

# Sistemas de Arquivos

Porque armazenar informações?

# Por que manter Sistemas de Arquivos?

- Suprir a deficiência de espaços de memória física (utilizando-se de técnicas como a paginação);
- Memórias de longo prazo (não voláteis) que permitam manter códigos e dados mesmo após o desligamento do equipamento, ou desconexão;
- Permitir a criação de espaços de memória, sem restrições, que possibilite sua utilização por todos os processos que estiverem sendo executados simultaneamente.



# Sistemas de Arquivos - Requisitos

Assim, temos três requisitos essenciais para armazenamento de informações a longo prazo:

1. Deve ser possível armazenar uma quantidade muito grande de informações.
2. A informação deve sobreviver ao término do processo que esta utilizando-a.
3. Múltiplos processos devem poder acessar as informações de uma só vez.

# Sistemas de Arquivos

Pense em um disco como uma sequência linear de blocos de tamanho fixo e suporte à leitura e gravação de blocos. Perguntas que surgem rapidamente:

- Como você encontra as informações?
- Como você impede que um usuário leia os dados de outro?
- Como você sabe quais blocos estão livres (liberados para registro)?



# Arquivos

Um arquivo é um mecanismo de abstração. Ele fornece uma maneira de armazenar informações no disco e lê-las novamente mais tarde. Isso deve ser feito de maneira a proteger o usuário dos detalhes de como e onde as informações são armazenadas e de como os discos realmente funcionam.

# Nome dos Arquivos

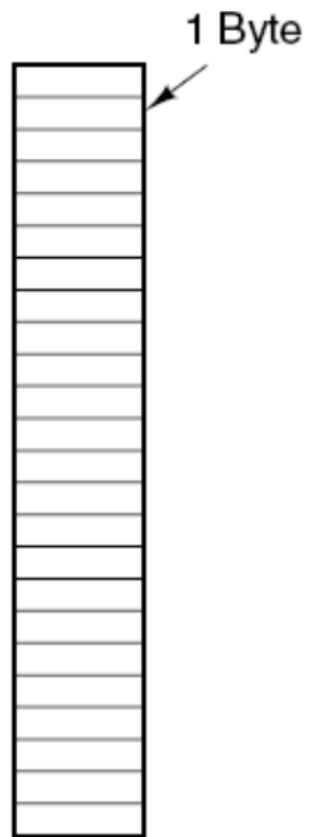
- Arquivos são mecanismos de abstração
  - ✓ Para armazenar informações no disco e lê-lo de volta
  - ✓ Quando um processo cria um arquivo, ele fornece um nome ao arquivo; e o arquivo pode ser acessado pelo nome
- Nome do arquivo de duas partes
  - ✓ Extensão do arquivo: indicando características do arquivo
  - ✓ No Unix, a extensão de arquivo é apenas uma convenção; (Compilador C é exceção)
  - ✓ No Windows, as extensões de arquivo especificam qual programa “possui” essa extensão; ao clicar duas vezes, o programa atribuído a ele é iniciado



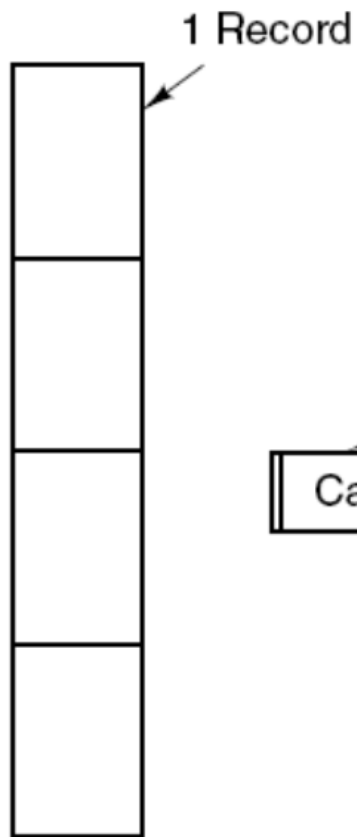
# Nome dos Arquivos - Exemplos

Extension	Meaning
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

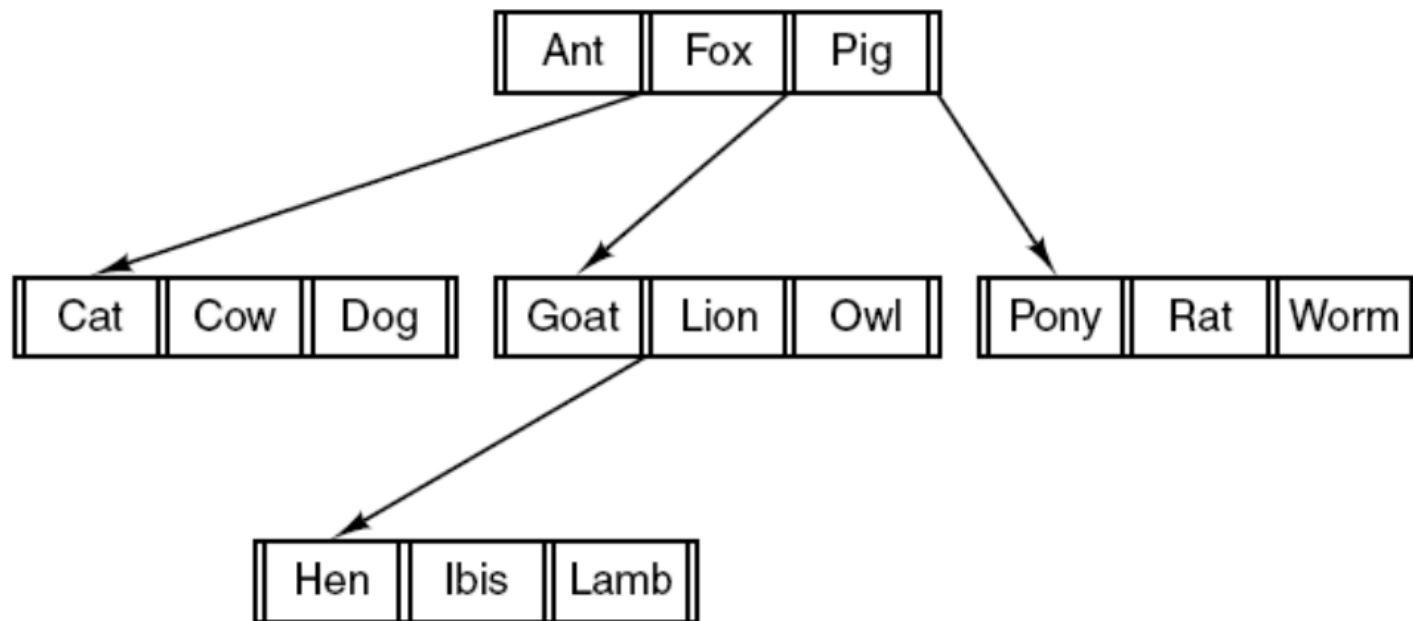
# Estrutura dos Arquivos



(a)



(b)



(c)

