

INE5412 Sistemas Operacionais I

L. F. Friedrich

SA – Estudo de caso - EXT2/3

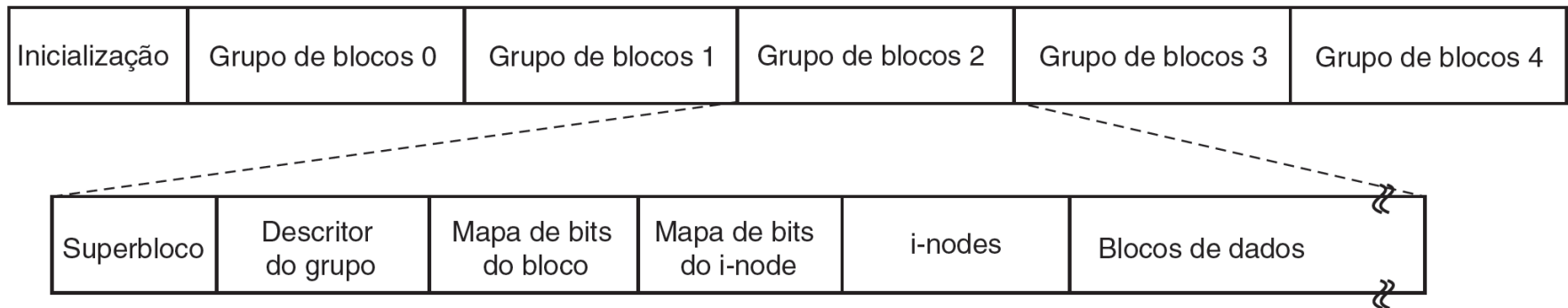
SA Ext2/3 - Layout

O SA Ext2/3 FS é particionado em **grupos de blocos**

Cada grupo de blocos tem a mesma estrutura interna.

Cada **grupo de blocos** é composto da forma que segue:

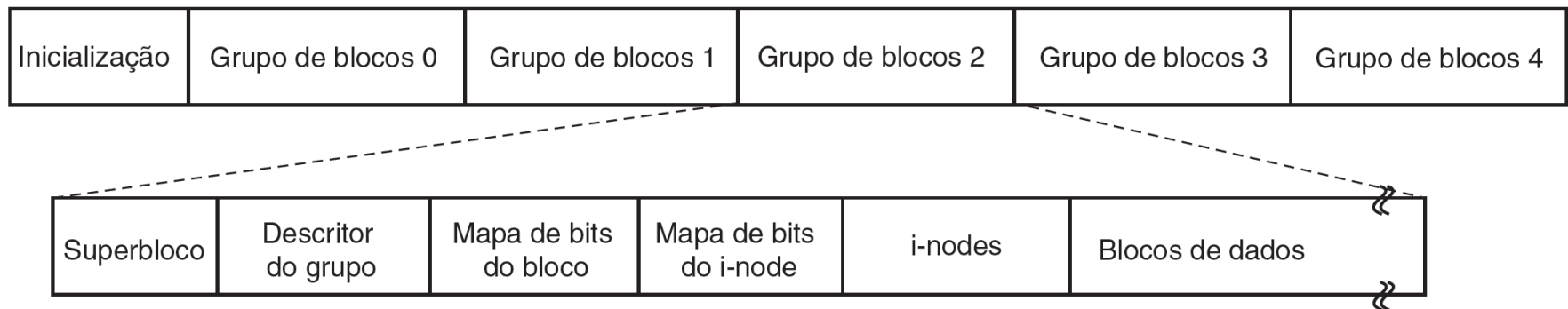
- Superbloco, descritor do grupo,
- Mapa de bits dos blocos, mapa de bits dos inodes,
- tabela de inodes, blocos de dados.



■ **Figura 10.19** Organização de disco do sistema de arquivos ext2 do Linux.

Ext2/3 File System Layout

- Porque?
 - **Desempenho** e **Confiabilidade**.
 - Manter informações (metadados) e conteúdo dos arquivos próximos de forma que a cabeça de l/e do disco não percorra distancias muito grandes.
 - As informações (metadados) estão espalhadas eliminando único ponto de falha.



■ **Figura 10.19** Organização de disco do sistema de arquivos ext2 do Linux.

Superbloco

```
# tune2fs -l /dev/sda1
tune2fs 1.40-WIP (14-Nov-2006)
```

```
.....
```

```
Filesystem OS type:      Linux
```

```
Inode count:             122624
```

```
Block count:             244983
```

```
Reserved block count:    12249
```

```
Free blocks:             228306
```

```
Free inodes:             122586
```

```
First block:             0
```

```
Block size:              4096
```

```
.....
```

```
Blocks per group:        32768
```

```
Fragments per group:     32768
```

```
Inodes per group:        15328
```

```
Inode blocks per group:  479
```

```
.....
```

```
First inode:             11
```

```
Inode size:              128
```

```
Journal inode:           8
```

```
First orphan inode:      27
```

```
.....
```

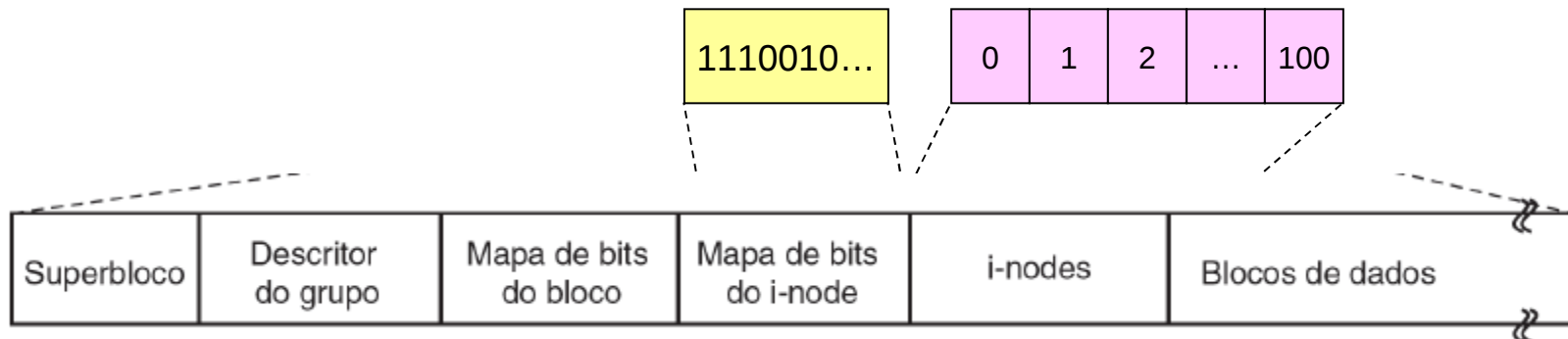
← Informação do SA

← layout do SA

← Utilização dos inodes.

