00 00 6e 2c f6 60 0a f3 01 00  ........n,.`....
00000110: 04 00 01 00 00 00 00 00 00 00 00 00 ff 7f d8 01  ................
00000120: 00 00 00 00 00 80 00 00 00 80 00 00 00 00 d9 01  ................
00000130: 00 00 01 00 00 80 00 00 00 80 d9 01 00 80 01 00  ................
00000140: 00 80 00 00 00 00 da 01 00 00 00 00 00 00 00 40  ...............@
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 20  ............. . .
00000160: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000170: 00 00 00 00 04 01 00 00 a7 63 e9 e9 02 00 00 00  .........c......
```

0x68: **7f 66 9e 0f 9b 54 4d b4 97 d9 28 ac 89 67 9e e3** é um UUID!

dumpe2fs

- Imprime as **informações do superbloco** de **cada grupo de blocos**.

```
$ sudo dumpe2fs /dev/sda2 | less
Last mounted on:          /
Filesystem UUID:          7f669e0f-9b54-4db4-97d9-28ac89679ee3
Filesystem OS type:       Linux
Inode count:              15597568
Block count:              62383360
Free blocks:              25315955
Free inodes:              14622474
First block:              0
First inode:              11
Block size:               4096
Blocks per group:         32768
Inodes per group:         8192
Inode blocks per group:   512
Inode size:               256
Filesystem created:       Mon Jul 19 22:52:46 2021
Last mount time:          Sun Oct 27 18:01:16 2024
```

dumpe2fs

- No meu sistema, há **1904 block groups**, cada um com seus metadados.

```
$ sudo dumpe2fs /dev/sda2
Grupo 0: (Blocos 0-32767) csum 0xdf87 [ITABLE_ZEROED]
  Primária superbloco em 0, Descritores de grupo em 1-30
  Blocos GDT reservados em 31-1054
  Bitmap de bloco em 1055 (+1055), csum 0xbf212fcf
  bitmap de inode em 1071 (+1071), csum 0x5c1c7110
  tabela de inode em 1087-1598 (+1087)
  1316 livre blocos, 8174 inodes livres, 2 pastas
  Blocos livres: 9286-9303, 9437-9857, 11421-11447
  inodes livres: 19-8192

Grupo 1903: (62357504-62383359) [INODE_UNINIT, ITABLE_ZEROED]
  Bitmap de bloco em 61865999 (bg 1888 + 15), csum 0x676f2306
  bitmap de inode em 61866015 (bg 1888 + 31), csum 0x00000000
  tabela de inode em 61873696-61874207 (bg 1888 + 7712)
  25542 livre blocos, 8192 inodes livres, 0 pastas
  Blocos livres: 62357504-62359551, 62359587-62359871
  inodes livres: 15589377-15597568
```

Um inode por arquivo

- **Cada arquivo** da **área de dados** possui um **inode que o controla**.

Um inode por arquivo

Isso poderá ser visto com o comando **`$ ls -i`**:

~/Documents/Aulas/SysOpe/5-Filesystems$ ls -i

```
7614760 5-Filesystems.aux
7614754 5-Filesystems.log
7614764 5-Filesystems.nav
7614761 5-Filesystems.out
7614759 5-Filesystems.pdf
7614766 5-Filesystems.snm
7614750 5-Filesystems.tex
7614765 5-Filesystems.toc
7614762 5-Filesystems.vrb
7614772 beamerthemeDreuw.sty
7615207 drink1.jpeg
7615206 drink2.jpeg
7615209 drink3.jpeg
7615244 drink4.jpeg
7614776 ref.bib
7615239 sbc.bst
```

debugfs

- Utilitário de **depuração** acessa **infos de inode** e **blocos de dados**.

```
$ sudo debugfs /dev/sda2
debugfs: stat ~/me.jpeg
Inode: 3408881   Type: regular    Mode: 0664   Flags: 0x80000
User:  1000      Group:  1000     Size: 4762
Links: 1         Blockcount: 16 (16 x 512B = 8KB)
 ctime: 0x66db8c06:689ea9ac -- Fri Sep  6 20:11:02 2024
 atime: 0x671ecfb5:3a2dee88 -- Sun Oct 27 20:41:41 2024
 mtime: 0x613f56ee:0b1c6f4c -- Mon Sep 13 10:49:34 2021
crtime: 0x613f56ee:09340874 -- Mon Sep 13 10:49:34 2021
Inode checksum: 0x011ca9e2
EXTENTS:(0-1):13664276-13664277 [logical blocks:physical ones]
debugfs: imap ~/me.jpeg
Inode 3408881 block group 416 located at block 13631583
debugfs: bmap ~/me.jpeg 0
13664276
debugfs: bmap ~/me.jpeg 1
13664277
debugfs: bmap ~/me.jpeg 2
0
```

MacTIME

- O gerenciamento é feito por **inodes** através do **sistema Mactime**.

MacTIME

`M` : **`Modify`** ou **`mtime`**: **Modificação do interior do arquivo** (criação).

`A` : **`read Access`** ou **`atime`**: Refere-se ao **acesso ao conteúdo**.

`C` : **`status Change`** ou **`ctime`**: **Modificação de atributos/metadados**.

MacTIME

- Com **`$ls -l`**, vemos a **data/hora de criação** ou **última modificação**:

~/Documents/Aulas/SysOpe/5-Filesystems$ ls -l

```
total 10152
-rw-rw-r-- 1 jr jr     3263 sept. 10 16:57 5-Filesystems.aux
-rw-rw-r-- 1 jr jr   114191 sept. 10 16:57 5-Filesystems.log
-rw-rw-r-- 1 jr jr     1040 sept. 10 16:57 5-Filesystems.nav
-rw-rw-r-- 1 jr jr        0 sept. 10 16:57 5-Filesystems.out
-rw-rw-r-- 1 jr jr  4707859 sept. 10 16:57 5-Filesystems.pdf
-rw-rw-r-- 1 jr jr        0 sept. 10 16:57 5-Filesystems.snm
-rw-rw-r-- 1 jr jr    30842 sept. 10 16:56 5-Filesystems.tex
-rw-rw-r-- 1 jr jr        0 sept. 10 16:57 5-Filesystems.toc
-rw-rw-r-- 1 jr jr     3918 sept. 10 16:57 5-Filesystems.vrb
-rw------- 1 jr jr     5985 sept.  4 18:24 beamerthemeDreuw.sty
-rw------- 1 jr jr      543 janv. 28  2007 beamerthemeWarsaw.sty
-rw-rw-r-- 1 jr jr   303432 sept.  9 20:11 drink1.jpeg
-rw-rw-r-- 1 jr jr   320434 sept.  9 19:46 drink2.jpeg
-rw-rw-r-- 1 jr jr   332097 sept.  9 19:52 drink3.jpeg
-rw-rw-r-- 1 jr jr   392911 sept.  9 20:00 drink4.jpeg
-rw-rw-r-- 1 jr jr     1329 mai   12 19:30 ref.bib
-rw-rw-r-- 1 jr jr    21864 mai    1  2016 sbc.bst
-rw------- 1 jr jr     2141 août  18  2007 tangocolors.sty
```

MacTIME

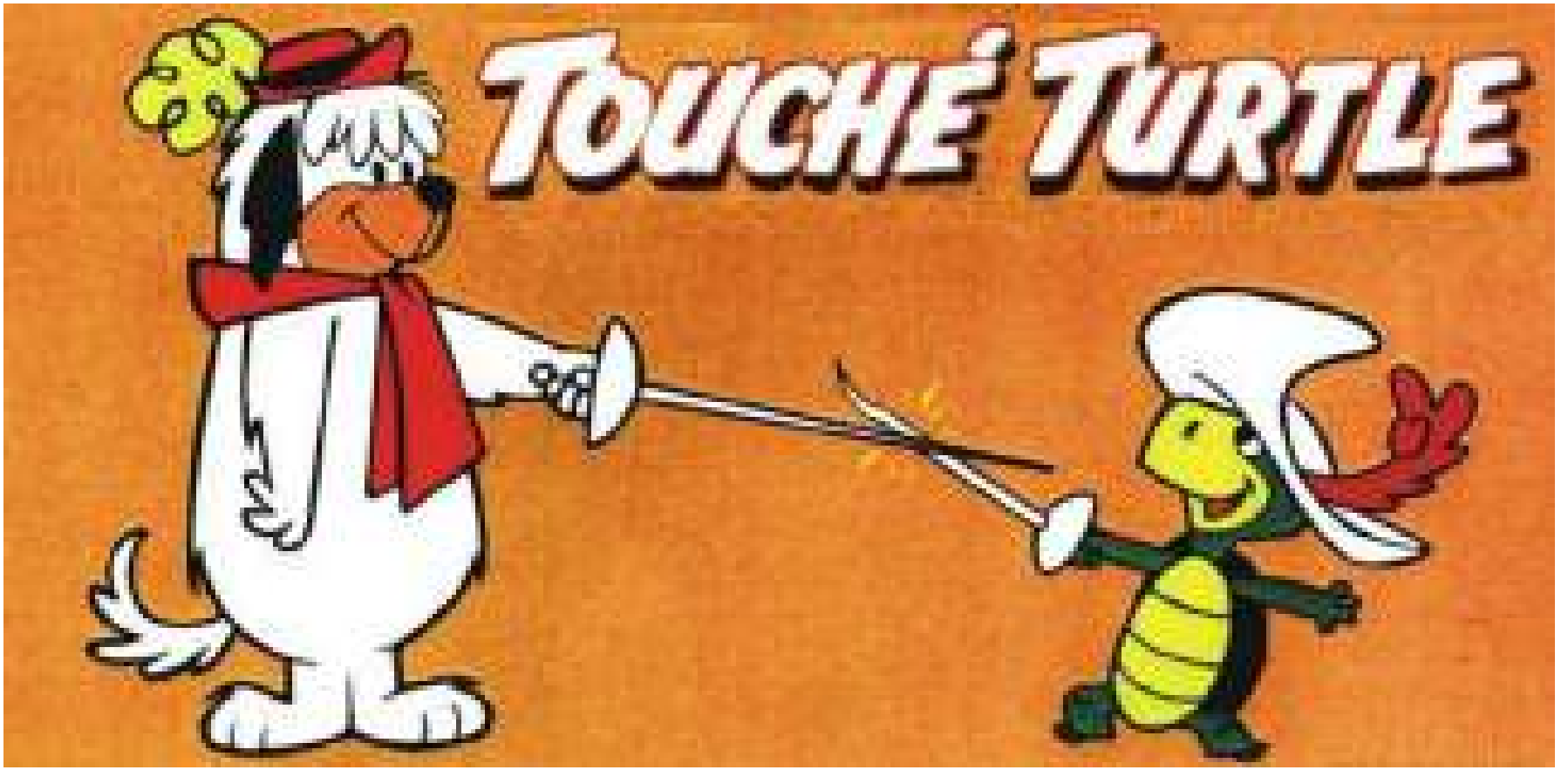
- Para vermos a **data e a hora do último acesso**, utiliza-se **`$ls -lu`**:

~/Documents/Aulas/SysOpe/5-Filesystems$ ls -lu

```
total 10152
-rw-rw-r-- 1 jr jr    3263 sept. 10 16:57 5-Filesystems.aux
-rw-rw-r-- 1 jr jr  114191 sept. 10 02:42 5-Filesystems.log
-rw-rw-r-- 1 jr jr    1040 sept. 10 16:57 5-Filesystems.nav
-rw-rw-r-- 1 jr jr       0 sept. 10 16:57 5-Filesystems.out
-rw-rw-r-- 1 jr jr 4707859 sept. 10 16:57 5-Filesystems.pdf
-rw-rw-r-- 1 jr jr       0 sept. 10 02:42 5-Filesystems.snm
-rw-rw-r-- 1 jr jr   32081 sept. 10 16:56 5-Filesystems.tex
-rw-rw-r-- 1 jr jr       0 sept. 10 02:42 5-Filesystems.toc
-rw-rw-r-- 1 jr jr    3918 sept. 10 16:57 5-Filesystems.vrb
-rw------- 1 jr jr    5985 sept. 10 16:44 beamerthemeDreuw.sty
-rw------- 1 jr jr     543 sept.  9 16:41 beamerthemeWarsaw.sty
-rw-rw-r-- 1 jr jr  303432 sept.  9 20:11 drink1.jpeg
-rw-rw-r-- 1 jr jr  320434 sept.  9 19:46 drink2.jpeg
-rw-rw-r-- 1 jr jr  332097 sept.  9 19:59 drink3.jpeg
-rw-rw-r-- 1 jr jr  392911 sept.  9 20:03 drink4.jpeg
-rw-rw-r-- 1 jr jr    1329 sept.  9 16:41 ref.bib
-rw-rw-r-- 1 jr jr   21864 sept.  9 16:41 sbc.bst
-rw------- 1 jr jr    2141 sept. 10 16:44 tangocolors.sty
```

touch

- **Atualiza o MACtime** de um **arquivo ou diretório**.
- Caso o **arquivo/diretório não existam**, um **arquivo vazio** é **criado**.



touch

Chave	Função
`-a`	Altera a data e hora de acesso para a atual.
`-m`	Altera a data e hora de última modificação para a atual.
`-t`	Altera para data e hora personalizados (formato padrão).
`-d`	Altera para data e hora personalizados (formato flexível).

- **`-t`**: o **padrão de data e hora** é `[[CC]YY]MMDDhhmm[.ss]`.
- **`-d`**: yesterday, tomorrow, x hours/days/weeks ago, ±x, next Friday, last...

```
# touch file
# ls -l file
# touch -m file
# ls -l file
# touch -a -d "2 weeks ago" file
# ls -lu file
```

* Ao alterar o M ou o A o C é automaticamente configurado.

Blocos

- Filesystems são **compostos por blocos** que são como caixas ou copos!

Blocos

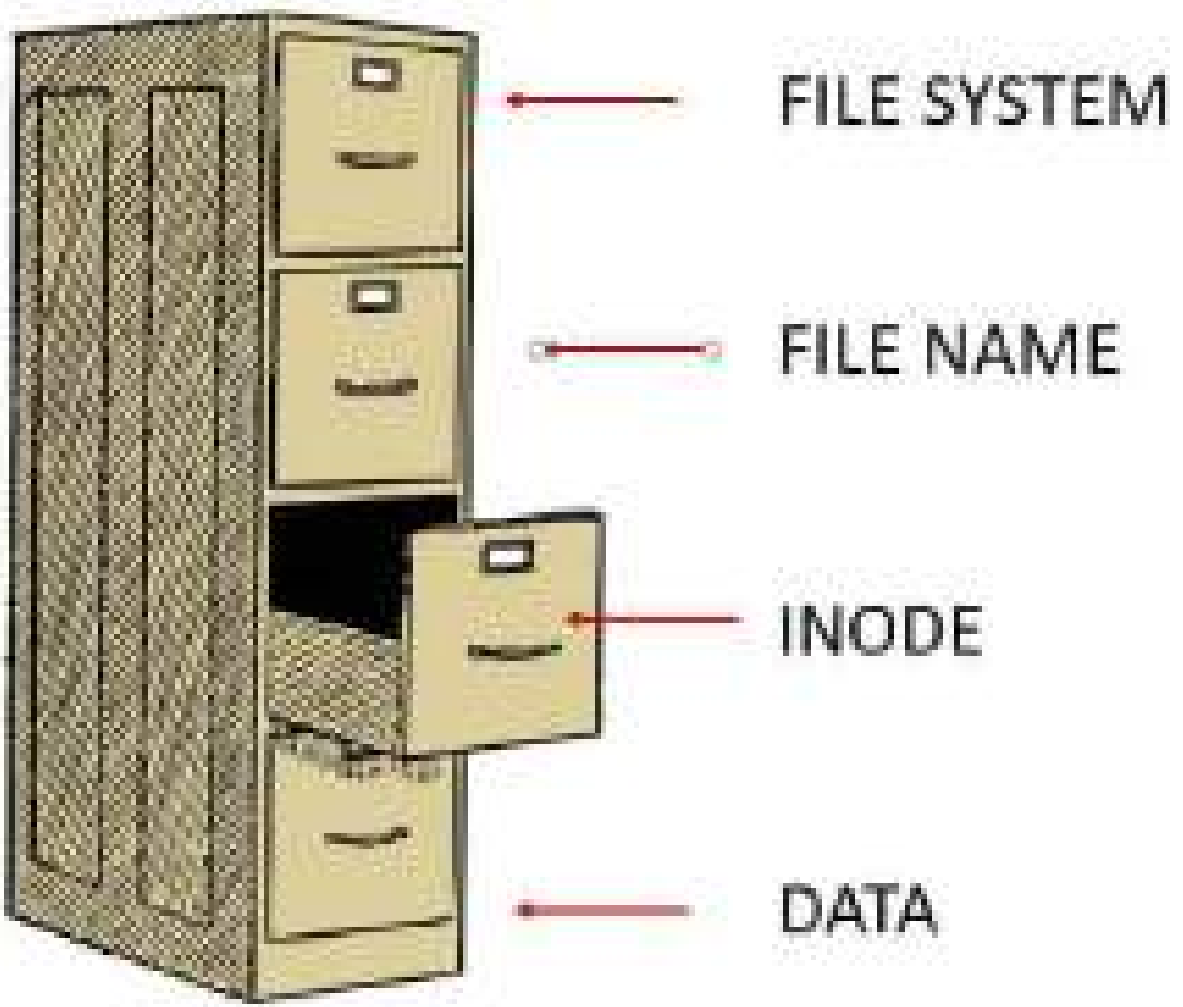
- **Cada bloco tem 4096 bytes** (4KB) de capacidade por padrão.

inodes ou i-nodes (index nodes ou nós índices)

- Cada **arquivo ou diretório** será **controlado por um único inode**.

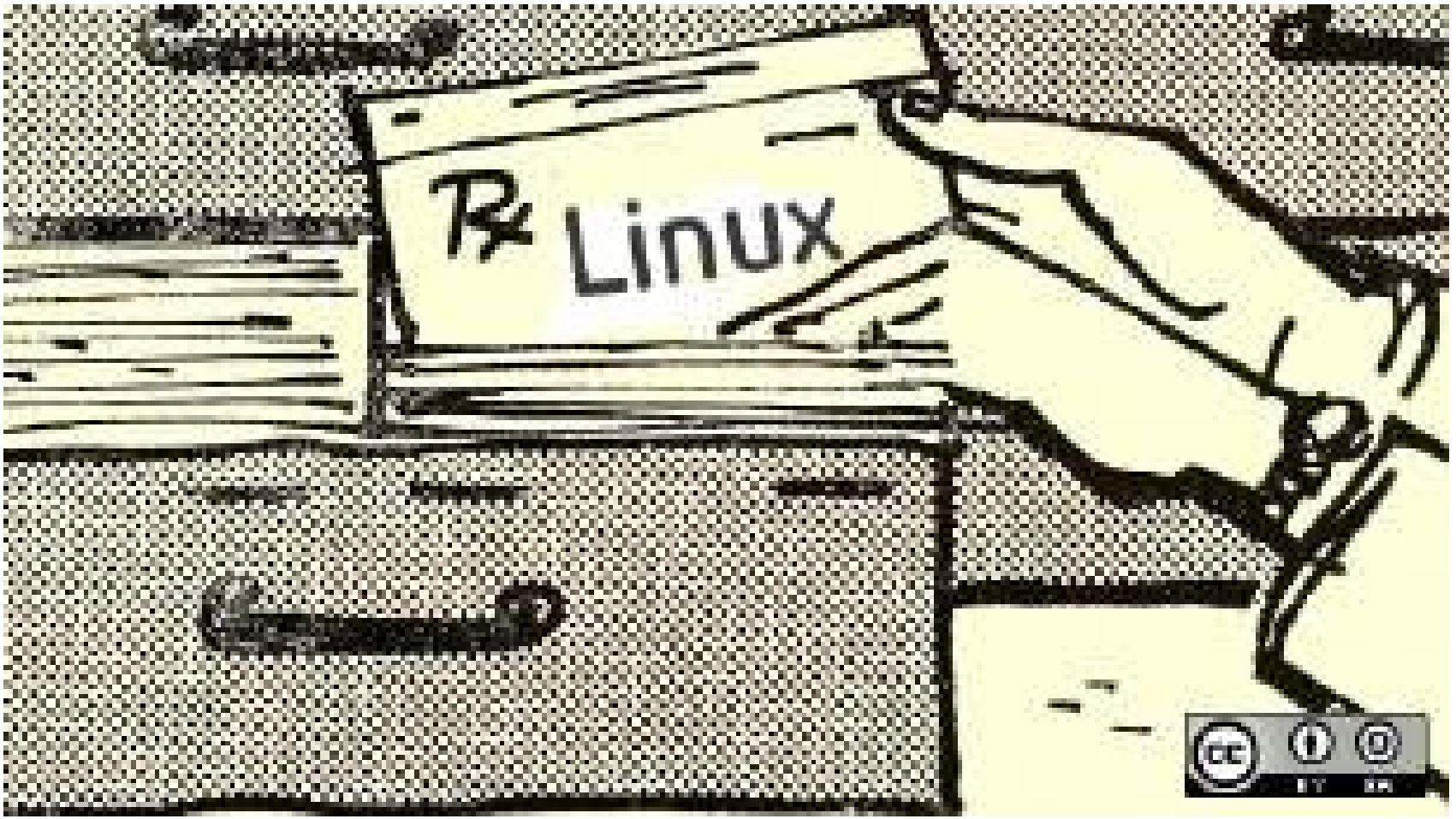
inodes ou i-nodes (index nodes ou nós índices)

- Os **inodes armazenam** todos os **metadados do file exceto nomes**.



inodes ou i-nodes (index nodes ou nós índices)

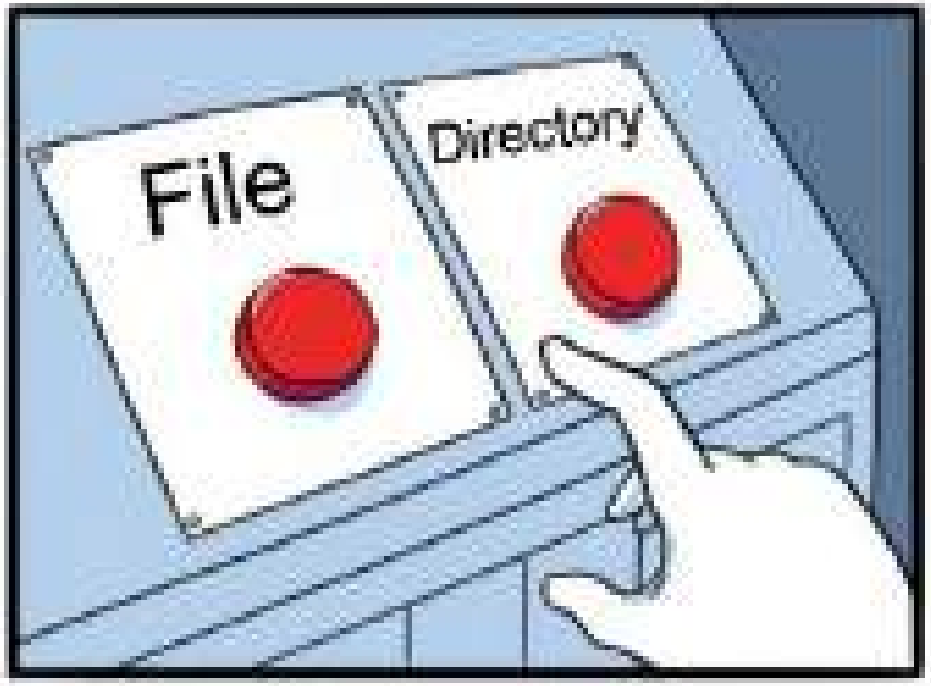
- O **nome dos arquivos** e **diretórios** sao **armazenados em diretório**.



Diretórios

- Cada **diretório** será controlado por um inode...