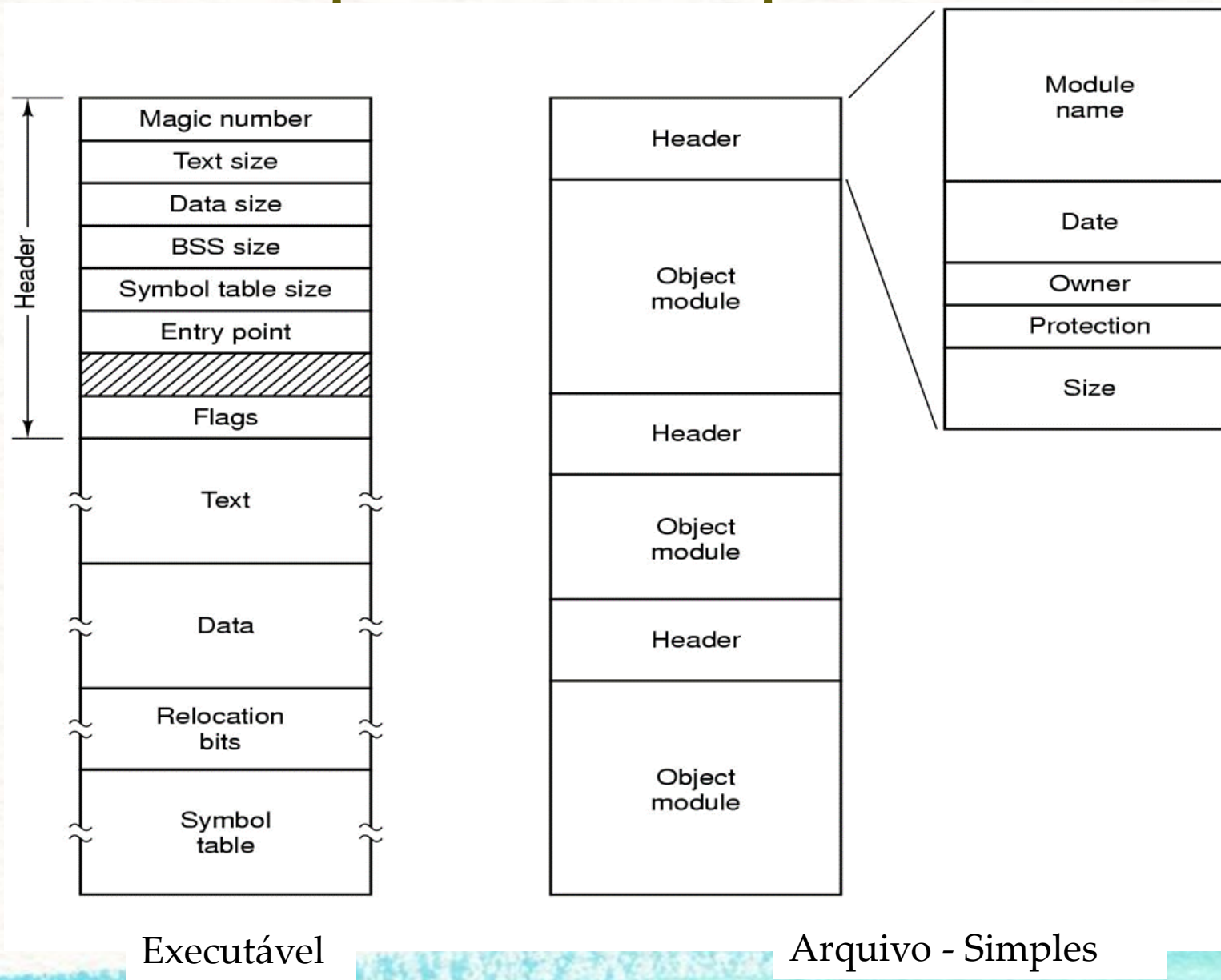
Três tipos de arquivos. (a) sequência de bytes. (b) Registro da sequência. (c) árvore.



# Tipos de Arquivos

- Arquivos Regulares:
  - ✓ Arquivos ASCII ou arquivos binários
  - ✓ ASCII consiste em linhas de texto; pode ser exibido e impresso
  - ✓ Binário, tem alguma estrutura interna conhecida pelos programas que os usam
- Diretório
  - Arquivo (pasta) para organizar arquivos
- Arquivos especiais de caracteres (um arquivo de dispositivo de caractere)
  - ✓ Relacionado a dispositivos de E/S e padrão serial de E/S
- Bloco de arquivos especiais (um arquivo de dispositivo de bloco)
  - ✓ Principalmente para modelar discos

# Tipos de Arquivos





# Acesso aos Arquivos

- Descritor de arquivo

Um descritor de arquivo é um inteiro pequeno representando um objeto gerenciado pelo kernel que um processo pode ler ou gravar em:

  - ✓ Todo processo tem um espaço privado de descritores de arquivos começando em 0
  - ✓ Por convenção, 0 é a entrada padrão, 1 é a saída padrão e 2 é o erro padrão
- Chamada do sistema: `read ()` e `write ()` para ler e gravar nos arquivos nomeados pelos descritores de arquivos

# Alguns atributo dos Arquivos


Attribute	Meaning
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 was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to



# Atributo dos Arquivos Unix/Linux



[www.linuxnaweb.com](http://www.linuxnaweb.com)


[www.linuxnaweb.com](http://www.linuxnaweb.com)

Usuário dono	Grupo dono	Outros							
-rw-	r--	r--	1	root	root	1528	Out	31	22:33 /etc/passwd
Tipo de objeto	Permissão		Número de links	Dono	Grupo Dono	Tamanho	Data	Hora	Caminho

# Atributo dos Arquivos Unix/Linux

drwxr-xr-x 2 root root 4096 Sep 24 2008 Unit2

drwxr-xr-x 2 root root 4096 May 26 19:21 a

-rwxr-xr-x 1 root root 10930 Aug 5 22:49 a.out

-rwxrwx--T 1 root root 81 Aug 2 2008 a.txt

-rwxr-x--- 1 root root 81 May 26 19:20 b.txt

-rwx----- 1 root root 81 Jul 30 19:28 c.txt

-rwxr-xr-x 1 root root 11193 Jul 30 19:27 cp



# Operação de Arquivos

- As chamadas de sistema mais comuns relacionadas a arquivos são:

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get Attributes
- Set Attributes
- Rename

# Exemplo de código de chamada de arquivo

**Ex. : Programa para  
Copiar arquivos:**

```
/* 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 */

    /* 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 */
}
```

