



Gerência de Arquivos

Leônidas Serra, Paulo Arthur, Isabela Oliveira,
Samuel Basilio

Sumário

Pontos para discutir:

- Arquivos 3
- Diretórios..... 11
- Implementação de Sistema de Arquivos..... 19
- Gerenciamento e Otimização..... 34
- Segurança..... 46
- Referências..... 56

Arquivos

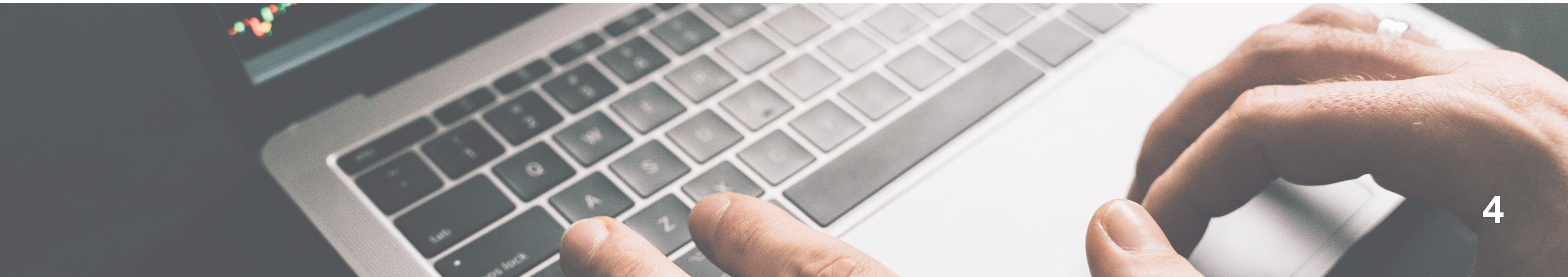
Explorando Arquivos em Sistemas de Computadores

O que são arquivos?

Um arquivo é uma coleção de dados ou informações que são armazenados juntos sob um nome único em um dispositivo de armazenamento, como um disco rígido, uma unidade USB, um cartão de memória ou mesmo em um servidor remoto na nuvem.

Nomeação de arquivos

É o processo de atribuir um identificador único a um arquivo em um sistema de arquivos



QUAIS OS TIPOS DE ARQUIVOS?

Arquivos de Texto: São arquivos simples contendo palavras e letras, como documentos ou notas. Exemplo: "Carta.txt".

Arquivos de Imagem: Guardam fotografias ou ilustrações digitais. Exemplo: "Foto.jpg".

Arquivos de Áudio: Armazenam músicas ou gravações de voz. Exemplo: "Música.mp3".

Arquivos de Vídeo: Contêm vídeos digitais, como filmes ou vídeos curtos. Exemplo: "Filme.mp4".

Arquivos Executáveis: São programas que o computador pode rodar. Exemplo: "Programa.exe".

Arquivos de Documentos: Utilizados para criar textos, planilhas e apresentações. Exemplo: "Documento.docx".

Arquivos de Backup: São cópias de segurança dos dados importantes. Exemplo: "Backup.zip".

Arquivos de Configuração: Guardam configurações de programas. Exemplo: "Configurações.ini"



O que são estruturas de arquivos?

Estrutura de arquivos descreve como os dados são organizados e acessados dentro de um sistema de arquivos, incluindo a **hierarquia de diretórios**, a **nomeação de arquivos**, os **metadados associados** e a organização física dos dados no dispositivo de armazenamento.



Formas de acesso aos arquivos

Acesso Sequencial

Quando um programa lê um arquivo sequencialmente, ele lê os dados em blocos ou segmentos, seguindo a ordem em que estão armazenados no arquivo. Por exemplo, se estamos lidando com um arquivo de texto, o programa lerá cada linha do arquivo na ordem em que foram escritas

Acesso Aleatório

No acesso aleatório, os dados podem ser lidos ou gravados em qualquer posição específica dentro do arquivo, sem a necessidade de seguir uma ordem predefinida. Isso significa que o programa pode acessar diretamente qualquer parte do arquivo, sem percorrer todos os dados anteriores.

Atributos de arquivos

1. Nome do Arquivo
2. Tipo de Arquivo
3. Tamanho do Arquivo
4. Data e Hora de Criação
5. Data e Hora de Modificação
6. Data e Hora de Acesso
7. Permissões de Acesso
8. Proprietário do Arquivo
9. Atributos de Sistema
10. Localização Física

Operações com arquivos

Referem-se às ações realizadas para criar, abrir, ler, gravar, modificar e fechar arquivos em um sistema de computador. Permitem aos usuários e programas interagir com os dados armazenados em arquivos de maneira eficiente e segura.

PRINCIPAIS CHAMADAS

1. Create: O arquivo é criado sem dados. A finalidade dessa chamada é anunciar que o arquivo está vindo e estabelecer alguns dos atributos.
2. Delete: Quando o arquivo não é mais necessário, ele tem de ser removido para liberar espaço para o disco. Há sempre uma chamada de sistema para essa finalidade.
3. Open: Antes de usar um arquivo, um processo precisa abri-lo. A finalidade da chamada open é permitir que o sistema busque os atributos e lista de endereços do disco para a memória principal a fim de tornar mais rápido o acesso em chamadas posteriores.
4. Close: Quando todos os acessos são concluídos, os atributos e endereços de disco não são mais necessários, então o arquivo deve ser fechado para liberar espaço da tabela interna. Muitos sistemas encorajam isso impondo um número máximo de arquivos abertos em processos. Um disco é escrito em blocos, e o fechamento de um arquivo força a escrita do último bloco dele, mesmo que não esteja inteiramente cheio ainda.

