

# Sistemas de Ficheiros

Ficheiros

Diretórios

Implementação de sistemas de ficheiros

Exemplos de sistemas de ficheiros

# Armazenamento de Informação de Longo Prazo

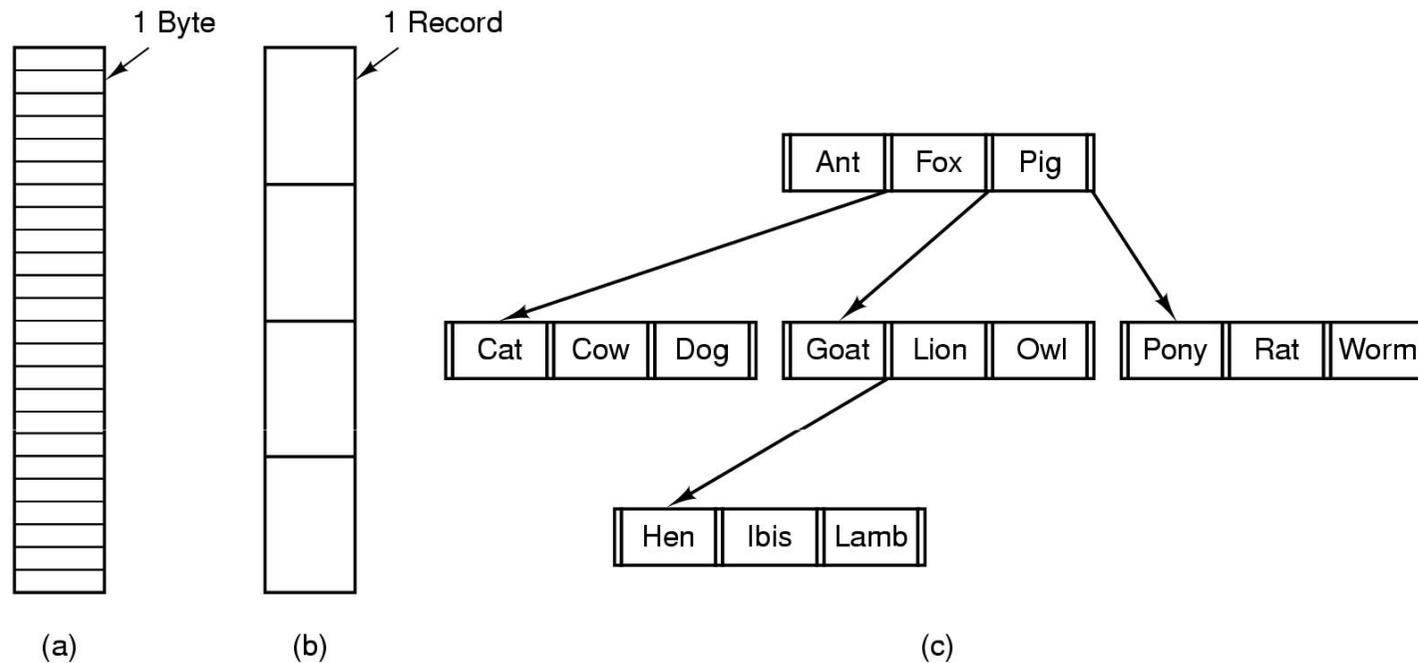
1. Deve armazenar grandes massas de dados
2. Informação armazenada deve ser persistente, isto é, deve se manter após o término do processo que a estava utilizando
3. Múltiplos processos devem poder aceder os dados de forma concorrente

# Nomes de ficheiros

<b>Extension</b>	<b>Meaning</b>
file.bak	Backup file
file.c	C source program
file.gif	CompuServe Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

## Extensões típicas de ficheiros

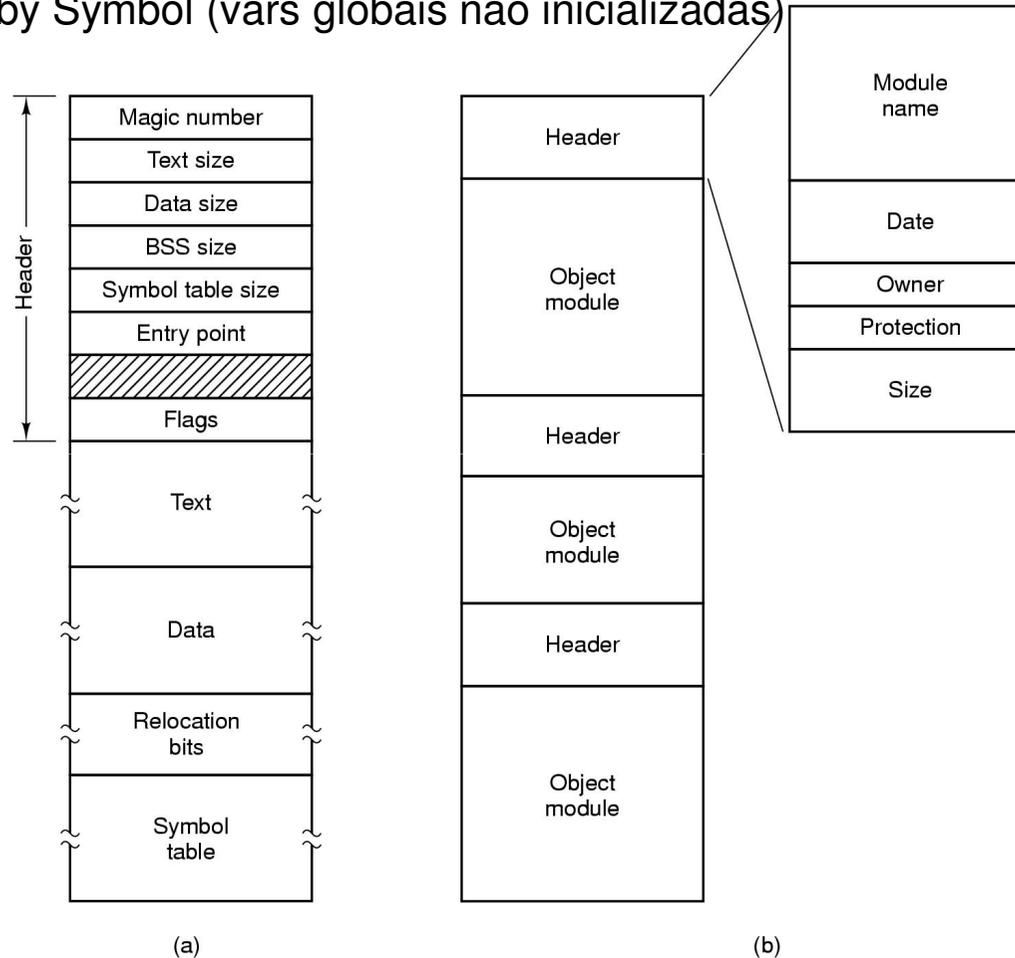
# Estrutura de ficheiros



- 3 tipos de ficheiros
  - Sequência de bytes
  - Sequência de registos
  - árvores

# Tipos de ficheiros

BSS = Block Started by Symbol (vars globais não inicializadas)



(a) Ficheiro executável (b) Um *archive*

# Métodos de acesso

- Sequencial
  - Lê todos os bytes/records a partir do início do ficheiro
  - Não pode ler qualquer byte sem passar pelos precedentes, pode voltar para trás
  - Conveniente quando o meio de armazenamento era fita magnética
- Aleatório
  - bytes/records são lidos em qualquer ordem
  - essencial para sistemas de bases de dados
  - Leitura pode ser ...
    - Mover marcador do ficheiro (seek), então ler ou ...
    - Ler e só então mover o marcador

# Atributos de ficheiros

<b>Attribute</b>	<b>Meaning</b>
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

## Atributos possíveis para um ficheiro

# Operações em ficheiros

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

# Exemplo de Programa usando chamadas ao sistema de ficheiros (1/2)

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>           /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]); /* ANSI prototype */

#define BUF_SIZE 4096           /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700       /* protection bits for output file */

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);     /* syntax error if argc is not 3 */
```

# Exemplo de Programa usando chamadas ao sistema de ficheiros (2/2)

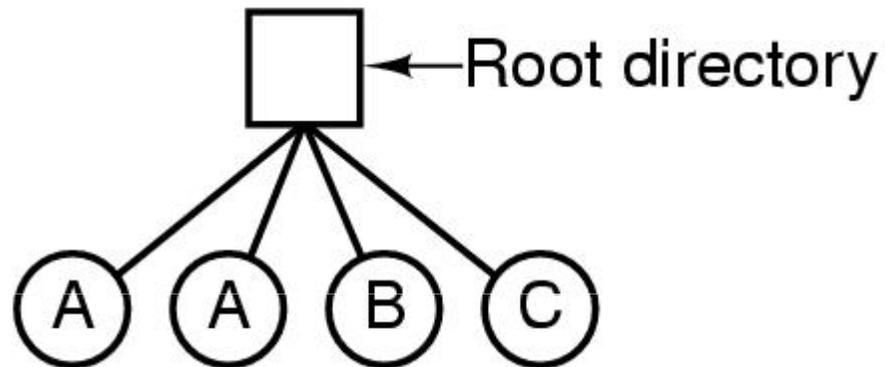
```
/* Open the input file and create the output file */
in_fd = open(argv[1], O_RDONLY); /* open the source file */
if (in_fd < 0) exit(2);          /* if it cannot be opened, exit */
out_fd = creat(argv[2], OUTPUT_MODE); /* create the destination file */
if (out_fd < 0) exit(3);        /* if it cannot be created, exit */

/* Copy loop */
while (TRUE) {
    rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
if (rd_count <= 0) break;                    /* if end of file or error, exit loop */
    wt_count = write(out_fd, buffer, rd_count); /* write data */
    if (wt_count <= 0) exit(4);              /* wt_count <= 0 is an error */
}