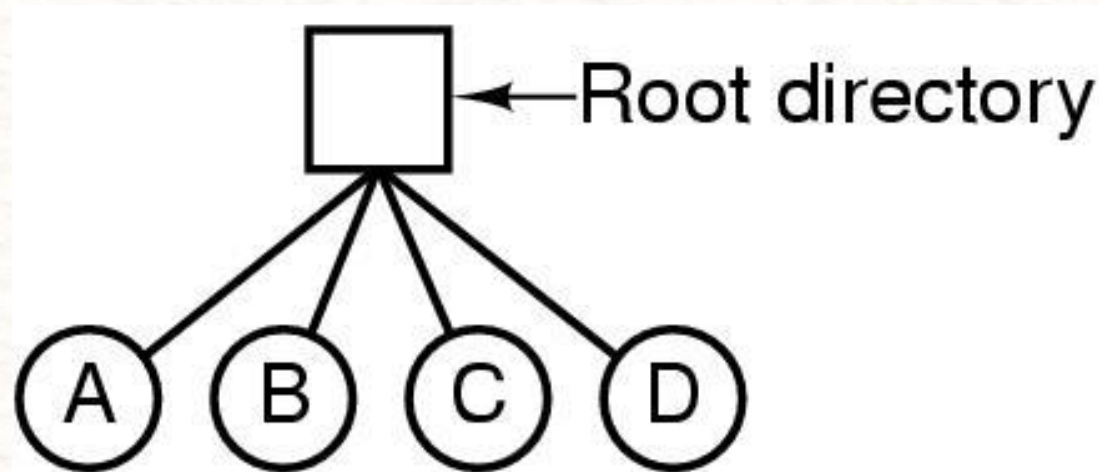
**Continua...**





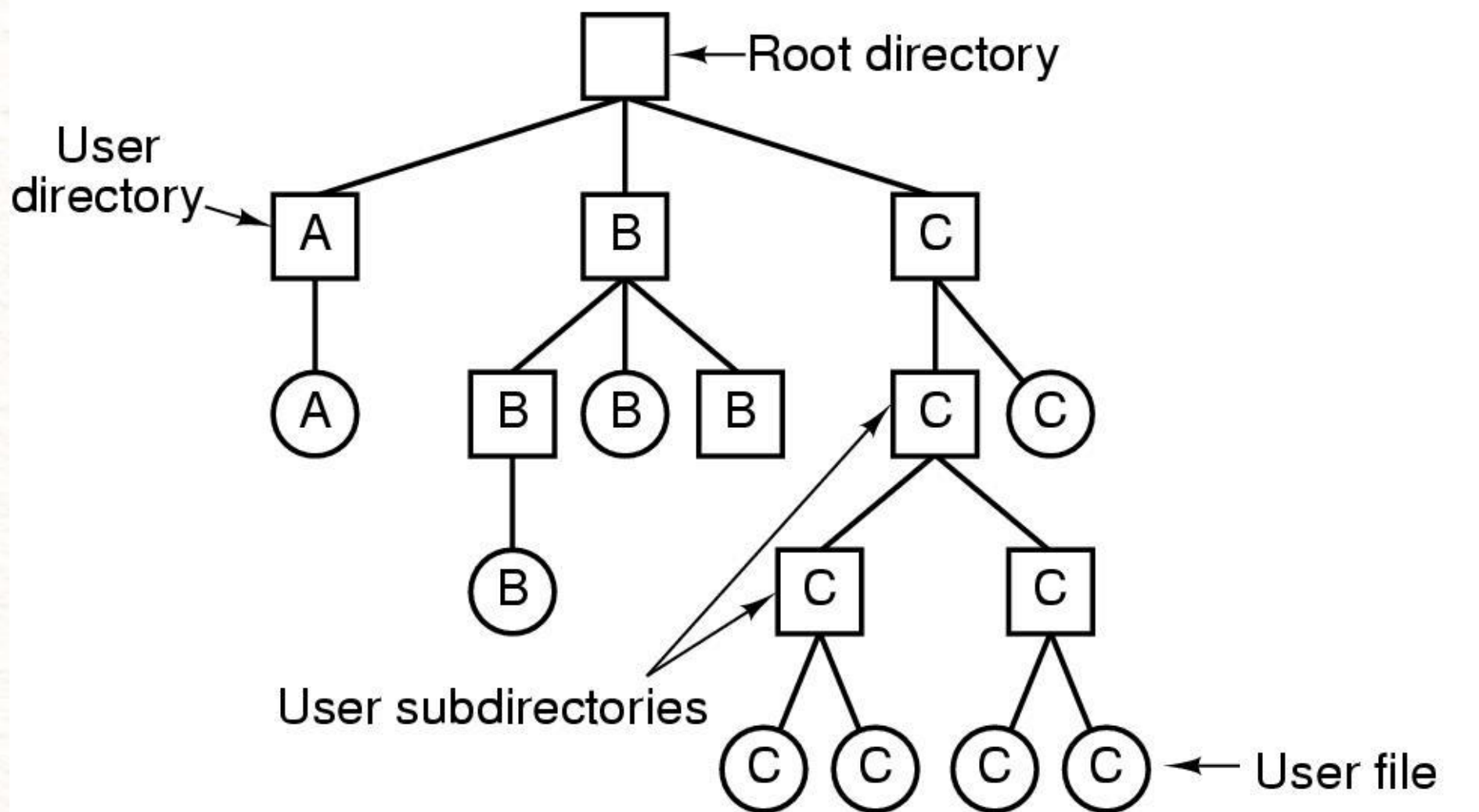
# Hierarquia de Diretórios

- Sistema de diretório de nível único:



Um sistema de diretório de nível único contendo quatro arquivos

# Hierarquia de Diretórios

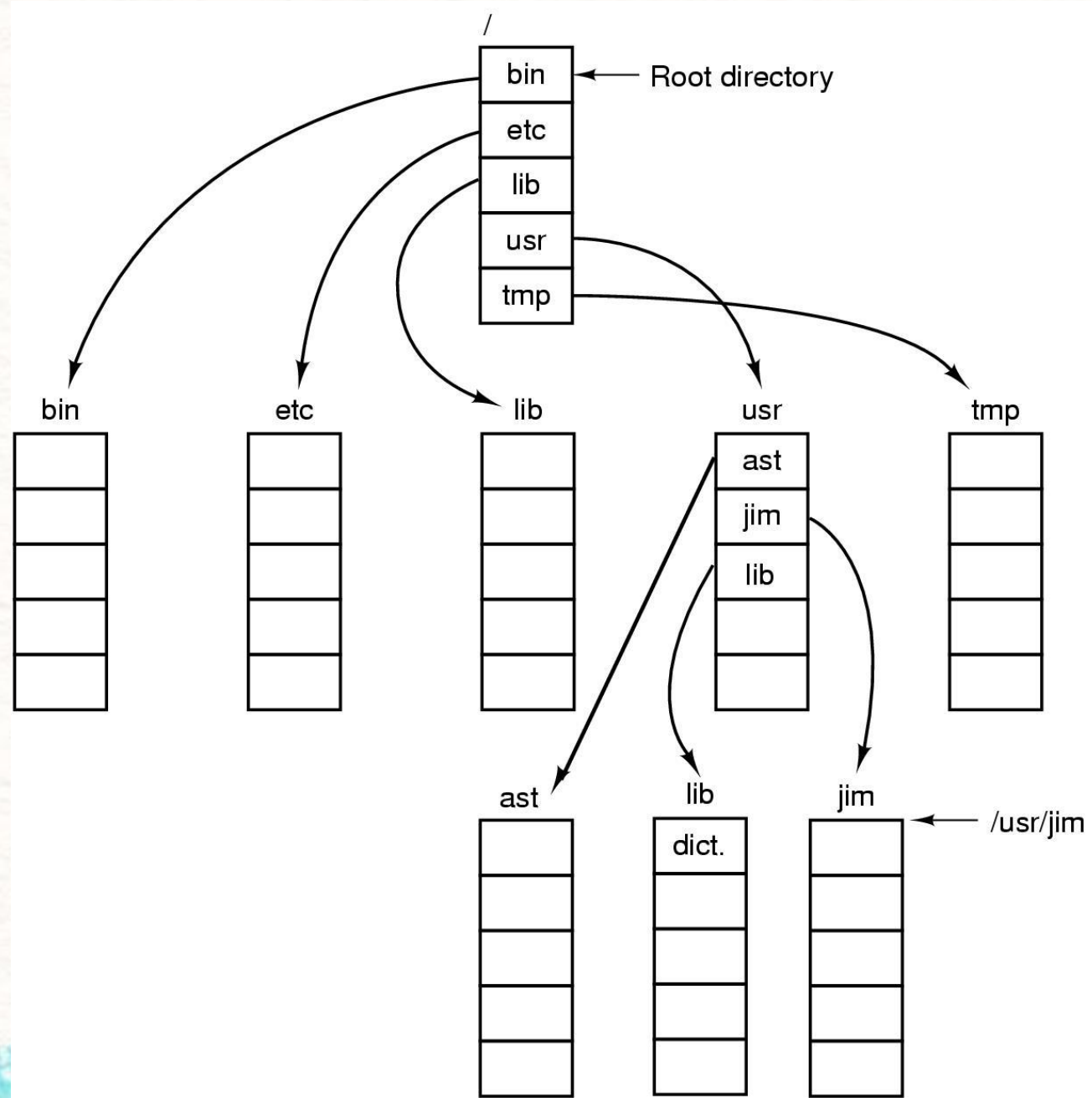


Hierarquia de um sistema de diretórios



# Nome dos caminhos (path names)

Árvore de diretórios do  
Unix/Linux



# Nome dos caminhos (path names)

- Nome do caminho absoluto
  - ✓ Começa da raiz e é único
- Nome do caminho relativo
  - ✓ Conjunto de trabalho atual: todos os nomes de caminho não iniciados no diretório raiz (root) são obtidos em relação ao diretório de trabalho.



# Operações com Diretórios

- As chamadas de sistema mais comuns gerenciamento de diretórios são:

- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

# Operações com Diretórios

- Hard Link

Este link permite que um arquivo apareça em mais de um diretório; incrementa o contador no arquivo i-node

- Link simbólico

Um nome é criado apontando para um pequeno arquivo nomeando outro arquivo



# Implementação de sistemas de arquivos

- Questões dos Usuários:  
Como os arquivos são nomeados, quais operações são permitidas neles, como é a árvore de diretórios.
- Questões dos Implementadores:  
Como arquivos e diretórios são armazenados, como o espaço em disco é gerenciado e como fazer tudo funcionar de forma eficiente e confiável.

