SISTEMAS OPERACIONAIS

AULA 17 – GERENCIAMENTO DE ARQUIVOS, PARTE 2

Prof.ª Sandra Cossul, Ma.

RELEMBRANDO

- **Sistema de arquivos** permite armazenar e acessar dados e programas.
- **Atributos de arquivos**
  - nome, identificador, tipo, localização, tamanho, proteção, timestamp
- **Operações em arquivos**
  - Criar, abrir, escrever, ler, deletar, renomear, atualizar, etc.
- **Tipos de arquivos** (estrutura interna e programas de aplicação)
- **Métodos de acesso**
  - sequencial, direto, indexado ou acesso por chave, mapeado em memória

DIRETÓRIOS DE ARQUIVOS

- Associado ao sistema gerenciador de arquivos e uma coleção de arquivos temos o **diretório de arquivos.**

- **Contém informação sobre os arquivos, incluindo seus atributos, localização e propriedades.**

- Ponto de vista do usuário: diretório provê um mapeamento entre os nomes de arquivos.

- A **organização por diretórios** é o modo como o SO organiza logicamente os diversos arquivos contidos em um dispositivo físico de armazenamento.

ELEMENTOS DE UM DIRETÓRIO DE ARQUIVO

Basic Information	
File Name	Name as chosen by creator (user or program). Must be unique within a specific directory
File Type	For example: text, binary, load module, etc.
File Organization	For systems that support different organizations
Address Information	
Volume	Indicates device on which file is stored
Starting Address	Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
Size Used	Current size of the file in bytes, words, or blocks
Size Allocated	The maximum size of the file

ELEMENTOS DE UM DIRETÓRIO DE ARQUIVO

	Access Control Information
Owner	User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges.
Access Information	A simple version of this element would include the user's name and password for each authorized user.
Permitted Actions	Controls reading, writing, executing, and transmitting over a network
	Usage Information
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
Current Usage	Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, and whether the file has been updated in main memory but not yet on disk

OPERAÇÕES EM DIRETÓRIOS

- **Pesquisa –** permite a busca de um arquivo (por nome, por extensão)
- **Criar arquivo –** quando um novo arquivo é criado, uma nova entrada deve ser adicionada ao diretório
- **Deletar arquivo -** quando um novo arquivo é deletado, uma entrada deve ser removida do diretório
- **Listar diretório –** mostrar tudo ou uma parte do diretório (listar todos os arquivos com alguns atributos principais – tipo, tamanho, etc.)
- **Atualizar diretório –** como os atributos dos arquivos estão guardados no diretório, uma mudança nos atributos requer uma mudança na entrada do diretório correspondente.

ESTRUTURAS DE DIRETÓRIOS

- **Nível único**
- **Dois níveis**
- **Estruturado em árvores**

ESTRUTURAS DE DIRETÓRIOS
NÍVEL UNICO

- Formato mais **simples**
- Todos os arquivos estão no **mesmo diretório**
  - devem ter nome único
- **Limitações** ocorrem com o **aumento do número de arquivos** ou quando o sistema tem **mais de um usuário**