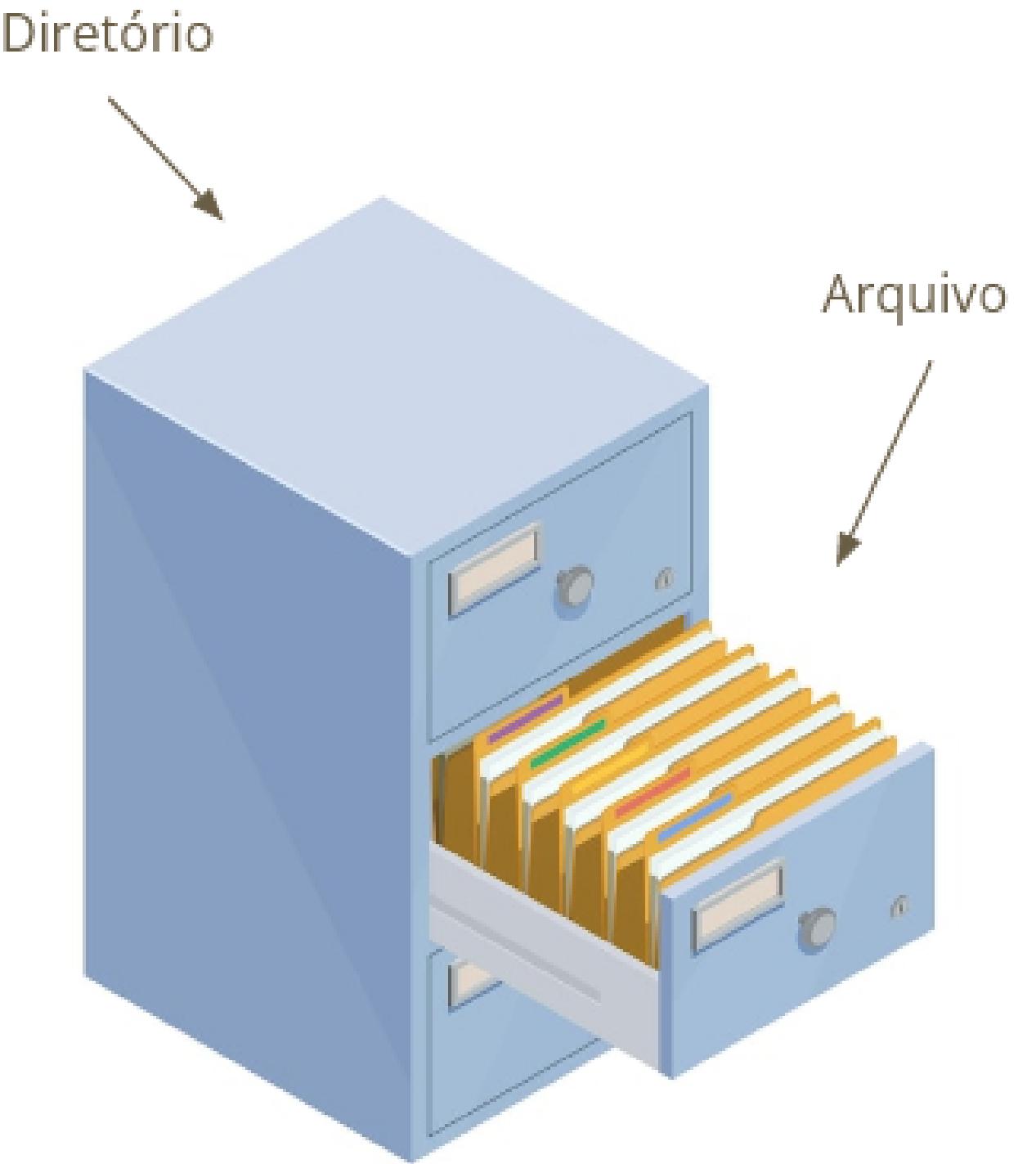
Diretórios

O que são diretórios

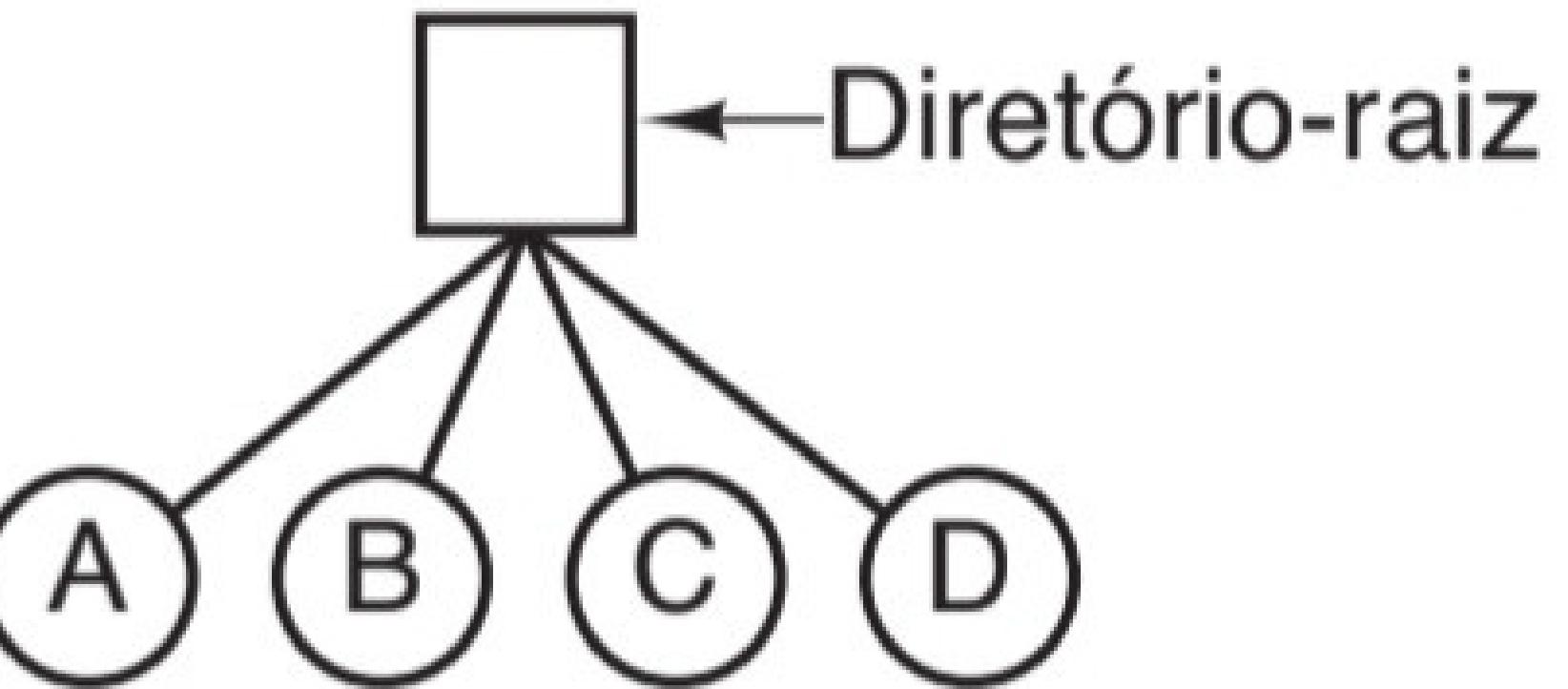
Estruturas que organizam e controlam arquivos ou sistemas de arquivos;

Também são arquivos

Existe mais de um modelo de implementação



Diretório-raiz



Só há um diretório

Geralmente chamado de diretório-raiz, ele é a única pasta a qual os arquivos são guardados.



Rapidez na busca

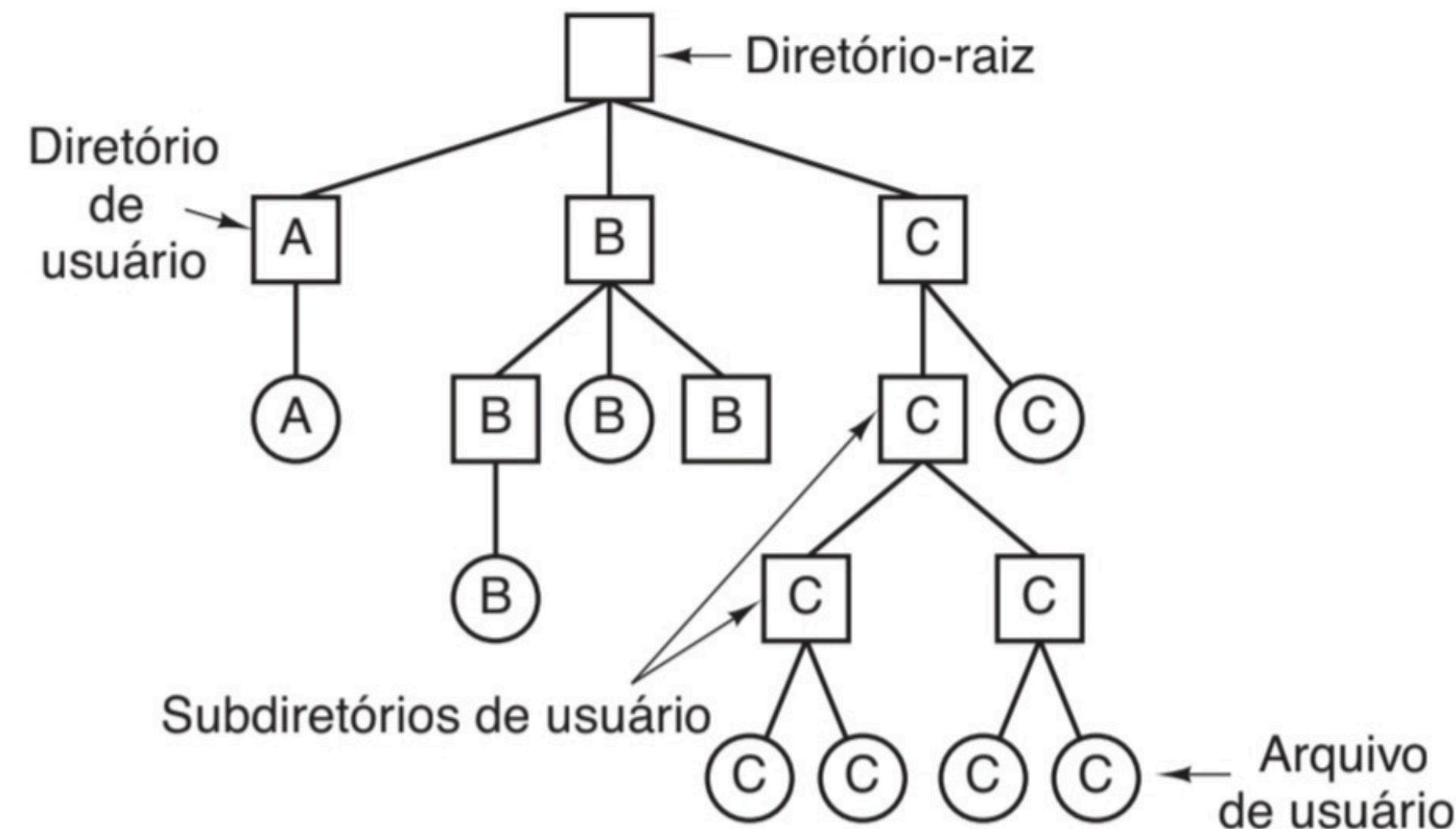
Devido sua estrutura simples e capaz de localizar arquivos rapidamente