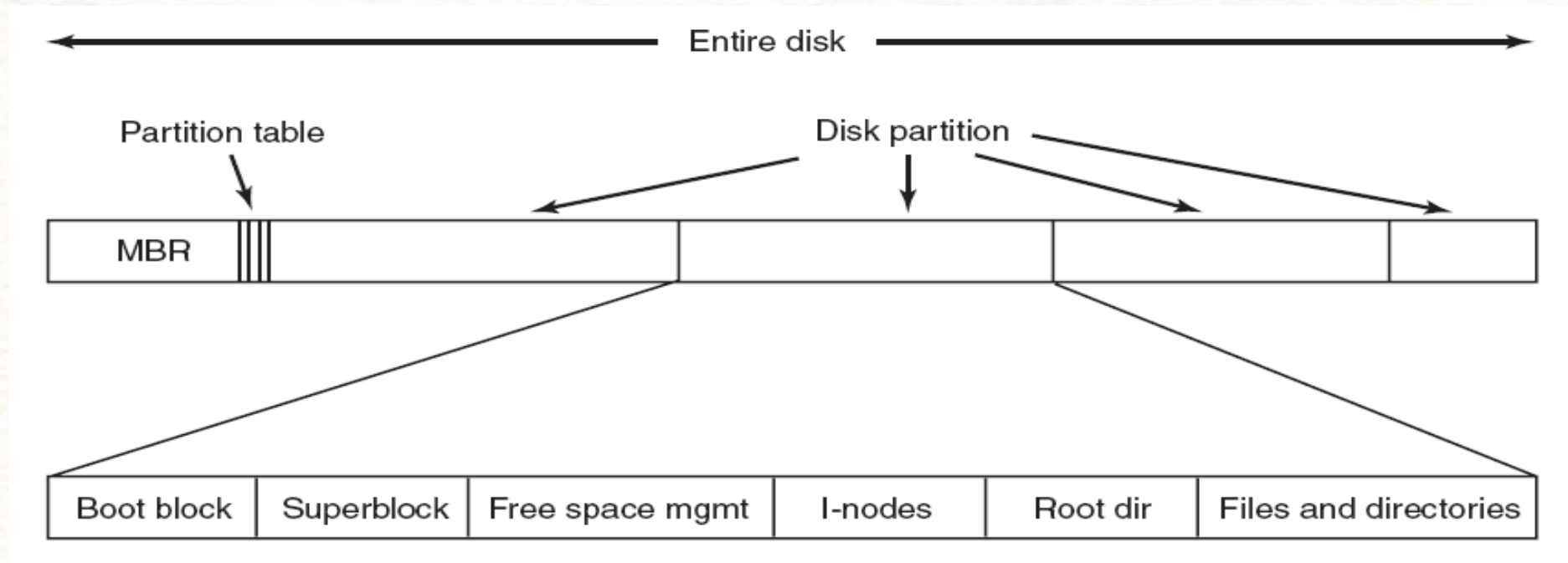
# Layout dos sistemas de arquivos

1. O sistema de arquivos é armazenado em discos;
2. A maioria dos discos é dividida em várias partições;
3. O setor 0 é chamado MBR (master boot record), para inicializar o computador;
4. BIOS lê e executa MBR, o MBR localiza a partição ativa, lê o bloco de inicialização e executa;
5. O bloco de inicialização lê no sistema operacional contido na partição.



# Layout dos sistemas de arquivos

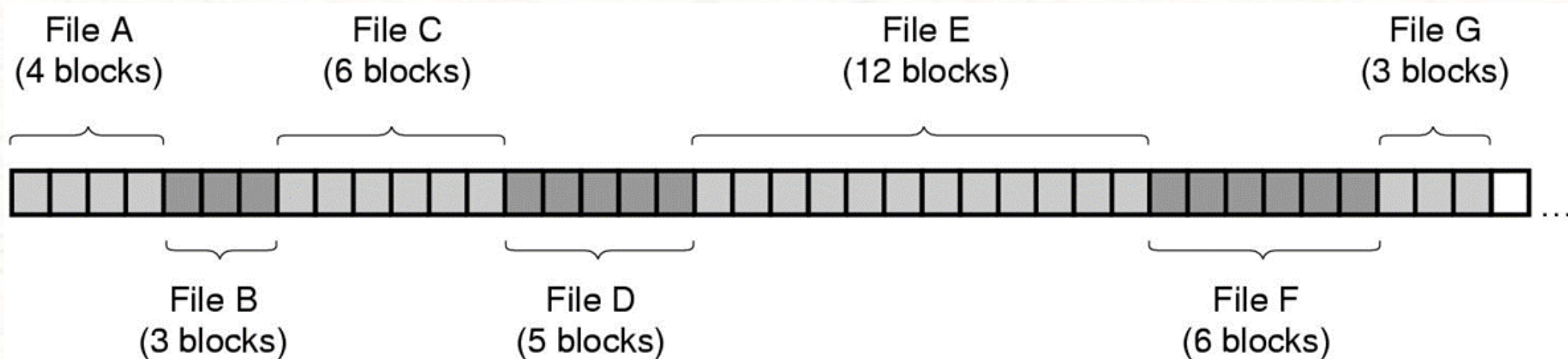
Superblock: contém todos os parâmetros chave sobre um sistema de arquivos; é lida na memória o inicializado (booted) ou no FS (file system) utilizado.



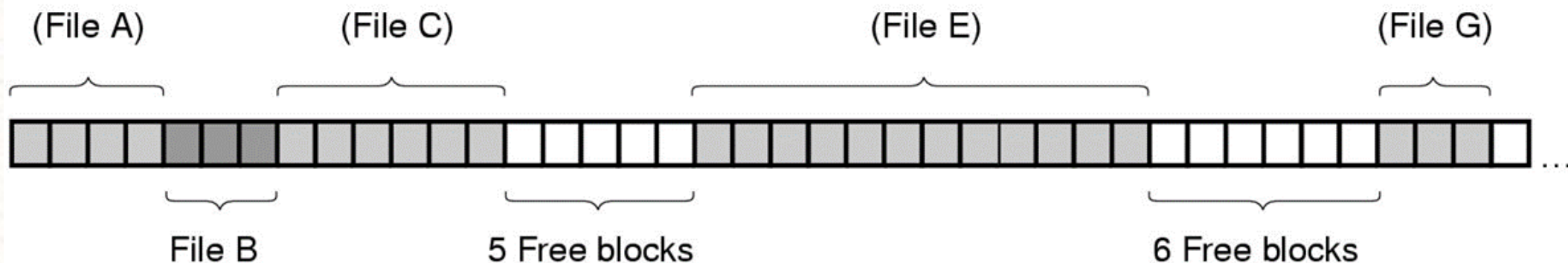
Um exemplo de layout de sistema de arquivos



# Alocação Contígua



(a)



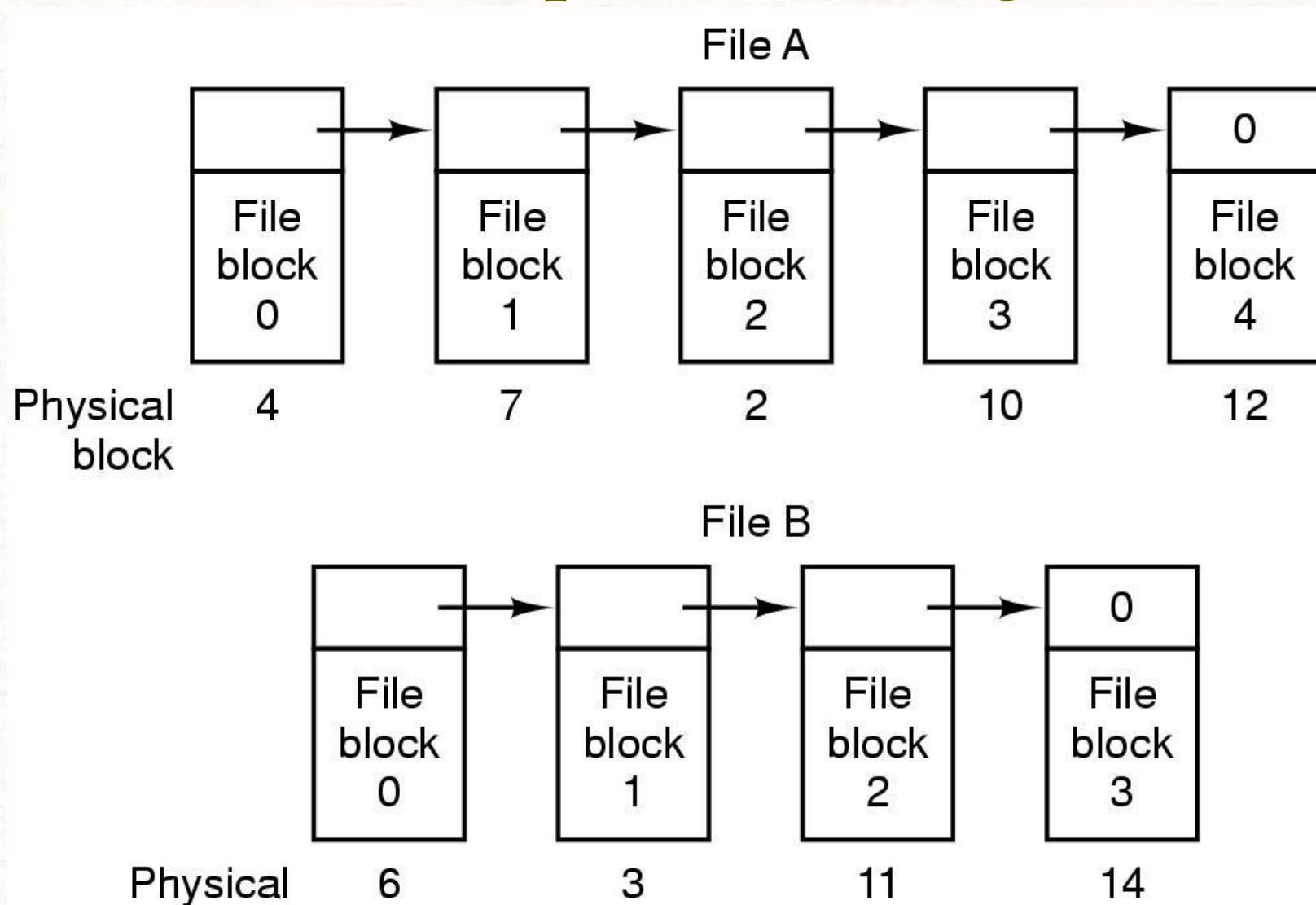
(b)

(a) Alocação contígua de espaço em disco para 7 arquivos.  
(b) O estado do disco depois que os arquivos D e F foram removidos

# Alocação Contígua

- Vantagem:
  - ✓ Simples de implementar; para gravar o primeiro bloco e comprimento
  - ✓ Fácil de ler; apenas uma busca é necessária
- Desvantagem:
  - ✓ Fragmentação

# Alocação por Lista Ligada



Armazenando um arquivo como uma lista ligada de blocos de disco.