```
$ ls -i1 / | sort -n
  1 dev
  1 proc
 13 bin
 18 sbin
```



Diretórios

- **Diretórios** são **abstrações** em um filesystem: **Eles são arquivos!!!**

Diretórios

- Um **diretório é um arquivo agrupador** (lista) **dentro de um FS**.

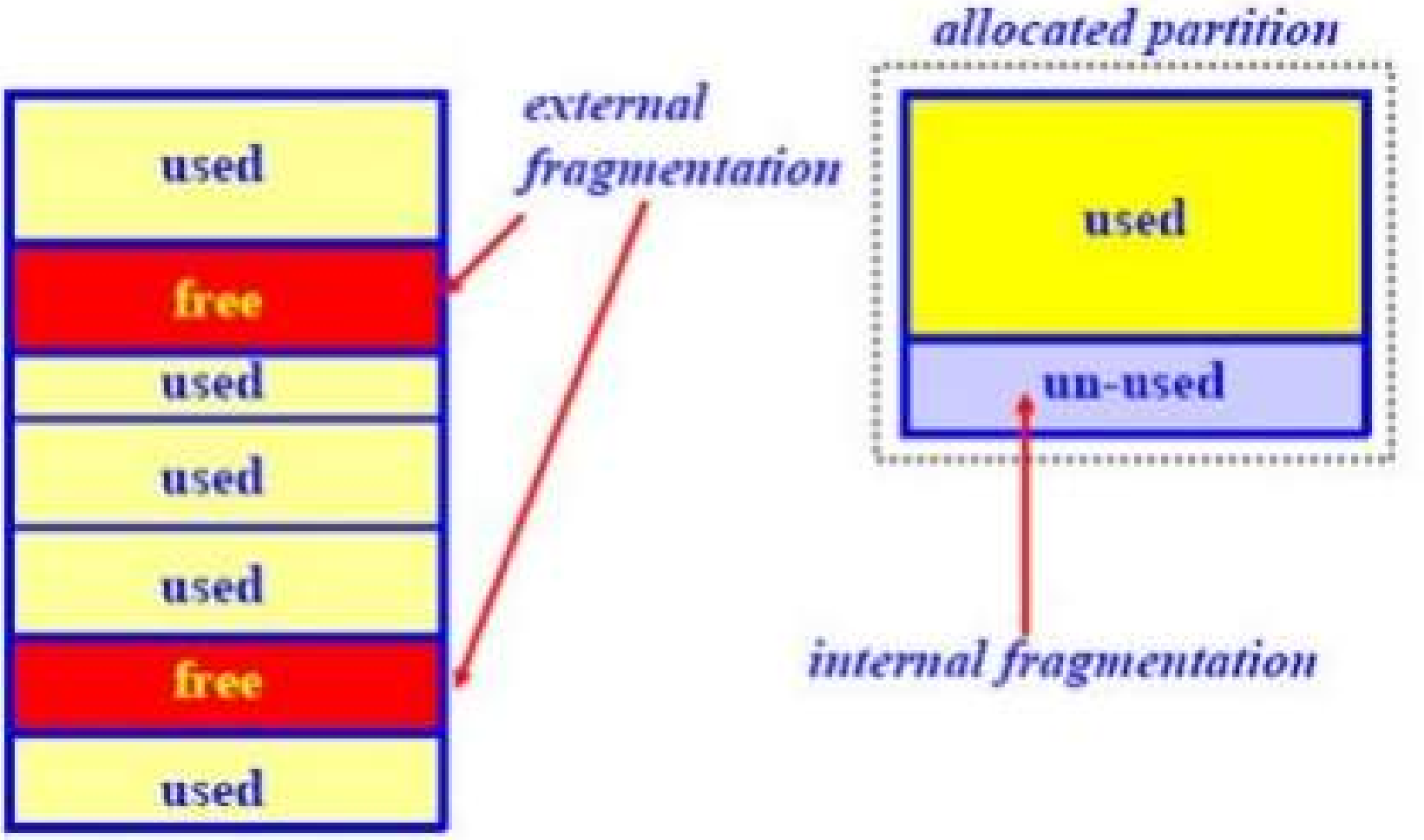
https://www.youtube.com/watch?v=oHrlU3b1ZAw

Fragmentação de arquivos ou de disco/espaço

- **Fragmentar arquivo**: quebra na sequência dos dados.
- **Fragmentar disco**: quebra na sequência de espaços livres contíguos.

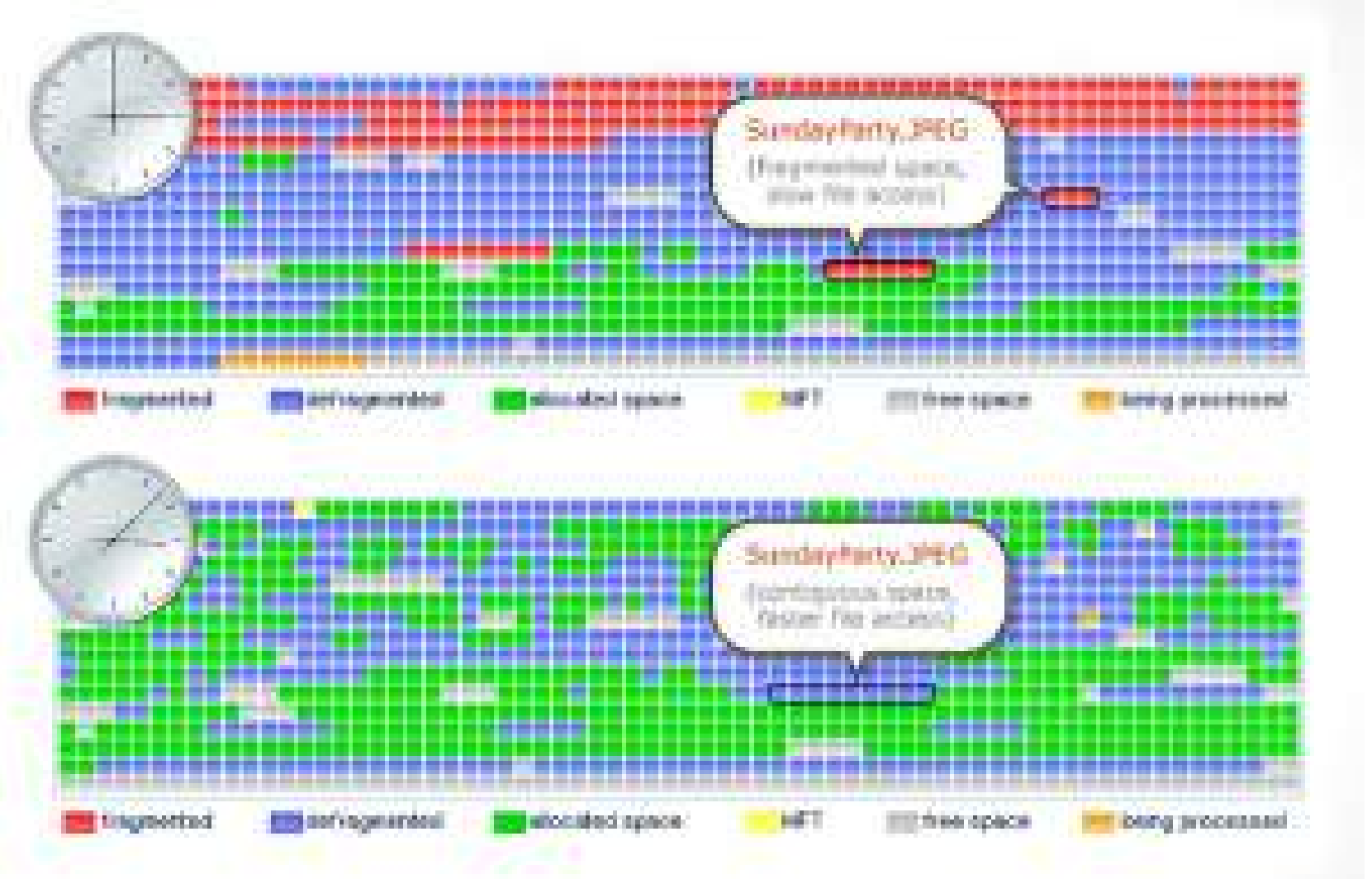
Fragmentação de arquivos ou de disco/espaço

- **Fragmentação interna**: **Espaços desperdiçados** dentro de blocos (slack).
- **Fragmentação externa**/de disco: Espaços livres em blocos não contíguos.



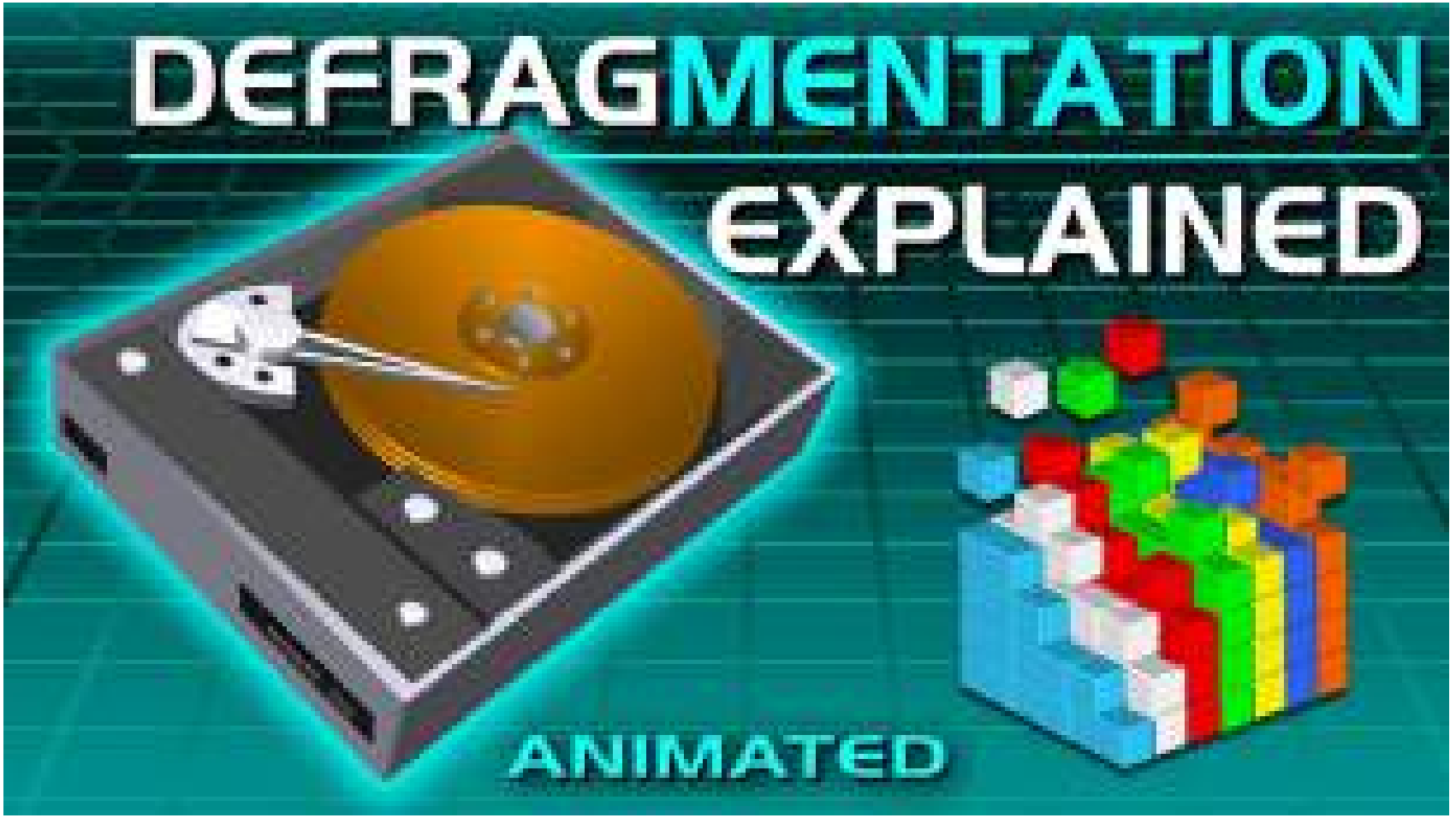
Fragmentação de arquivos ou de disco/espaço

- **Fragmentação** de HDD **muda a velocidade de acesso** aos arquivos.



Fragmentação de arquivos ou de disco/espaço

- É por isso que existe o **processo de desfragmentação** em HDD.



Fragmentação de arquivos ou de disco/espaço

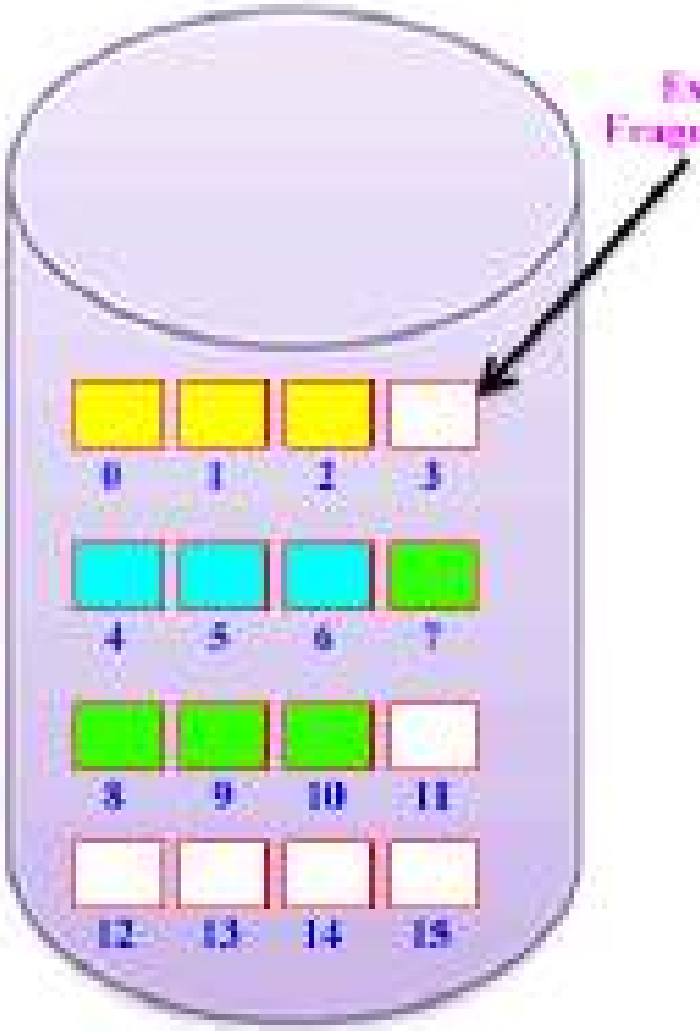
- A **fragmentação** da **memória flash** não fará a menor diferença.
- Um **disco de memória** não usa **cabeça de leitura** que se move.

Tipos de alocações

- Existem **varios tipos de alocações** suportando o **acesso aleatório**.

Alocação contígua

- Localizar arquivo com **endereço do 1^o bloco** e com seu tamanho.

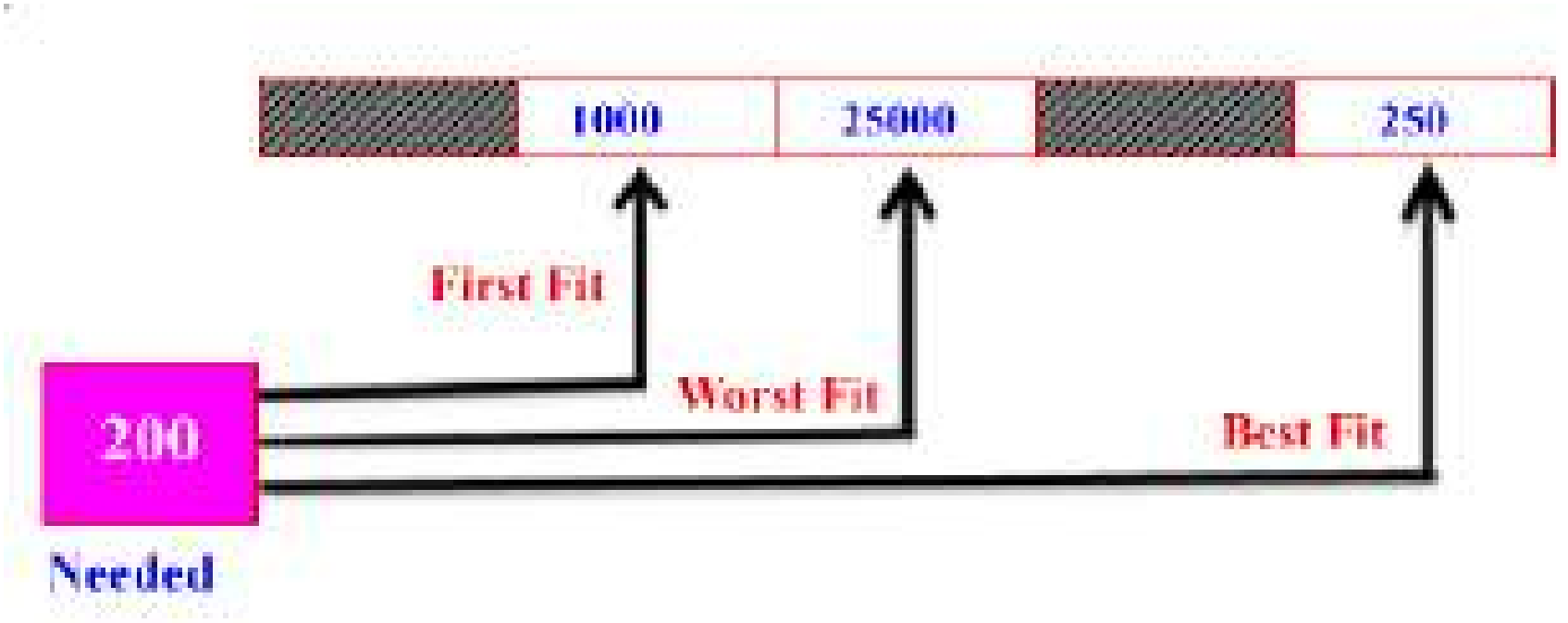


Directory

File	Start	Length
A	0	3
B	4	3
C	7	4

Alocação contígua

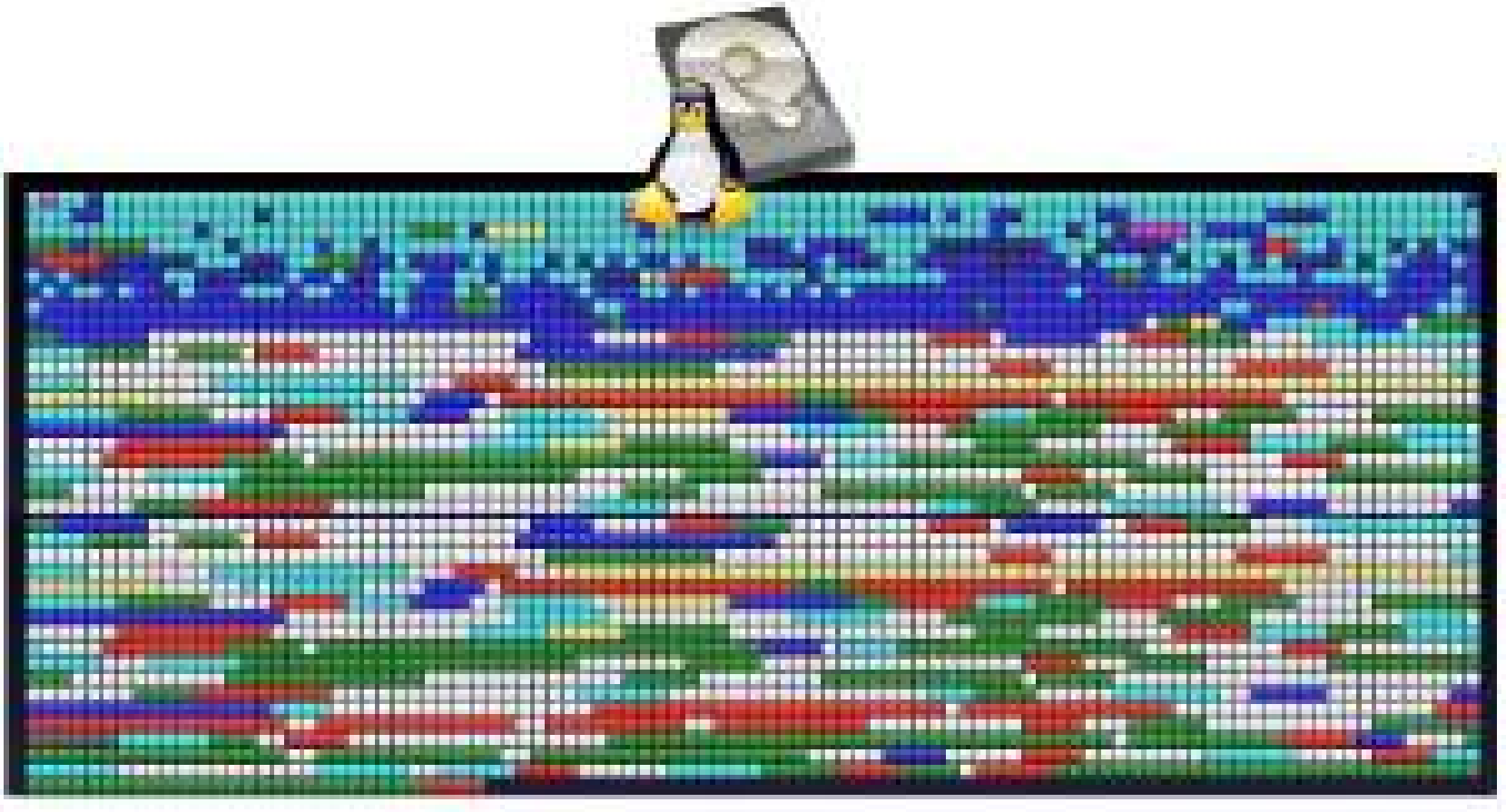
- **First-fit**: Busca do 1^o segmento livre com o tamanho suficiente
- **Best-fit**: Seleciona o menor segmento livre com o tamanho suficiente.
- **Worst-fit**: Seleciona o maior segmento livre.



* Esses algos também podem ser usados em alocações não contíguas.

Alocação contígua

- **Problema** na alocação contígua é a **fragmentação de espaço**!

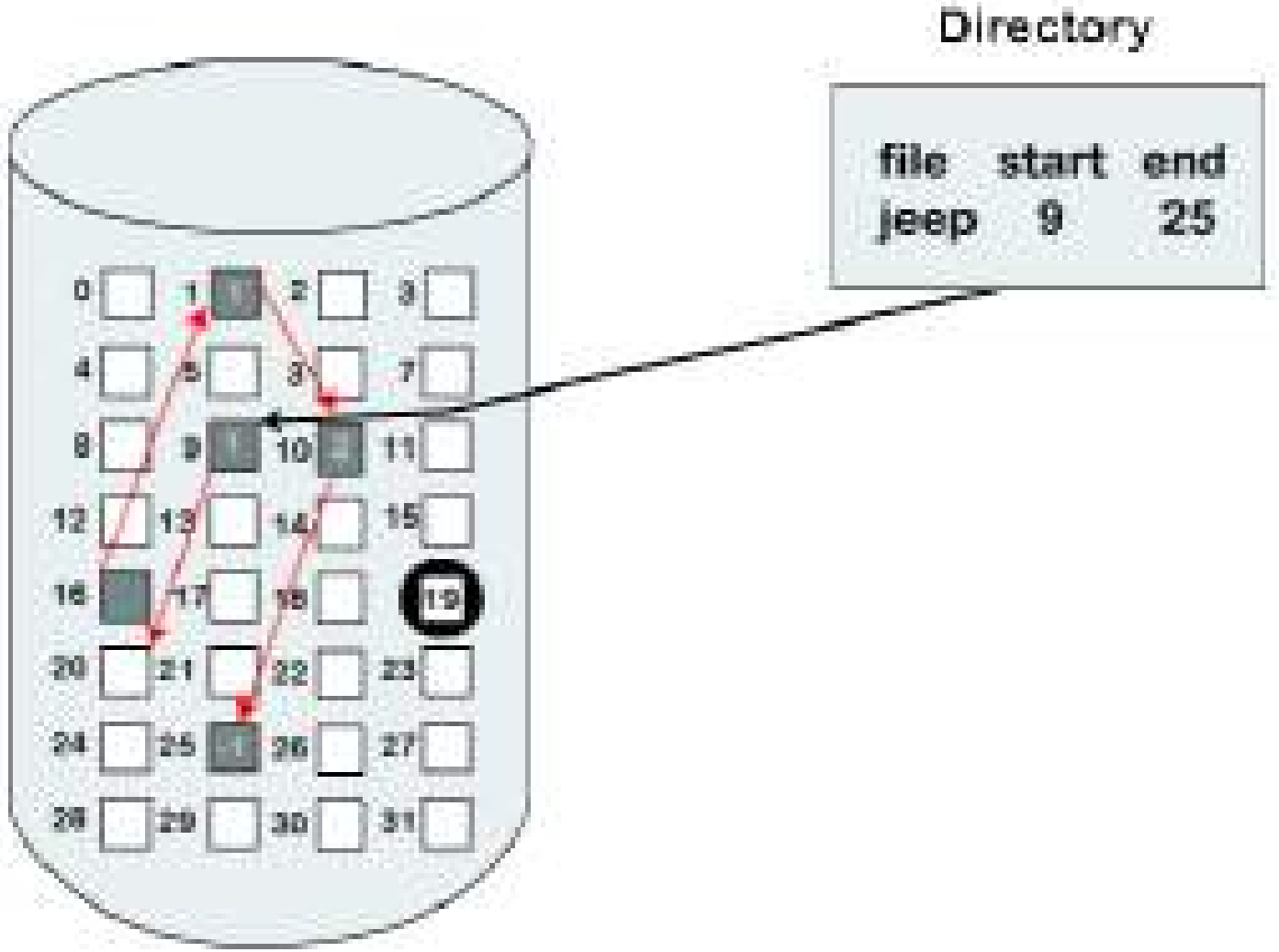


Alocação contígua

- Os **CD-ROMs** utilizam um FS a alocação contígua: a **ISO 9660**.

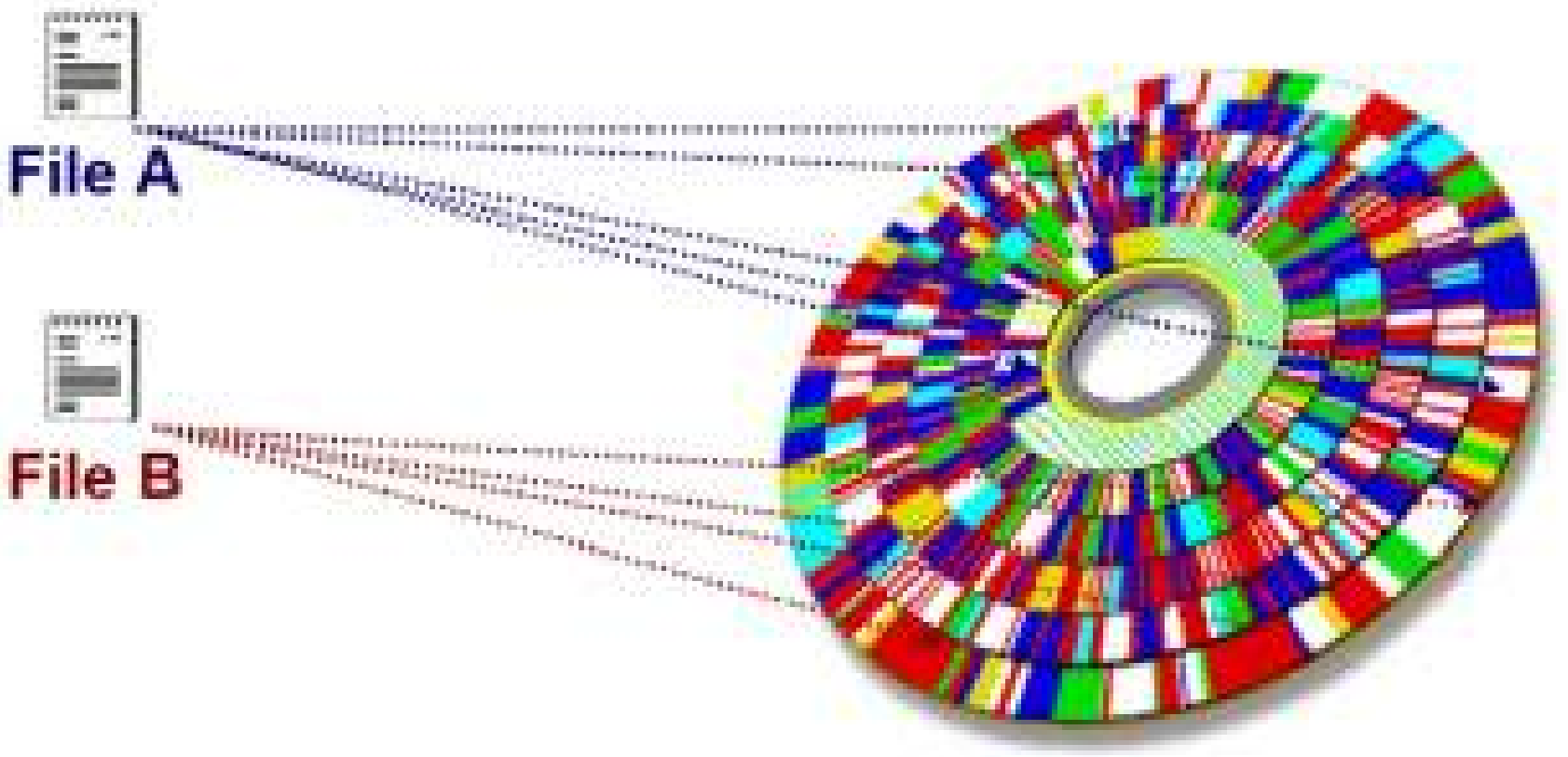
Alocação Encadeada

- Cada bloco possui um **ponteiro** para o **bloco seguinte** do arquivo.



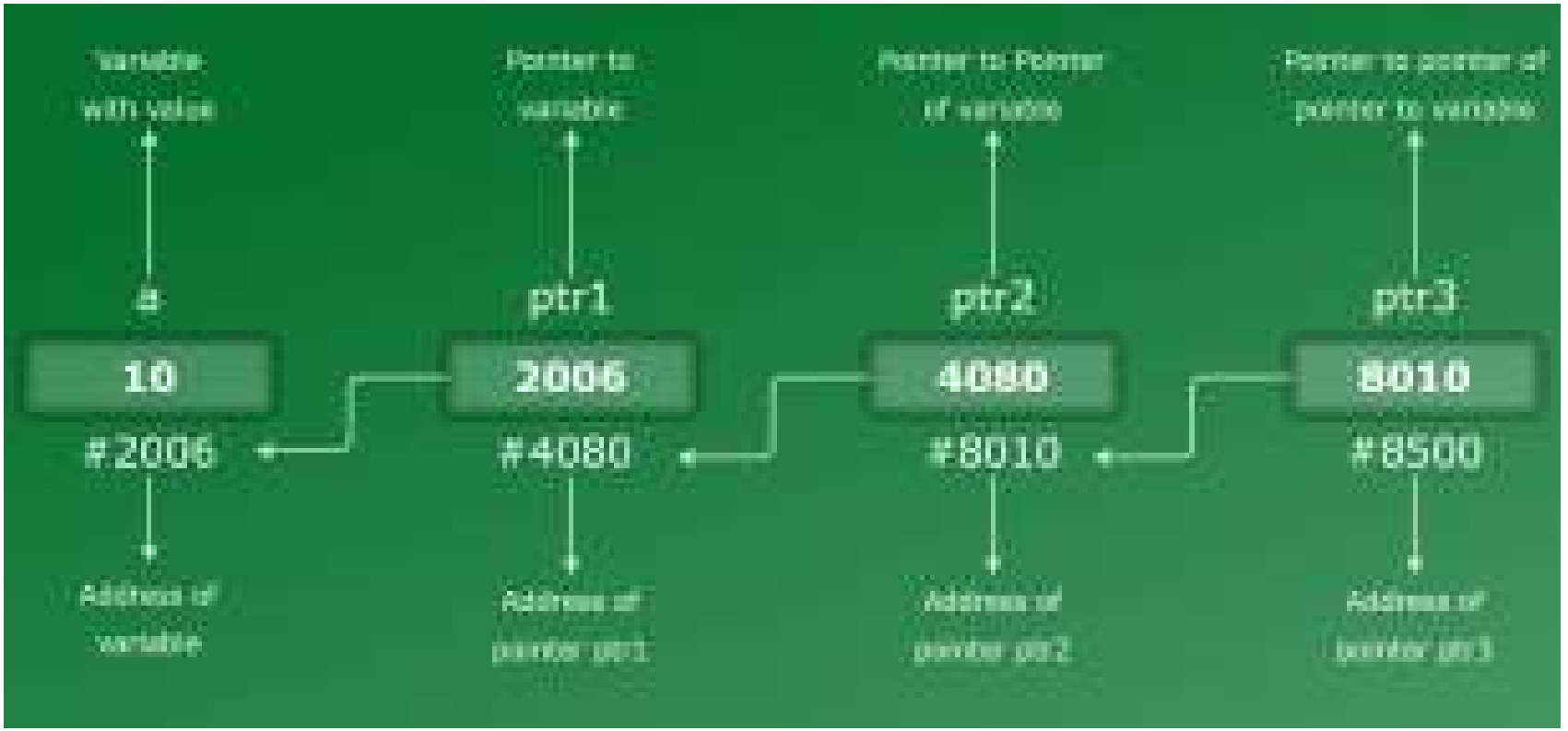
Alocação Encadeada