/* Close the files */
close(in_fd);
close(out_fd);
if (rd_count == 0)                          /* no error on last read */
    exit(0);
else
    exit(5);                                /* error on last read */
}
```

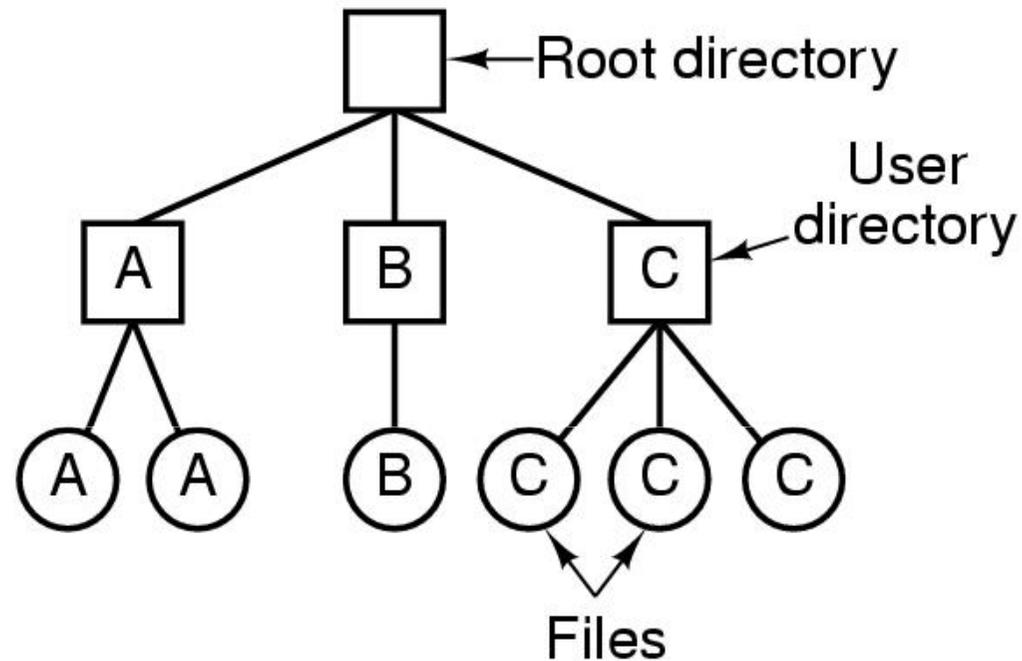
# Diretórios

## Sistemas de diretório com um único nível



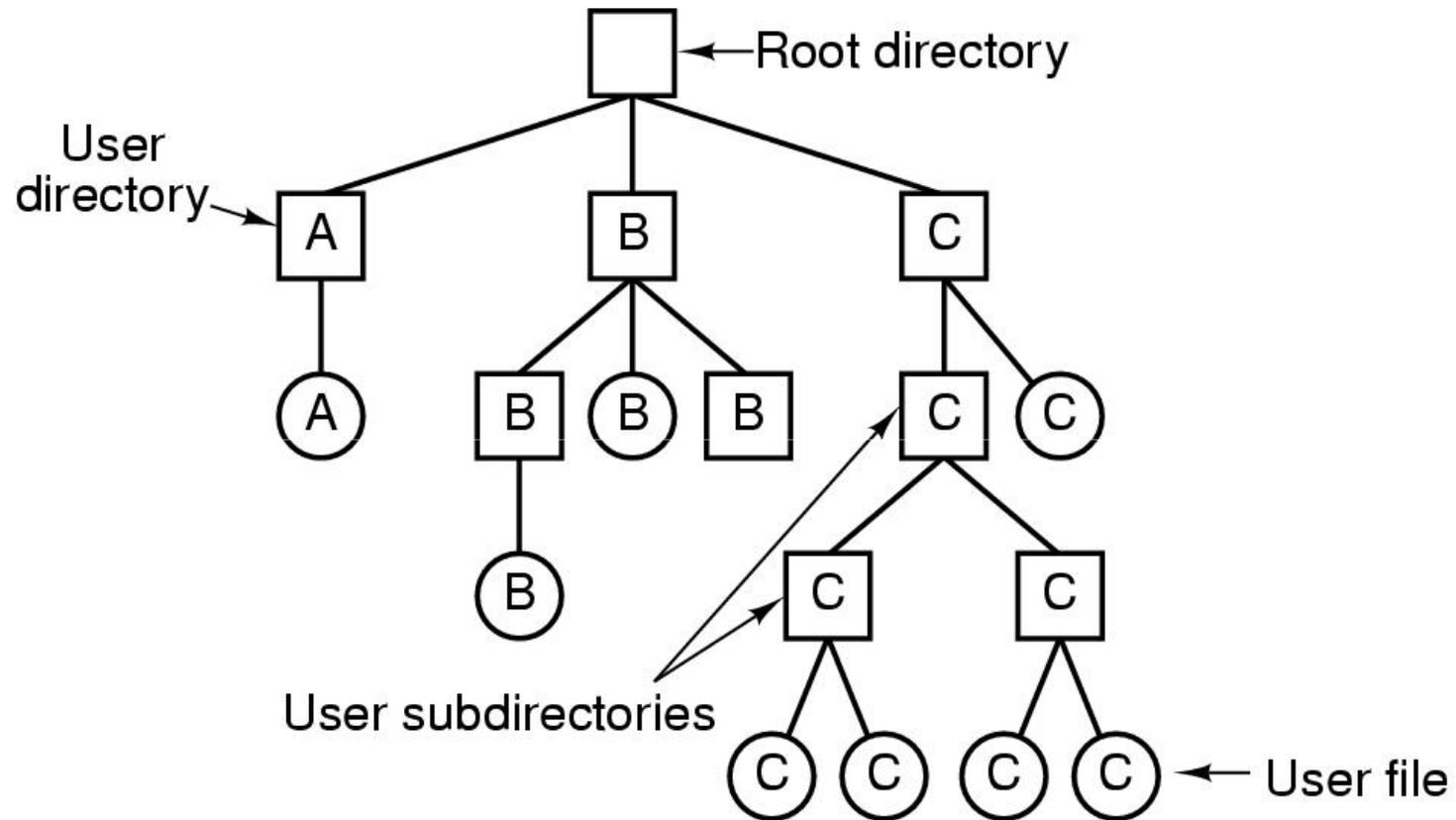
- Exemplo de sistema de diretório com um nível
  - contém 4 ficheiros...
  - ...que pertencem a 3 pessoas diferentes: A, B, and C

# Sistemas de diretórios de 2 níveis

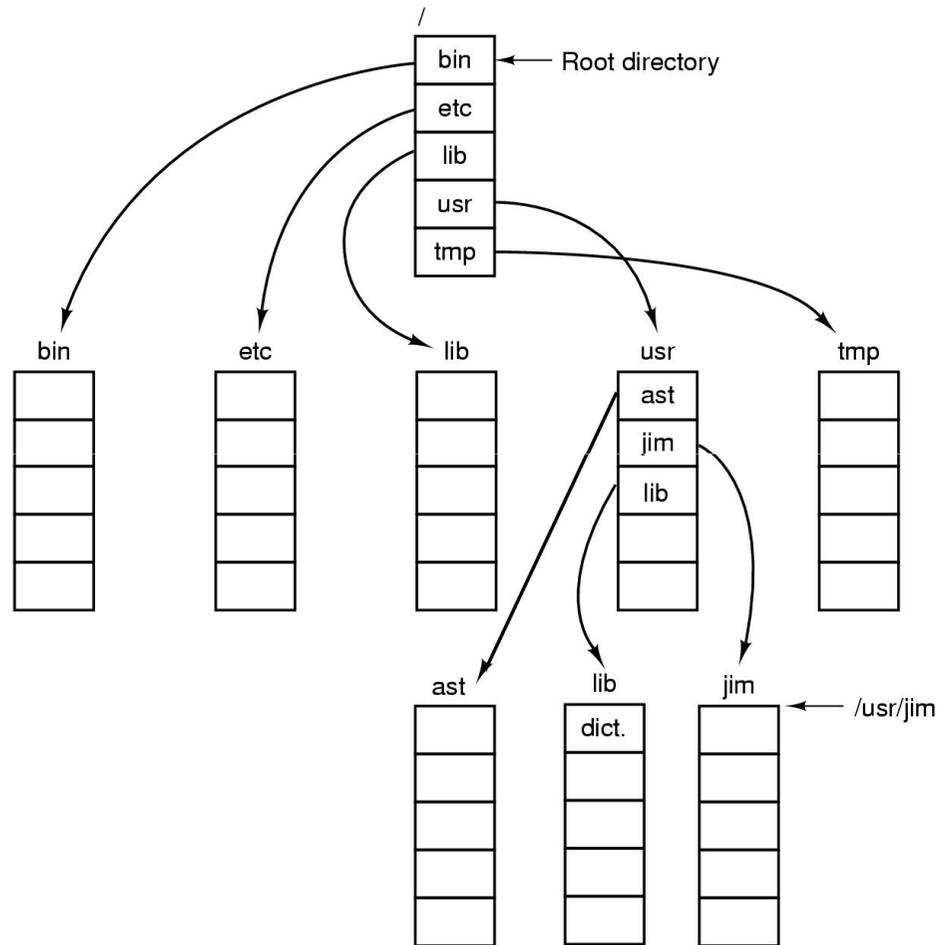


Letras indicam os donos dos diretórios e ficheiros

# Sistemas hierárquicos de diretórios



# Nomes dos caminhos em sistemas com organização hierárquica de diretórios



Uma árvore de diretórios UNIX

# Operações com Diretórios

1. Create

2. Delete

3. Opendir

4. Closedir

5. Readdir

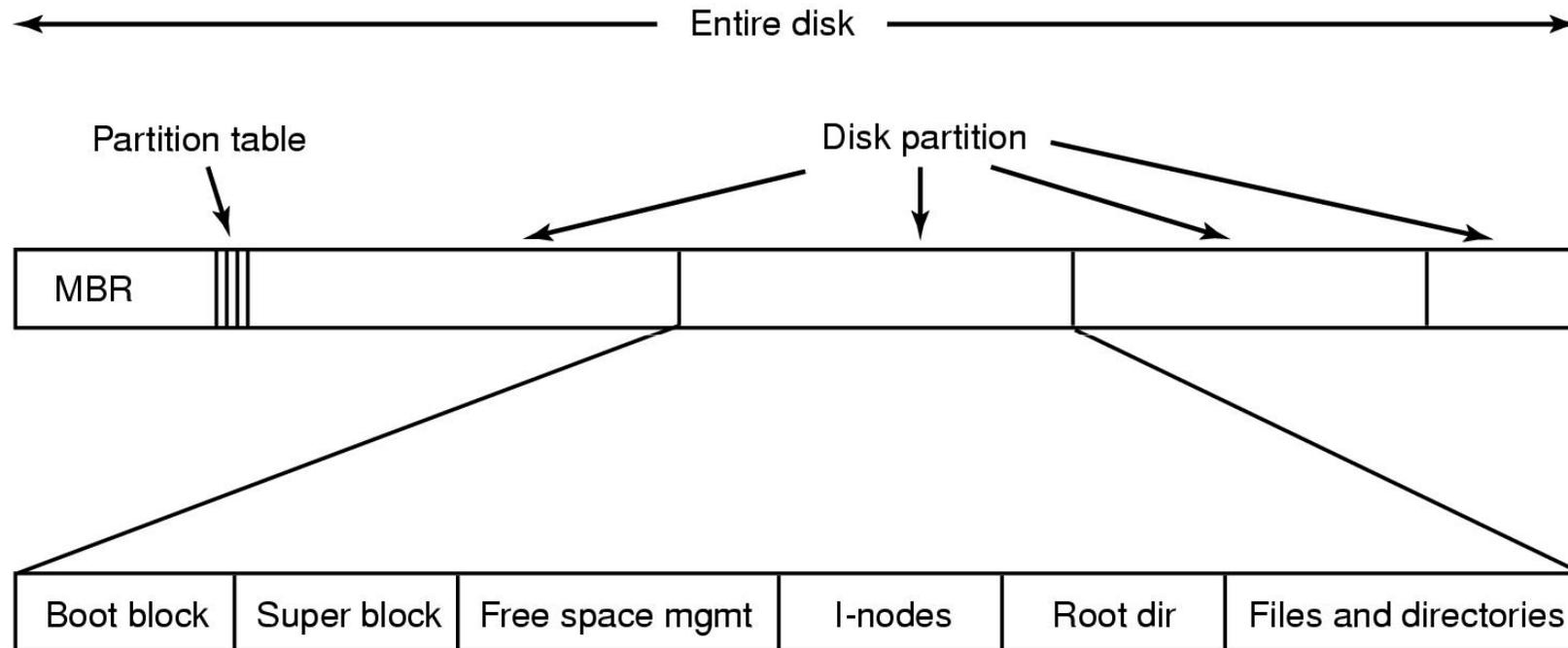
6. Rename

7. Link

8. Unlink

# Implementação de sistemas de ficheiros

MBR = Master Boot Record

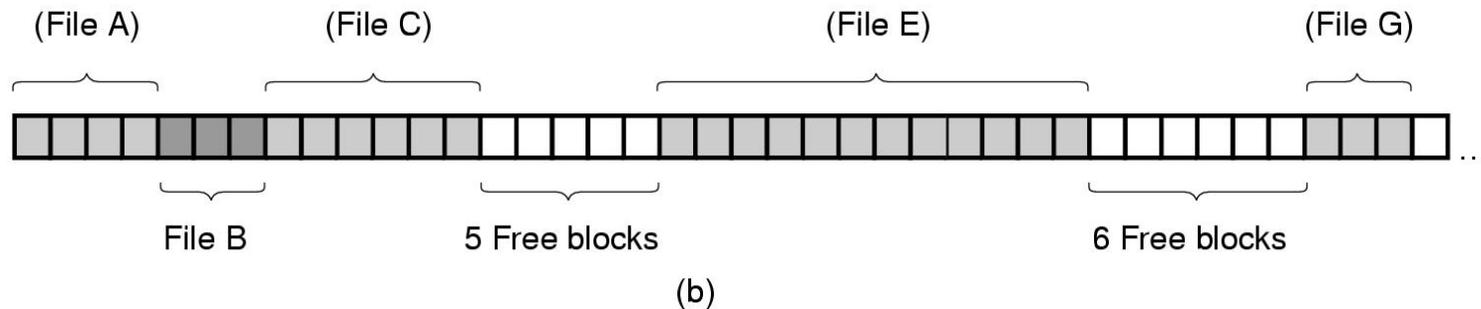
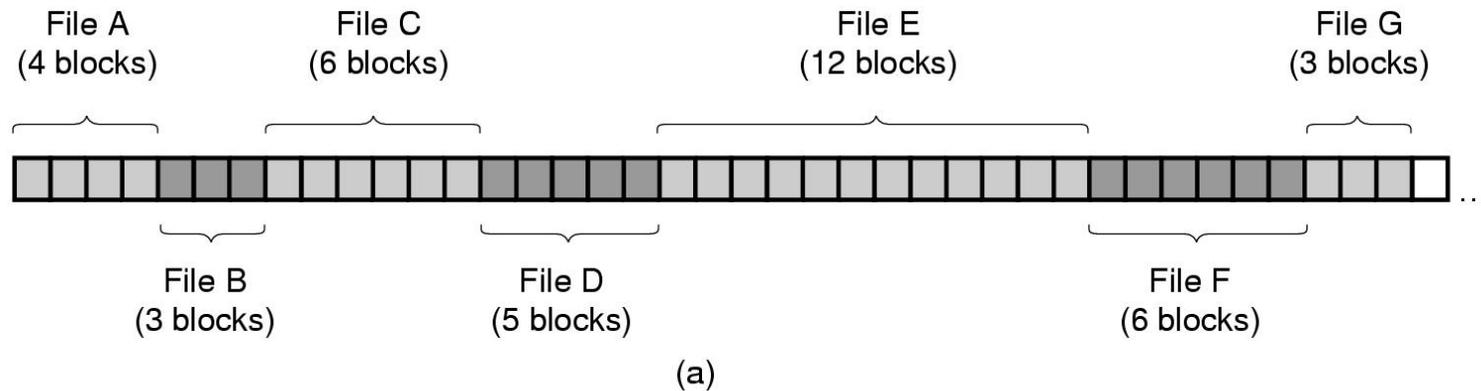


Um layout possível para um sistema de ficheiros

# Implementação de sistemas de ficheiros

- Organização dos ficheiros
  - Alocação contígua
  - Alocação em listas encadeadas

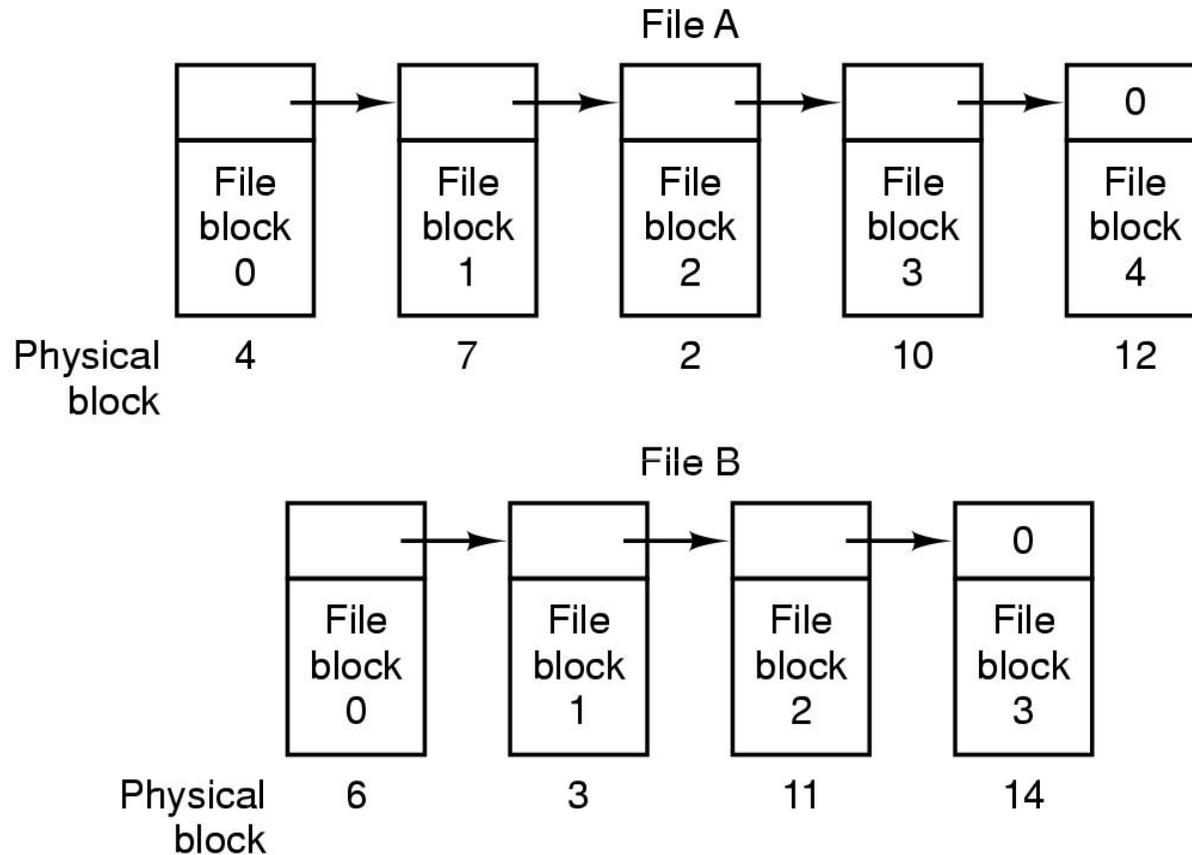
# Implementação de ficheiros (1)



(a) Alocação Contígua de espaço para 7 ficheiros