Admite um único usuário

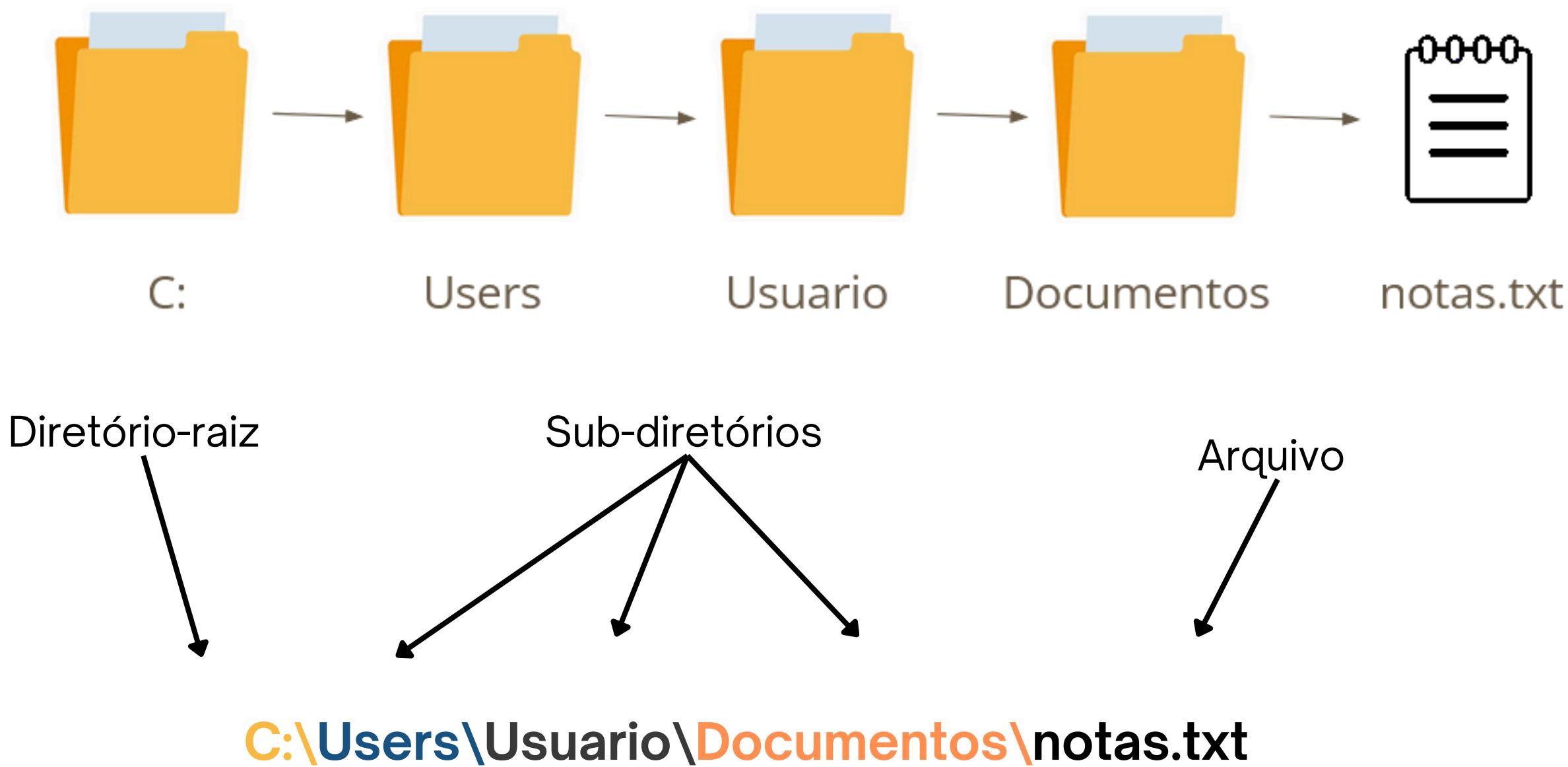
A estrutura não permite que arquivos sejam separados, o que impossibilita multiusuários

Diretório-Hierárquico



- 01 É uma árvore de diretórios**
- 02 Estrutura lógica por meio de subdiretórios**
- 03 Admite vários usuários**
- 04 Maior privacidade**

Diretório-Hierárquico



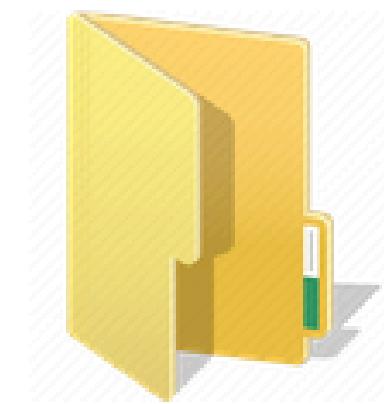
Operações com diretórios



Create



Delete



Open



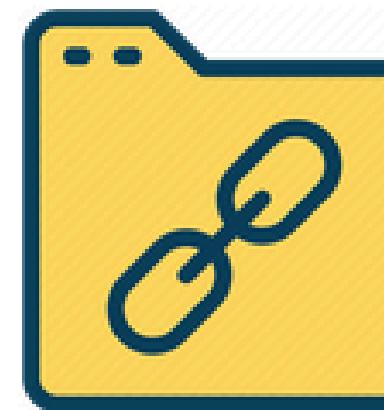
Close



Read



Rename



Link



Unlink

Exemplo

```
33     self.logduplicates = True
34     self.debug = debug
35     self.logger = logging.getLogger(__name__)
36
37     if path:
38         self.file = open(path, 'wt')
39         self.file.seek(0)
40         self.fingerprints.update(fp)
41
42     @classmethod
43     def from_settings(cls, settings):
44         return cls(settings.get('FINGERPRINTS_PATH'),
45                    settings.get('FINGERPRINTS_DEBUG'),
46                    settings.get('FINGERPRINTS_LOGDUPES'))
47
48     def request_seen(self, request):
49         fp = self.request_fingerprint(request)
50         if fp in self.fingerprints:
51             return True
52         self.fingerprints.add(fp)
53         if self.file:
54             self.file.write(fp + os.linesep)
```

```
def request_fingerprint(self, request):
    fingerprint(request)
```

C SD Hierarquico.c > ⚙ adicionarSubdiretorio(Diretorio *, Diretorio *)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define MAX_NOME_TAMANHO 50
6
7 // Estrutura para um diretório
8 typedef struct Diretorio {
9     char nome[MAX_NOME_TAMANHO];
10    struct Diretorio *subdiretorios;
11    struct Diretorio *pai;
12 } Diretorio;
13
14 // Função para criar um novo diretório
15 Diretorio* criarDiretorio(char nome[MAX_NOME_TAMANHO], Diretorio *pai) {
16     Diretorio *novoDiretorio = (Diretorio*)malloc(sizeof(Diretorio));
17     if (novoDiretorio == NULL) {
18         printf("Erro ao alocar memória para o diretório!\n");
19         exit(1);
20     }
21     strcpy(novoDiretorio->nome, nome);
22     novoDiretorio->subdiretorios = NULL;
23     novoDiretorio->pai = pai;
24     return novoDiretorio;
25 }
26
27 // Função para adicionar um subdiretório a um diretório
28 void adicionarSubdiretorio(Diretorio *pai, Diretorio *subdiretorio) {
29     if (pai->subdiretorios == NULL) {
30         pai->subdiretorios = subdiretorio;
31     } else {
32         Diretorio *temp = pai->subdiretorios;
33         while (temp->subdiretorios != NULL) {
34             temp = temp->subdiretorios;
35         }
36         temp->subdiretorios = subdiretorio;
37     }
38 }
```

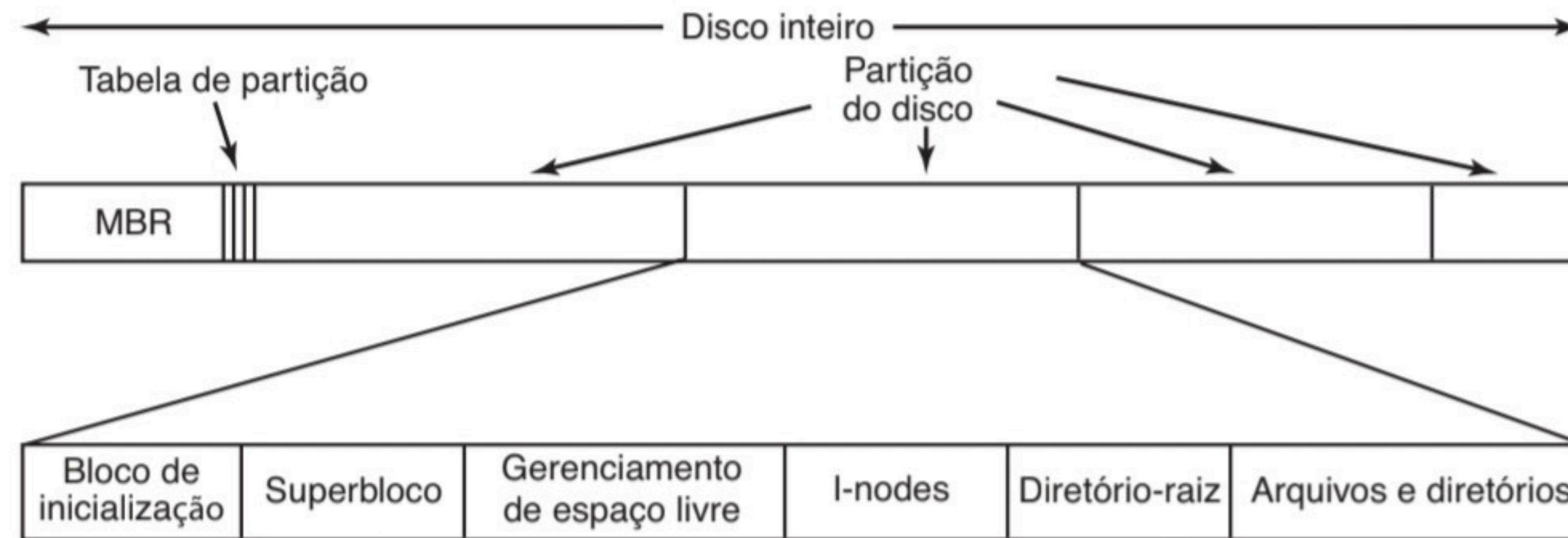
Implementação de Sistemas de arquivos