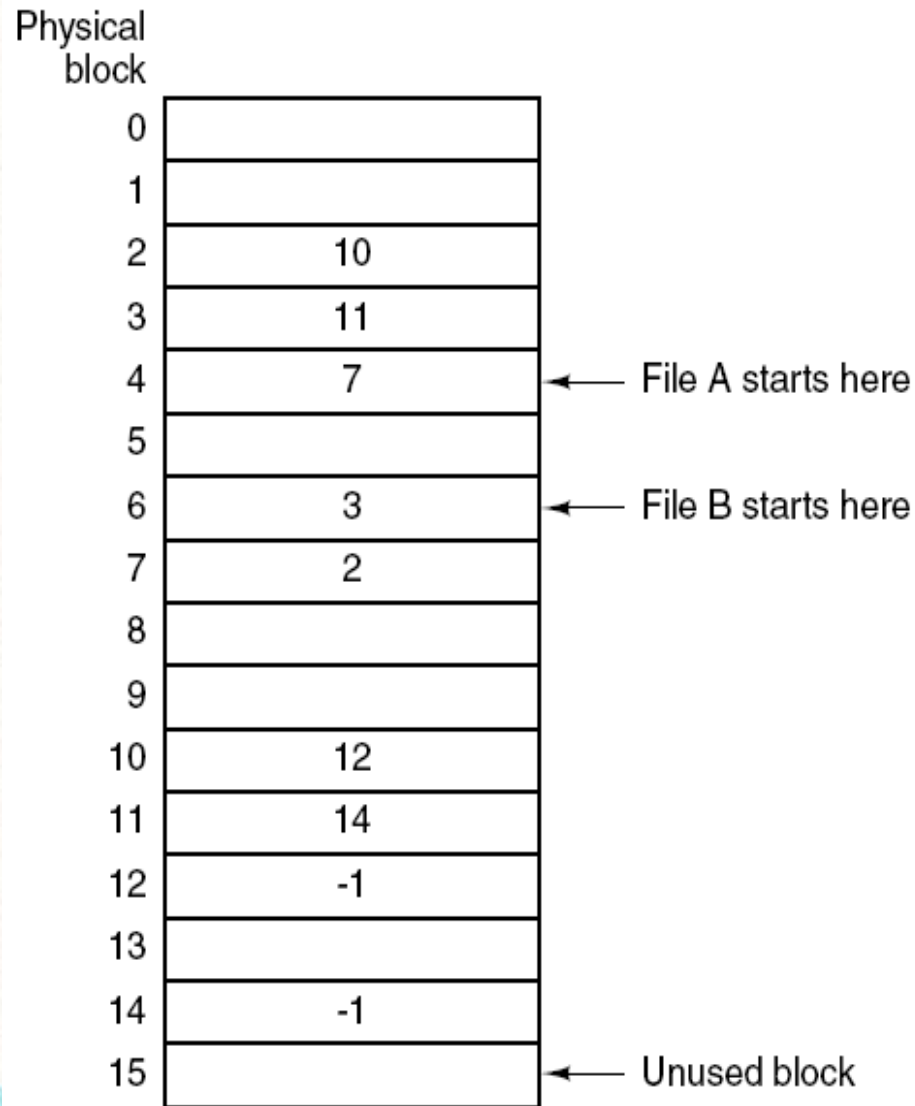


# Alocação por Lista Ligada

Para manter cada arquivo como uma lista vinculada de blocos de disco temos vantagens e desvantagens:

- Vantagem
  - ✓ Apenas fragmentação interna
- Desvantagem
  - ✓ Acesso aleatório é difícil
  - ✓ Adicionando um ponteiro na cabeça do bloco; sobrecarga extra ao copiar

# Alocação por lista ligada utilizando uma tabela na memória principal



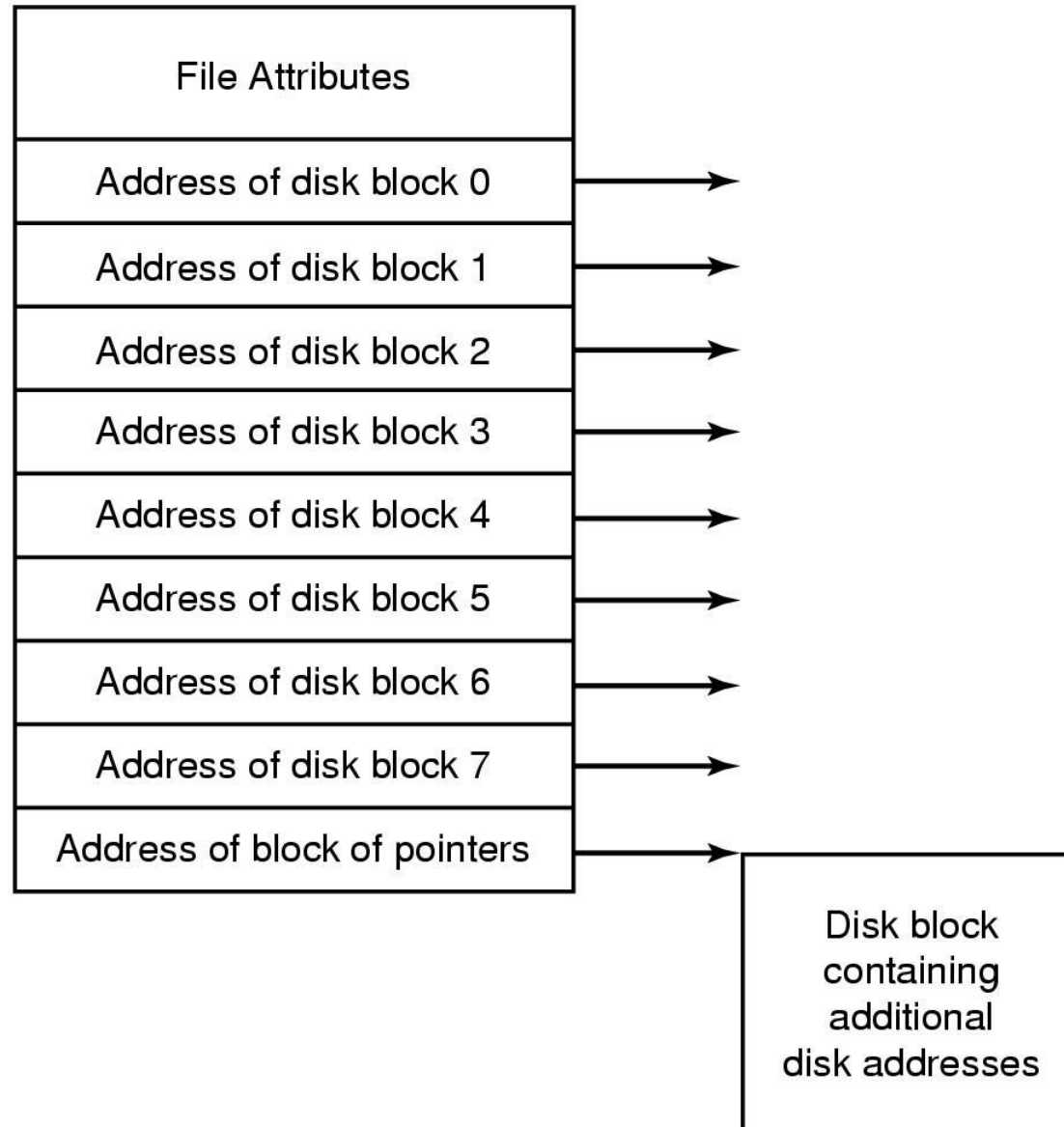
# Alocação por lista ligada utilizando uma tabela na memória principal

## Tabela de alocação de arquivos **FAT (File Allocation Table)**

- Vantagem
  - ✓ Pode usar todo o bloco
  - ✓ Acesso aleatório é fácil
  - ✓ Apenas necessita armazenar o número do bloco inicial
- Desvantagem
  - ✓ Necessita manter a tabela inteira na memória
  - ✓ Não consegue escalar bem