- **Fragmentação do arquivo aumenta tempo de acesso** em HDD.



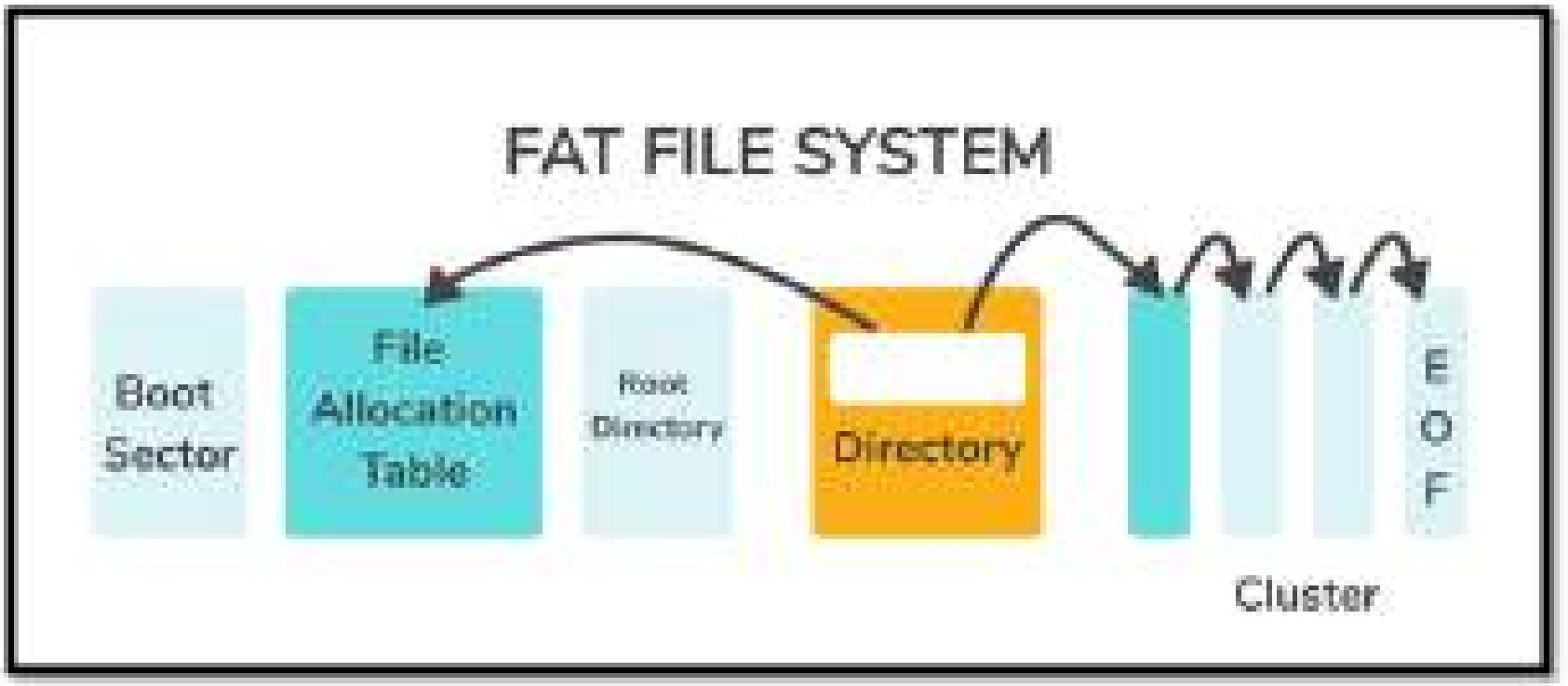
Alocação Encadeada

- Sem **acesso seqüencial** aos blocos, gastamos **espaços de ponteiros**.



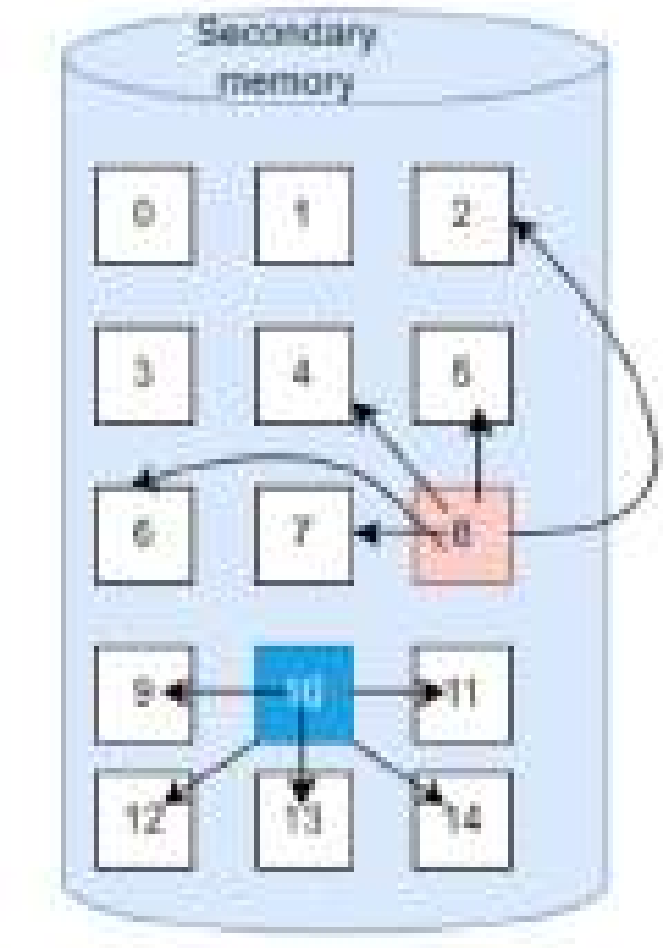
Alocação Encadeada

- FAT usa **lista encadeada separada** diminuindo a proba de erros.



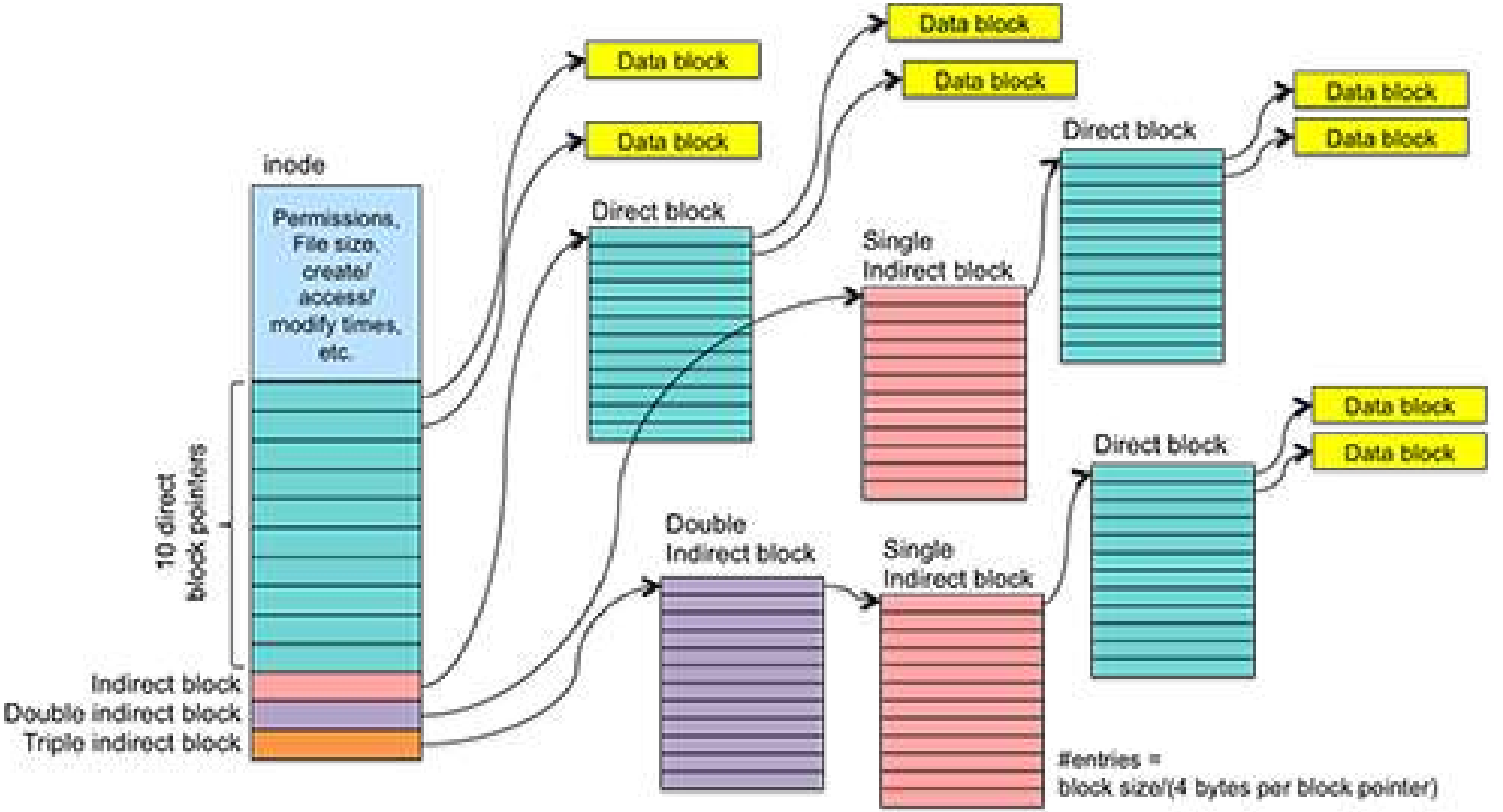
Alocação Indexada

- **Ponteiros (max 12) de todos os blocos** do (small) file no **inode**.



File name	Index block
os.pdf	8
dbms.doc	10

Alocação Indexada



Aumentar a velocidade de acesso

Escondendo a latencia ou usando índices

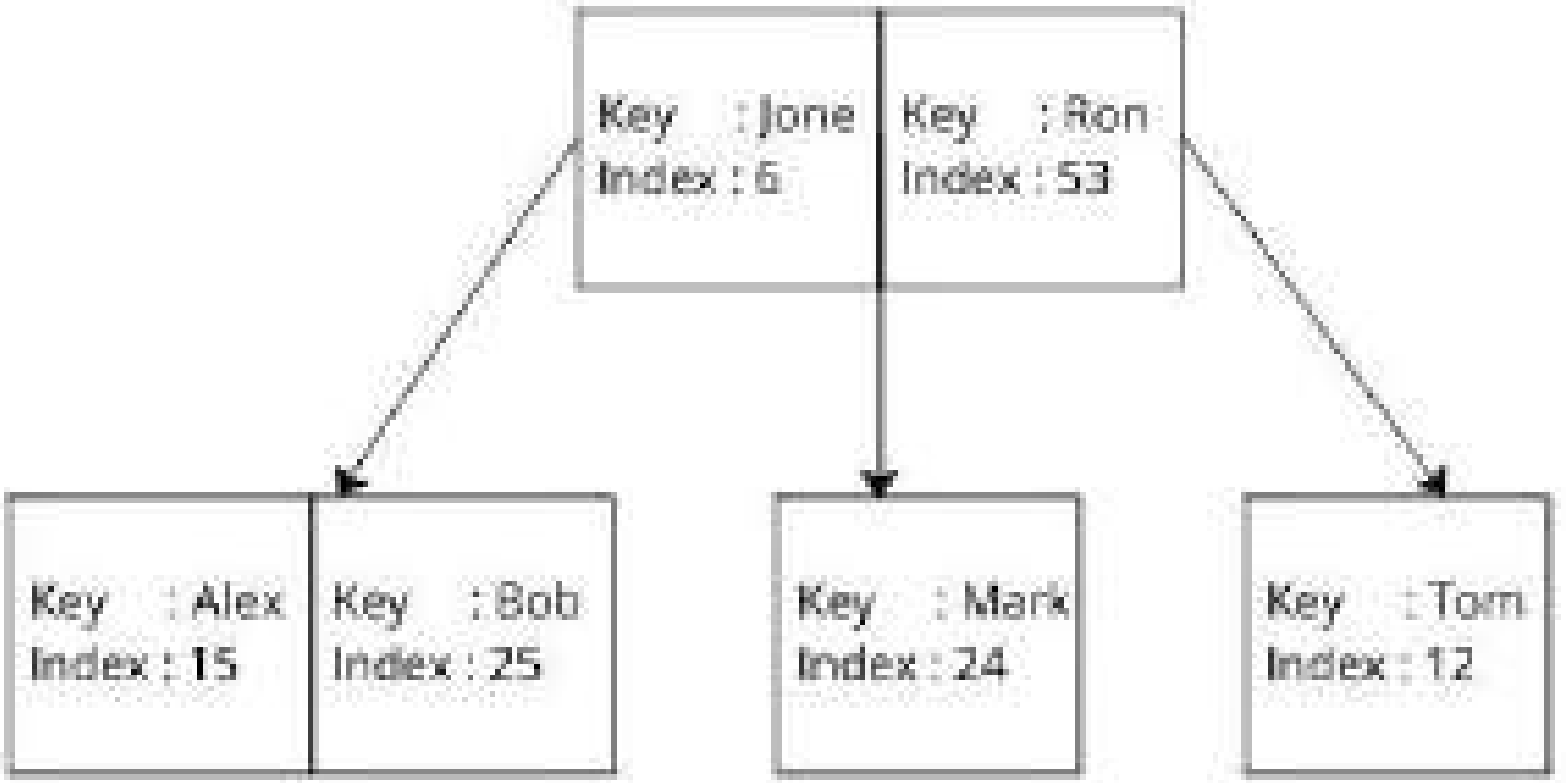
- Blocos de índice podem ser **mantidos em cache** na memória principal (RAM).

2. Alocações encadeadas e indexadas

Aumentar a velocidade de acesso

Escondendo a latencia ou usando índices

- **NTFS** e **EXT4** usam **B+Tree** para **indexar files** em diretórios grandes.



FAT

- **FAT**: **F**ile **A**llocation **T**able passou por várias encarnações **desde 1977**.

FAT

- Um **Boot sector**, uma **FAT**, um **Root Directory** e uma **área de dados**.
- **Root directory** está no **cluster 0**, ponto de partida para explorar.

FAT

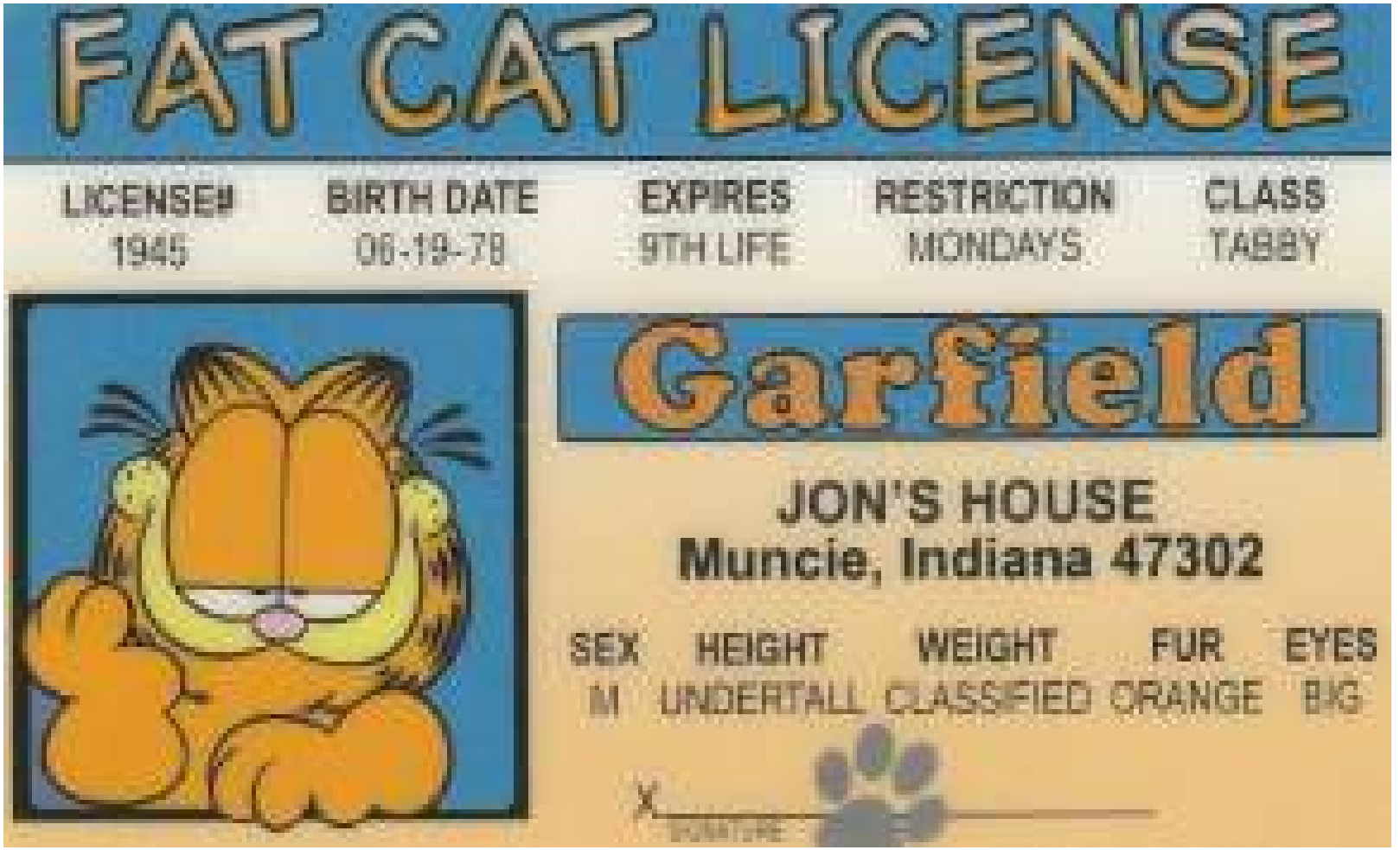
- **Cada diretório** é representado por uma **tabela de entradas** (32B cada).
- Entrada: **nome**, **tamanho**, **MACtime** e **endereço do primeiro cluster**.

Root Directory SFN Entry Data Structure	
Bytes	Purpose
0	First character of file name (ASCII) or allocation status (0x00=unallocated, 0xE5=deleted)
1-10	Characters 2-11 of the file name (ASCII); the "." is implied between bytes 7 and 8
11	File attributes (see File Attributes table)
12	Reserved
13	File creation time (in tenths of seconds)[1]
14-15	Creation time (hours, minutes, seconds)[1]
16-17	Creation date[1]
18-19	Access date[1]
20-21	High-order 2 bytes of address of first cluster (0 for FAT12/16)[2]
22-23	Modified time (hours, minutes, seconds)
24-25	Modified date
26-27	Low-order 2 bytes of address of first cluster
28-31	File size (0 for directories)

fatcat

- Reparar e realizar **análises forenses** em **sistemas de arquivos FAT**.

  $ sudo apt install fatcat



fatcat

- **Cluster 3** contém ubuntu/ e BOOT/, **apontando para clusters** 4 e 1071.