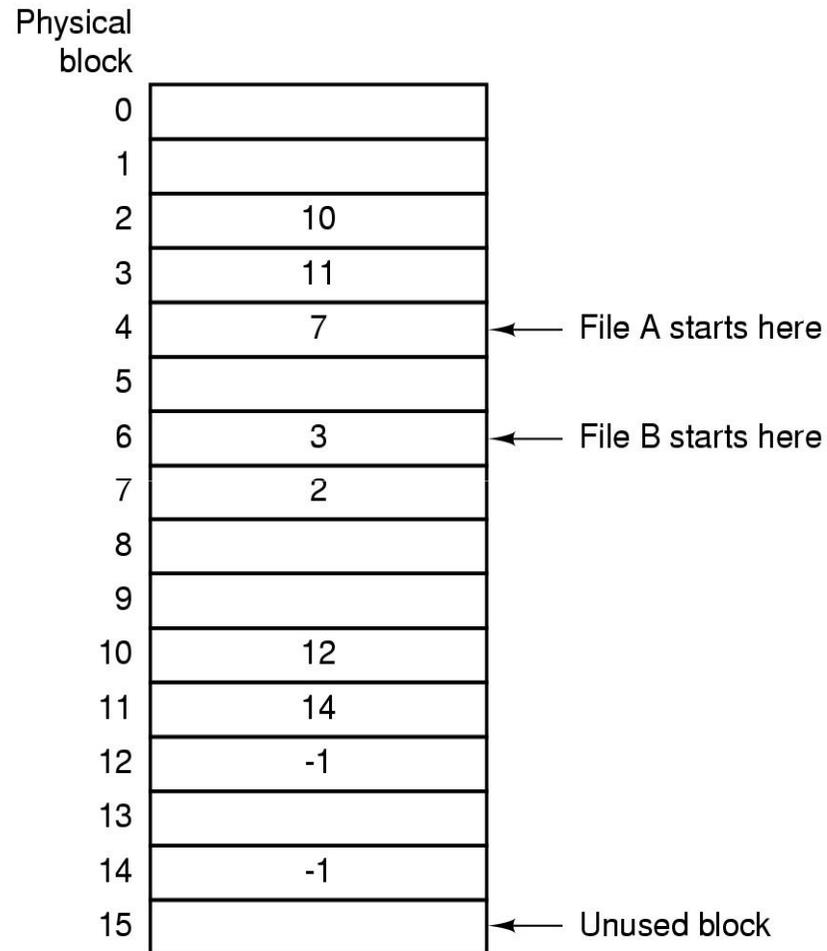
(b) Estado do disco após ficheiros *D* e *E* terem sido removidos

# Implementação de ficheiros (2)



Armazenamento de ficheiros utilizando listas encadeadas de blocos do disco

# Implementação de ficheiros (3)

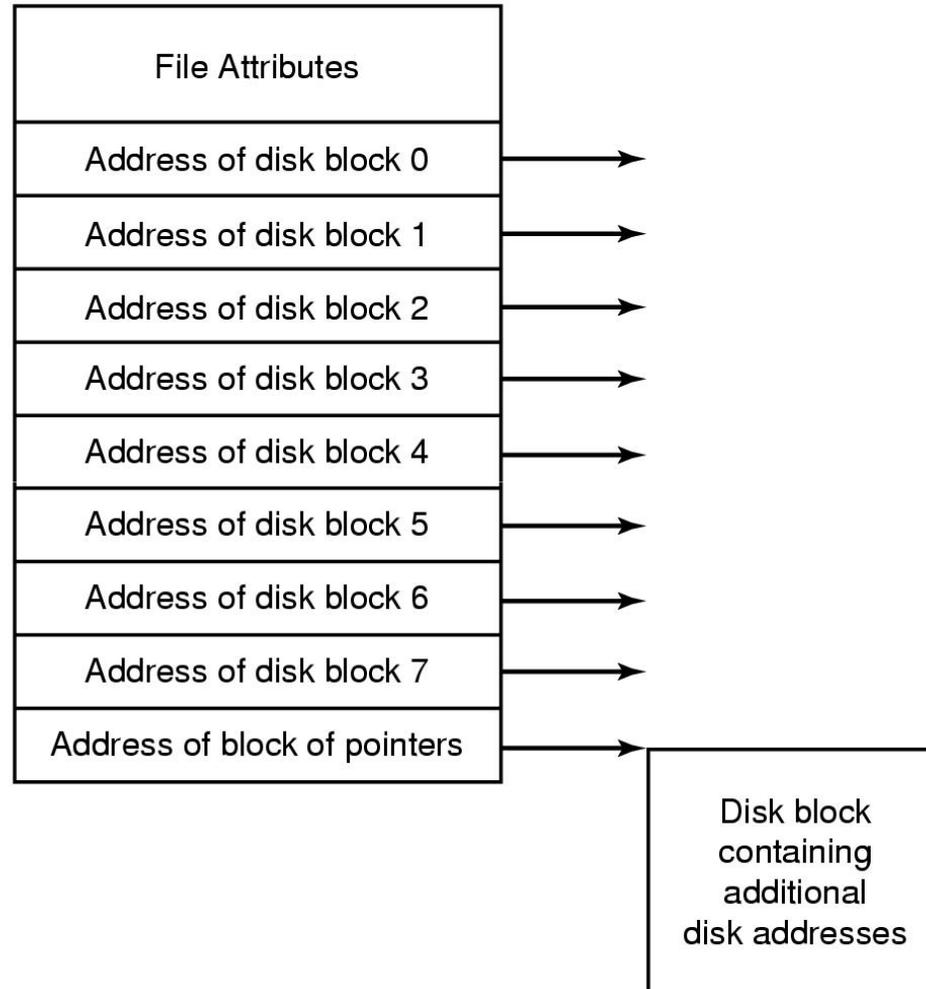


Implementação de listas encadeadas utilizando uma File Allocation Table (FAT) em RAM

# Implementação de ficheiros (4)

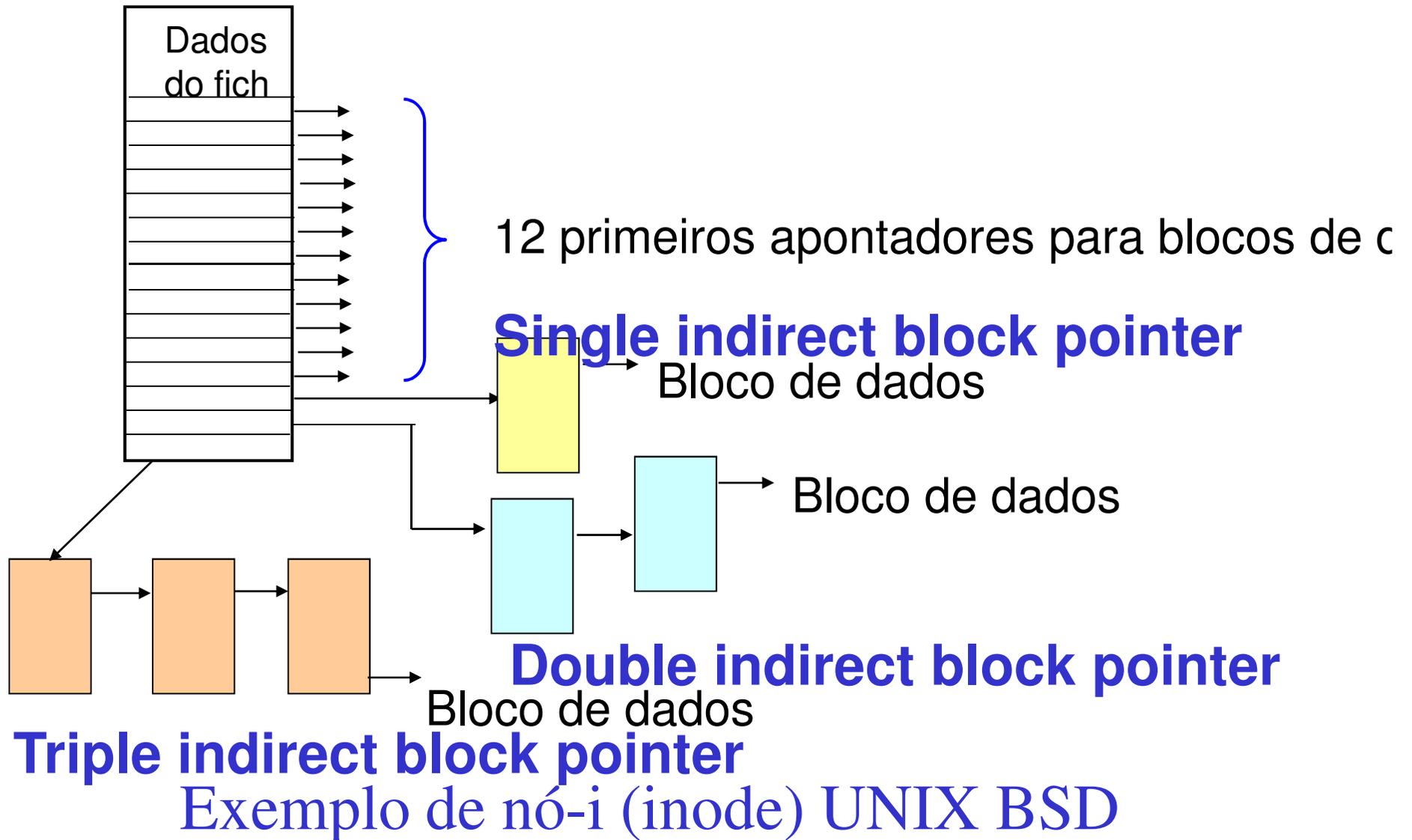
- Tamanho da FAT depende do seu tipo:
  - 8, 10, 12, 16, 32.
- Número de entradas numa FAT:
  - $(2^{\text{tipo\_fat}}) - 1$

# Implementação de ficheiros (5)



Exemplo de nó-i (inode) usado em UNIX

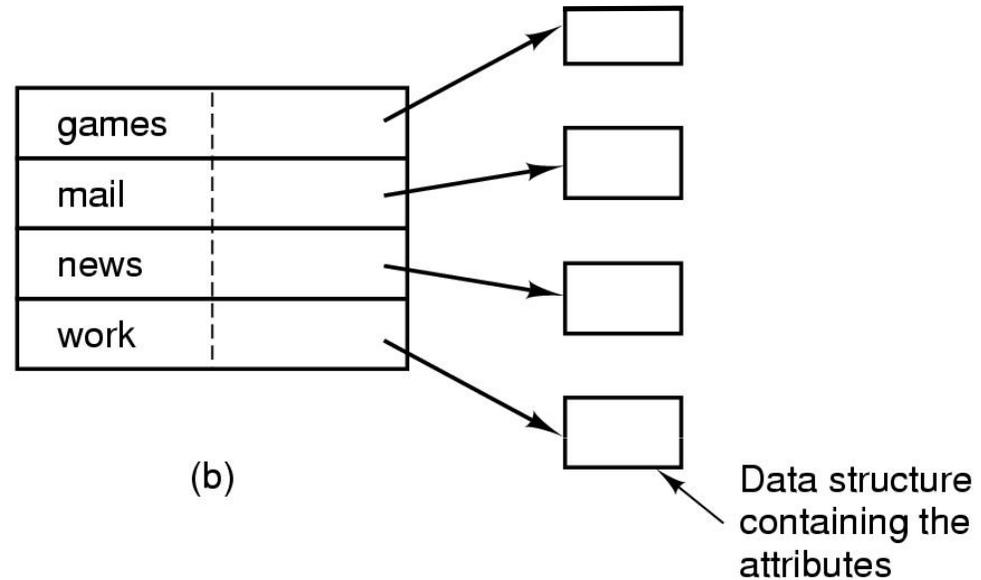
# Implementação de ficheiros (6)



# Implementação de diretórios (1)

games	attributes
mail	attributes
news	attributes
work	attributes

(a)



(b)

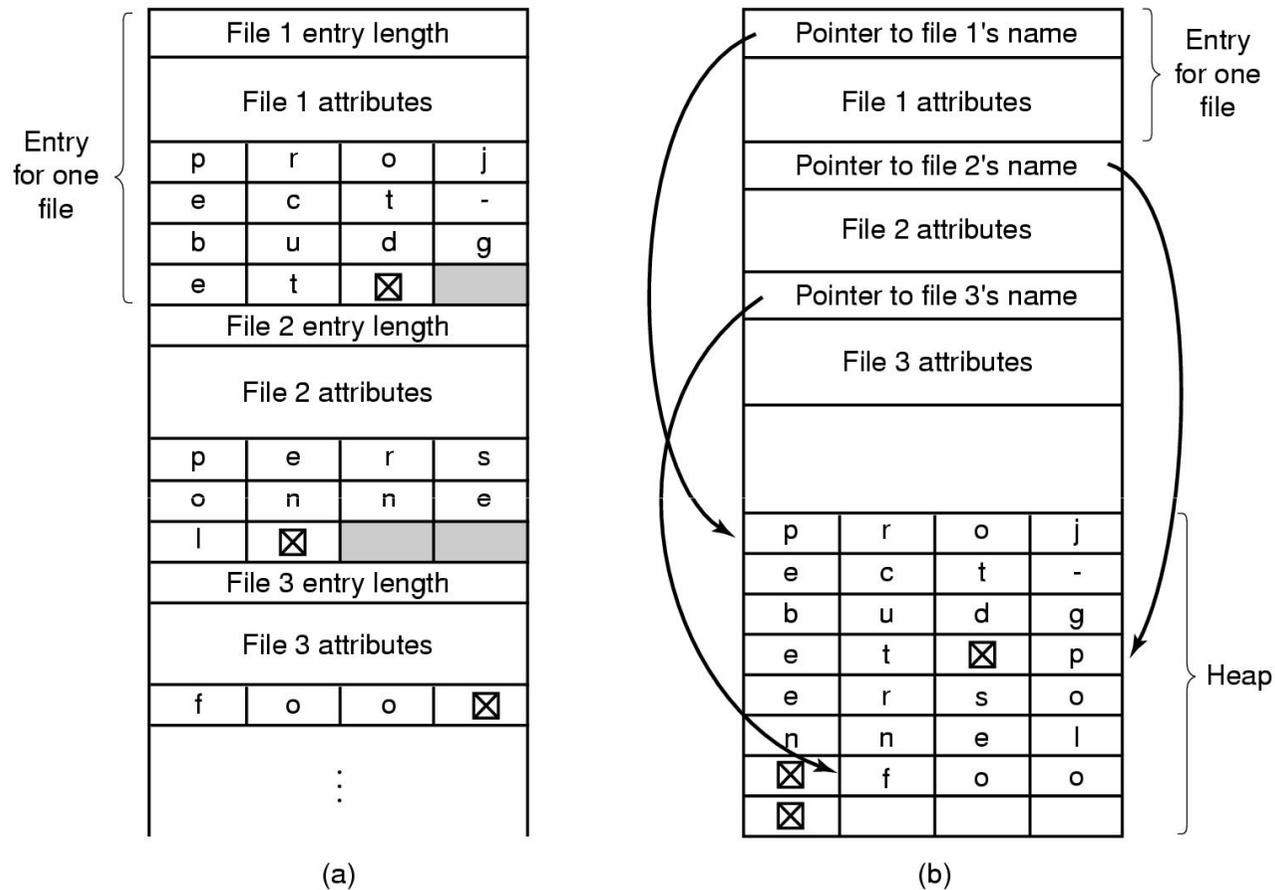
## (a) diretório simples

Entradas de tamanho fixo

Endereços e atributos na entrada de diretório

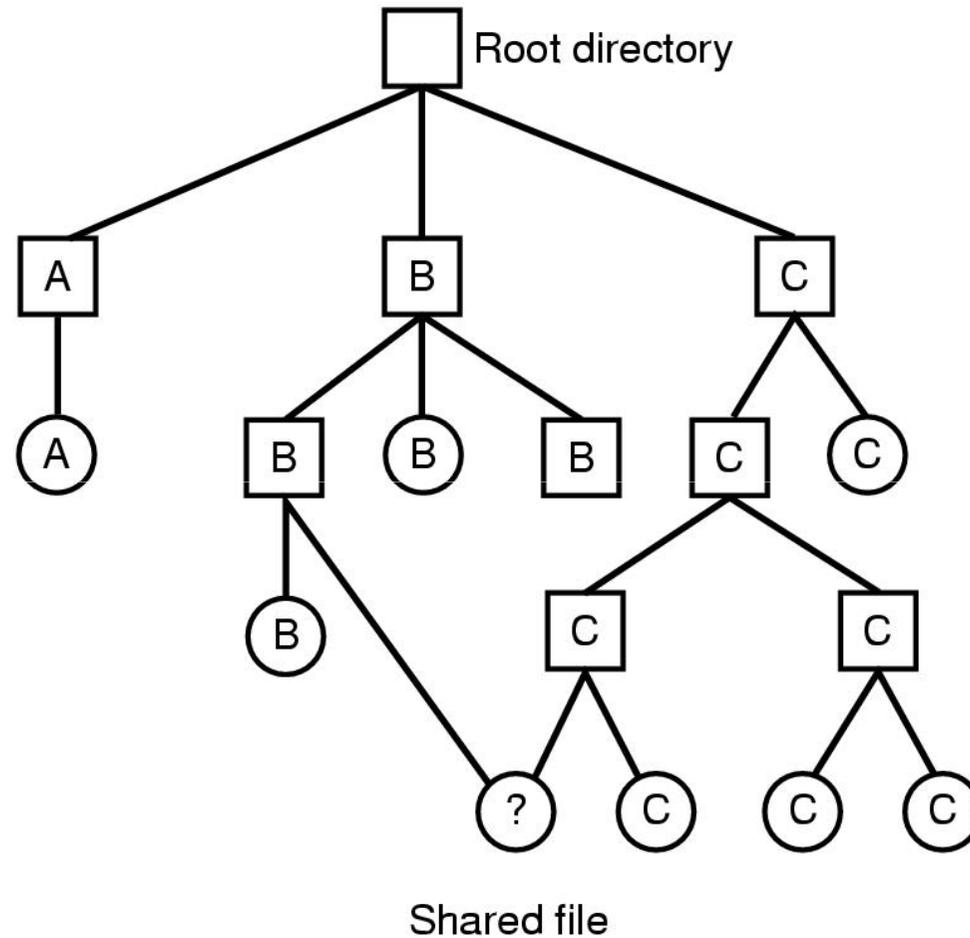
## (b) Diretório onde cada entrada refere a um nó-i

# Implementação de Diretórios (2)



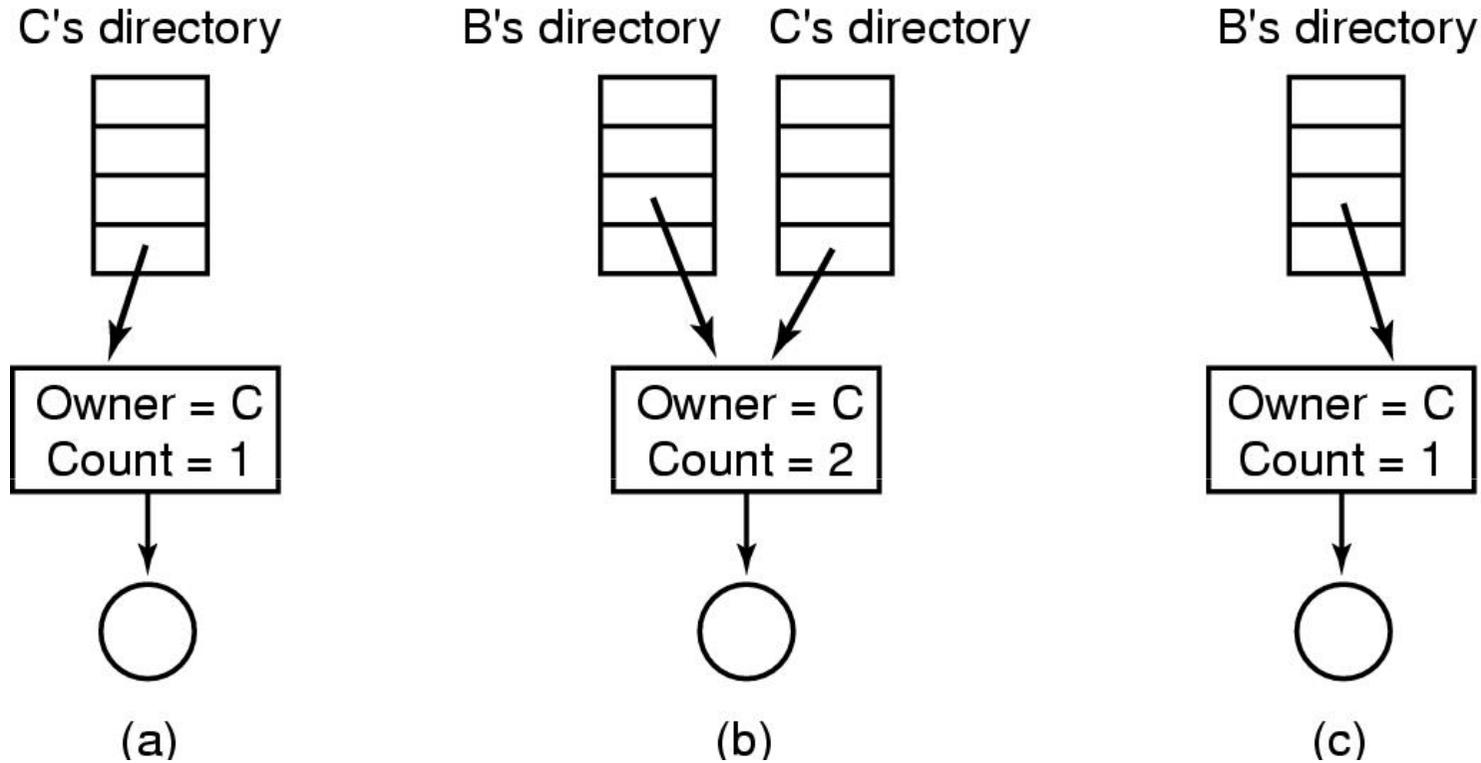
- Como armazenar nomes de diretórios muito longos?
  - (a) In-line
  - (b) In a heap

# Ficheiros compartilhados (1)



Sistema de ficheiros contendo um ficheiro compartilhado