Informações (Metadados)

- Bitmap blocos
 - Guarda o **estado de alocação dos blocos** do grupo.
- Bitmap
 - Guarda o **estado de alocação dos inodes** do grupo.
- Tabela de inodes
 - Guarda o conteúdo dos inodes do grupo.
 - Cada inode tem tamanho fixo, especificado no superbloco.
 - Os inodes são armazenados **sequencialmente**.



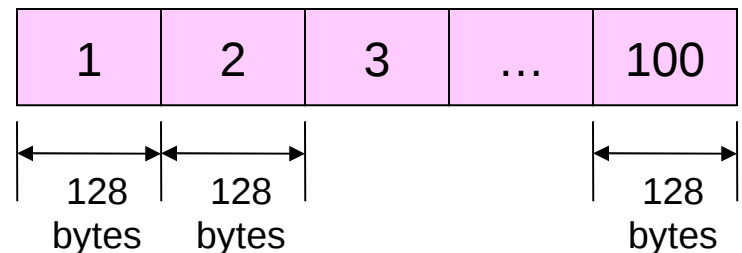
Estrutura do inode

Bytes	Value
0-1	File type and permission
2-3	User ID
4-7	Lower 32 bits of file sizes in bytes
8-23	Time information
24-25	Group ID
26-27	Link count
...	...
40-87	12 direct data block pointers
88-91	Single indirect block pointer
92-95	Double indirect block pointer
96-99	Triple Indirect block pointer
...	...
108-111	Upper 32 bits of file sizes in bytes

localizar o início do inode i :

$$128 \times (i-1) + \text{Endereço de Início da tabela.}$$

Tabela de inodes



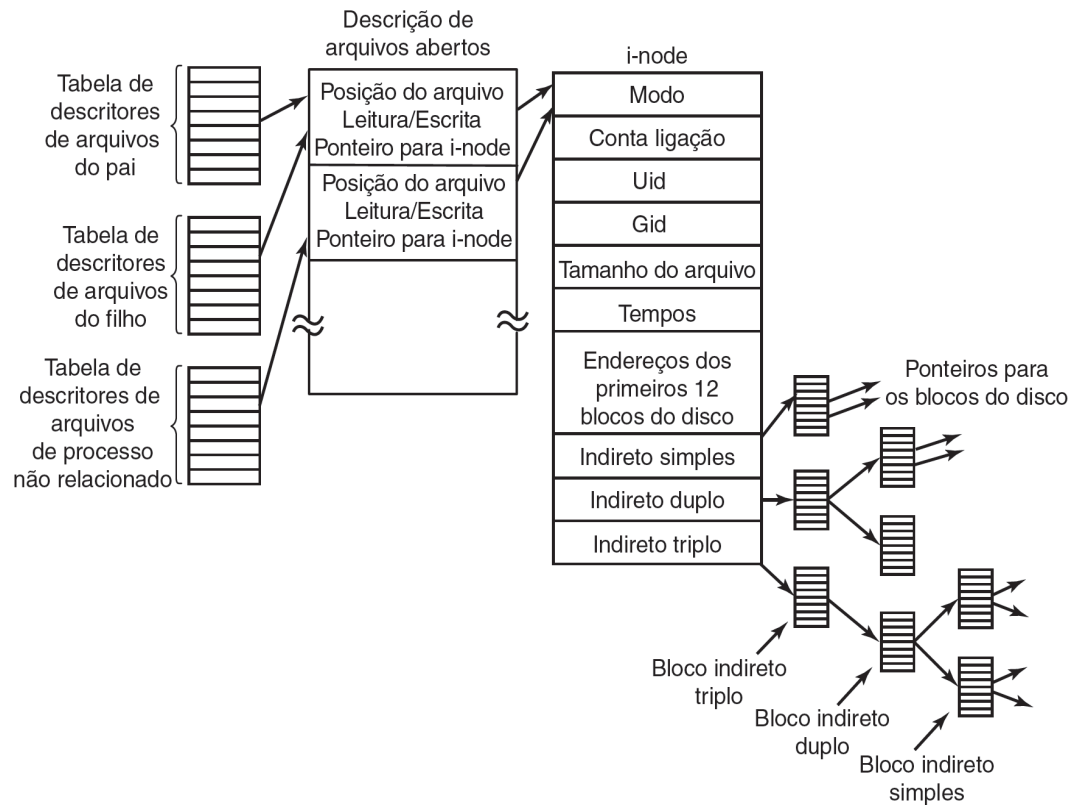
← Estrutura de 128-byte

Estrutura do inode

Campo	Bytes	Descrição
Mode	2	Tipo do arquivo, bits de proteção, setuid, bits setgid
Nlinks	2	Número de entradas no diretório apontando para esse i-node
Uid	2	UID do proprietário do arquivo
Gid	2	GID do proprietário do arquivo
Size	4	Tamanho do arquivo em bytes
Addr	60	Endereço dos primeiros 12 blocos do disco e de três blocos indiretos
Gen	1	Número de geração (incrementado cada vez que o i-node é reutilizado)
Atime	4	Hora do último acesso ao arquivo
Mtime	4	Hora da última modificação do arquivo
Ctime	4	Hora da última alteração do i-node (exceto as outras vezes)

■ **Tabela 10.13** Alguns campos na estrutura de i-nodes do Linux.

Estrutura do inode



■ **Figura 10.21** A relação entre a tabela de descritores de arquivos, a tabela de descritores de arquivos abertos e a tabela de i-nodes.

Inodes especiais

- A numeração de inodes começa em 1.
- Inode #1 sempre aponta para blocos ruins.
- Inode #2 sempre aponta para diretório raiz.
- No Ext3, inode #8 aponta para blocos de dados do sistema de journal.

Entrada de Diretório

- Um diretório é armazenado em blocos de dados.
- Um diretório é feito de pelo menos 2 entradas: pai e atual.

Inode (4 bytes)
Tam. entrada (2 bytes)
Tam. nome (2 bytes)
Nome (até 255 bytes)

Inode #	Tam. entrada	Tam. nome	Nome
123	12	2	.
2	12	3	..
5,086	16	7	12.jpg

$$4 + 2 + 2 + \boxed{7} = 15?!$$

não esquecer
'\0'.

Alinhamento de palavra

Link File

- Um **hard link é uma entrada de diretório** apontando para um arquivo existente.
 - Nenhum arquivo novo é criado!

```
# ls /dir1/12.jpg
12.jpg
# ln /dir1/12.jpg /my_link
# _
```

Uma nova
entrada de
Diretório é
criada.