```
$ sudo fatcat -L 3 /dev/sda1
Listing cluster 3
Directory cluster: 3
d 20/7/2021 01:58:14  ./                    c=3
d 20/7/2021 01:58:14  ../                   c=0
d 20/7/2021 01:58:14  ubuntu/               c=4
d 20/7/2021 01:58:14  BOOT/                 c=1071
```

- **Cluster 4** contém os arquivos de ubuntu e os **referentes clusters** deles.

```
$ sudo fatcat -L 4 /dev/sda1
Listing cluster 4
Directory cluster: 4
d 20/7/2021 01:58:14  ./                    c=4
d 20/7/2021 01:58:14  ../                   c=3
f 5/10/2023 14:56:26  grubx64.efi           c=6224 (2.4784M)
f 5/10/2023 14:56:26  shimx64.efi           c=6859 (933.258K)
f 5/10/2023 14:56:26  BOOTX64.CSV           c=7303 (108B)
f 5/10/2023 14:56:26  grub.cfg              c=7304 (117B)
```

fatcat

- **Acessar o cluster 6224** fornecendo **próximo cluster** na cadeia.

```
$ sudo fatcat -@ 6224 /dev/sda1
Cluster 6224 address:
26550272 (000000001952000)
Next cluster:
FAT1: 6225 (00001851)
FAT2: 6225 (00001851)
Chain size: 635 (2600960 / 2.48047M)
Chain is contiguous
```

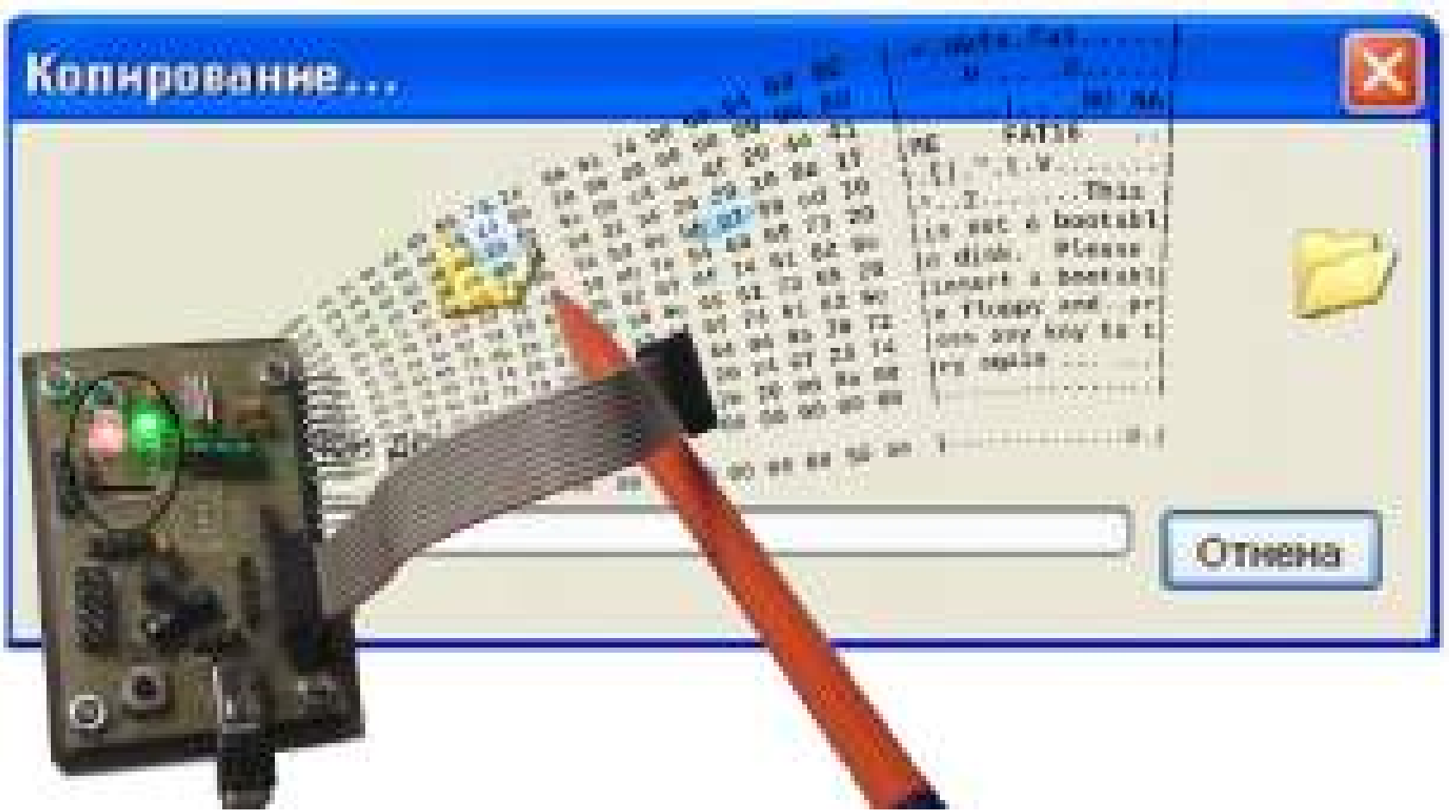
- **Acessar o cluster 6225** fornecendo **próximo cluster** na cadeia.