# Ficheiros compartilhados (2)

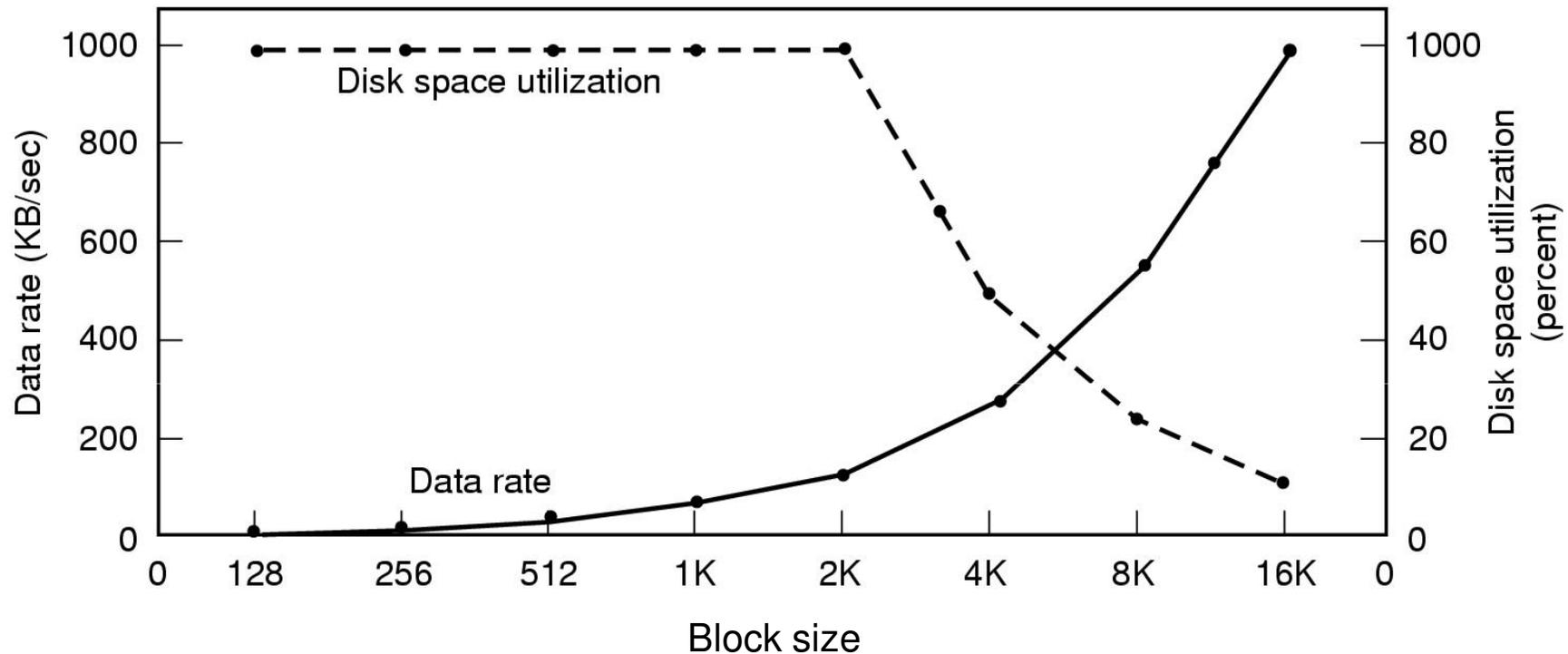


(a) Situação antes do link

(b) Após o link ser criado

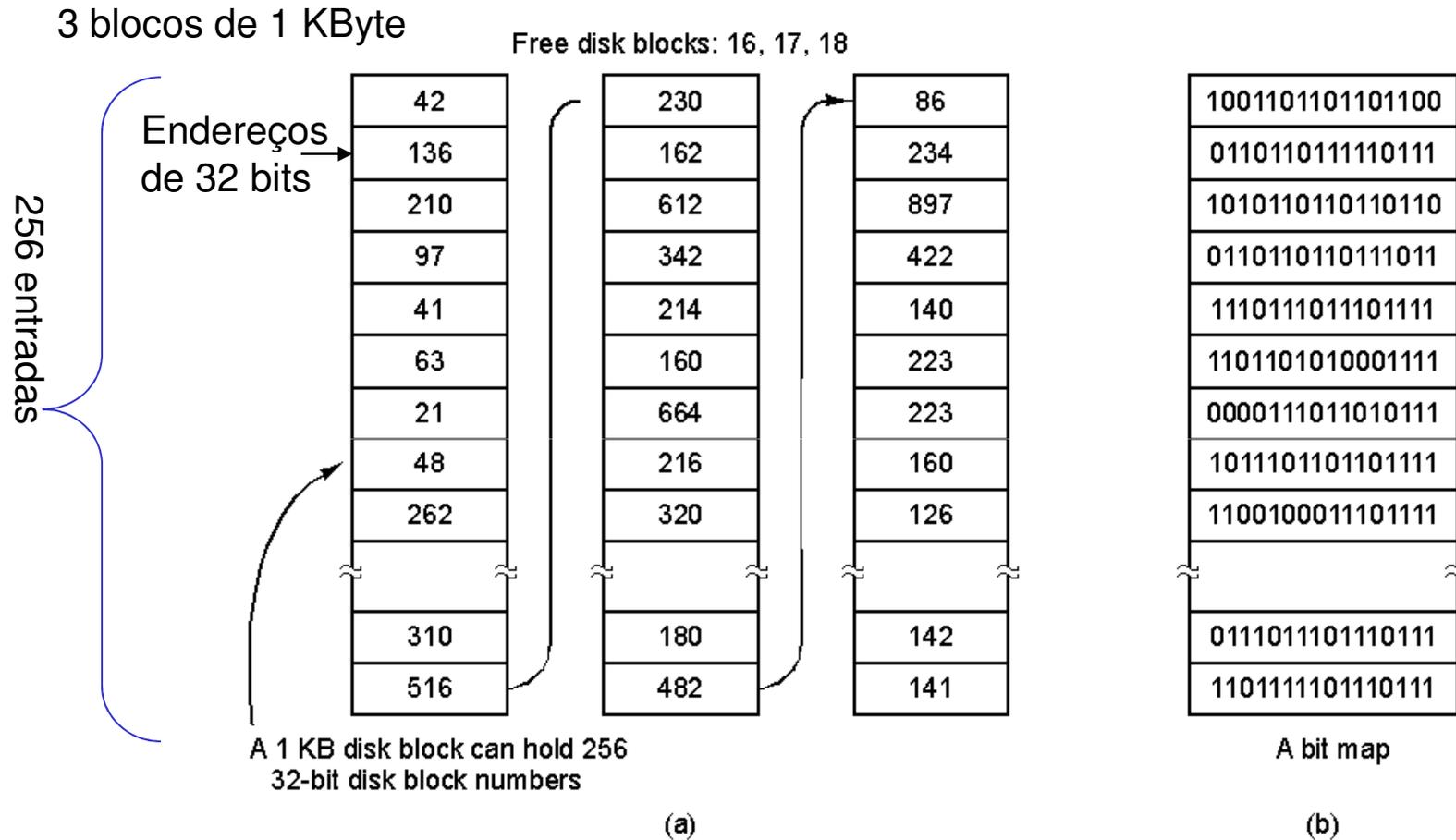
(c) Após fich original ser removido pelo dono

# Gestão de espaço em disco (1)



- Linha cheia (escala à esq) indica taxa de transf do disco
- Linha pontilhada (escala à dir) indica eficiência
- Todos os fichs 2 KBytes

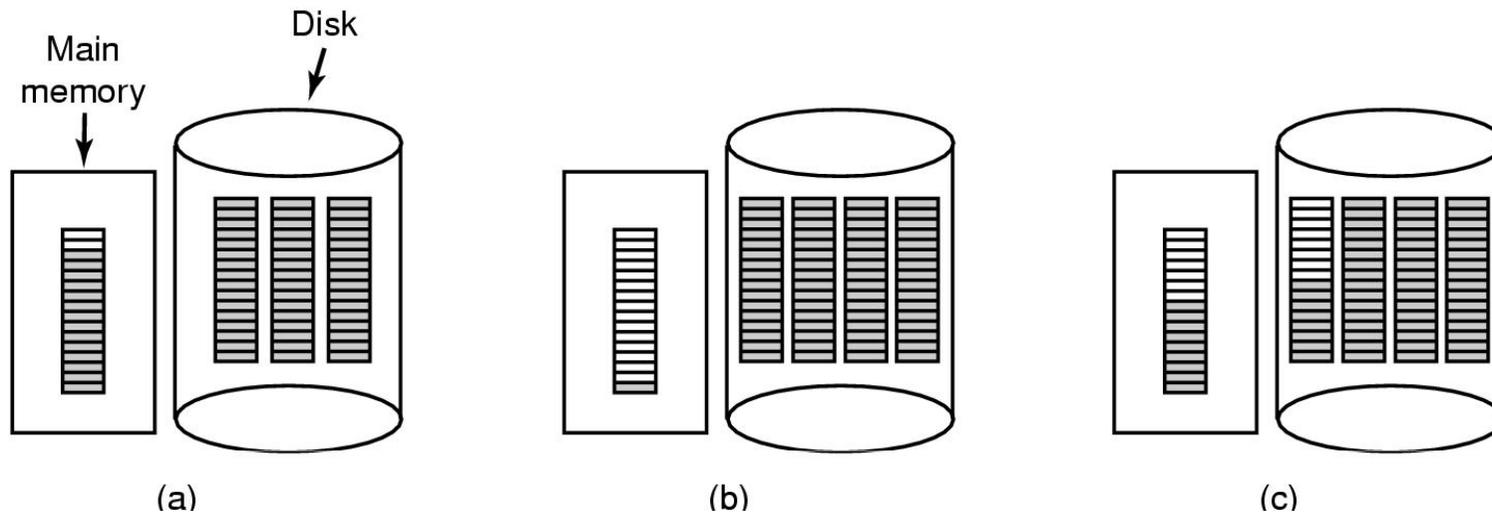
# Gestão de espaço em disco (2)



(a) Armazenando blocos livres numa lista encadeada

(b) Armazenando em um bitmap

# Gestão de espaço em disco (3)



(a) Bloco com apontadores para blocos livres no disco em memória RAM, quase cheio

- Três blocos de apontadores no disco

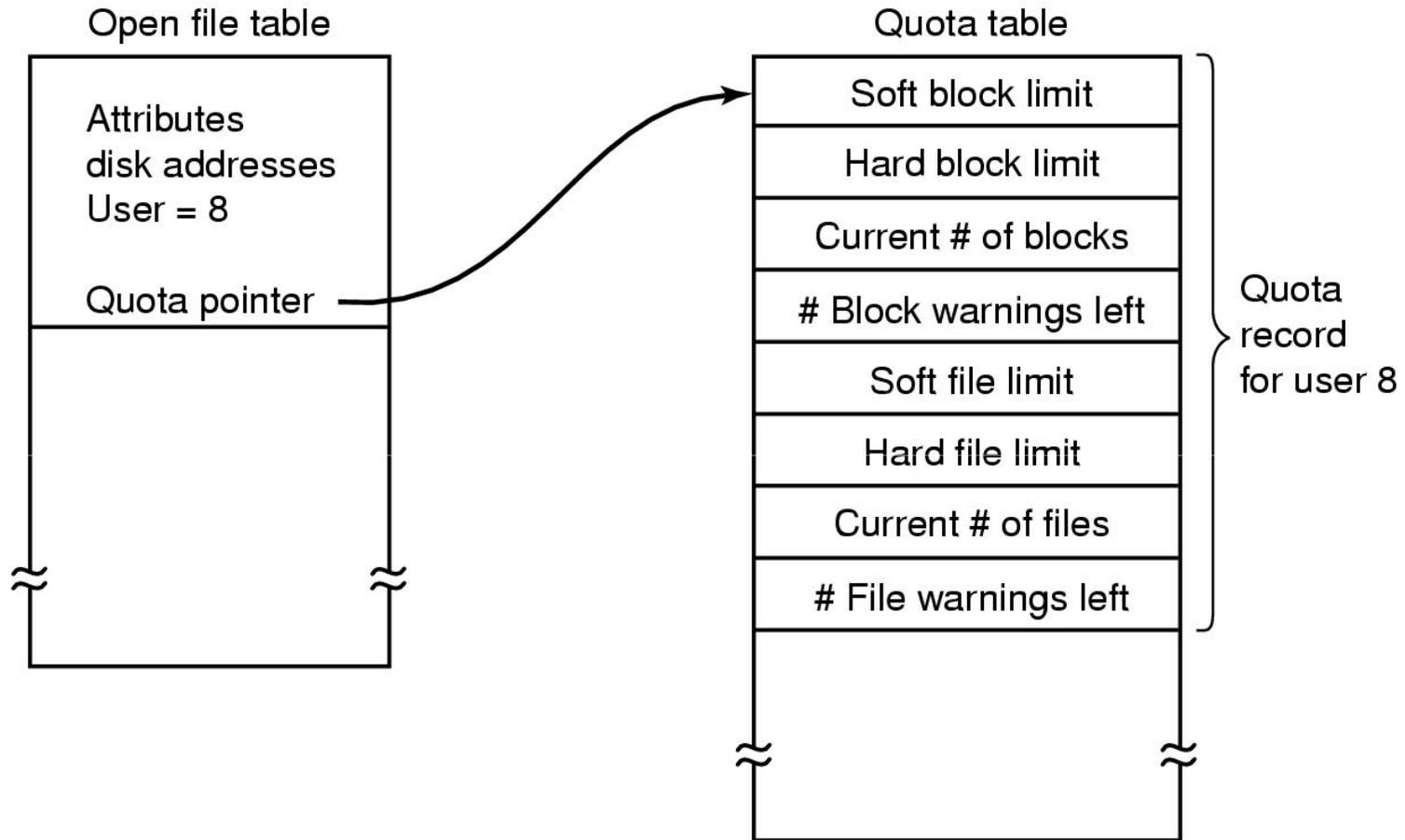
(b) Resultado após libertar um ficheiro de 3 blocos

(c) Estratégia alternativa para manipular os três blocos livres

– Alguns apontadores para blocos livres ficam já em memória

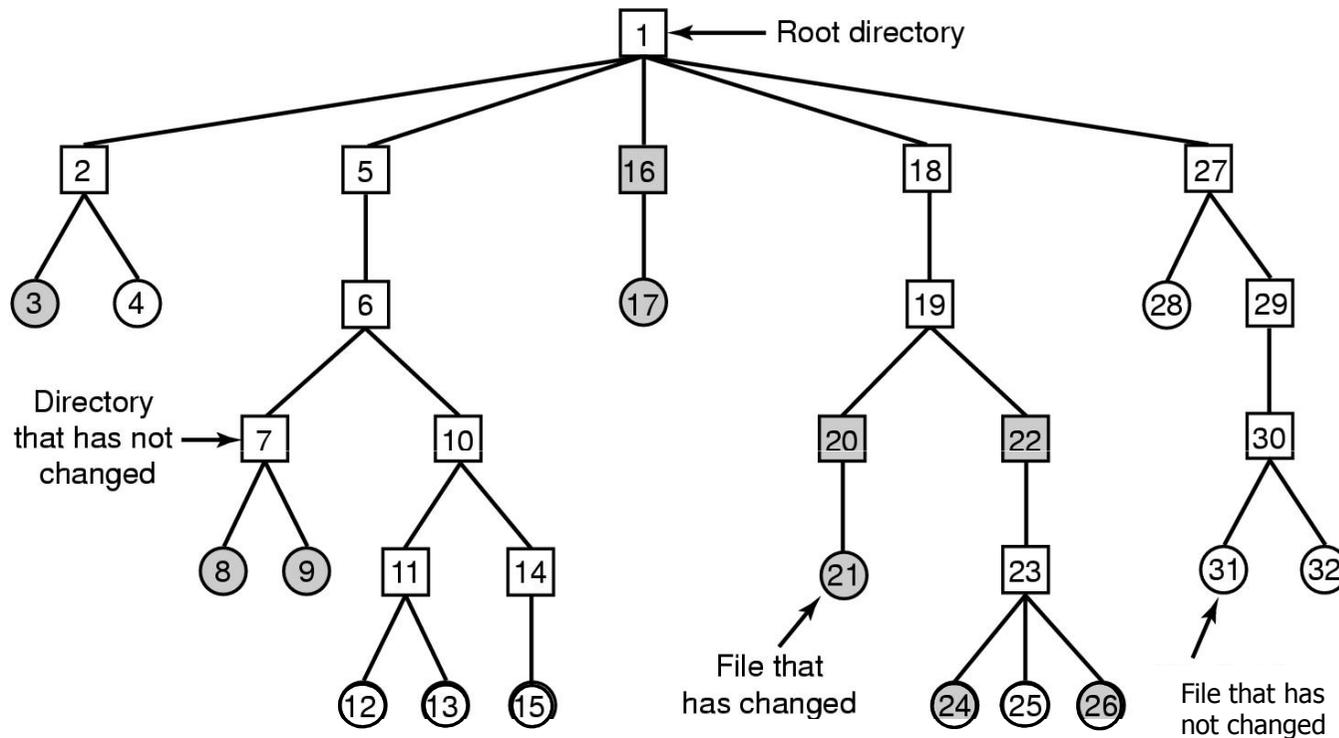
- Entradas em cinza são ponteiros para blocos livres

# Gestão de espaço em disco (4)



Esquema de quotas para gerir espaço consumido por utilizador

# “Backups” do sistema de ficheiros (1)



- Sistema de ficheiros a ser “dumped”
  - Quadrados são diretórios, círculos são ficheiros
  - Itens em cinza, modificados desde o último backup
  - Cada dir e fich com seu número de nó-i

# “Backups” do sistema de ficheiros (2)

(a) 