Diretório: /dir1

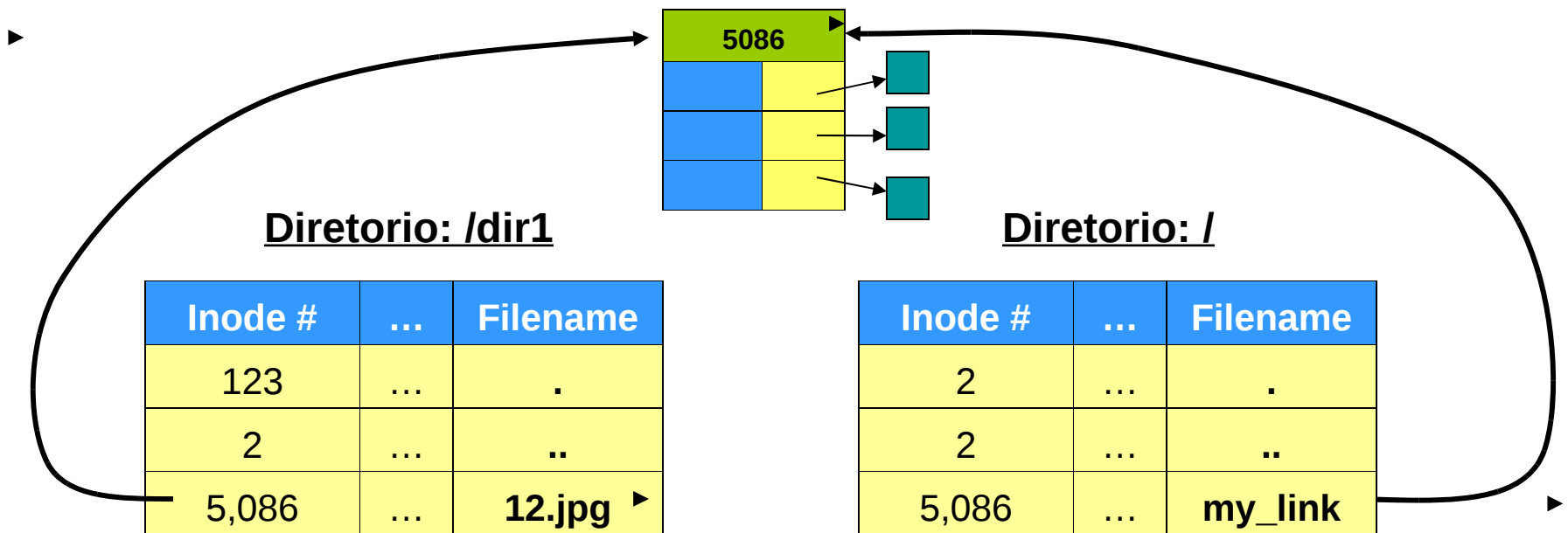
Inode #	...	Filename
123
2
5,086	...	12.jpg

Diretório: /

Inode #	...	Filename
2
2
5,086	...	my_link

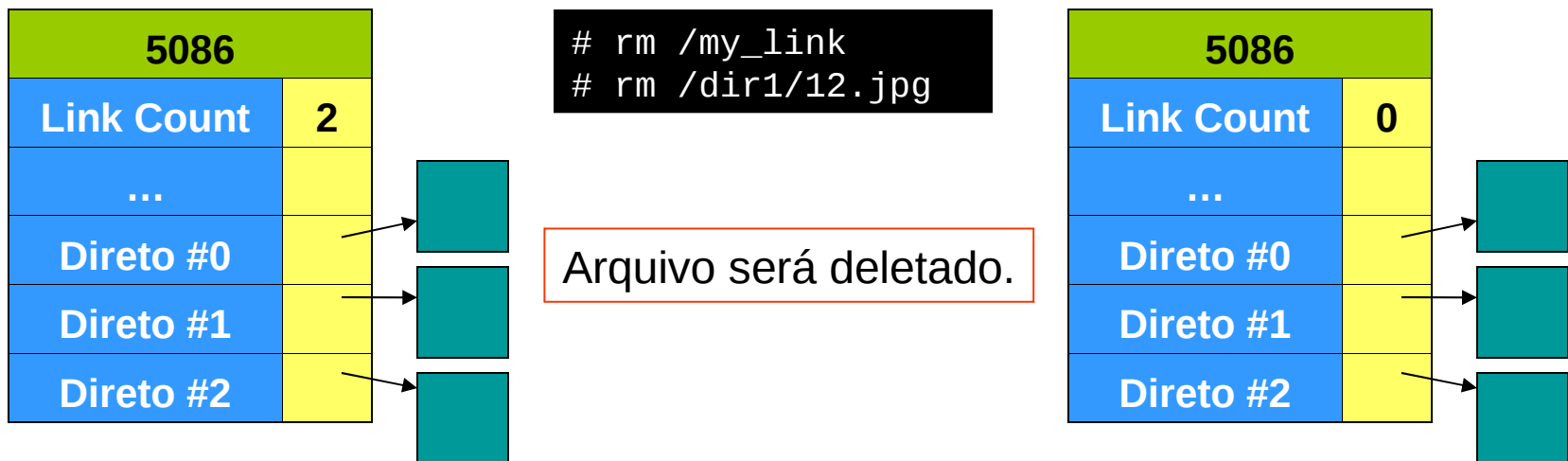
Link File

- Conceitualmente, isto cria um arquivo com 2 nomes.
- O interessante é que **deletar apenas uma das entradas de diretório não irá deletar o conteúdo do arquivo!**



Link File

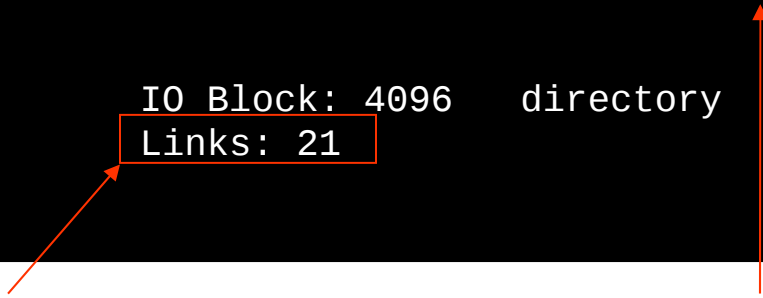
- O campo **link count** no inode informa quantas entradas de diretório apontam para o arquivo.
 - O exemplo tem um link count igual a 2.
 - Qdo um link é removido usando a chamada **unlink()**, o link count é decrementado de 1.
 - Qdo o **link count = 0**, o conteúdo do arquivo poderá ser removido, incluindo os blocos de dados e o inode.