```
$ sudo fatcat -@ 6225 /dev/sda1
Cluster 6225 address:
26554368 (000000001953000)
Next cluster:
FAT1: 6226 (00001852)
FAT2: 6226 (00001852)
Chain size: 634 (2596864 / 2.47656M)
Chain is contiguous
```

FAT Family

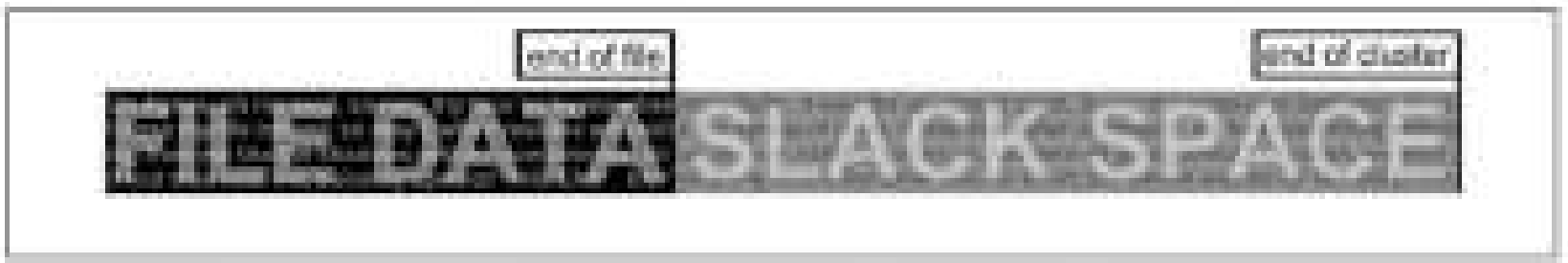
- **Suporte ao FAT** no Linux foi desenvolvido com **engenharia reversa**.
- **Patentes expirando**. **M$** entrou na **O**pen **I**nvention **N**etwork em **2019**.

- **16 bits** para o **endereçamento de dados**: $2^{16} = 65.536$ posições.
- Com cluster de tamanho du um setor (**512 B**): 65.536 x 512 B = **32 MB**.



- Tabela dos **tamanhos da partição** em relação aos **tamanhos do cluster**.

Tamanho de um cluster	Tamanho da partição em FAT 16
1 KB	64 MB
2 KB	128 MB
4 KB	256 MB
8 KB	512 MB
16 KB	1 GB
32 KB	2 GB

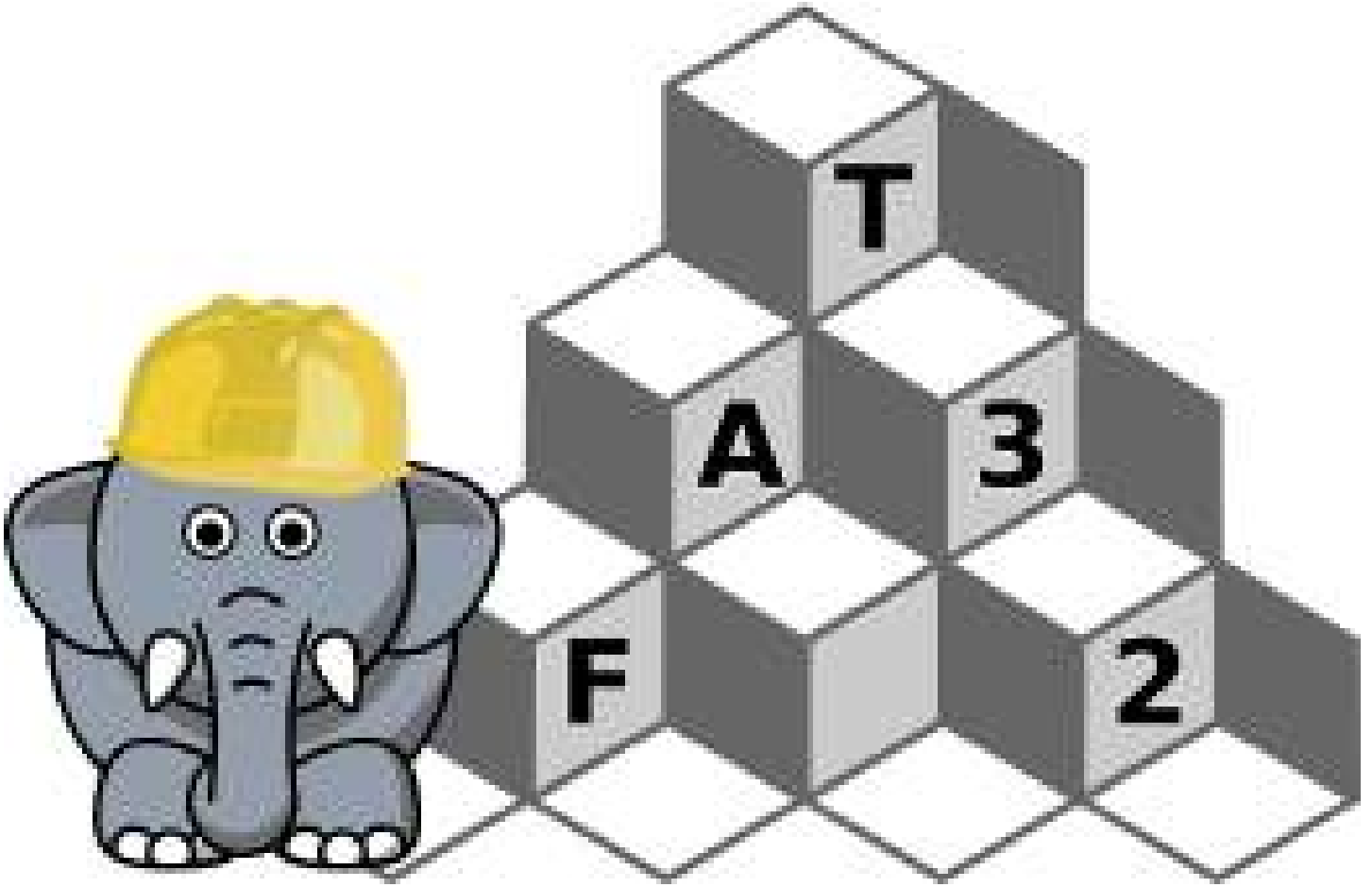


Chapter FS Section FAT Sub FAT16, Tables Generator, table e \listoftables

- O **slack space** pode chegar a **25% da capacidade total** do HD!
- Slack: **largo**, **solto**, **folga**, **margem**, **preguiçoso**, **irresponsável**...

FAT32

- Criado em **1996 pela M$** para **substituir FAT16** usado pelo MS-DOS.
- Endereço formado de **32 bits**, pode ter 2^{32} = 4294967296 endereços.



exFAT (FAT64) 2006 (Open Invention Network 2019)

- **Tamanhos** armazenados com **8B**: **arquivos** teoricamente **até 16 EB**.
- Endereçamento utiliza 64 bits: **partições** limitada por padrão **a 128 PB**.

vFAT

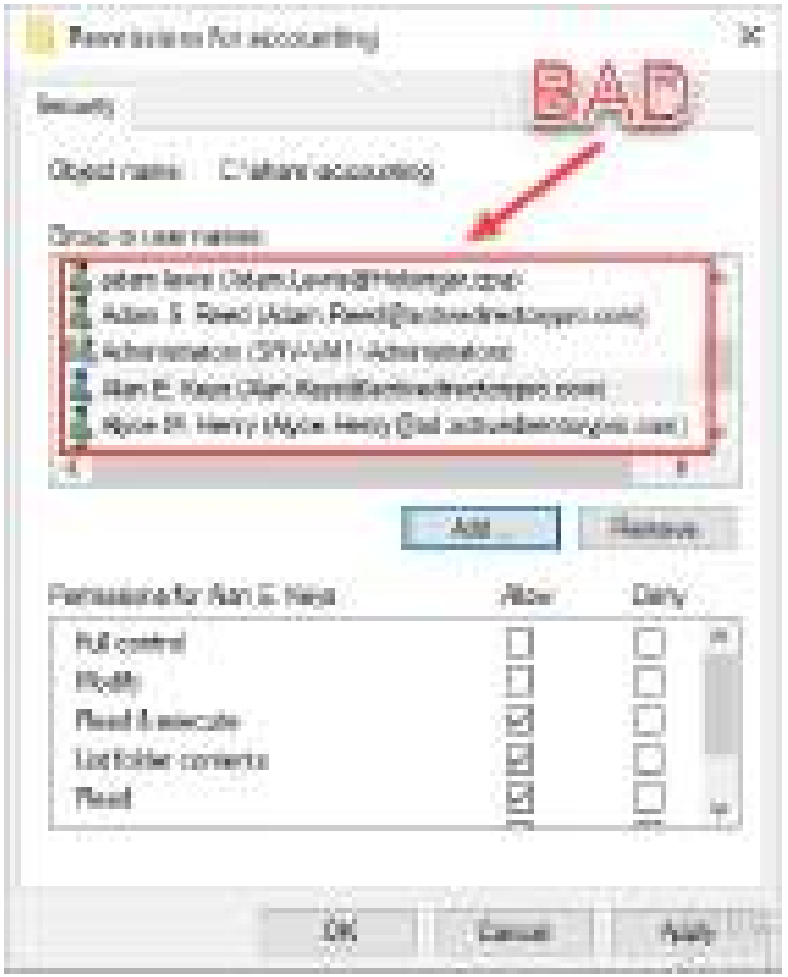
- O **truque do LFN** não pode ser facilmente adaptado ao campo size.
- **Compatibilidade** com **sistemas antigos** e **integração** com **tabela FAT**.

Primeiro desvantagem do FAT

- **Não é possível definir permissões** em arquivos que são partições FAT.
- Os arquivos podem ser **lidos ou escritos** por **qualquer utilizador**.

Primeiro desvantagem do FAT

- O **modelo de permissões POSIX** é amplamente em FS Unix/Linux.
- NTFS declara um **Access Control List** para cada arquivo/diretório.



Segunda desvantagem do FAT

- **A corrupção dos dados** causada por falhas de **energia** ou erros.
- **As alterações** na **área de controle** podem ser **interrompidas**.

Segunda desvantagem do FAT

- **Perda da consistência** entre a área de controle e a área de dados

Sincronização crítica

1. **Excluir os dados do arquivo original** na **indexação** do disco.
2. **Escrever o conteúdo** da **memória no disco**.
3. **Inserir os novos dados** de **nome e localização** na **área de controle**.

Perdas e danos por desligmanentos abruptos

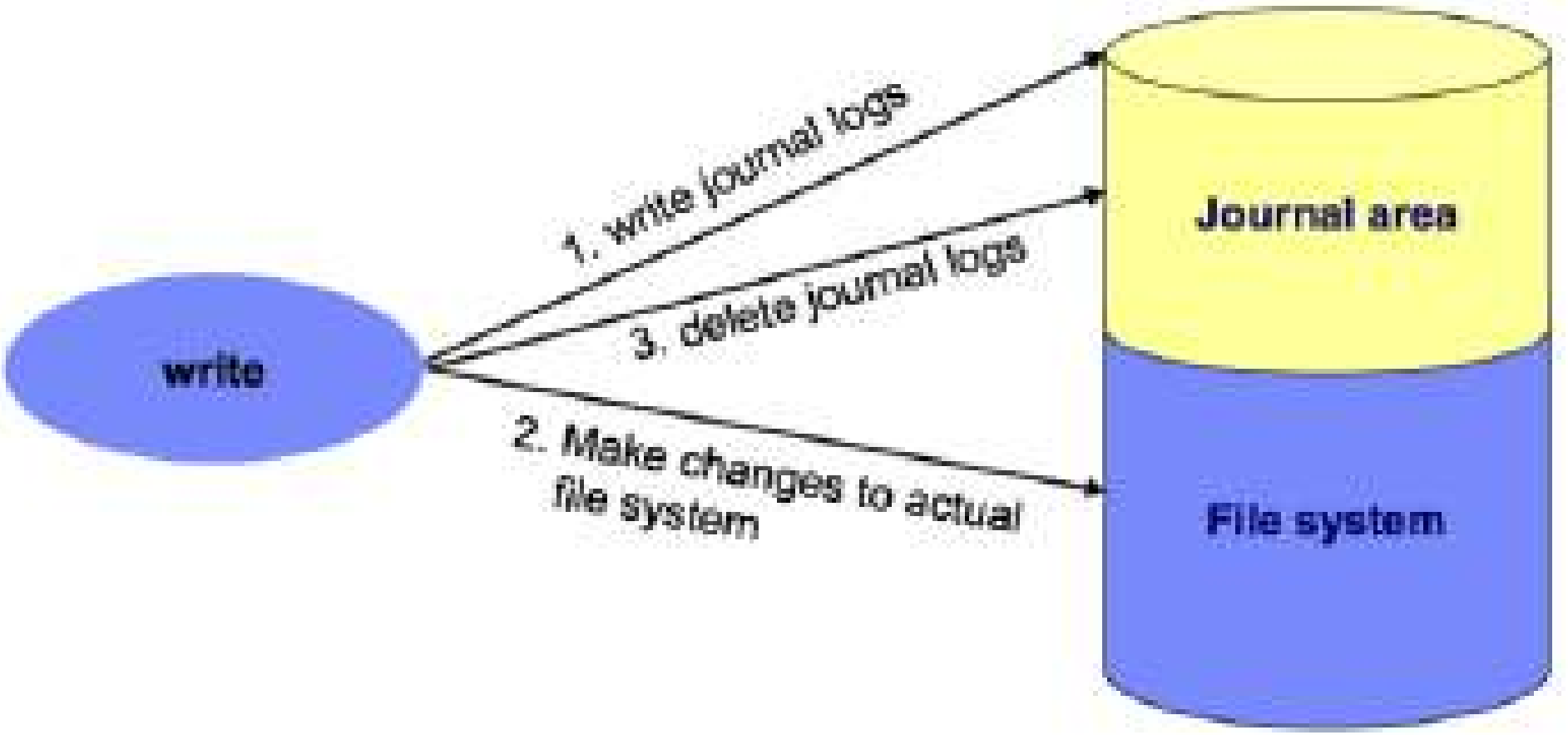
- **Dados poderão existir**, apesar do **inode** não **ter sido criado**!
- **Dados poderão não existir**, apesar do **inode já ter sido criado**!

Filesystems jornalados

- **Log de transações** é um **histórico de ações** executadas por um SGBD.

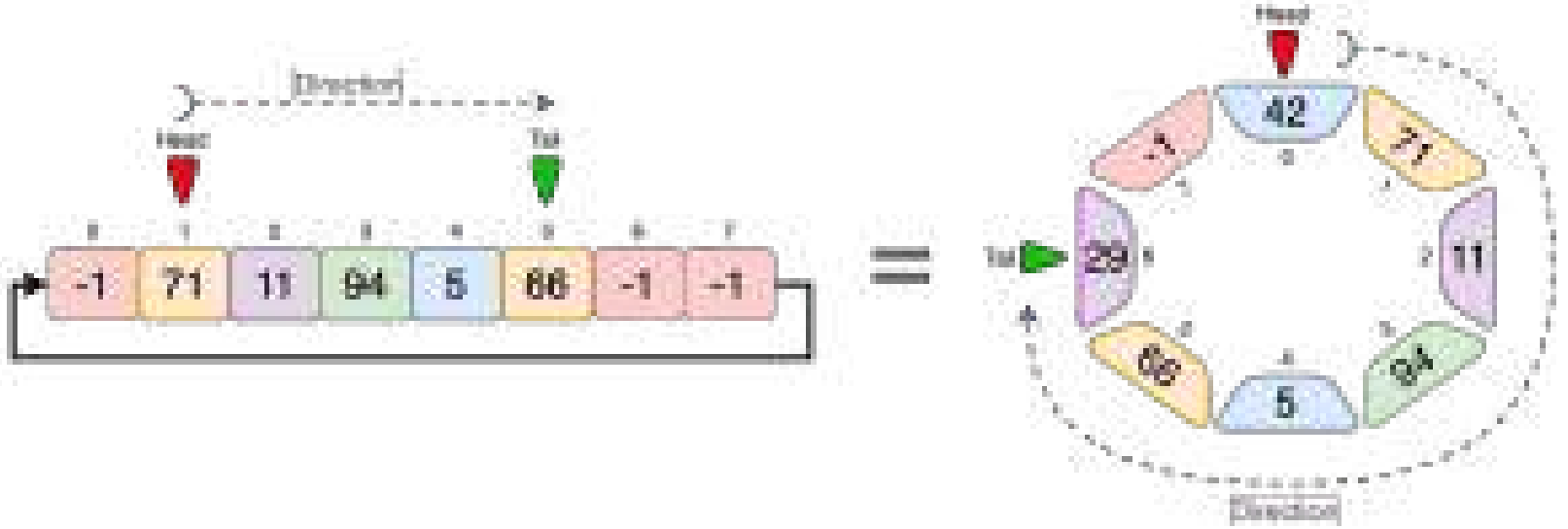
Filesystems jornalados

- Tais FS mantém **log de alterações** feitas durante a gravação no disco.



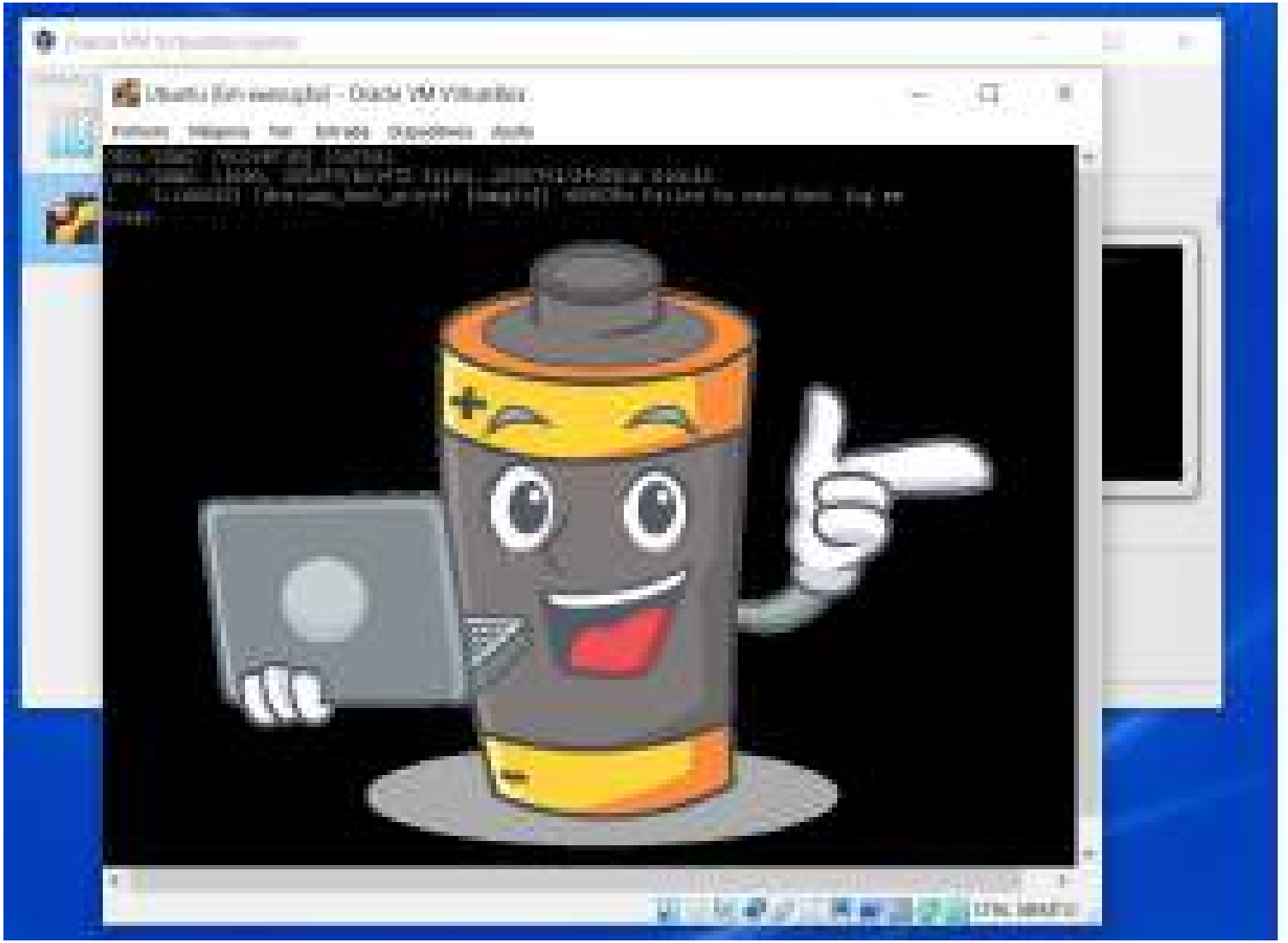
Filesystems jornalados

- Usa um **buffer circular** é uma estrutura de dados de **tamanho fixo**.



Filesystems jornalados

- **Protege seus dados** contra falhas durante o processo de gravação.



Filesystems jornalados

- Vai **reconstruir rapidamente corrupções** depois de falhas do sistema.

Filesystems jornalados

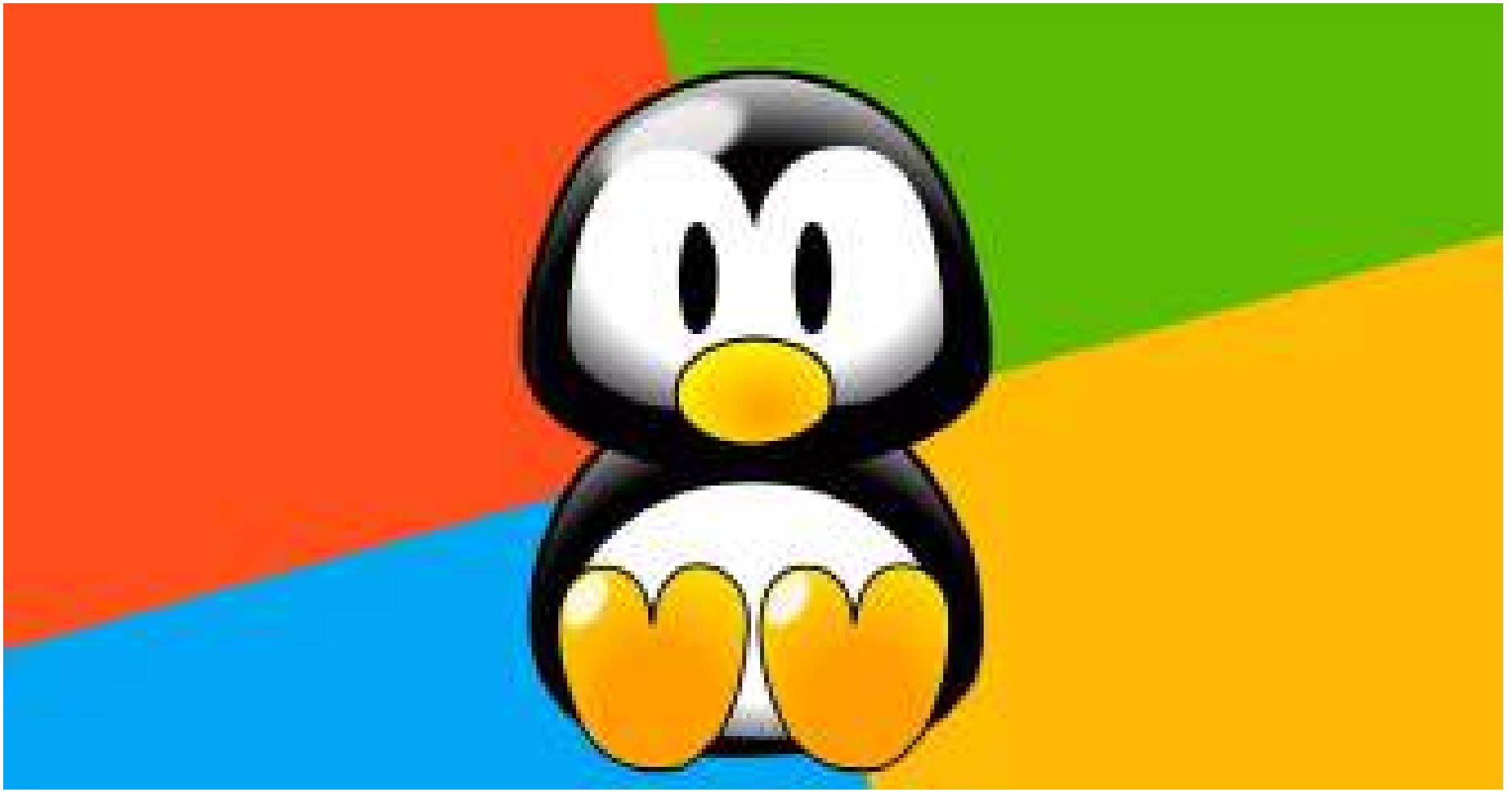
- **Reduz** a alguns segundos o que poderia levar **minutos** com fsck...

NTFS

- **N**ew **T**echnology **F**ile **S**ystem jornalado a partir do Windows NT.
- **Substituiu FAT** no Windows e mas é bem suportado no **Linux e BSD**.

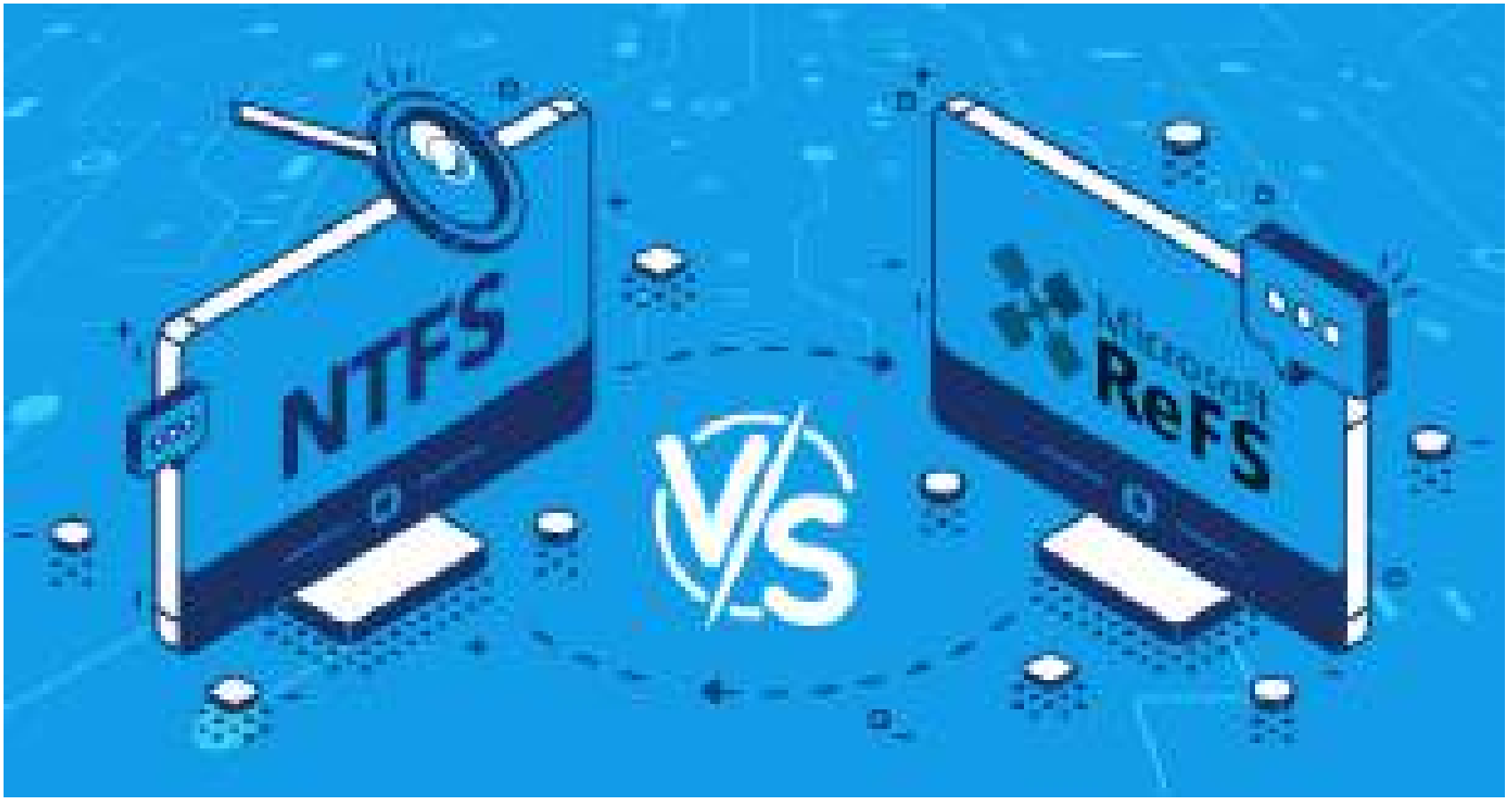
NTFS

- **100% compatível com POSIX** armazenar todas suas **permissões**.



NTFS

- O tamanho de **cluster de 4 KB** é o padrão para a **maioria dos NTFS**.
- **Volumes de até 16 TB** com **cluster de 4K**, de **até 256 TB** com **de 64K**.



NTFS

- **12,5% dos clusters** reservados para **M**aster **F**ile **T**able.
- A **MFT** é espelhada no final do FS e suas **entradas MFT reutilizáveis**.

NTFS

- **Cada arquivo** e **diretório** possui um **record** de 1 KB (= inode) no **MFT**.
- **Items** < 512 B podem estar **inteiramente contidos** no registro da MFT.

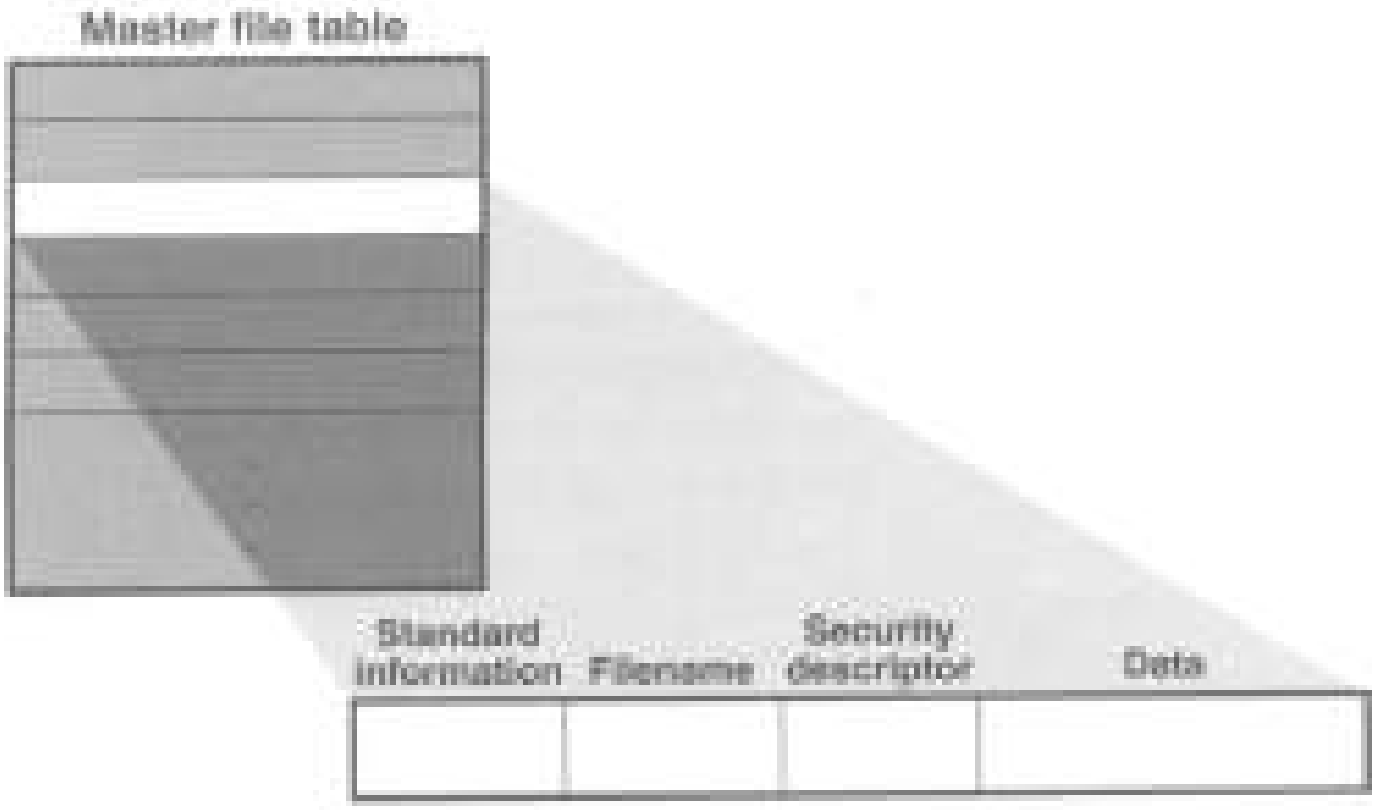
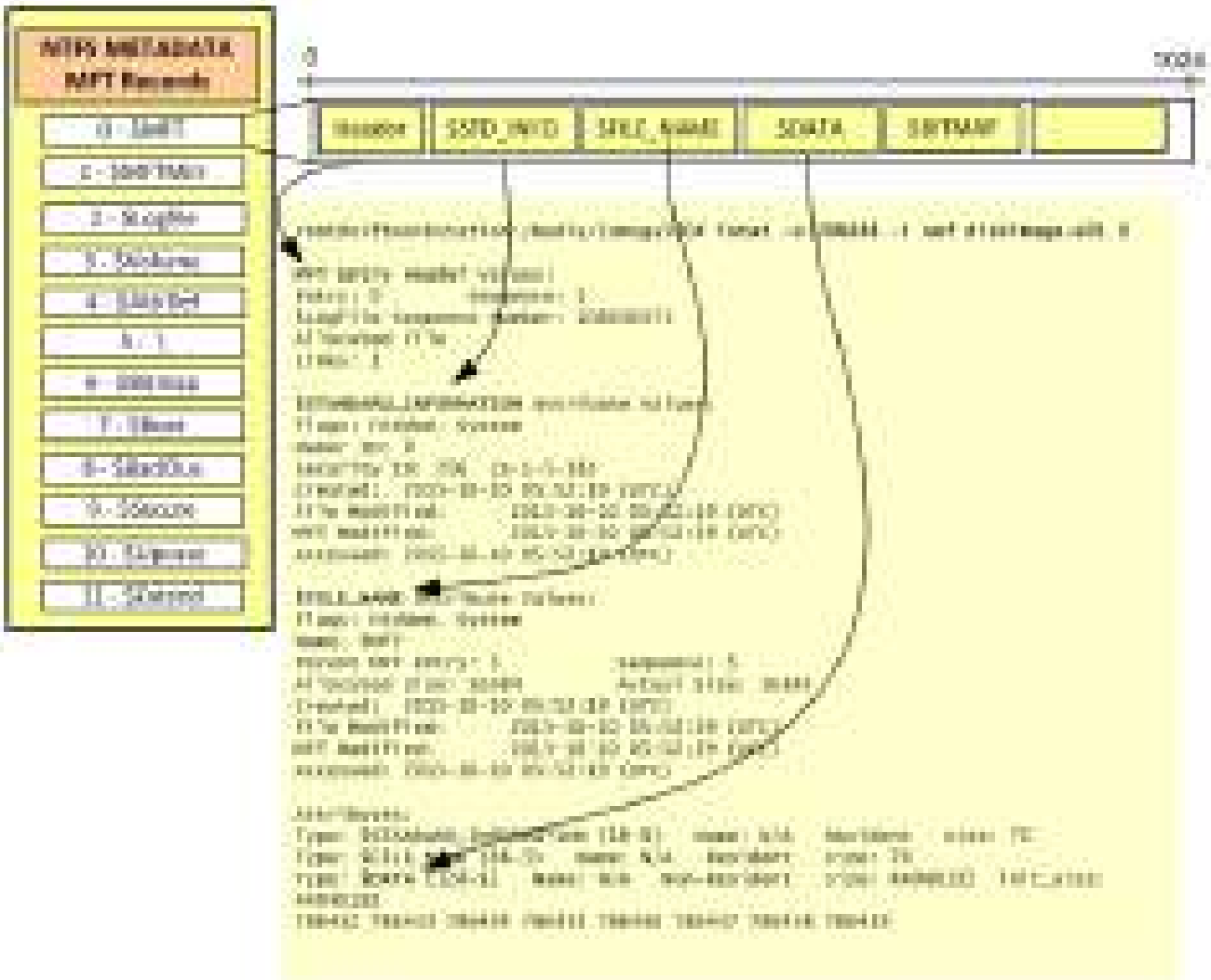


Figure 9-8