```
33
34     self.logduplicates = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(path, "w+")
39         self.file.seek(0)
40         self.fingerprints.update(fingerprint)
41
42     @classmethod
43     def from_settings(cls, settings):
44         config = settings.get("file")
45         if config:
46             return cls(config["path"])
47
48     def request_fingerprint(self, request):
49         fp = self.request_fp(request)
50         if fp in self.fingerprints:
51             return True
52         self.fingerprints.add(fp)
53         if self.file:
54             self.file.write(fp + os.linesep)
```

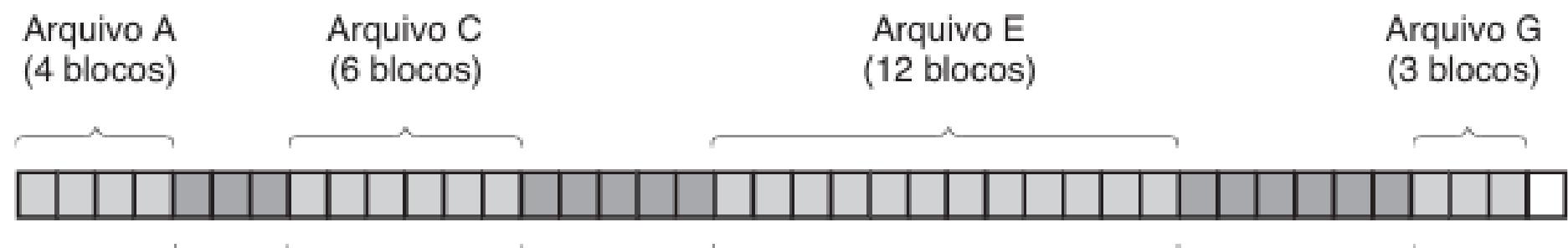
Explorando Arquivos em Sistemas de Computadores

Setor 0 do disco é o MBR(Master Boot Record), usado para inicialização

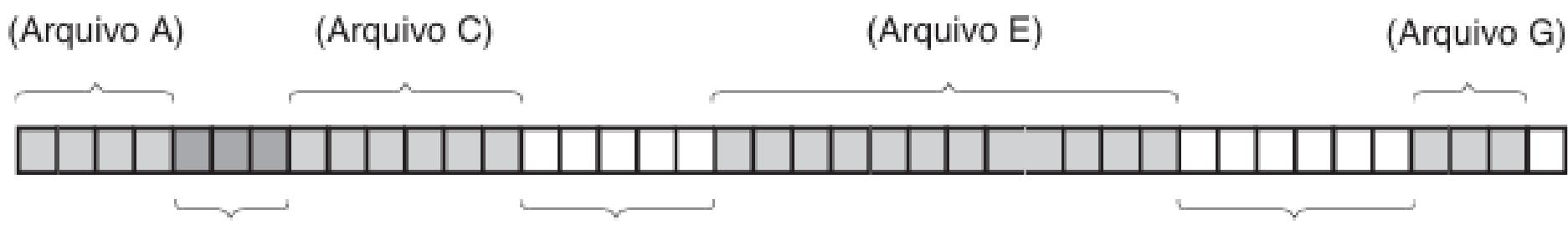
O super-bloco é um dos possíveis elementos de um sistema de arquivos, o qual contém todos os parâmetros chave sobre o sistema



Alocação contígua



(a)



(b)

01 Tipo mais simples de alocação

02 Armazenamento em sequencia
ininterrupta

Desvantagens

- Fragmentação do disco com o passar do tempo
- Mal uso do espaço
- Experiência do usuário ruim

Vantagens

- Leitura rápida
- Fácil implementação
- Fácil mapeamento

Implementação de diretórios

01 A entrada de diretório fornece a informação para encontrar os blocos de disco

02 Principal função é o mapear o nome do arquivo

O nome mapeado é em ASCII e a informação mapeada é necessária para localizar dados

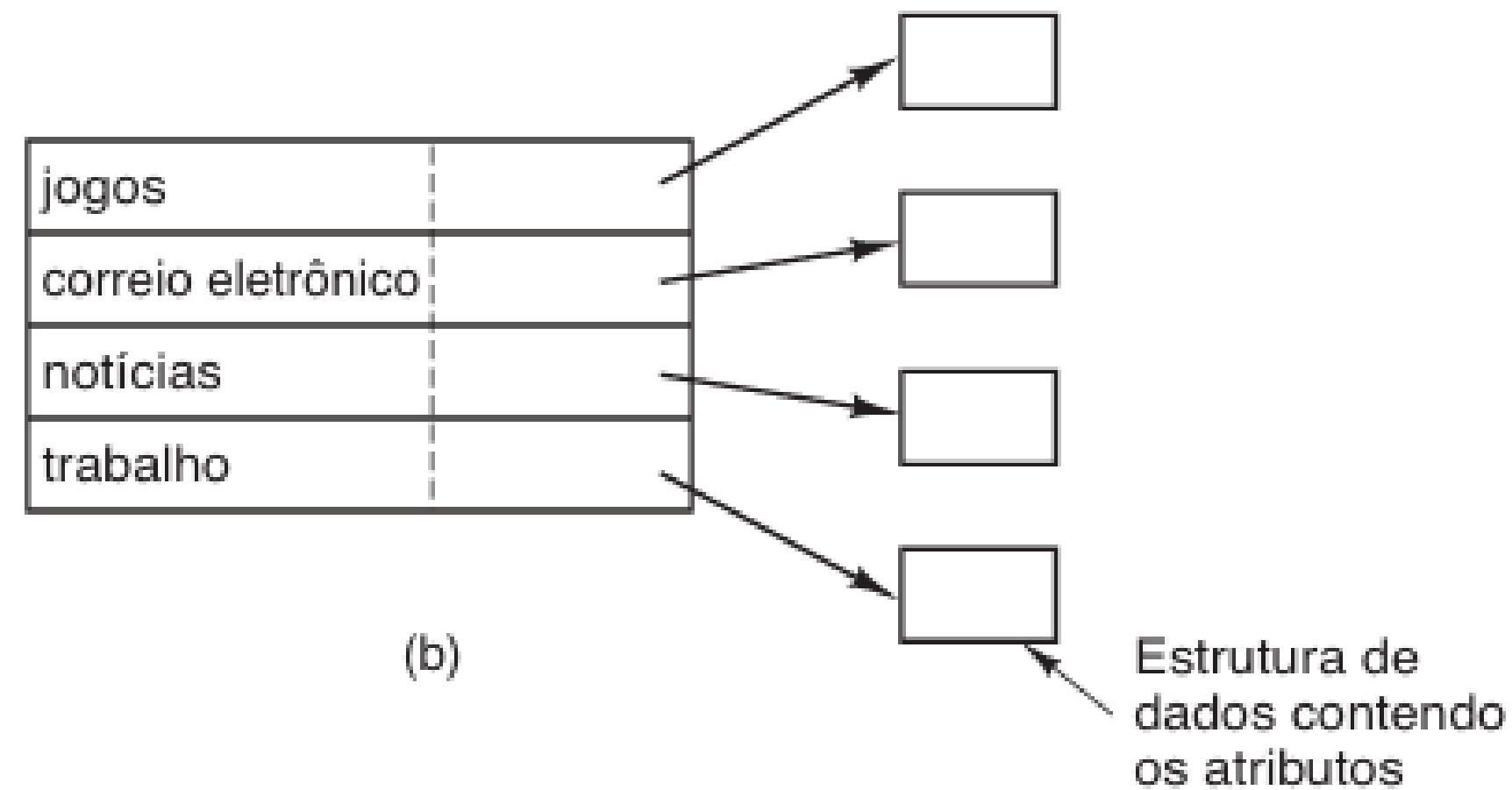
03 A informação depende do sistema

A informação varia de formas de acordo com as diferentes implementações, a exemplo da alocação contígua, lista encadeadas e i-nodes.

Implementação de diretórios

jogos	atributos
correio eletrônico	atributos
notícias	atributos
trabalho	atributos

(a)



Implementação de diretórios

Atributos e nomes de caminhos

01 Onde os atributos onde devem armazenados

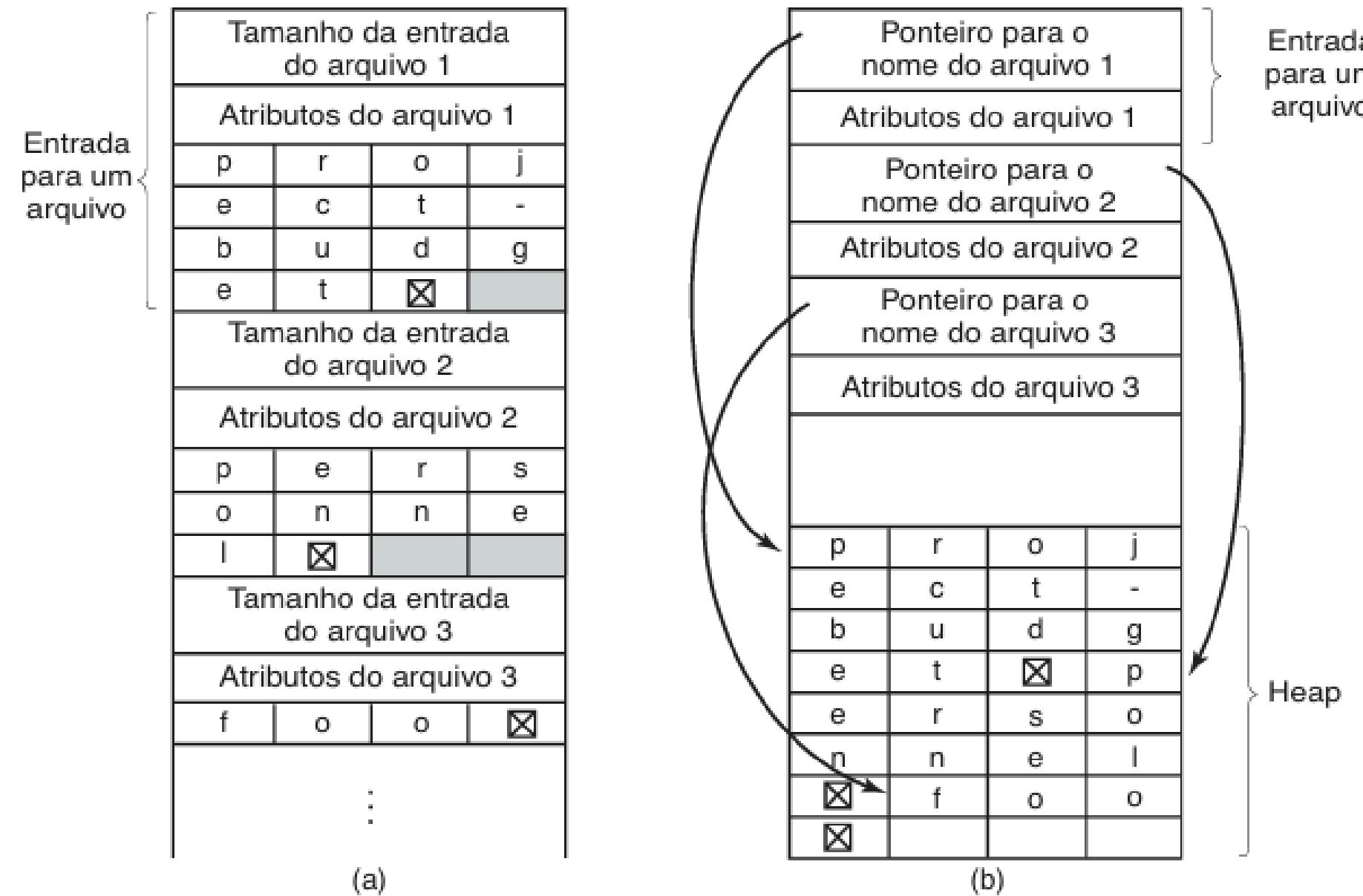
O sistema de arquivos deve manter os atributos de arquivos armazenados. Assim, torna-se necessário um lugar para tal.

02 A alternativa mais simples é direto na entrada do diretório

03 Há mais de um tipo de lista de entradas de diretório

Podem ser de tamanho fixo ou conter uma parte fixa que indica o tamanho da entrada de fato. Também é possível colocar os nomes dos diretórios no heap

Implementação de diretórios

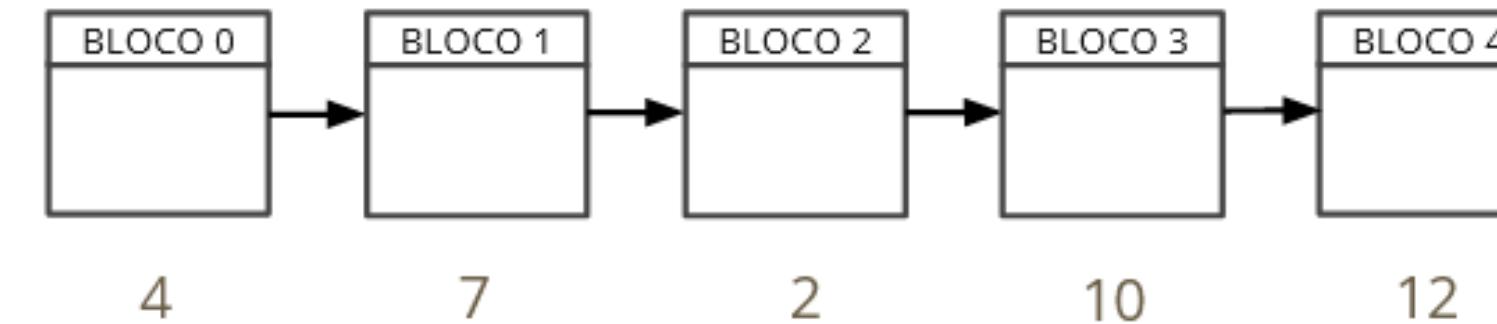


Alocação por lista encadeada

01 Cada arquivo é uma lista encadeada

02 Primeira palavra de cada bloco reservada para armazenar o ponteiro do próximo bloco

Arquivo A:



Desvantagens

- Acesso lento
- Menos eficiente