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

(b) 

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

(c) 

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

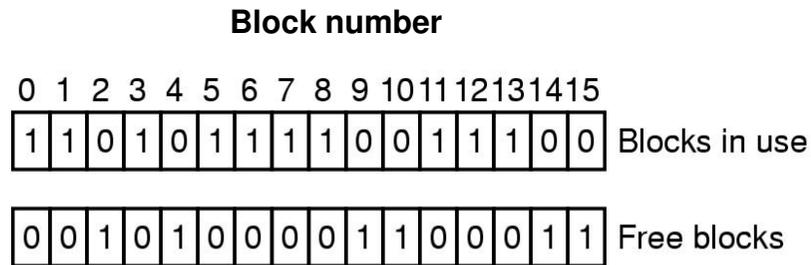
(d) 

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

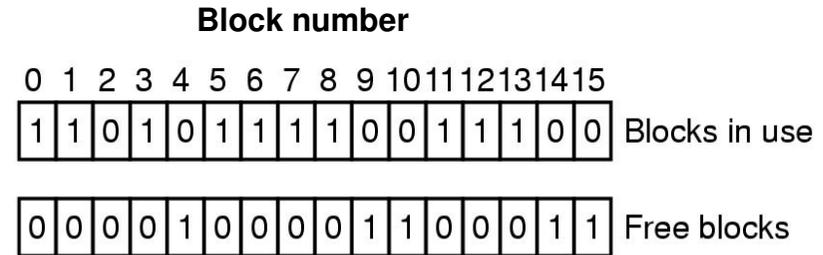
- (a) Ficheiros que não foram modificados (4, 12, 13...)
- (b) Começando pela raiz, nós-i de sub-árvores que foram modificadas (1,2, 3,...)
- (c) Diretórios modificados (1, 2, 5, 6,...)
- (d) Ficheiros modificados (3, 8, 9,...)

Bitmaps usados pelo algoritmo de “dumping” lógico

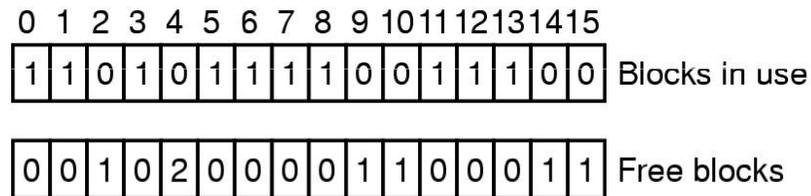
# Consistência do sistema de ficheiros(1)



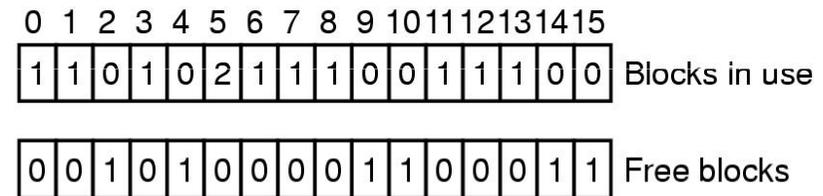
(a)



(b)



(c)



(d)

- Estados do sistema de ficheiros

(a) consistente

(b) bloco perdido (2)

(c) bloco duplicado na lista de blocos livres (4)

(d) Bloco de dados utilizado mais de uma vez (5)

# Consistência do sistema de ficheiros (2)

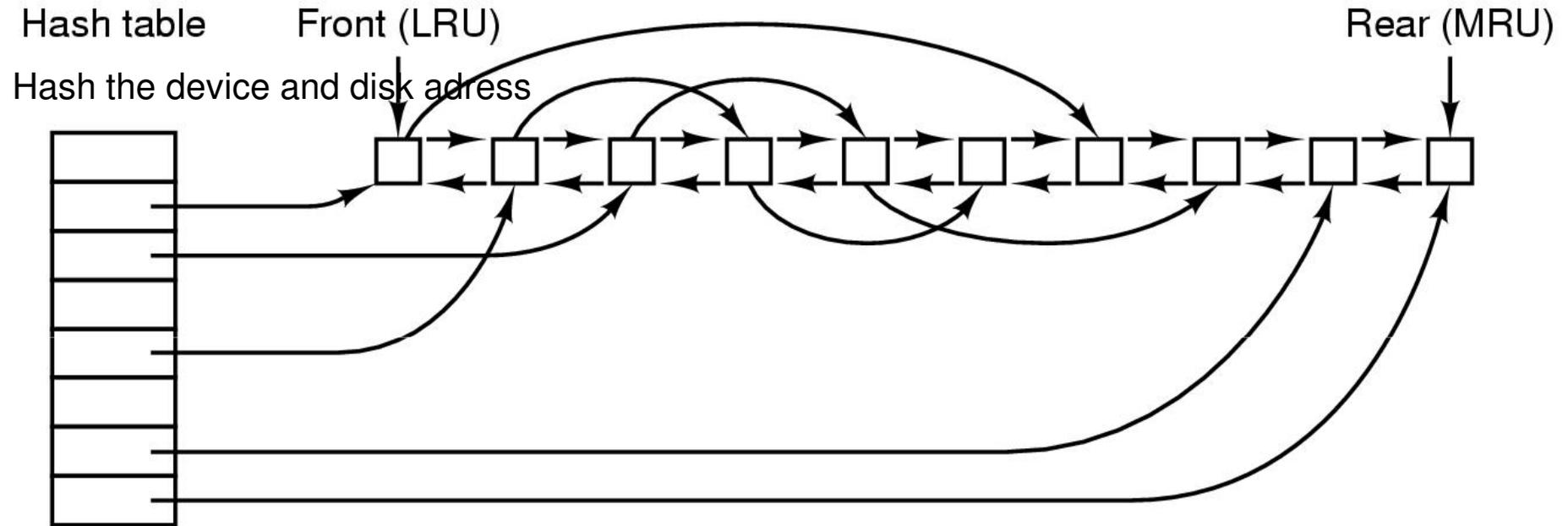
**FCK em Unix:**

**Compara os blocos em ficheiros com a lista de blocos livres. Todo bloco deve estar contado exatamente 1 vez.**

**Se este não for o caso:**

- bloco não aparece em lugar algum: ligar à lista de blocos livres.
- bloco aparece na lista de blocos livres e em um ficheiro: remover da lista de blocos livres.
- Aparece mais de uma vez em ficheiros – situação muito má!  