Link File

- Hard Links especiais
 - Diretório “.” é um hard link para ele próprio.
 - Diretório “..” é um hard link para o diretório pai.

```
# ls -F /
bin/    dev/    initrd/    lib/    mnt/    root/    sys/    var/
boot/   etc/    initrd.img@    lost+found/    opt/    sbin/    tmp/    vmlinuz@
cdrom@  home/   initrd.img.old@    media/    proc/    srv/    usr/    vmlinuz.old@
# stat /
  File: `/'
  Size: 4096          Blocks: 8          IO Block: 4096    directory
Device: 802h/2050d   Inode: 2           Links: 21
...
```



The terminal output shows the command `ls -F /` listing the root directory. The file `vmlinuz.old@` is highlighted with a red box. Below the listing, the command `stat /` is executed, showing file statistics. The `Links: 21` field is also highlighted with a red box. Two red arrows point from explanatory boxes below to these two fields: one from the box containing the calculation for the 21 links, and another from the box stating it is a symbolic link.

“..” em 19 sub-diretórios + “.” do “/” + “..” do “/” = 21.

um link simbólico

Link File

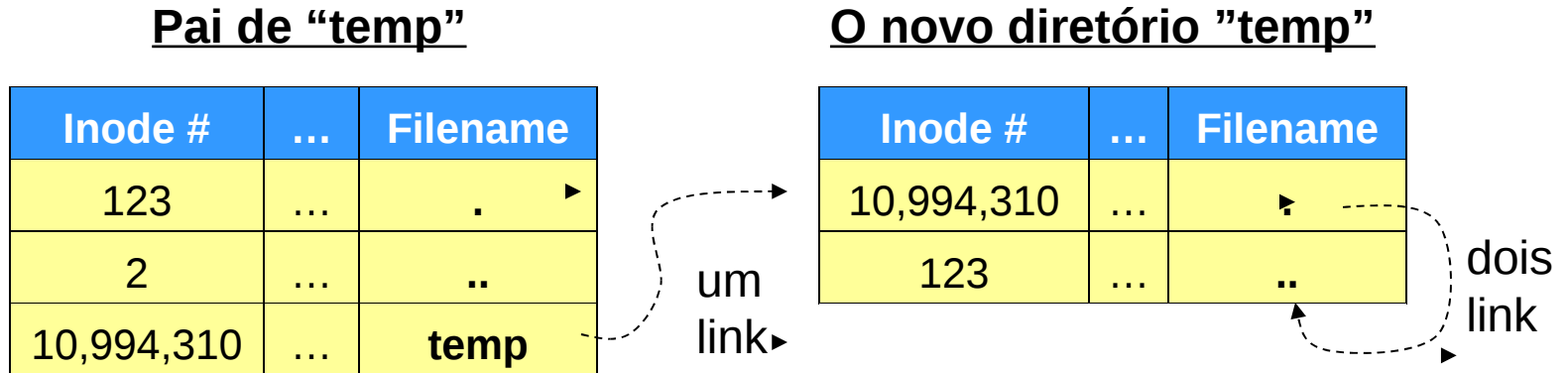
- Quando um arq. regular é criado, o link count é **1**.

```
# stat /boot/grub/menu.lst
File: `/boot/grub/menu.lst'
  Size: 4552          Blocks: 16          IO Block: 4096   regular file
Device: 801h/2049d   Inode: 30669       Links: 1
.....
```

- Quando um diretório é criado, o link count é **2**.

```
# mkdir temp
# stat temp
File: `temp'
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: 804h/2052d   Inode: 10994310    Links: 2
.....
```

Link File



- Quando um diretório é criado, o link count é 2.

```
# mkdir temp
# stat temp
  File: `temp'
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: 804h/2052d   Inode: 10994310    Links: 2
. . . . .
```


Link File

- Um link simbólico é um **arquivo**.
 - Diferente do hard link, um **novο inode é criado** para um link simbólico.

```
# ls /dir1/12.jpg
12.jpg
# ln -s /dir1/12.jpg /my_link
# ls -l /mylink
/mylink -> /dir1/12.jpg
#
```

uma nova
entrada de
diretório é
criada.

Diretório: /dir1

Inode #	...	Filename
123
2
5,086	...	12.jpg

Diretório: /

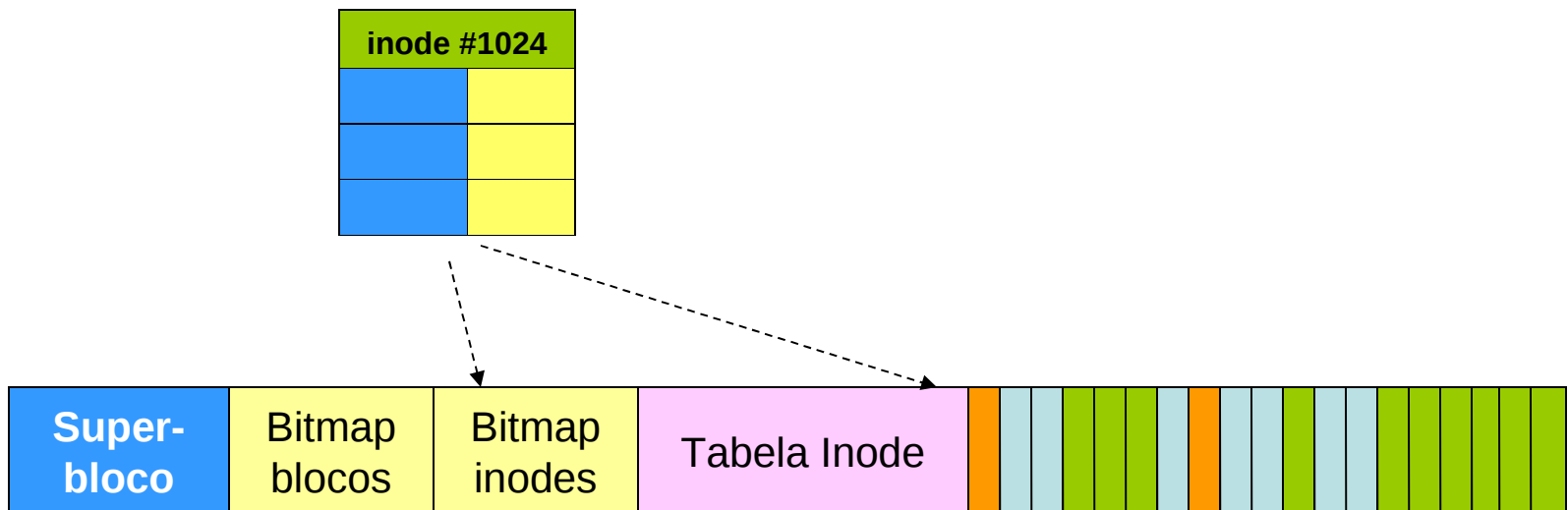
Inode #	...	Filename
2
2
6,120	...	my_link

Link File

- A localização de armazenagem do caminho (nome) depende do tamanho do nome.
 - Se o nome é menor que 60 char, é armazenado no espaço destinado aos 12 blocos diretos e os 3 blocos indiretos.
 - $(12 + 3) \times 4 = 60$.
 - Se o nome é maior que 60 char, um bloco de dados é alocado para armazená-lo.