MFT record for a small file

MFT

- **Série de atributos** detalhando **metadados** e **estrutura dos dados**.

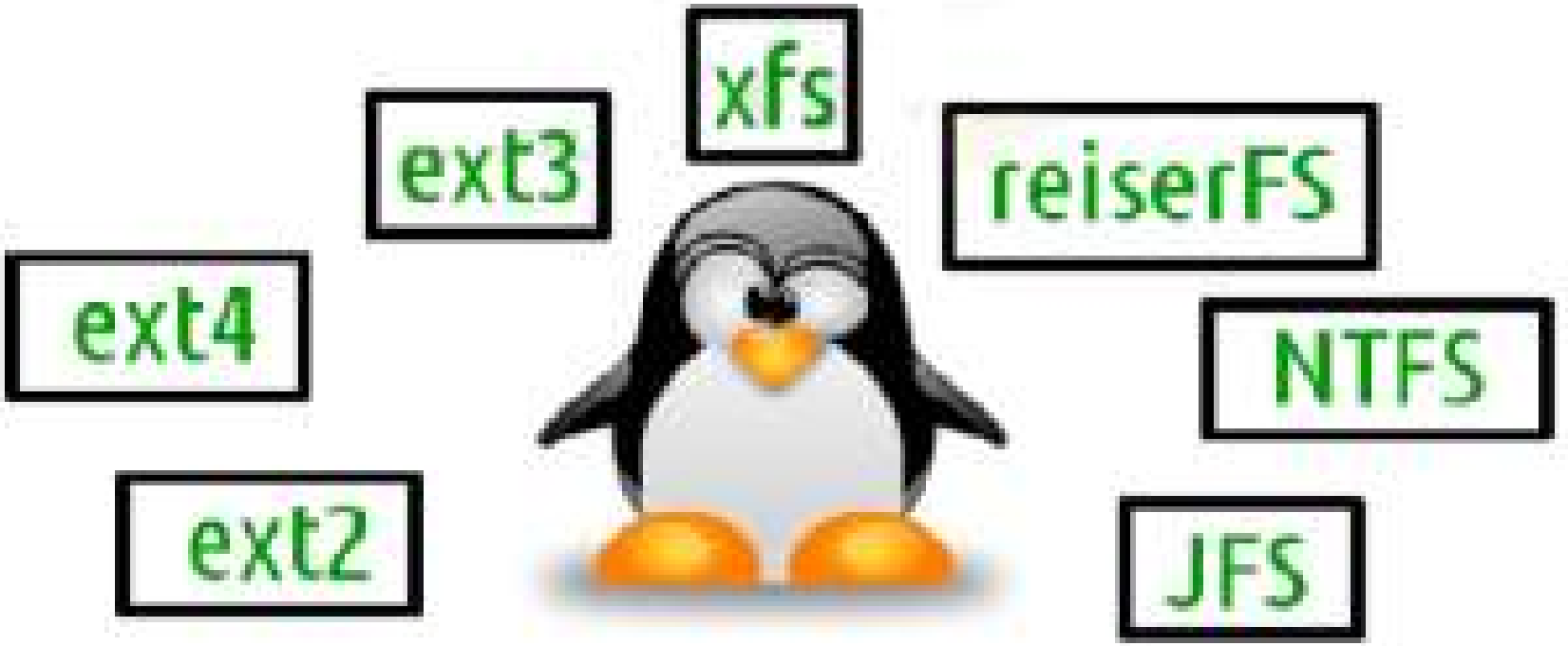


MFT

- **$STANDARD_INFORMATION**: Timestamps, Tamanho, Permissões. Estados: READONLY,ARCHIVE,HIDDEN,SYSTEM,COMPRESSED,ENCRYPTED.
- **$FILE_NAME**: Nome do arquivo/diretório e referência para localizar.
- **$ATTRIBUTE_LIST**: Atributo opcional quando precisa de + espaço.
- **$SECURITY_DESCRIPTOR**: Permissões de segurança (ACL).
- **$INDEX_ROOT**: Indexar o conteúdo de diretórios (entry point).
- **$INDEX_ALLOCATION**: Índices de alocação conteúdo de diretórios.
- **$BITMAP**: Gerencia os clusters utilizados pelo arquivo/diretório.
- **$DATA**: apontadores para clusters externos armazenando conteúdo.
- **$OBJECT_ID**: Identificador único global (GUID) do arquivo/diretório.

Outros Filesystems jornalados

- **JFS**: criado pela IBM (`http://www.ibm.com`).
- **XFS**: criado pela SGI (`http://oss.sgi.com`).
- **Ext 2**: Rémy Card. **Ext 3**: Stephen Tweedie, **Ext 4** sua extensão.
- **ReiserFS**: criado por Hans Reiser (`http://www.namesys.com`).
- **ZFS**: craido pela Sun Microsystems (por Solaris).

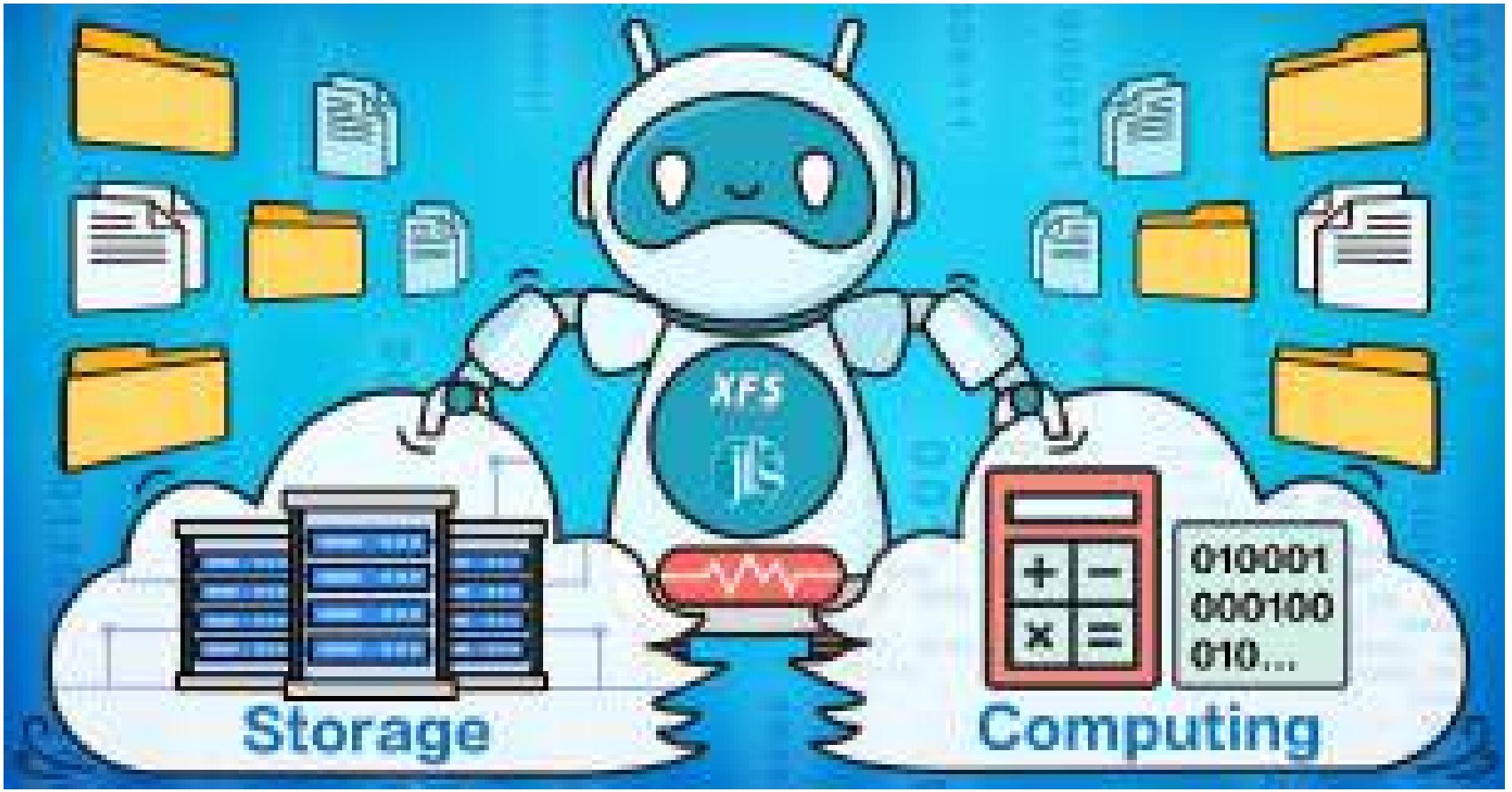


JFS (Journaled File System)

- Em **1990**, Filesystem de **64 bits** com **journaling da IBM** para BDs.
- **Arquivos grandes** em **pouca quantidade** (**área de controle pequena**).

XFS

- Em **1993**, FS jornalado 64 bits da **Silicon Graphics** para **IRIX** (BSD).
- Explicitamente projetado para lidar com **arquivos grandes** (como JFS).



Ext2

- Em **1993**, **Ext2** (**não jornalado**) por Rémy Card a **partir do Ext**.
- **Primeiro FS Linux**, ideal para **pendrives** (realiza menos escritas).

Ext3

- Em **2001**, **Ext2 jornalado** (Stefen Tweedie) suportado pelo **kernel 2.4+**.

Ext4

- Em **2006**, **aumento** muito o **limite de armazanamento** e **performance**.

+ Aumento do **tamanho do arquivo de 2 para 16 TB** (blocos 4 KB).
+ Aumento do **tamanho de partição** possível **de 32 TB para 1 EB**.

ReiserFS

- Escrito por **Hans Reiser**, em **forte decadência** e possível extinção!
- Baseadas num sistema **"fast balanced trees"** e **"dancing trees"** (v4).

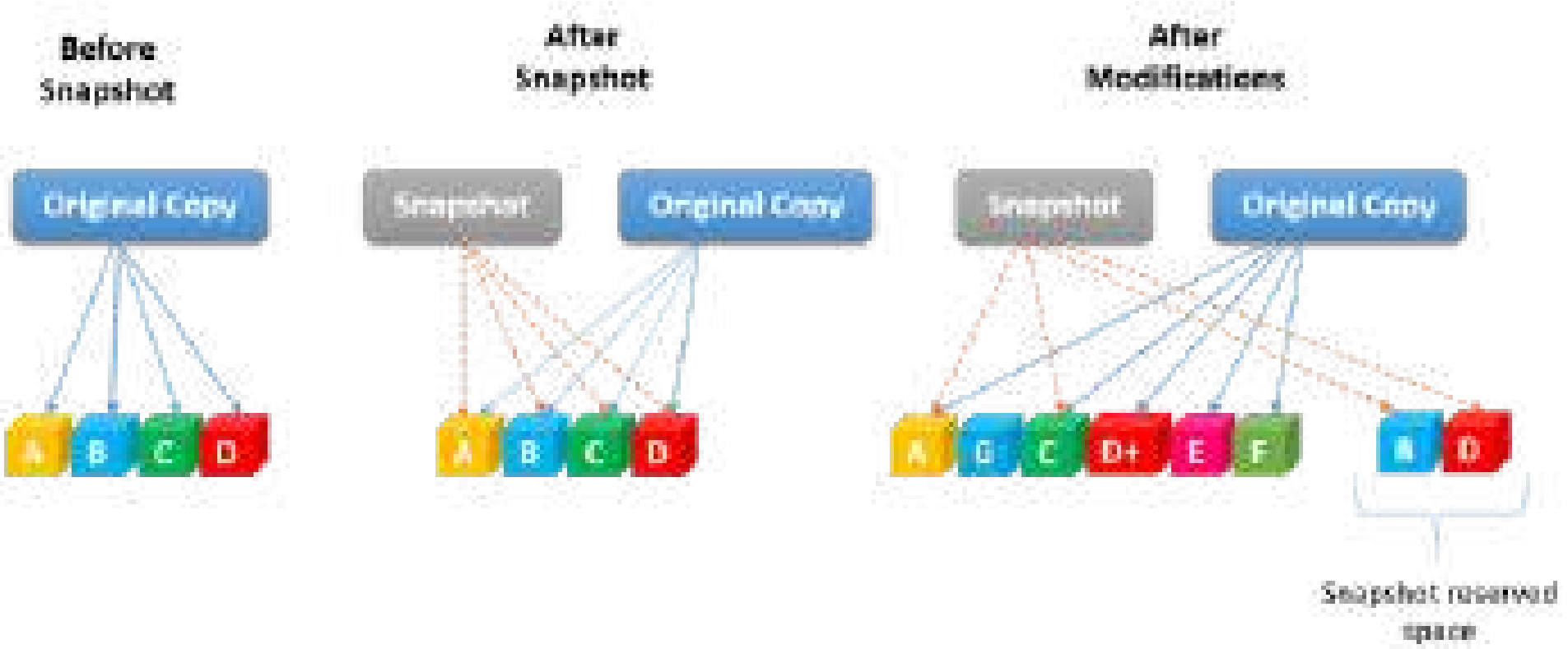
1. os **arquivos menores** do que o **tamanho de um bloco** e;
2. os **bytes finais** de arquivos que **não preencherem** o **último bloco**.

Nova geração de Filesystems

- Porém uma **nova geração de FS** foram desenvolvidos nos anos 2010.

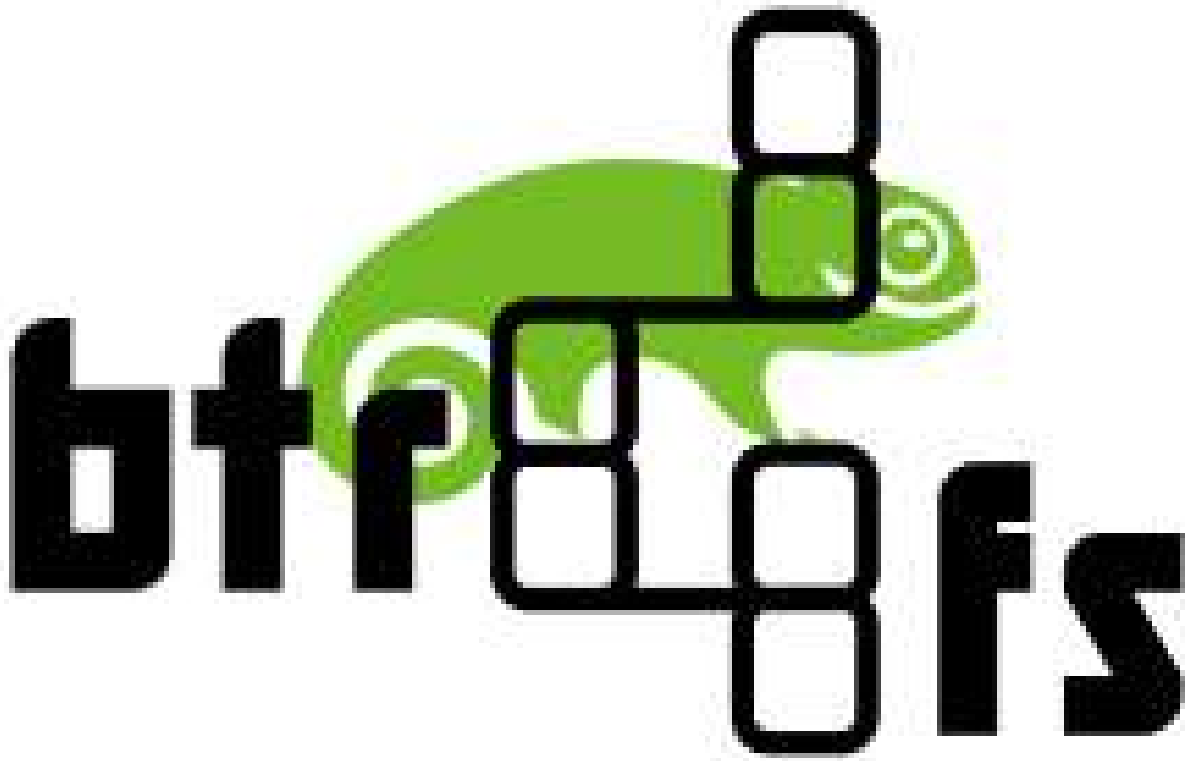
Copy-on-write

- **Metadados** controlam **versões das mudanças** de um arquivo.



BtrFS: B-tree File System

- Desenvolvido pela **Oracle a partir 2007** para Linux (estável em 2014).
- Usa árvores B para **organizar dados e metadados** de forma eficiente.

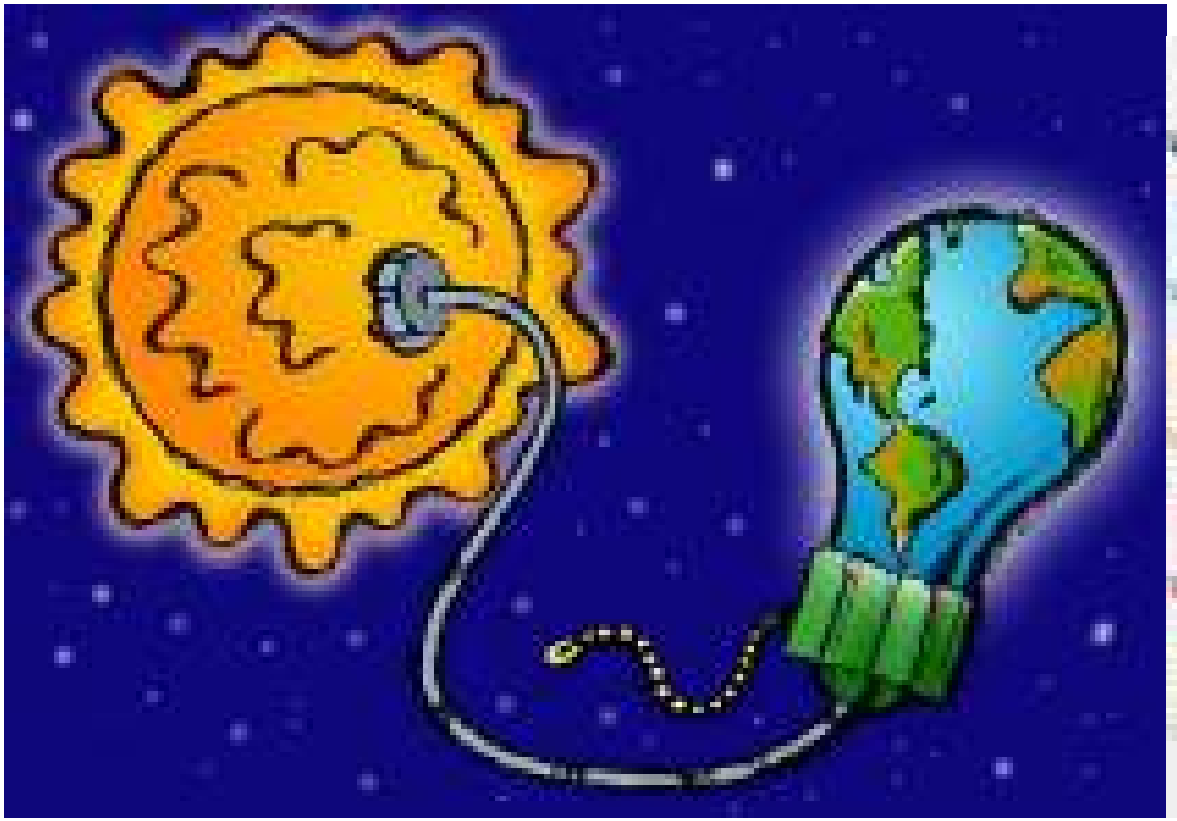


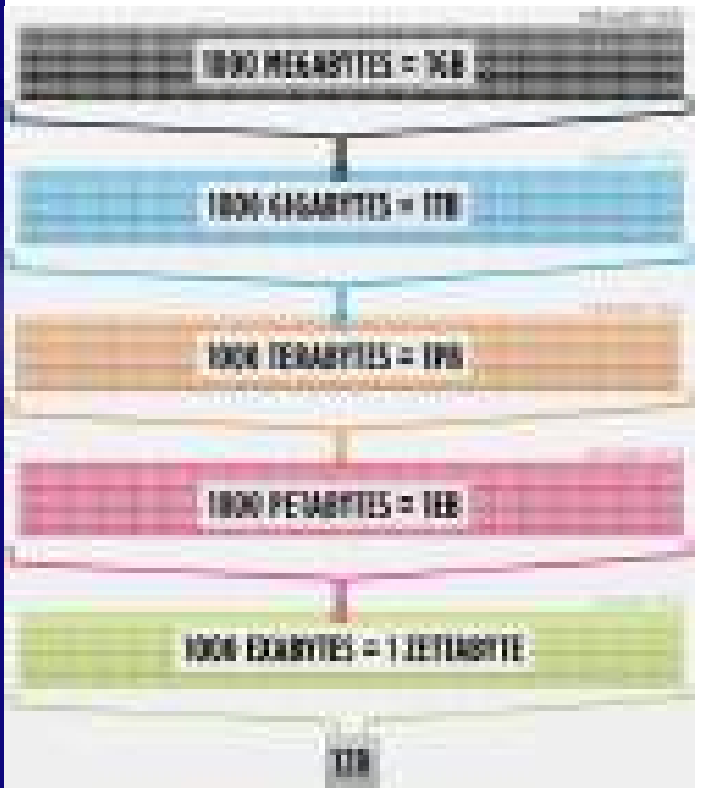
* Chris Mason que trabalhava no ReiserFS para SUSE se juntou a btrFS.

ZFS: Zettabyte File System

OpenZFS (2010)

- **Sun Microsystem** (2005) monta partição 10^{24} **unidades** de **3 TB**!
- **FS de 128 bits**, endereça $1,84x10^{19}$ vezes + dados do que um 64 bits.





"Montar exigiria mais energia do que ferver os oceanos" **Jeff Bonwick** (criador)

Apagando e formantando

- Quando mandamos **apagar um arquivo de 4GB**, ocorre em **< 10 s**.
- Conseguimos realmente **remover todos os dados** referente a **4 GB**???

Apagando e formantando

- **Depois da deleção**, o **arquivo permanece** na **área de dados**!
- Caso haja a necessidade, **OS poderá escrever outro arquivo nele**!

Hard Link e Soft Link

- Em Linux, os **links de arquivos** são quase idênticos aos **atalhos**!

Hard Link e Soft Link

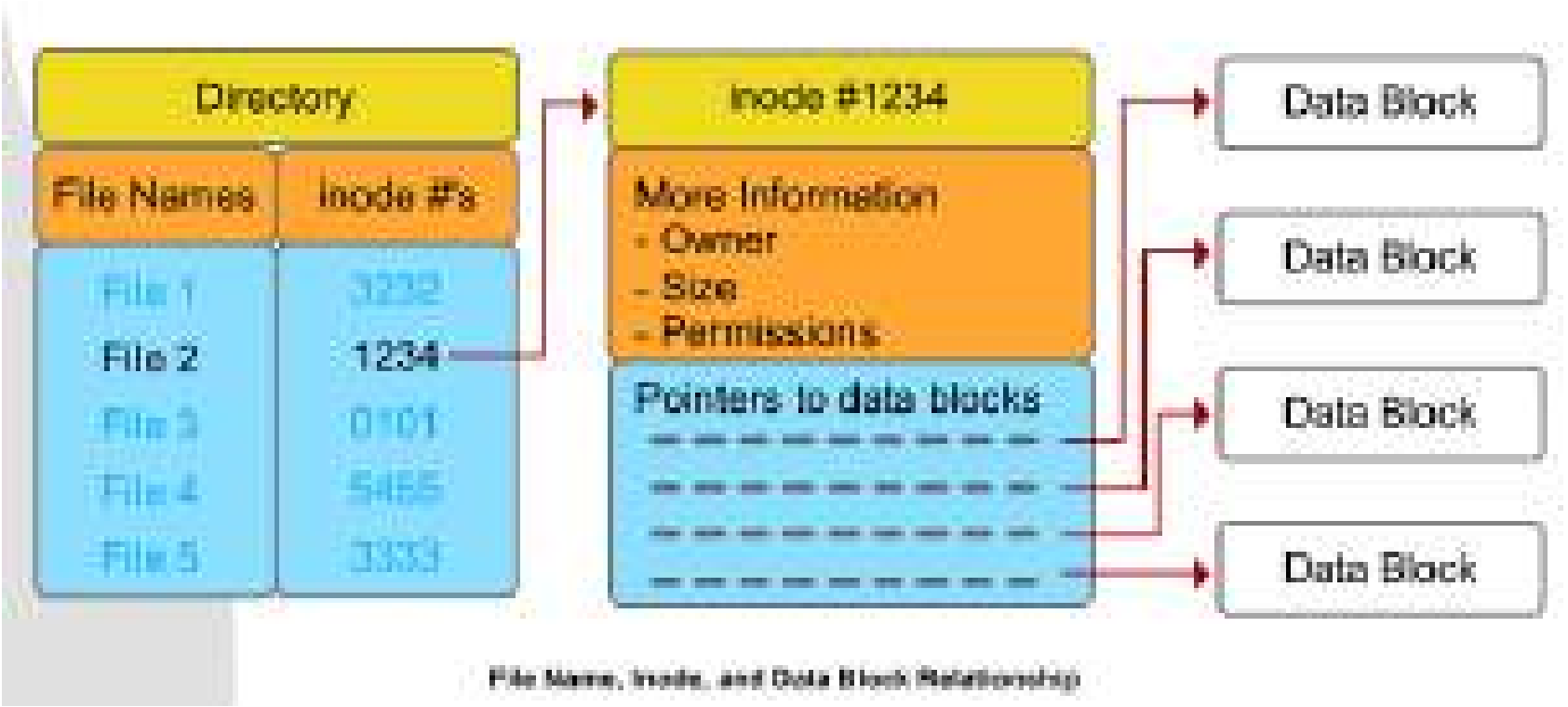
- Cria **links (atalhos)** de **dois tipos**: **hard link ou soft link**.

Chave	Função
`-s`	Cria um link simbólico (ou soft link).

https://www.youtube.com/watch?v=L0xvJWRN_r0

Hard Link (Link absoluto)

- O **controle do hard link** é feito **pelo diretório**.
- É um **segundo, terceiro, quarto, quinto... nome** para um **arquivo**!



Hard Link (Link absoluto)

- O **hard link não pode ser aplicado** a **diretórios**.
- Por ser o **mesmo inode**, só pode ser **criado na mesma partição**!

Hard Link (Link absoluto)

- O **hard link** pode ser criado com o **comando `ln`**.