– associar o bloco a um dos ficheiros e notificar.

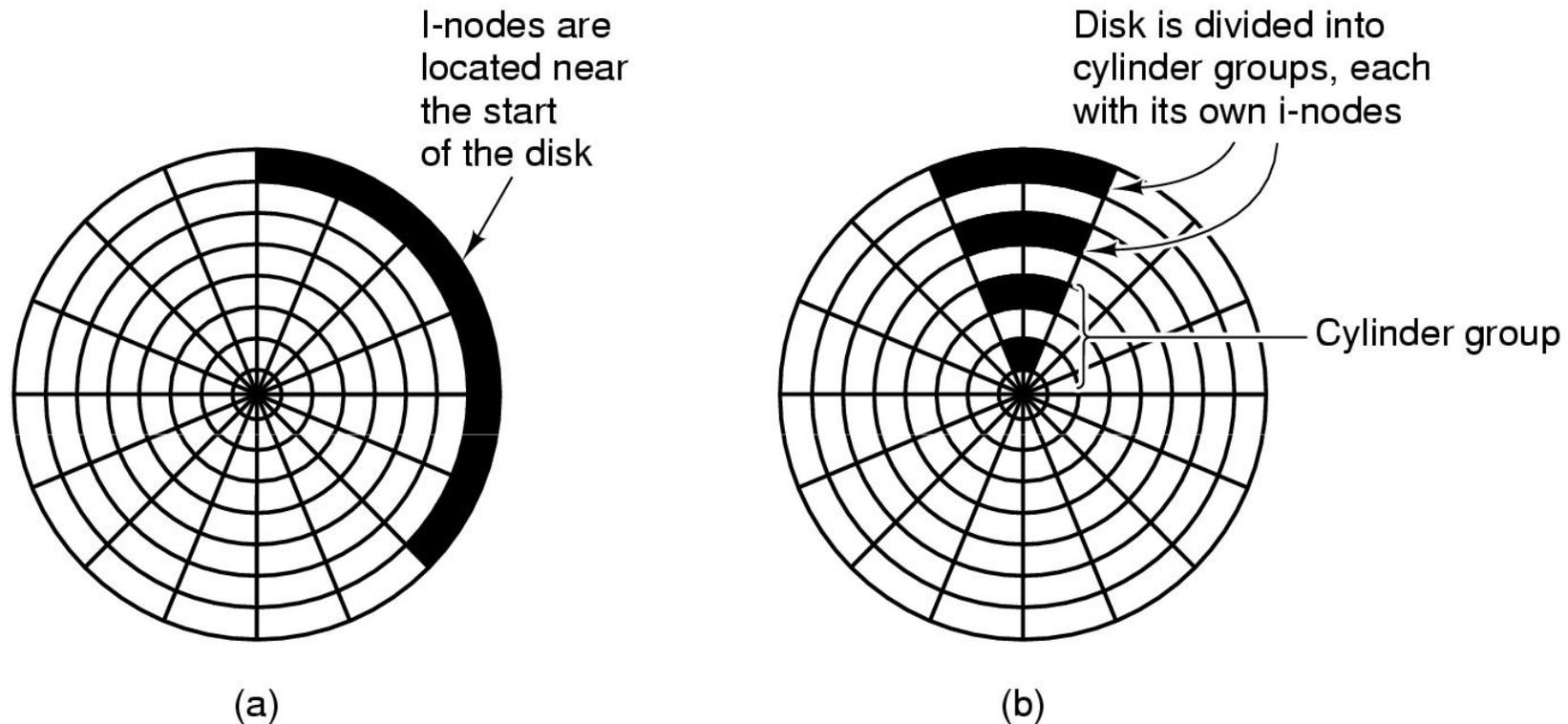
# Desempenho do sistema de ficheiros (1)



## Estruturas de dados do "buffer (block) cache"

- caching
- leitura do próximo bloco
- redução do movimento do braço do disco

# Desempenho do sistema de ficheiros (2)



- (a) Inodes colocados no início do disco
- (b) Disco dividido em grupos de cilindros
  - Cada qual com seus blocos e inodes

# Sistemas de ficheiros “log-structured”

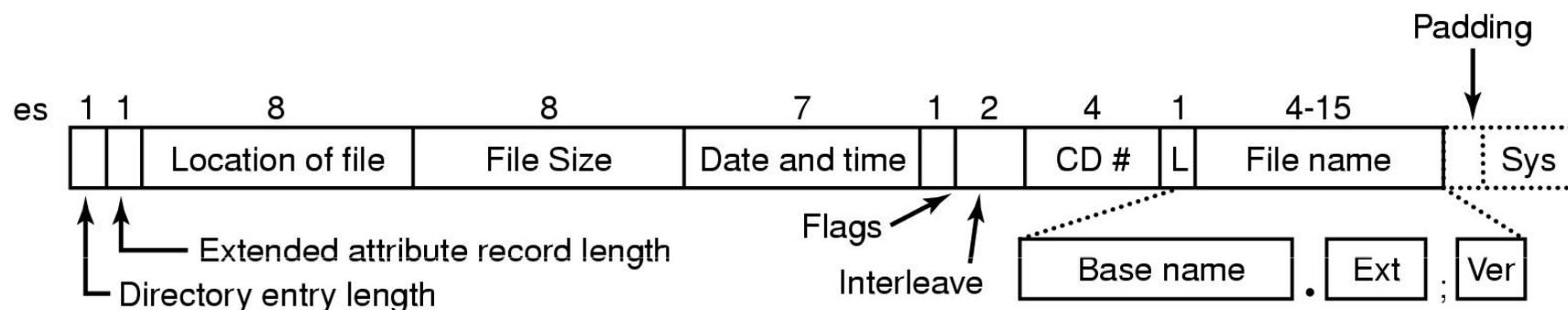
- Com CPUs mais rápidas e memórias maiores
  - Caches de disco podem também ser grandes
  - Número maior de pedidos pode ser resolvido pela cache
  - ...o que indica que a maior parte de acessos a disco será para escritas
- LFS (Log-structured File System) organiza o disco como se fosse um log
  - Escritas são guardadas em memória (buffered)
  - Escreve periodicamente no fim do log
  - Inodes, blocos de dados, diretórios etc estão todos misturados ocupando um segmento
  - Qdo fich é aberto, localiza i-node, e os blocos

# Sistemas de ficheiros “journaling”

- Utilizados para manter sistemas de ficheiros consistentes até um determinado momento
- NTFS (Win), Linux ext2, ReiserFS
- procedimento
  - Anota as três ações de remoção de ficheiros em log:
    - Remover a entrada do diretório
    - Libertar o nó-i (ligar à lista de livres)