# I-nós (I-nodes)



# I-nós (I-nodes)

- Vantagem
  - ✓ O nó  $i$  só precisa estar na memória quando o arquivo correspondente estiver aberto; tabela de arquivos cresce linearmente com o disco
- Desvantagem
  - ✓ Cada nó  $i$  tem tamanho fixo

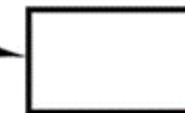
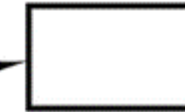
# Implementando Diretórios (1)

games	attributes
mail	attributes
news	attributes
work	attributes

(a)

games	
mail	
news	
work	

(b)

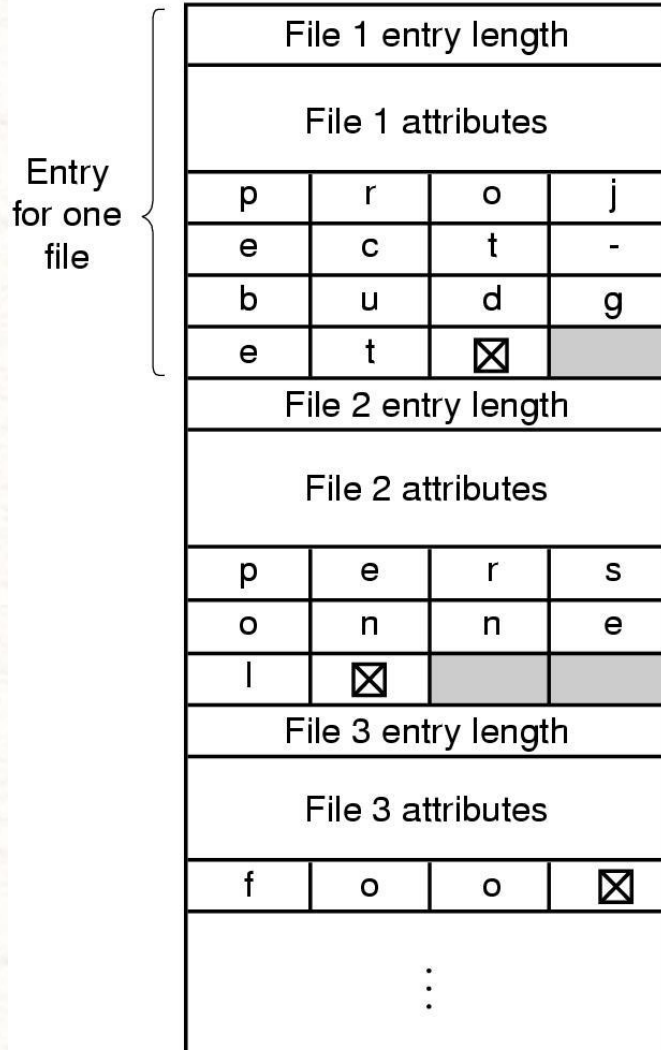


Data structure  
containing the  
attributes

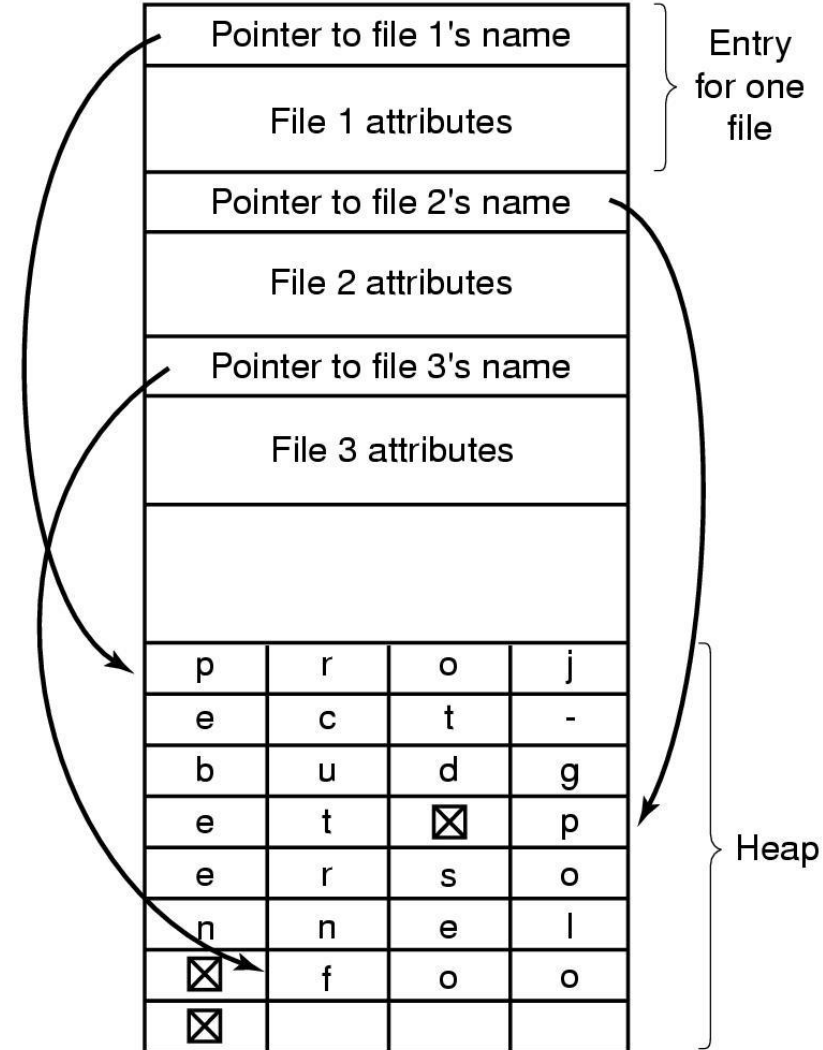
- (a) Um diretório simples contendo entradas de tamanho fixo com os endereços de disco e atributos na entrada de diretório.
- (b) Um diretório no qual cada entrada apenas se refere a um nó i.



# Implementando Diretórios (2)



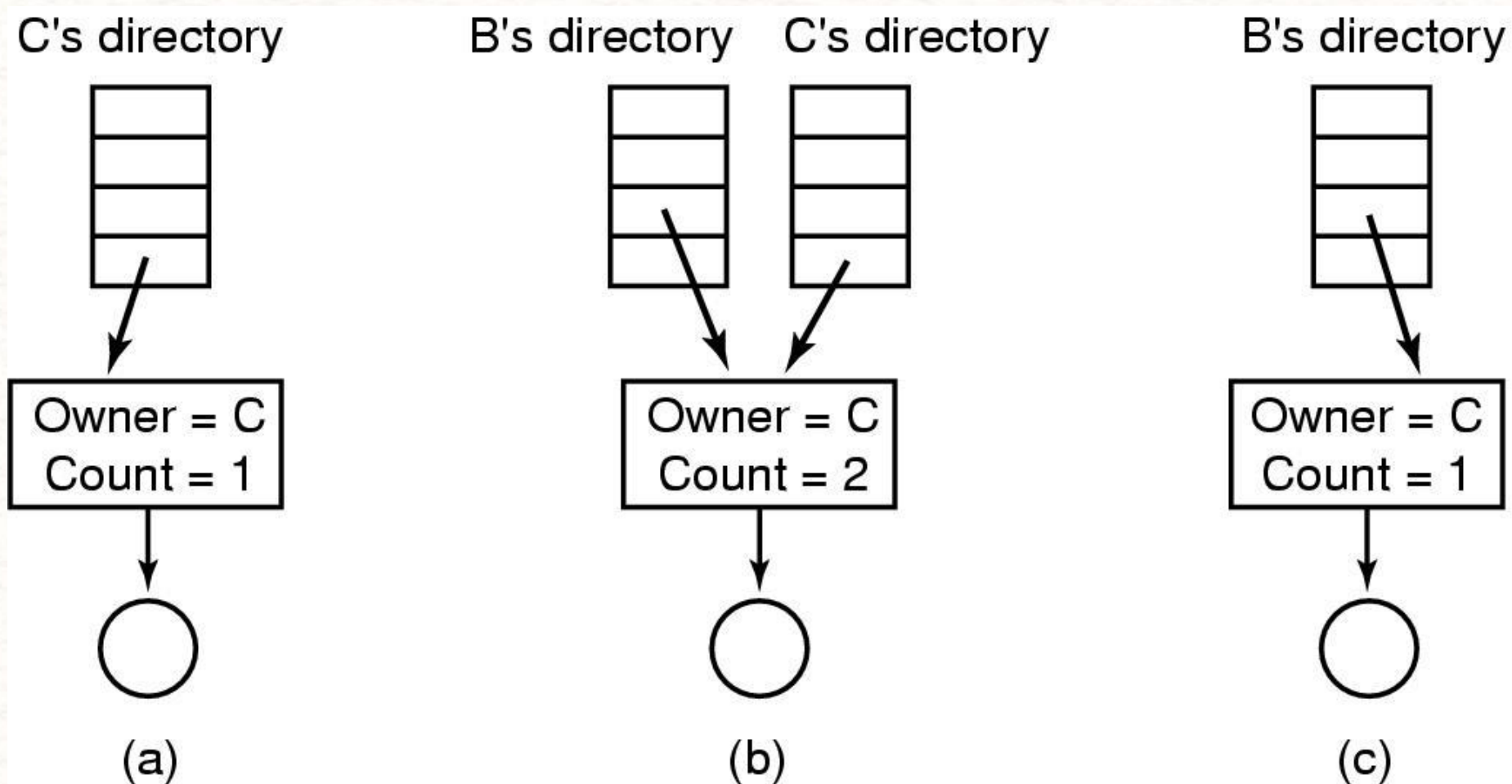
(a)