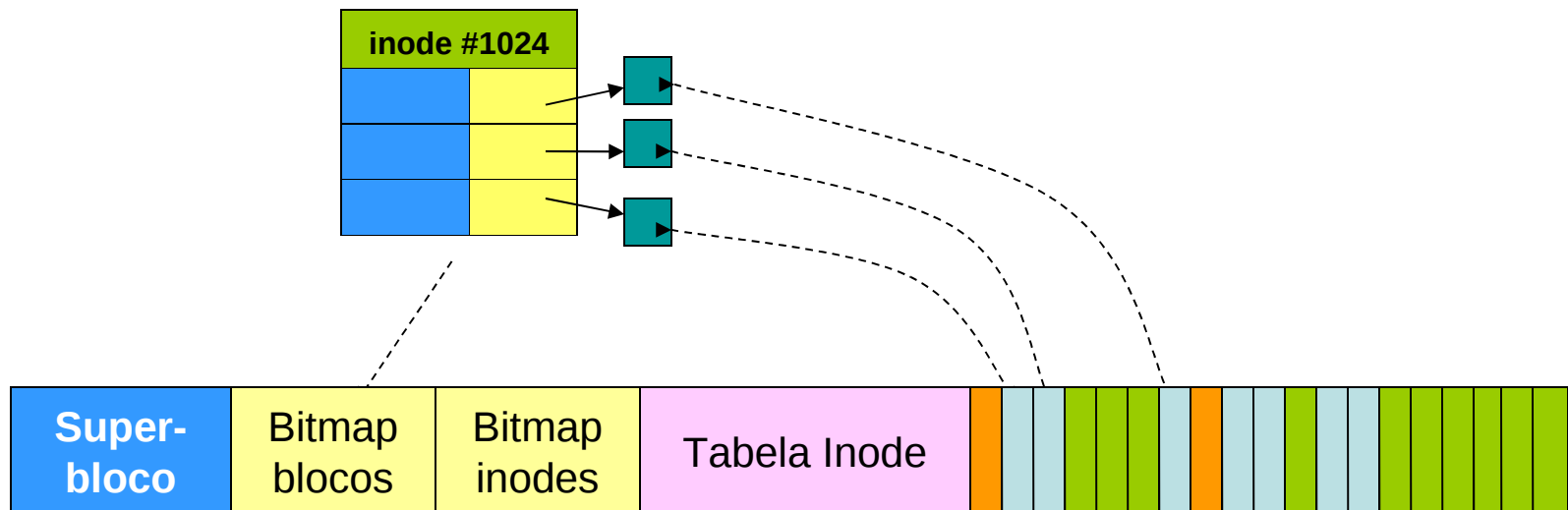
Como escrever um arquivo no Ext2/3

- Criar um arquivo “/dir1/hello.txt”.
 - passo (1): buscar no bitmap de inodes um inode não alocado para um novo arquivo.



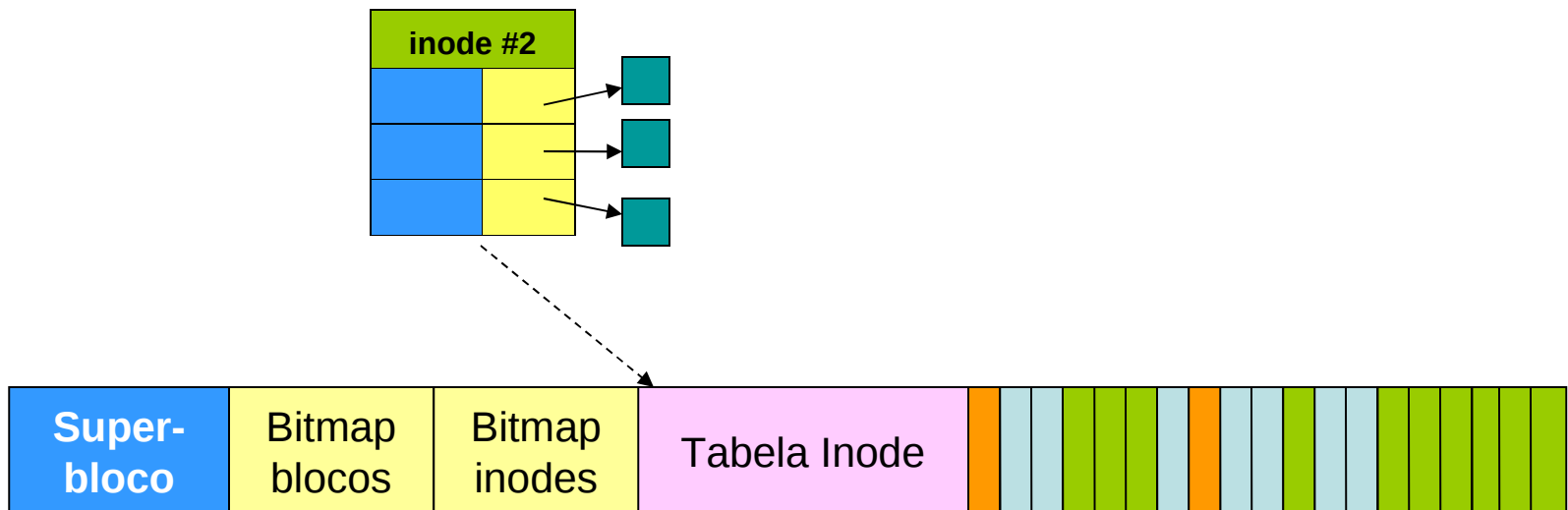
Como escrever um arquivo no Ext2/3

- Criar arquivo “/dir1/hello.txt”.
 - passo (2): buscar no bitmap blocos de dados não alocados para o novo arquivo.