```
$ cat 2001.bib
@proceedings{DBLP:conf/alsc/2001,
  editor       = {Bryan C. Andregg},
  title        = {5th Annual Linux Showcase {\&} Conference 2001, USA},
  publisher    = {{USENIX} Association},
  year         = {2001},
  url          = {https://www.usenix.org/conference/als01},
  biburl       = {https://dblp.org/rec/conf/alsc/2001.bib}
}
$ ln 2001.bib 2002.bib
$ cat 2002.bib
@proceedings{DBLP:conf/alsc/2001,
  editor       = {Bryan C. Andregg},
  title        = {5th Annual Linux Showcase {\&} Conference 2001, USA},
  publisher    = {{USENIX} Association},
  year         = {2001},
  url          = {https://www.usenix.org/conference/als01},
  biburl       = {https://dblp.org/rec/conf/alsc/2001.bib}
}
$ ls -i 200*
7615420 2002.bib  7615420 2001.bib
```

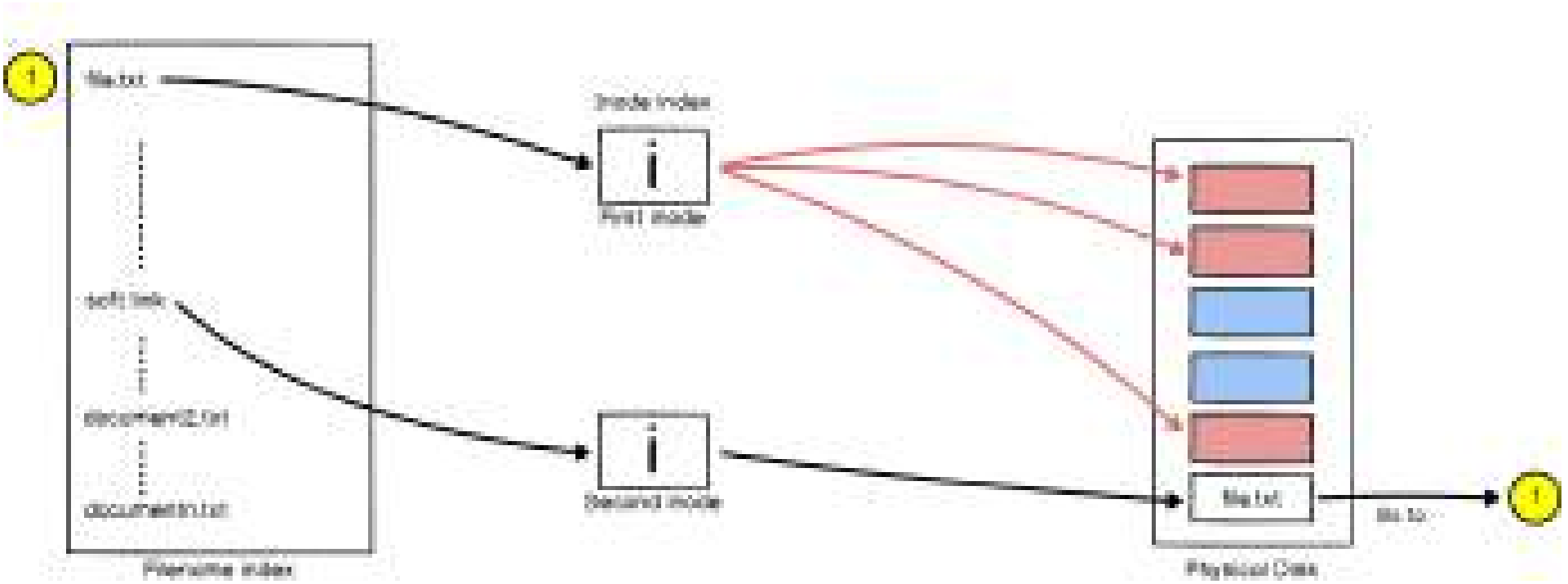
Soft Link (Link simbólico)

- As **setas apontam para destinos** (somente no caso do soft links).
- A **letra l** no início representa a **existência de links simbólicos**.