- Tamanho peculiar que não é potência de dois

Vantagens

- Uso de todos os blocos
- Diminuição da fragmentação do disco
- Facilidade de acesso do diretório

Alocação por lista encadeada com tabela de memória

- 01** As palavras dos ponteiros são armazenadas em uma tabela de memória, chamada **FAT(File Allocation Table)**.
- 02** O encadeamento é concluído com uma marcação especial, como o -1 utilizado na tabela

FAT:	
1	
2	5
3	
4	7
5	8
6	
7	
8	-1

arquivo começa

arquivo termina

Desvantagens

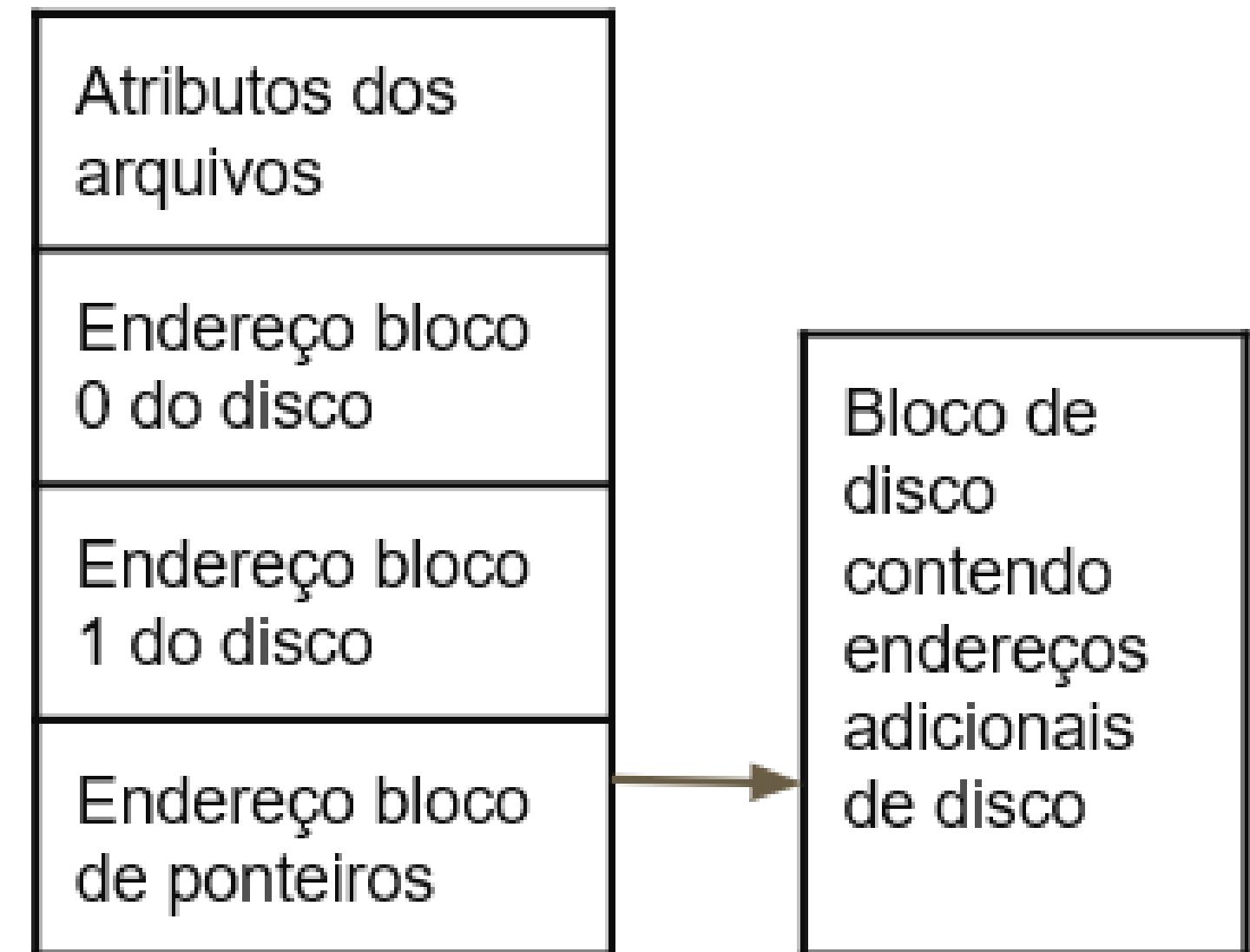
- Uso constante da memória
- Uso ruim em discos grandes
- Tabela inteira na memória

Vantagens

- Uso de todos os blocos
- Diminuição da fragmentação do disco
- Acesso aleatório mais fácil

I-nodes

- 01** Nesse método associa-se cada arquivo à uma estrutura de dados chamada i-node, que lista os endereços de cada bloco do arquivo.
- 02** Diferente da lista encadeada cada i-node só é aberto se o arquivo correspondente a ele for aberto.



Desvantagens

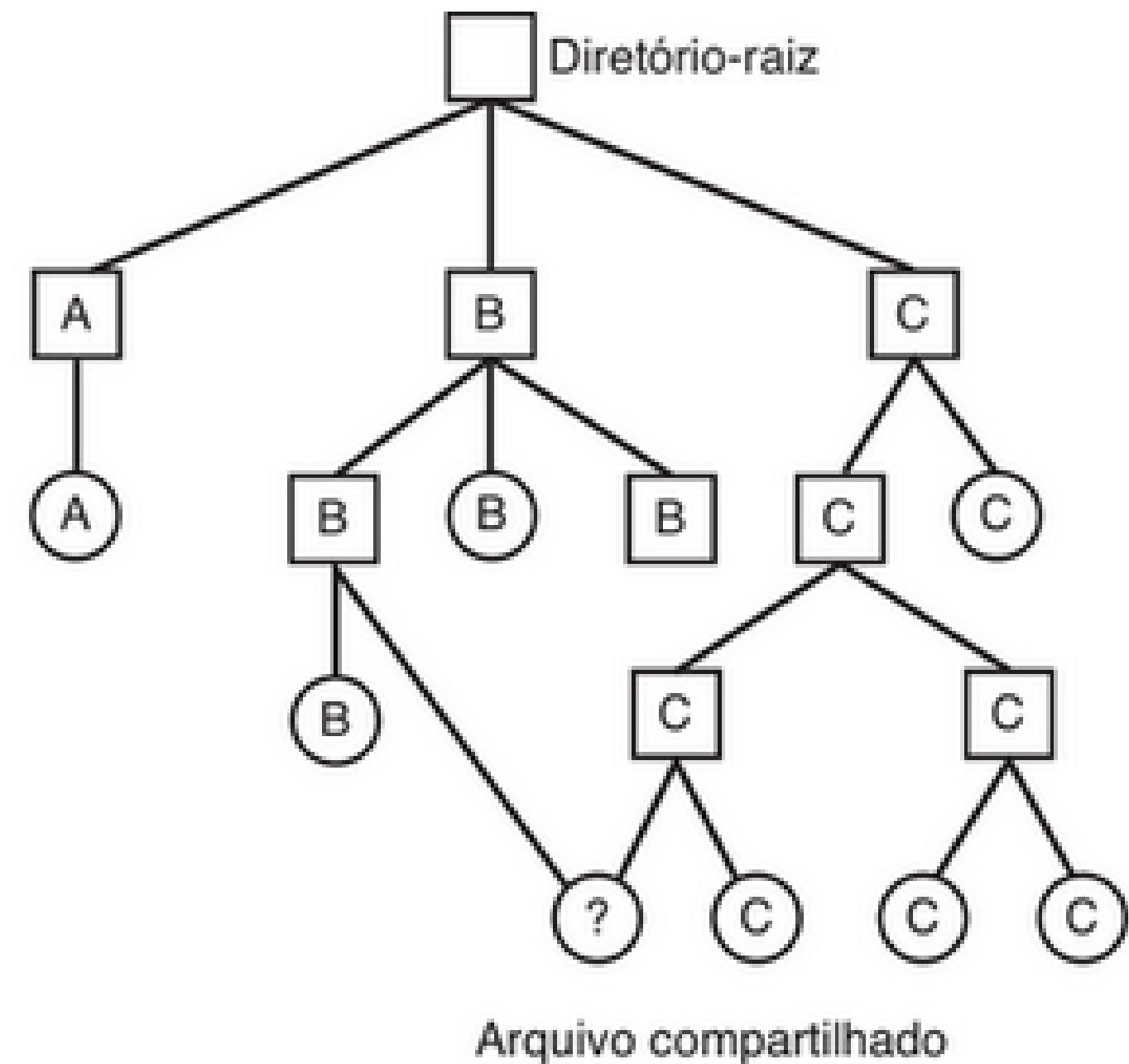
- Limite de tamanho do arquivo
- Complexidade de uso

Vantagens

- Tabela na memória é somente as que estão sendo usadas
- Menos memória usada
- Todos os blocos são usados

Arquivos compartilhados

- 01** Quando há vários usuários compartilhando um mesmo arquivo, muitas vezes é necessário que esse arquivo esteja em diretórios diferentes de usuários diversos.
- 02** A conexão entre o diretório de um usuário e um arquivo compartilhado é chamada de ligação.
- 03** O sistema de arquivos é um Gráfico Acíclico Orientado.



Gerenciamento e otimização



GERENCIAMENTO DE ESPAÇO EM DISCO

- 01** Existem duas estratégias gerais para armazenar um arquivo de n bytes:
 - são alocados n bytes consecutivos de espaço;
 - o arquivo é dividido em uma série de blocos, contíguos ou não.
- 02** Quase todos os sistemas de arquivo os dividem em blocos de tamanho fixo que não são necessariamente adjacentes.
- 03** Quanto ao tamanho do bloco, deve-se ter em mente informações sobre a distribuição do tamanho do arquivo.

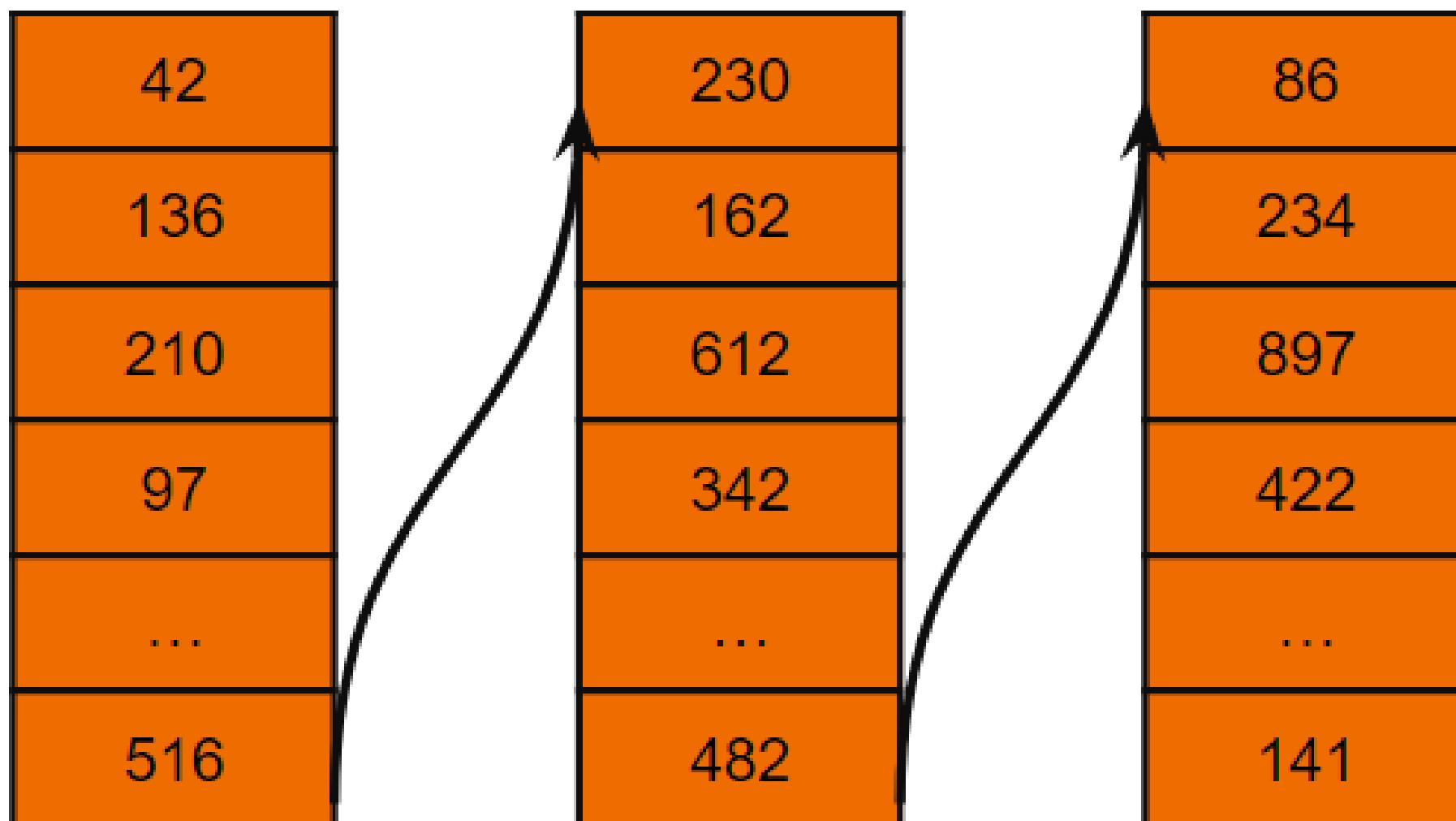
MONITORAMENTO DE BLOCOS LIVRES

Pode ser feito por dois métodos amplamente utilizados:

- lista encadeada de blocos livres de disco;
- mapa de bits.



Lista encadeada



- 01** Cada bloco contém identificadores de outros blocos livres;
- 02** Uma entrada é reservada para o ponteiro do bloco seguinte;
- 03** Com um bloco de 1 KB e um número de bloco de disco de 32 bits, cada bloco na lista livre contém os números de 255 blocos livres.

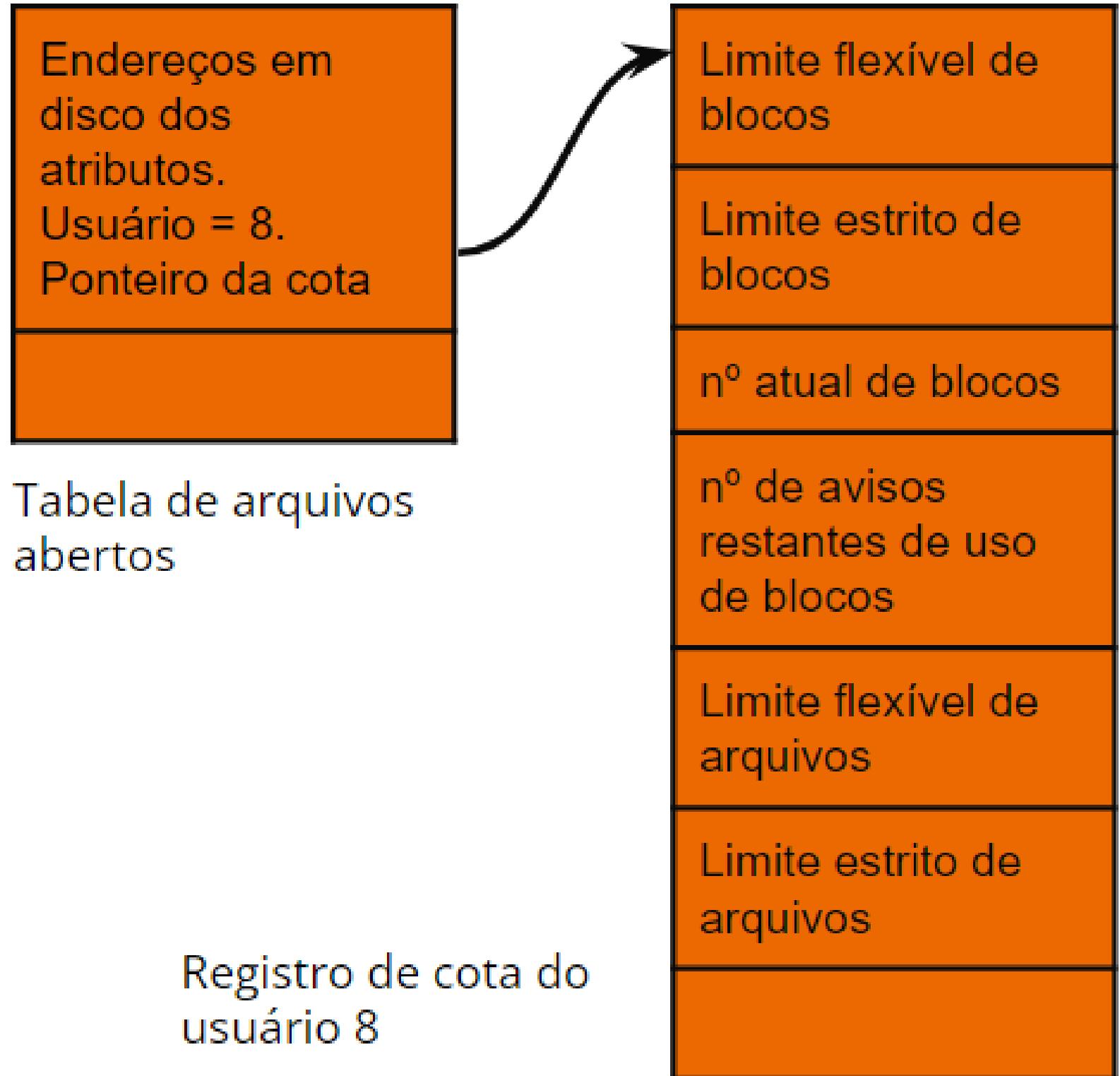
Mapa de Bits

1001101101101100
011011011110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