(b)

Duas maneiras de lidar com nomes de arquivos longos em um diretório.  
(a) em linha. (b) em uma pilha.

# Compartilhamento de arquivos



(a) Situação anterior à vinculação. (b) Depois que o link é criado. (c) Após o proprietário original remover o arquivo.

# Gerenciamento do espaço em disco (Block Size)

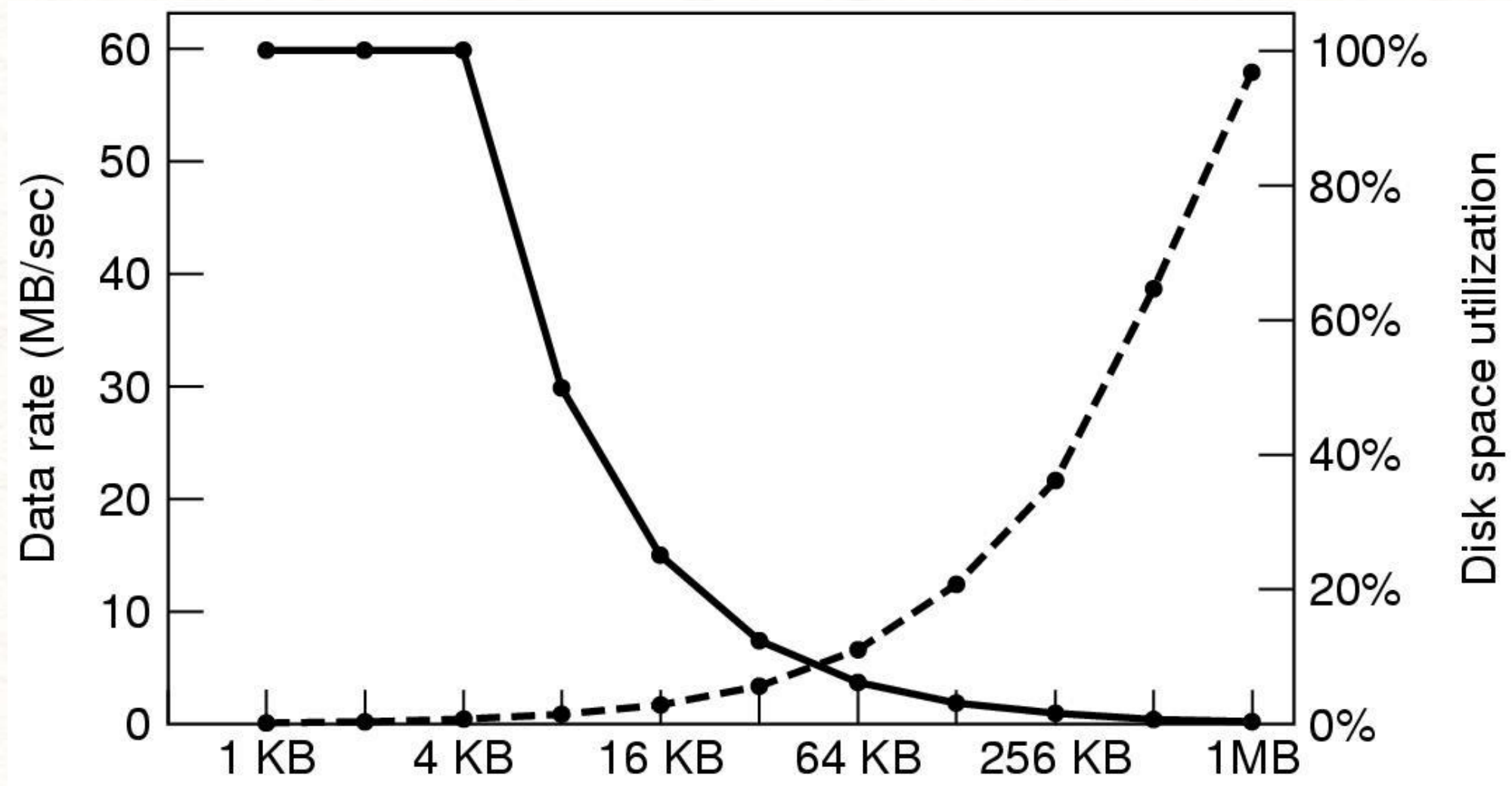
Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69

Length	VU 1984	VU 2005	Web
16 KB	92.53	78.92	86.79
32 KB	97.21	85.87	91.65
64 KB	99.18	90.84	94.80
128 KB	99.84	93.73	96.93
256 KB	99.96	96.12	98.48
512 KB	100.00	97.73	98.99
1 MB	100.00	98.87	99.62
2 MB	100.00	99.44	99.80
4 MB	100.00	99.71	99.87
8 MB	100.00	99.86	99.94
16 MB	100.00	99.94	99.97
32 MB	100.00	99.97	99.99
64 MB	100.00	99.99	99.99
128 MB	100.00	99.99	100.00

Porcentagem de arquivos menores que um determinado tamanho (em bytes).