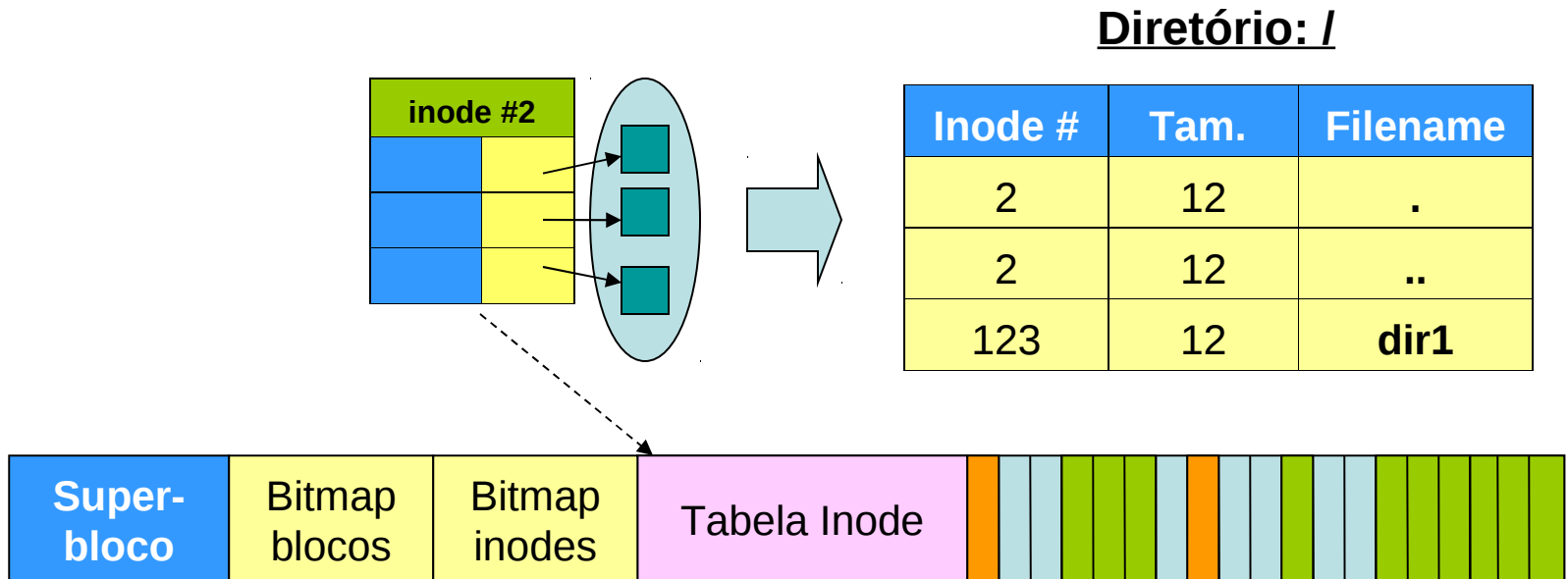
Como escrever um arquivo no Ext2/3

- Criar arquivo “/dir1/hello.txt”.
 - passo (3): ler o inode do diretório raiz, inode #2.



Como escrever um arquivo no Ext2/3

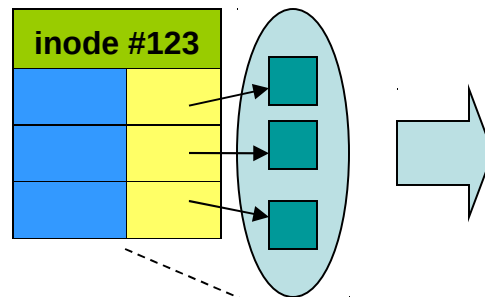
- Criar arquivo “/dir1/hello.txt”.
 - passo (4): conforme ponteiros dos blocos, ler a estrutura da entrada de diretório “dir1”.



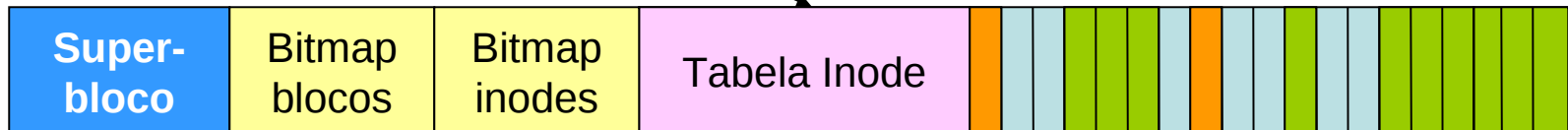
Como escrever um arquivo no Ext2/3

- Criar arquivo “/dir1/hello.txt”.
 - passo (5): ler inode #123 e os blocos de dados.
Construir as entradas de diretório.

Diretório: /dir1



Inode #	Tam.	Filename
123	12	.
2	12	..
456	16	panic.c
894	20	picture.jpg

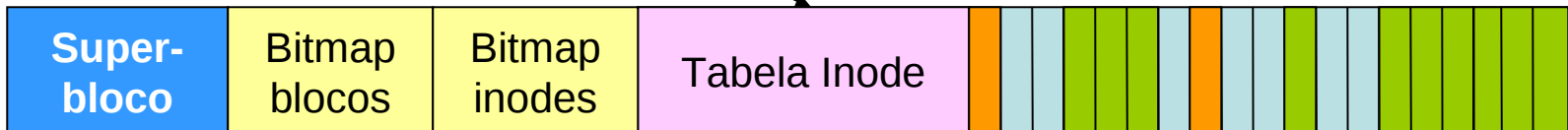
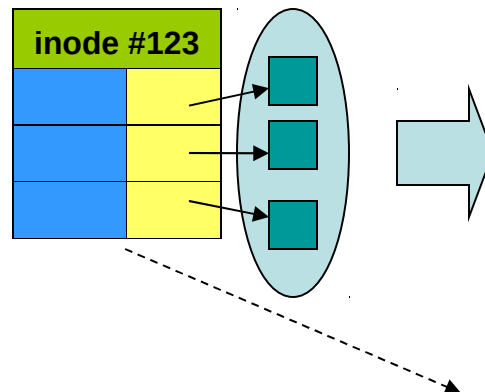


Como escrever um arquivo no Ext2/3

- Criar arquivo “/dir1/hello.txt”.
 - passo (6): Adicionar uma nova entrada de diretório com inode #1024 em “/dir1”.