...
110111101110111

- 01** Blocos livres são representados por 1s e blocos alocados por 0s (ou vice-versa);
- 02** Usa 1 bit para cada bloco de disco;
- 03** Em geral, exige menos espaço do que a lista encadeada.

COTAS DE DISCO

- 01 Utilizadas para limitar o uso do espaço de disco por usuários;
- 02 Sistema operacional garante que os usuários não excedam suas respectivas cotas;
- 03 Limite flexível pode ser excedido, mas o estrito não.



Backup



São cópias de segurança do sistema de arquivos.



Empresas, em geral, oferecem suporte e costumam realizar backups frequentes dos dados.



O Windows, por exemplo, apresenta a “lixeira”, que contém arquivos que foram removidos.

CONSISTÊNCIA DE SISTEMA DE ARQUIVOS

- 01 A maioria dos sistemas operacionais apresenta um programa que confere a consistência de arquivos.
 - Ex: Windows apresenta o sfc e o UNIX tem o fsck.
- 02 Existem dois tipos de verificação de consistência: de blocos e de arquivos.

CONSISTÊNCIA DE BLOCOS

- 01** Programa constrói duas tabelas: uma monitora quantas vezes o bloco está presente em um arquivo e outra quantas vezes o bloco aparece na lista de livres;
- 02** Se o sistema for consistente, cada bloco terá 1 na primeira ou na segunda tabela.

CONSISTÊNCIA DE BLOCOS

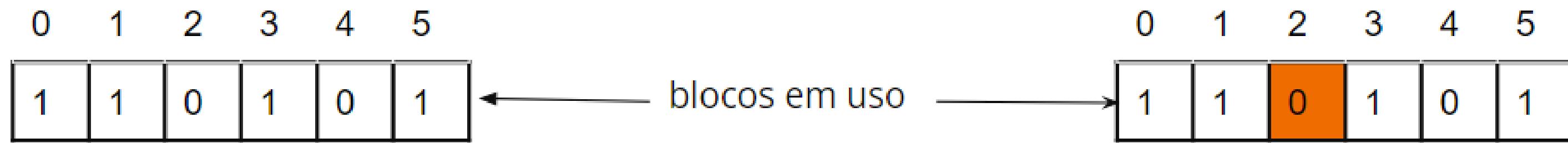


Imagen 1 (consistente)

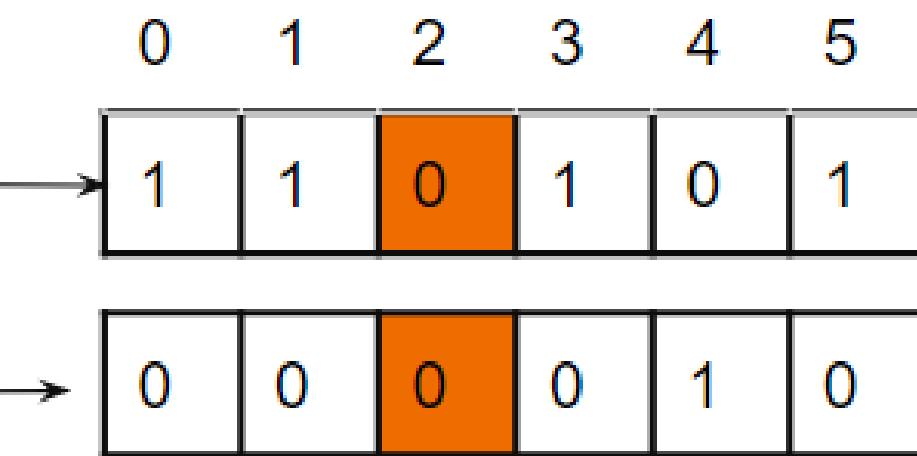


Imagen 2 (bloco desaparecido)

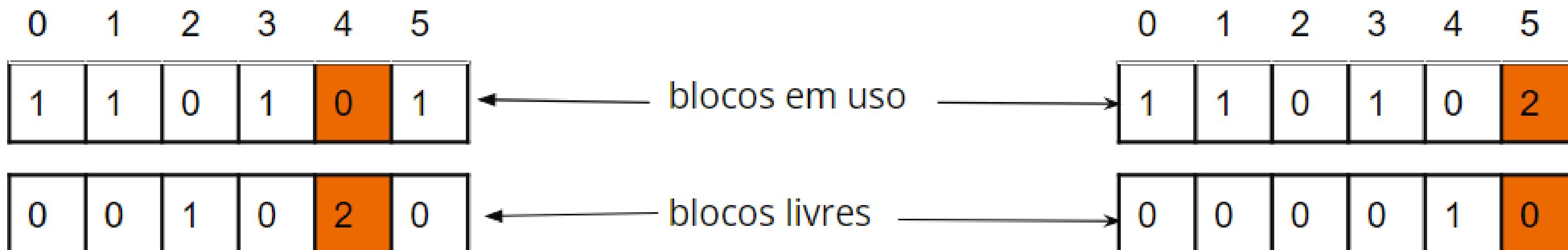


Imagen 3 (Bloco duplicado na lista de livres)

Imagen 4 (bloco de dados duplicado)

CONSISTÊNCIA DE ARQUIVOS

- 01** O verificador confere o sistema de diretórios;
- 02** Para cada i-node em cada diretório, o verificador incrementa um contador
- 03** Depois, ele compara uma lista que mostra quantos diretórios tem cada arquivo com as contagens de ligações armazenadas nos i-nodes;

DESEMPENHO

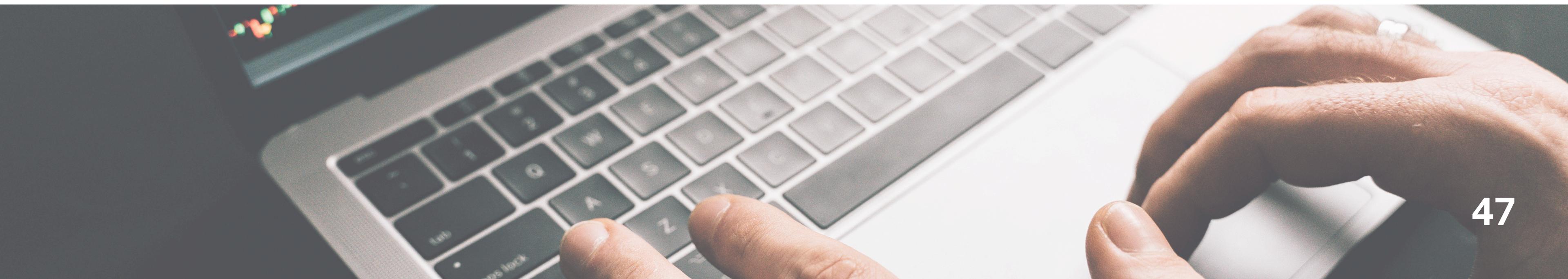
- 01 Em geral, o tempo para ler um bloco no disco é, no mínimo, 4 vezes mais lento do que ler uma palavra na memória.
- 02 Devido a essa diferença de tempo, muitos sistemas de arquivos foram projetados com otimizações para melhorar o desempenho.
- 03 As principais delas são: cache de blocos, leitura antecipada de blocos e redução do movimento do braço do disco.

Segurança

Segurança de Dados

O que é?

A segurança de dados é fundamental para proteger informações confidenciais e garantir a integridade e disponibilidade dos sistemas. Essa abordagem abrange diversos aspectos, desde a prevenção contra malware até a conformidade regulatória.



Confidencialidade