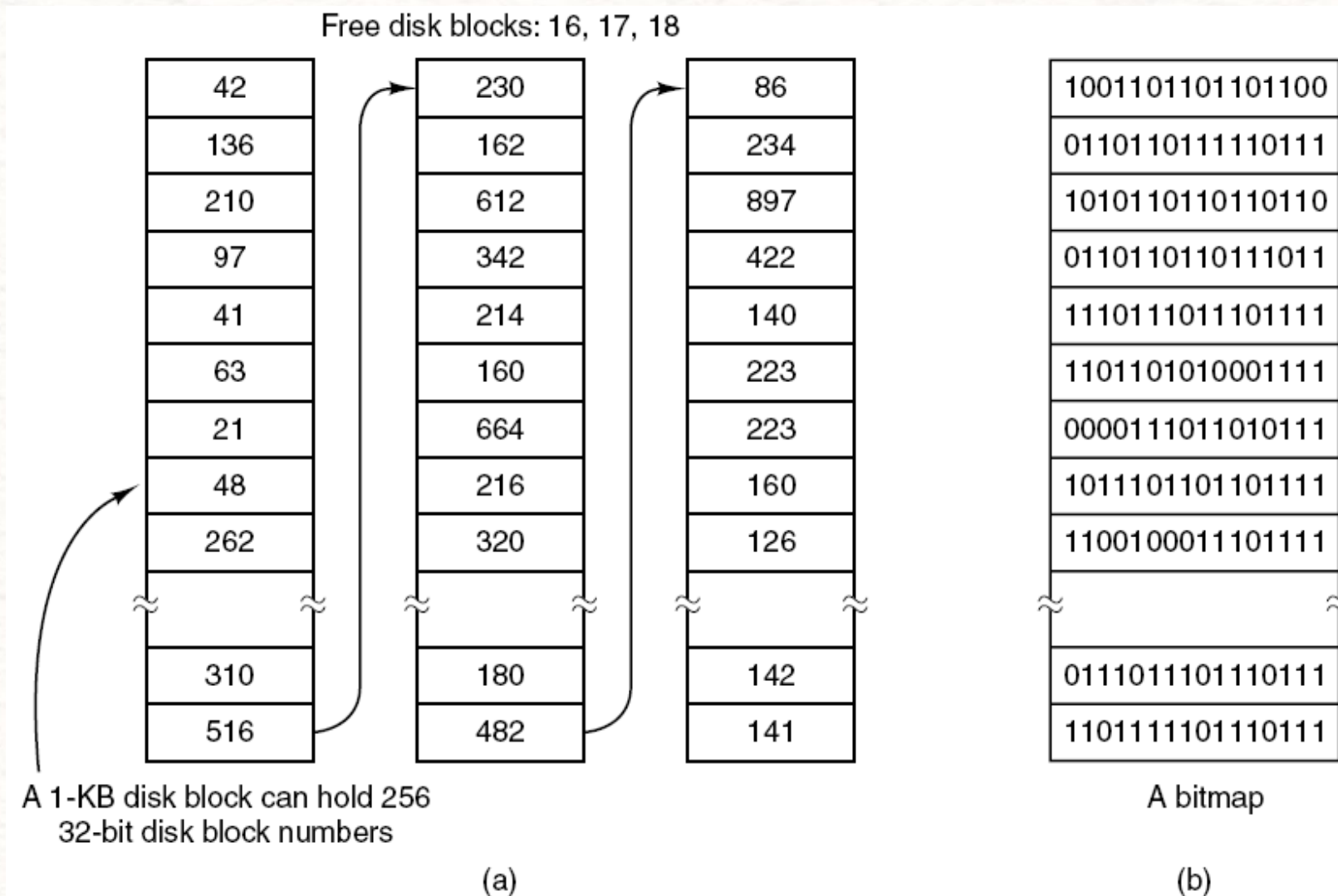


# Gerenciamento do espaço em disco (Block Size)



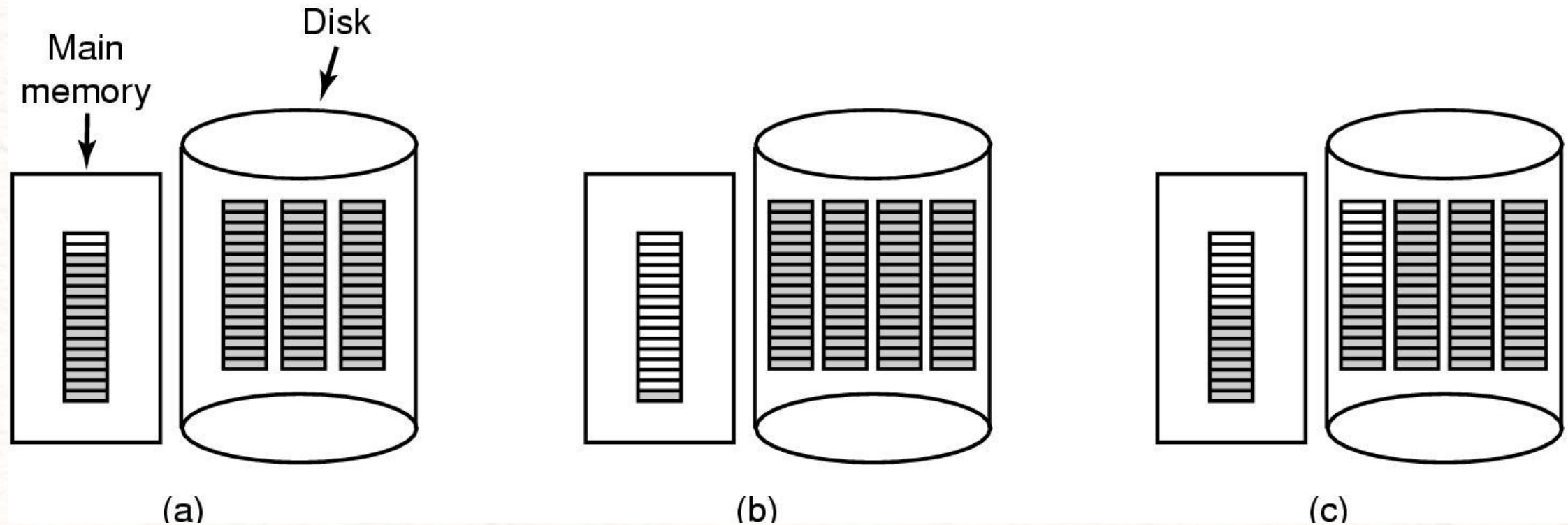
A curva (escala da esquerda) fornece a taxa de dados de um disco. A curva tracejada (escala à direita) fornece a eficiência do espaço em disco. Todos os arquivos são de 4 KB.

# Mantendo o controle dos blocos livres (1)



(a) Armazenar a lista livre em uma lista encadeada. (b) um bitmap

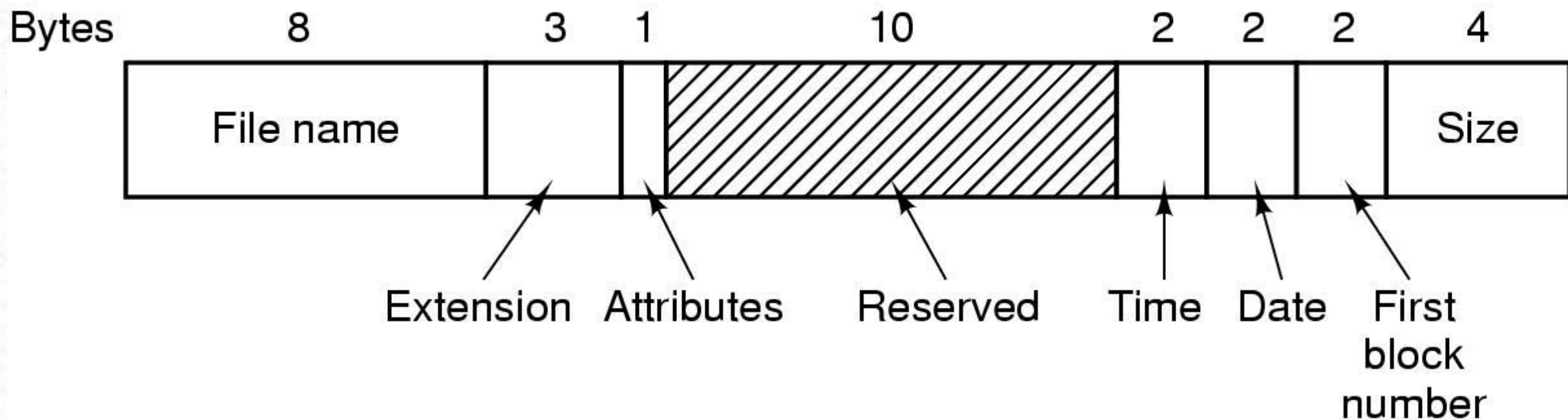
# Mantendo o controle dos blocos livres (2)



(a) Um bloco quase completo de ponteiros para liberar blocos de disco na memória e três blocos de ponteiros no disco. (b) Resultado da liberação de um arquivo de três blocos. (c) Uma estratégia alternativa para lidar com os três blocos livres. As entradas sombreadas representam ponteiros para liberar blocos de disco.



# Sistema de Arquivo do MS-DOS



# Cluster

Um cluster é a menor parcela do HD que pode ser acessada pelo sistema operacional. Cada cluster tem um endereço único, um arquivo grande é dividido em vários clusters, mas um cluster não pode conter mais de um arquivo, por menor que seja.

O tamanho dos clusters pode variar em relação ao sistema de arquivos que ele utiliza, quanto menores forem os clusters, menor será a quantidade de espaço desperdiçada no HD, sobretudo ao gravar vários arquivos pequenos, já que mesmo com apenas 1 byte de tamanho, qualquer arquivo ocupará um cluster inteiro.



# Pesquisa em grupo

Pesquisar os principais sistemas de arquivo utilizados pelos sistemas operacionais Mac OS, Linux, Unix, Windows, IOS e Android, listando as principais vantagens e desvantagens;  
Criar uma tabela comparativa entre eles.

Enviar, até a próxima aula, para: [paulo.silva323@fatec.sp.gov.br](mailto:paulo.silva323@fatec.sp.gov.br)



# Referências Bibliográficas

- TANENBAUM, Andrew S., BOSS, Herbert. **Sistemas Operacionais Modernos**, Pearson - 4ª ed., 2016.
- SILBERSCHATZ, A., GALVIN, P.B., GAGNE, G. **Fundamentos de Sistemas Operacionais**, Ed. LTC, 8ª ed., 2011
- DEITEL, H.M.; DEITEL, P.J.; CHOFFNES, D.R. – **Sistemas Operacionais**. Prentice Hall, Tradução da 3ª ed., 2005
- DEITEL, H.M.; DEITEL, P.J. – **C How to Program**. Prentice Hall, Tradução da 3ª ed., 2001
- MIZRAHI, Victorine Viviane. **Treinamento em Linguagem C – Curso Completo módulos 1 e 2**, Ed. Person Education.