```
# ls -l /
lrwxrwxrwx   1 root root       7 juil. 19  2021 bin -> usr/bin
drwxr-xr-x   4 root root    4096 mai   12 12:26 boot
```

Soft Link (Link simbólico)

- Representa um **novo arquivo possuindo inode próprio**!
- Atua como uma **referência para o nome** de um arquivo.



Soft Link (Link simbólico)

- Ele simplesmente **aponta para** um outro **arquivo ou diretório**.
- **sed soft link** vai **terminar o link**, o **editor edita** o **arquivo apontado**.

```
~/Documents/Aulas/SysOpe/5-Filesystems$ ln -s 2001.bib 2003.bib

~/Documents/Aulas/SysOpe/5-Filesystems$ cat 2003.bib
@proceedings{DBLP:conf/alsc/2001,
  editor        = {Bryan C. Andregg},
  title         = {5th Annual Linux Showcase {\&} Conference 2001, USA},
  publisher     = {{USENIX} Association},
  year          = {2001},
  url           = {https://www.usenix.org/conference/als01},
  biburl        = {https://dblp.org/rec/conf/alsc/2001.bib}
}
~/Documents/Aulas/SysOpe/5-Filesystems$ ls -l 2003.bib
lrwxrwxrwx 1 jr jr 7 sept. 20 14:36 2003.bib -> 2001.bib

~/Documents/Aulas/SysOpe/5-Filesystems$ ls -i 200*
7615420 2002.bib  7615421 2003.bib  7615420 2001.bib
```

https://www.youtube.com/watch?v=4-vye3QFTFo

Soft Link (Link simbólico)

- **Renomeando o arquivo original** irá **quebrar o soft link** em seguido.

```
$ ln -s 2002.bib 2004.bib && ls -l
$ rm 2001.bib && ls -l
$ mv 2002.bib 2005.bib && ls -l
```