A confidencialidade dos dados é crucial para evitar o acesso não autorizado a informações sensíveis. Isso envolve o uso de criptografia, controles de acesso e políticas rígidas de segurança para garantir que apenas pessoas autorizadas possam acessar os dados



Integridade dos dados

A integridade dos dados é fundamental para assegurar que as informações permaneçam precisas, completas e confiáveis ao longo do tempo. Isso requer a implementação de mecanismos de backup, redundância e validação de dados para prevenir alterações não autorizadas.



Disponibilidade de Dados

A disponibilidade dos dados é essencial para garantir que as informações estejam acessíveis quando necessário. Isso envolve a implementação de sistemas de redundância, planos de recuperação de desastres e monitoramento constante para evitar interrupções de serviço.



Prevenção contra malware

01 Antivírus

Uso de software antívirus atualizado e eficaz para detectar e neutralizar ameaças maliciosas

02 Atualizações

Manter todos os sistemas e aplicativos atualizados para corrigir vulnerabilidades conhecidas

03 Conscientização

Treinar os usuários sobre boas práticas de segurança para evitar a disseminação de malware

Conformidade Regulatória



Legislação

Estar em conformidade com as leis e regulamentações vigentes em relação à proteção de dados



Certificações

Obter certificações de segurança reconhecidas, como ISO 27001 ou SOC 2, para demonstrar compromisso com a segurança



Auditórias

Realizar auditórias regulares para garantir o cumprimento dos requisitos legais e normativos

Proteção Contra Acesso Não Autorizado

Autenticação

Implementar métodos de autenticação robustos, como login com senha, autenticação de dois fatores e biometria.

Controle de Acesso

Estabelecer políticas de controle de acesso para limitar o acesso aos dados apenas a pessoas autorizadas.

Monitoramento

Monitorar constantemente as atividades de acesso e detectar possíveis tentativas de invasão.

Segmentação

Segmentar a rede e os sistemas de informação para restringir o acesso a áreas específicas.

Recuperação de Desastres

01 Backup

Criar backups regulares e armazená-los em locais seguros para garantir a recuperação dos dados em caso de desastres.

02 Replicação

Implementar soluções de replicação de dados em sites remotos para manter a disponibilidade em caso de incidentes.

03 Plano de Recuperação

Desenvolver e testar um plano de recuperação de desastres para garantir a retomada das operações de forma rápida e eficaz.

Malware e Mecanismos de Proteção



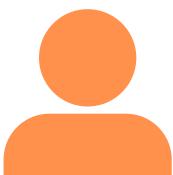
Antivírus

Software especializado em detectar, bloquear e remover malware.



Firewall

Barreira de segurança que monitora e controla o tráfego de rede.