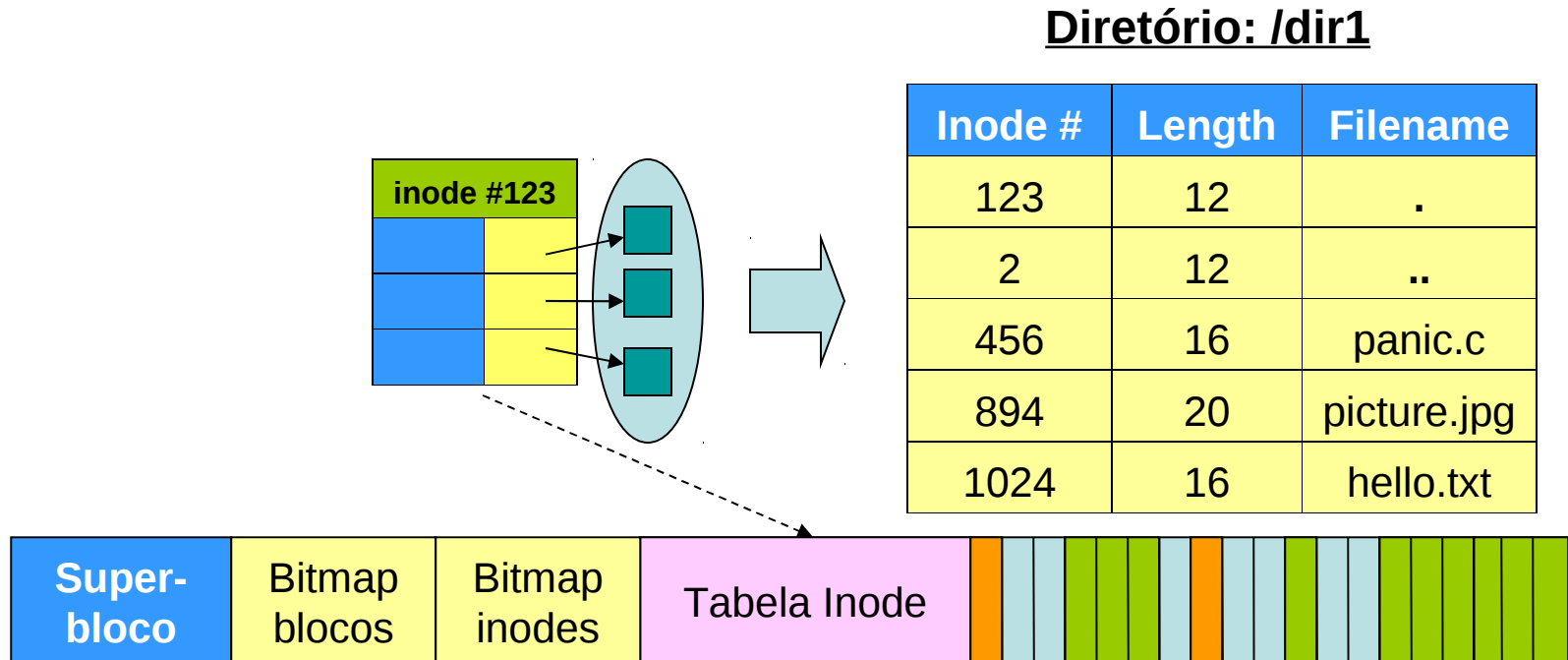
Directory: /dir1

Inode #	Tam.	Filename
123	12	.
2	12	..
456	16	panic.c
894	20	picture.jpg
1024	16	hello.txt



Como deletar um arquivo no Ext2/3

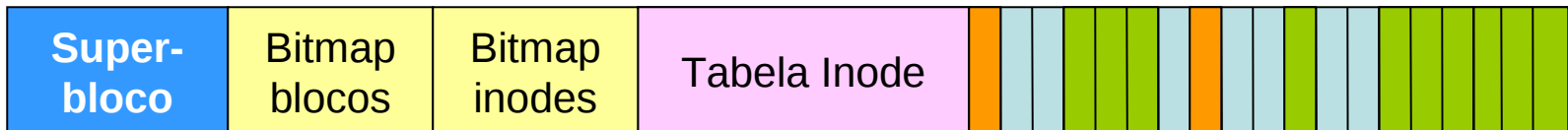
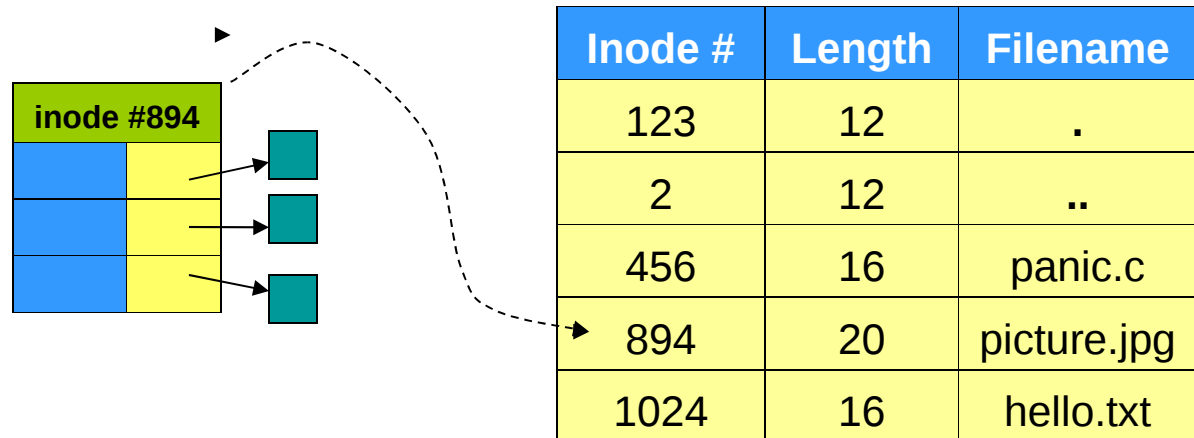
- Deletar arquivo “/dir1/picture.jpg”.
 - passo (1): ler o inode e os blocos de dados de “/dir1”, localizar o inode de “picture.jpg”.



Como deletar um arquivo no Ext2/3

- Deletar arquivo “/dir1/picture.jpg”.
 - passo (2): ler inode #894 e decrementar o contador “link count” em “picture.jpg”. O contador torna-se zero.