    - Libertar blocos de dados do ficheiro
  - Escreve o log no disco
  - Pode então começar as ações
  - Se bem sucedido: apaga o log
  - Senão, verifica o log para repetir as ações

# Exemplos de sistemas de ficheiros

## CD-ROM File Systems

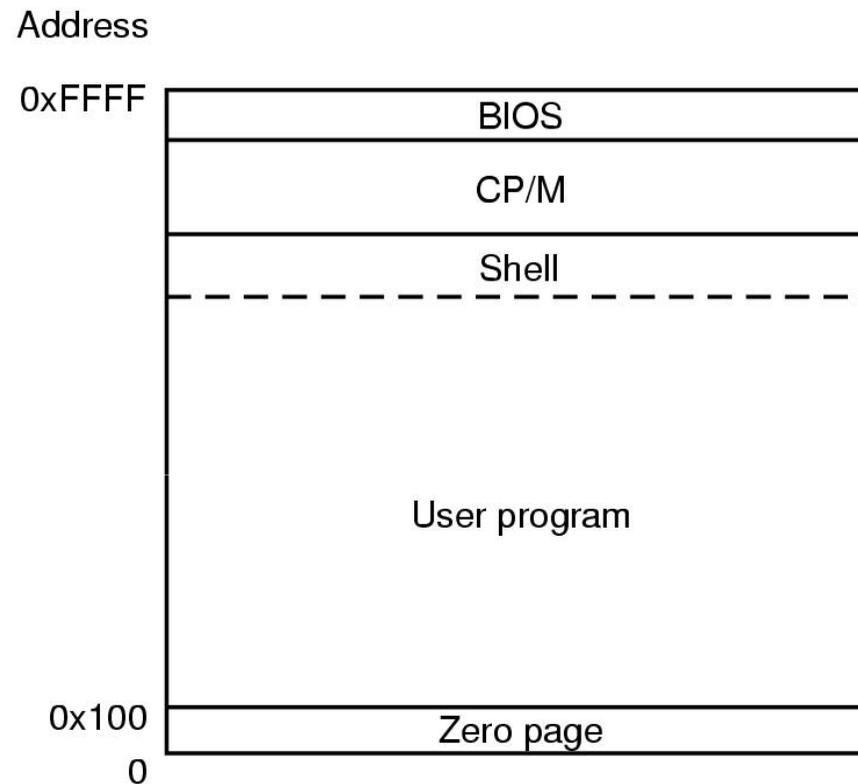


### Organização física:

- espiral de bits
- movimento através da espiral possível
- bits divididos em blocos (setor lógico) de tamanho 2532 bytes
- posição de um bloco: minutos e segundos, 1sec = 75 blocos
- CD para música tem extensões (leadin, leadout, intertrack gaps etc)

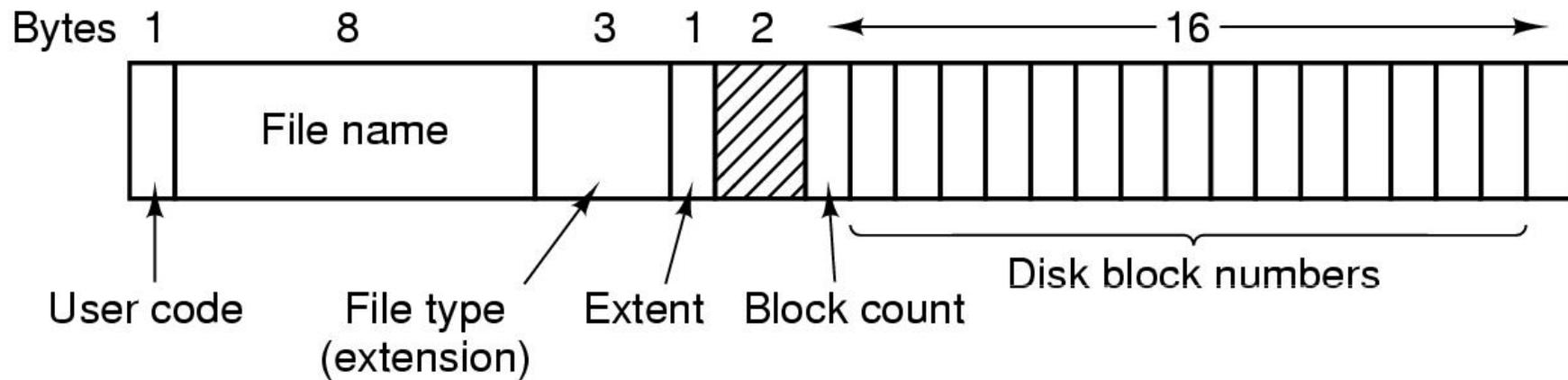
## Entrada de diretório do padrão ISO 9660

# CP/M (1)



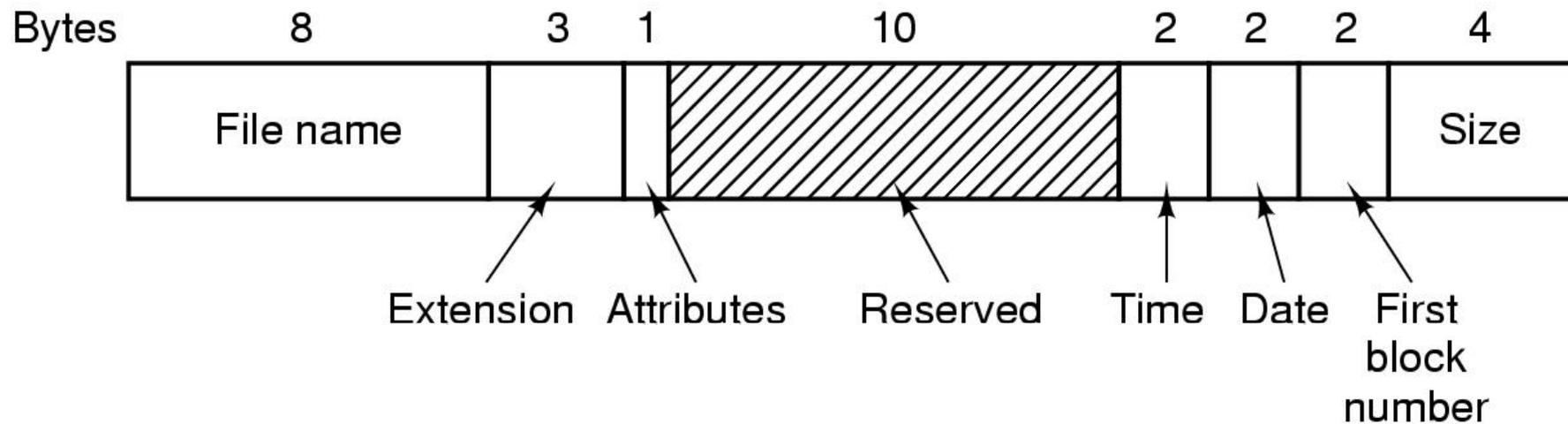
Organização de memória do CP/M

# O sistema de ficheiros CP/M (2)



Entrada de diretório do CP/M

# MS-DOS (1)



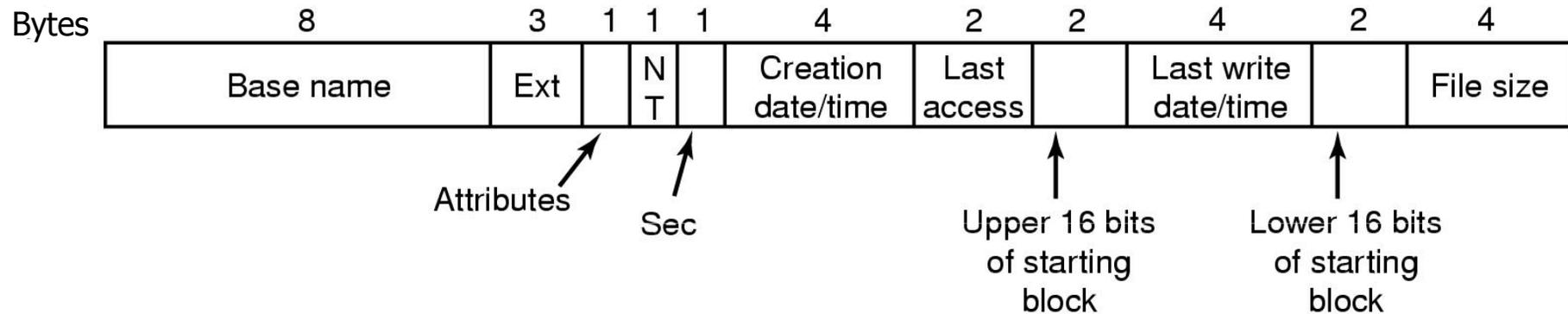
Entrada de diretório do MS-DOS

# MS-DOS (2)

<b>Block size</b>	<b>FAT-12</b>	<b>FAT-16</b>	<b>FAT-32</b>
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

- Partição máxima para diferentes tamanhos de bloco
- Células vazias são combinações não permitidas

# O sistema de ficheiros do Windows 98 (1)



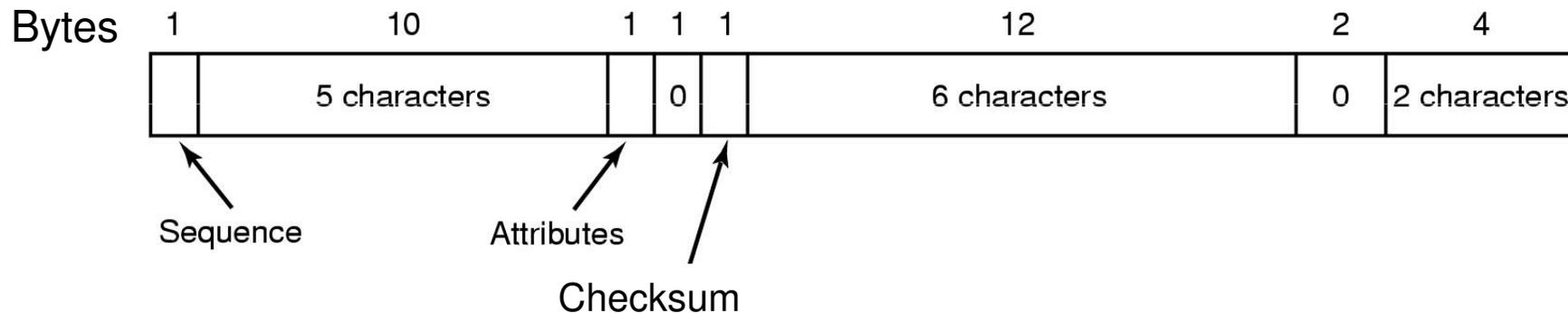
16 bits extra para ser compatível com FAT32 (32 bits de endereçamento por bloco)

Bit de compatibilidade com NT

Precisão de 10ms para o tempo de criação (via campo sec)