Anti-Malware

Ferramentas avançadas para identificar e eliminar ameaças complexas.



Gerenciamento de Atualizações

Manter o software atualizado para corrigir vulnerabilidades conhecidas.

REFERÊNCIAS

- 01 TANENBAUM, Andrew Stuart. Sistemas Operacionais Modernos. 4. ed. São Paulo: Pearson, 2010.
- 02 IMAGENS ARQUIVOS. Freepik, 2024. Disponível em: <https://br.freepik.com/>. Acesso em: 19, maio de 2024.
- 03 SILBERSCHATZ, Abraham; GAGNE, Greg; GALVIN, Peter B. Operating System Concepts. 9. ed. Hoboken: Wiley, 2012.
- 04 CARRIER, Brian. File System Forensic Analysis. Boston: Addison-Wesley Professional, 2005.

Reconhecimentos e Direitos Autorais

@autor: [Leônidas Serra, Isabela Oliveira, Paulo Arthur, Samuel Basílio]

@ contato: [leonidas.serra@discente.ufma.br, moreira.isabela@discente.ufma.br, paulo.alb@discente.ufma.br]

@data última versão: [14/06/2024]

@versão: 1.0

@outros repositórios: [URLs - apontem para o seus gits, se quiserem]

@Agradecimentos: Universidade Federal do Maranhão (UFMA), Professor Doutor Thales Levi Azevedo Valente, e colegas de curso.

@Copyright/License

Este material é resultado de um trabalho acadêmico para a disciplina SISTEMAS OPERACIONAIS, sobre a orientação do professor Dr. THALES LEVI AZEVEDO VALENTE, semestre letivo 2024.1, curso Engenharia da Computação, na Universidade Federal do Maranhão (UFMA). Todo o material sob esta licença é software livre: pode ser usado para fins acadêmicos e comerciais sem nenhum custo. Não há papelada, nem royalties, nem restrições de "copyleft" do tipo GNU. Ele é licenciado sob os termos da licença MIT reproduzida abaixo e, portanto, é compatível com GPL e também se qualifica como software de código aberto. É de domínio público. Os detalhes legais estão abaixo. O espírito desta licença é que você é livre para usar este material para qualquer finalidade, sem nenhum custo. O único requisito é que, se você usá-lo, nos dê crédito.

Copyright © 2024 Educational Material

Este material está licenciado sob a Licença MIT. É permitido o uso, cópia, modificação, e distribuição deste material para qualquer fim, desde que acompanhado deste aviso de direitos autorais.

O MATERIAL É FORNECIDO "COMO ESTÁ", SEM GARANTIA DE QUALQUER TIPO, EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS DE COMERCIALIZAÇÃO, ADEQUAÇÃO A UM DETERMINADO FIM E NÃO VIOLAÇÃO. EM HIPÓTESE ALGUMA OS AUTORES OU DETENTORES DE DIREITOS AUTORAIS SERÃO RESPONSÁVEIS POR QUALQUER RECLAMAÇÃO, DANOS OU OUTRA RESPONSABILIDADE, SEJA EM UMA AÇÃO DE CONTRATO, ATO ILÍCITO OU DE OUTRA FORMA, DECORRENTE DE, OU EM CONEXÃO COM O MATERIAL OU O USO OU OUTRAS NEGOCIAÇÕES NO MATERIAL.