Diretório: /dir1

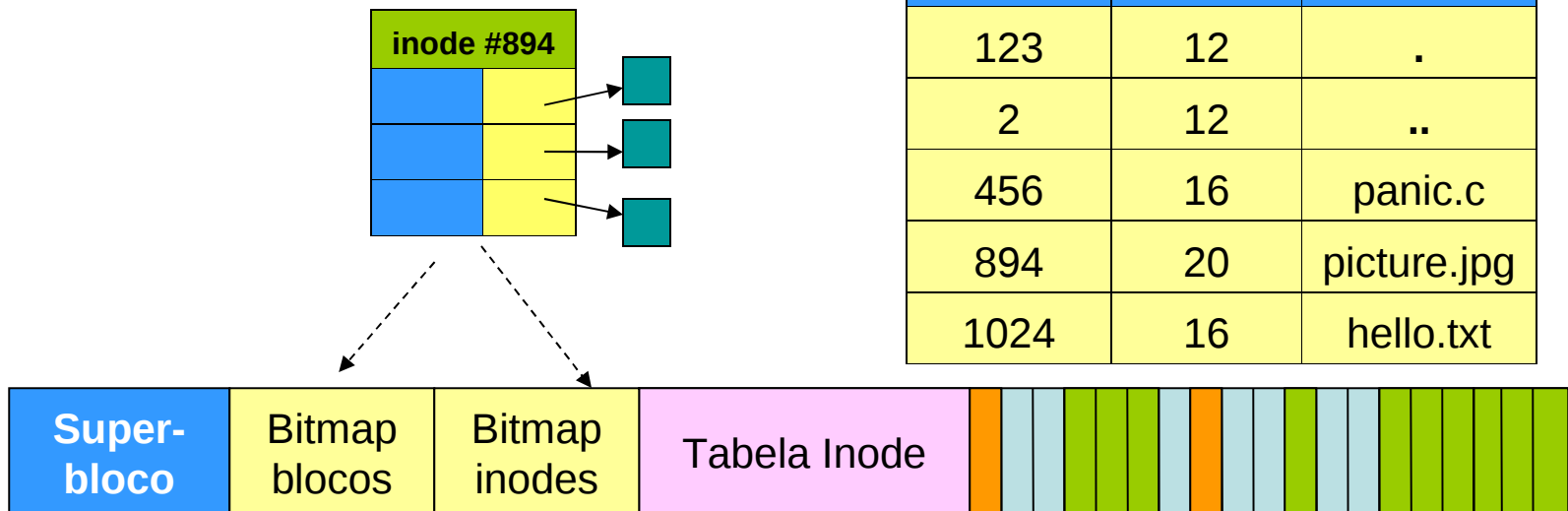


Como deletar um arquivo no Ext2/3

- Deletar arquivo “/dir1/picture.jpg”.
 - passo (3): liberar os blocos de dados e o inode nos bits correspondentes no bitmap blocos e no bitmap inodes.

Diretório: /dir1

Inode #	Length	Filename
123	12	.
2	12	..
456	16	panic.c
894	20	picture.jpg
1024	16	hello.txt



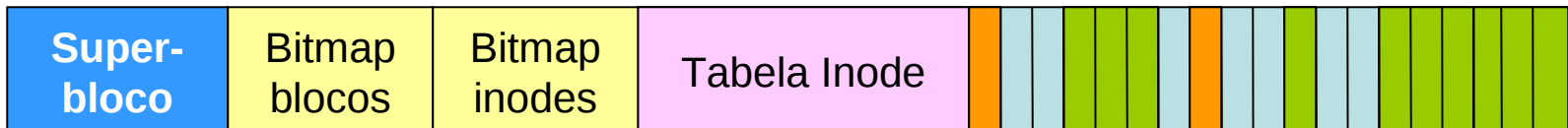
Como deletar um arquivo no Ext2/3

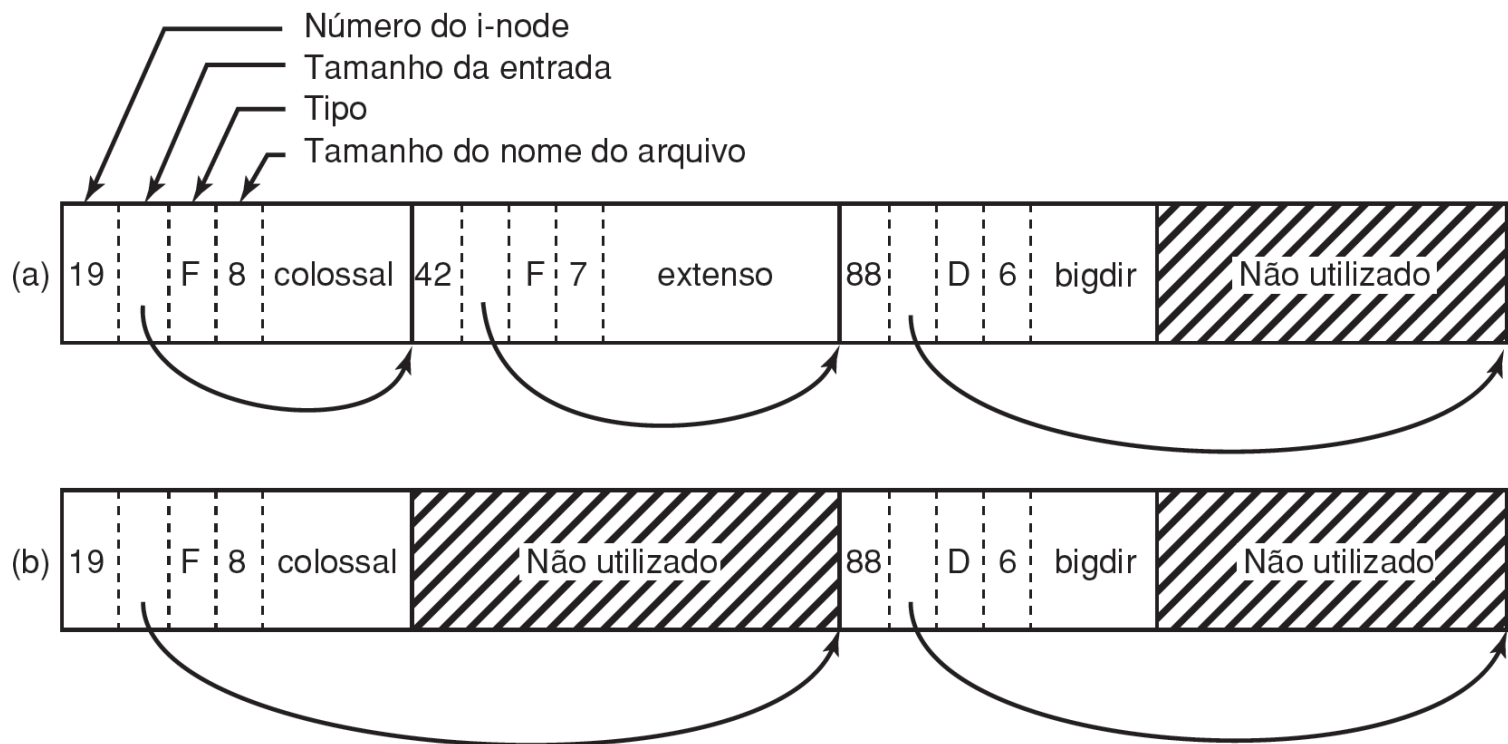
- Deletar arquivo “/dir1/picture.jpg”.
 - passo (4): mudar o tamanho da entrada do diretório anterior a “picture.jpg”.

Directory: /dir1

Inode #	Tam.	Filename
123	12	.
2	12	..
456	36	panic.c
894	20	picture.jpg
1024	16	hello.txt

novο tam. de “panic.c”
= tam. original de “panic.c” +
tam. original de “picture.jpg”
= 16 + 20
= 36.





■ **Figura 10.20** (a) Um diretório no Linux com três arquivos. (b) O mesmo diretório após a exclusão do arquivo *extenso*.

Como deletar um arquivo no Ext2/3

- observação:
 - A mudança no tamanho da entrada objetiva pular a entrada deletada.
 - Esta mudança produz **buracos** entre as entradas do diretório.
 - Quando um novo arquivo é criado neste diretório, o SA deve verificar se existe um buraco que é grande o suficiente para armazenar a nova entrada no diretório.
 - Aqui, o arquivo **não** é realmente **deletado**!
 - No Ext2, os ponteiros para blocos de dados não são removidos.
 - Já no Ext3, os ponteiros para blocos de dados serão removidos.