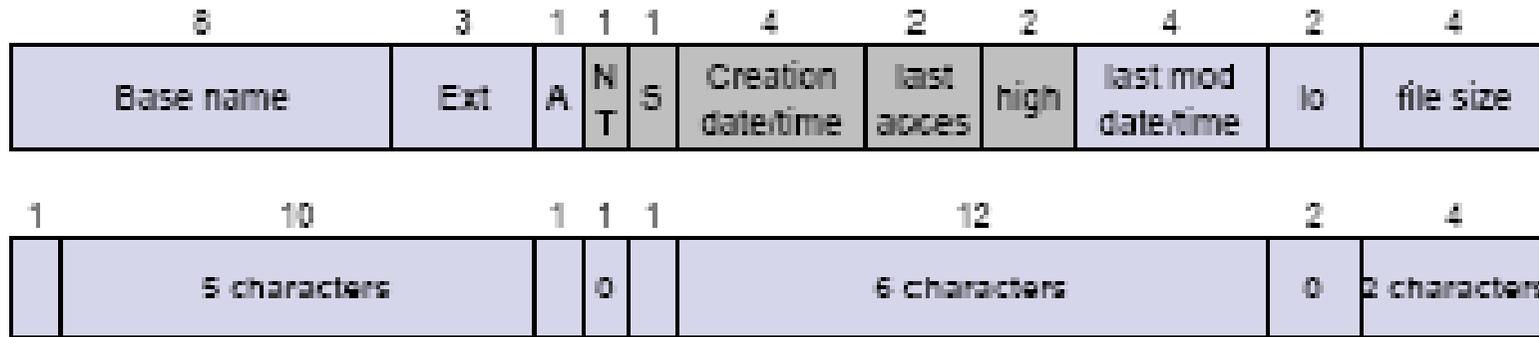
Extensão de entrada de diretório do MS-DOS usada no Windows 98

# O sistema de ficheiros do Windows 98 (2)



Entrada (parte de) para um ficheiro com nome longo no Windows 98  
Extensão da entrada padrão com 8.3 caracteres  
Pode utilizar várias destas extensões

# O sistema de ficheiros do Windows 98 (3)

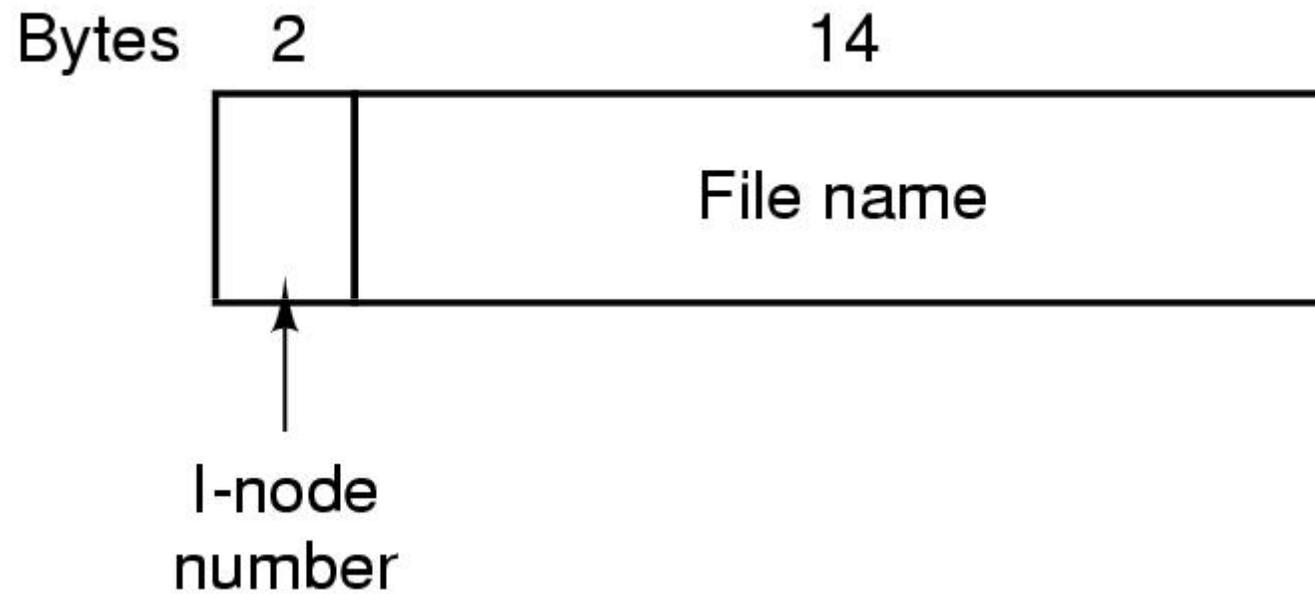


THISIS~1	TXT	A	N	S	Creation date/time	last acces	high	last mod date/time	lo	file size	stored last
1	T h i s		0		i s		a	v	0	e r	
2	y l o n		0		g f i l e				0	n a	
67	m e !		0						0		stored first

"This is a very long filename!.TXT" == THISIS~1.TXT

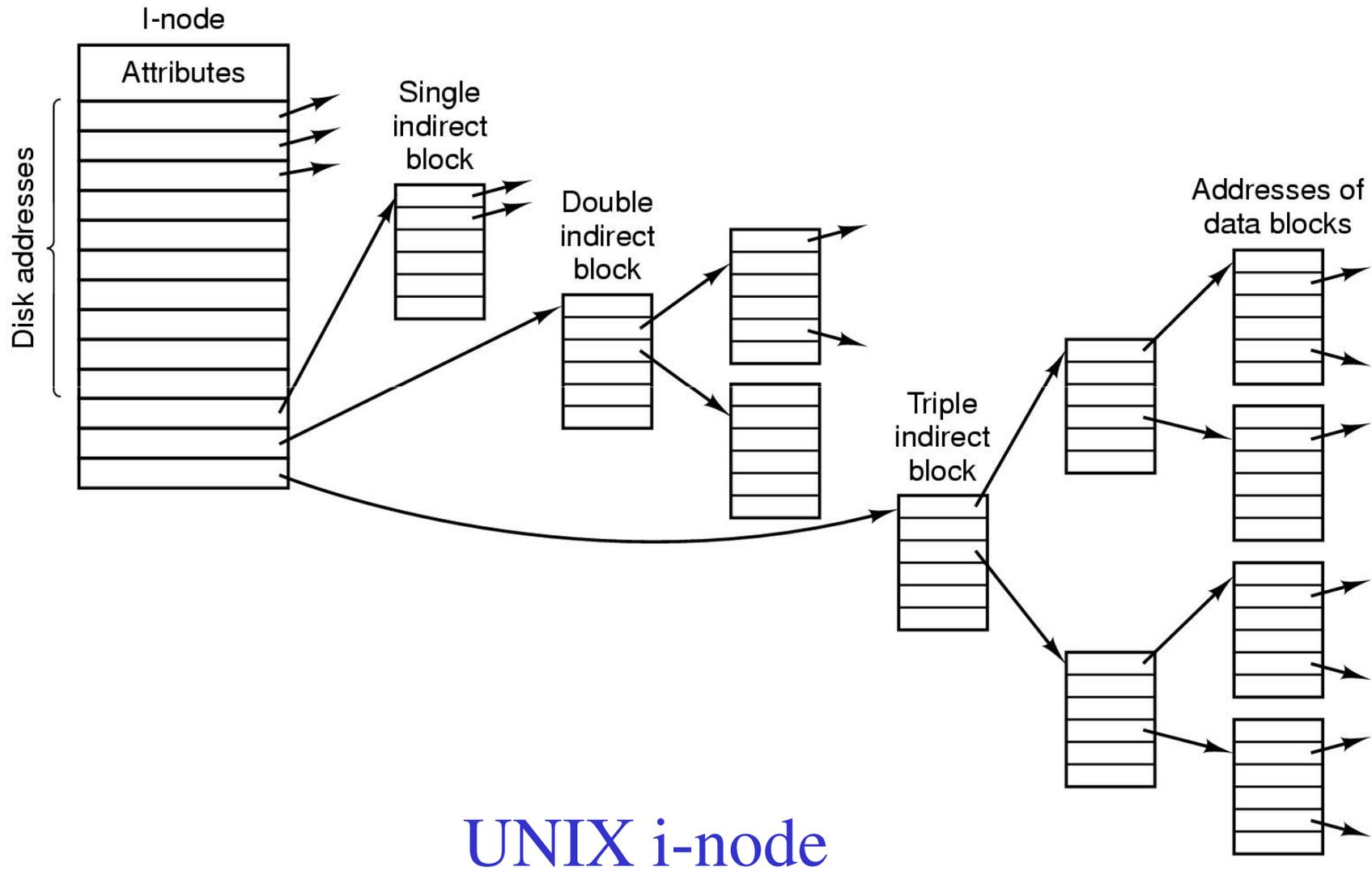
Exemplo de nome longo armazenado no Windows 98

# O sistema de ficheiros UNIX V7 (1)



Entrada de diretório

# O sistema de ficheiros UNIX V7 (2)



# UNIX V7 (3)

Root directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up  
usr yields  
i-node 6

I-node 6  
is for /usr

Mode
size
times
132

I-node 6  
says that  
/usr is in  
block 132

Block 132  
is /usr  
directory

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast  
is i-node  
26

I-node 26  
is for  
/usr/ast

Mode
size
times
406

I-node 26  
says that  
/usr/ast is in  
block 406

Block 406  
is /usr/ast  
directory

26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox  
is i-node  
60

Passos para encontrar */usr/ast/mbox*