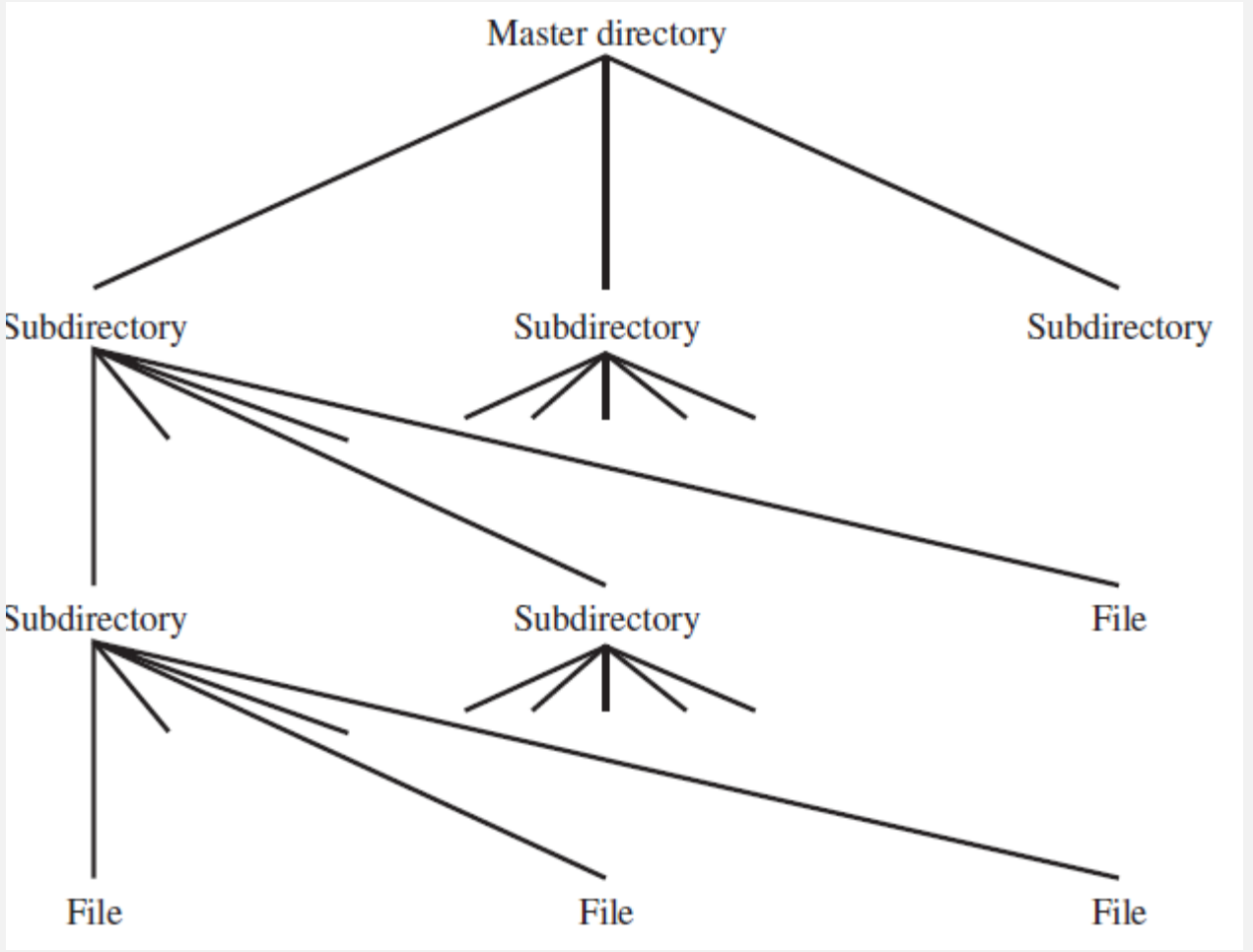
ESTRUTURAS DE DIRETÓRIOS
DOIS NÍVEIS

- Um **diretório para cada usuário** e um diretório **master**
- O **diretório master** tem uma **entrada** para cada diretório de usuário, fornecendo **endereço** e acesso a **informações de controle**
- Cada **diretório de usuário** é uma lista simples dos arquivos desse usuário
- **Limitação**: estruturar coleções de arquivos

ESTRUTURAS DE DIRETÓRIOS
ESTRUTURADO EM ÁRVORES

- Um **diretório master** com vários **diretórios de usuário**.
- Cada **diretório de usuário**, pode ter **subdiretórios** e **arquivos** como entradas
  - diretórios de usuário → nós
  - arquivos → folhas
- Quando se referencia a um arquivo, é necessário especificar seu **nome**, bem como o **diretório** onde ele se encontra, referência chamada **PATH**.
- **Diretório de trabalho** (user home directory – current directory para um processo)

ESTRUTURAS DE DIRETÓRIOS ESTRUTURADO EM ÁRVORES



PROTEÇÃO

- Quando uma informação é guardada no computador, queremos manter ela **segura de danos físicos** (confiabilidade) e **acesso indevido** (proteção)

- **Confiabilidade** → duplicar cópias de arquivos (muitos sistemas fazem isso de forma automática)

- A proteção de acesso aos arquivos visa possibilitar o **compartilhamento seguro de arquivos entre usuários**, quando desejado. Em geral, exige **concessão ou não de acessos** como leitura, gravação, execução e eliminação.

PROTEÇÃO

- Existem diferentes mecanismos de níveis de proteção:
  - **Lista de controle de acesso**
  - **Grupo de usuários**
  - **Senha de acesso**

PROTEÇÃO – LISTA DE CONTROLE DE ACESSO

- **Especifica nomes de usuários e os tipos de acesso permitidos para cada**
  - Quando um usuário requisita acesso a um arquivo, o SO checa a lista associada daquele arquivo. Se o usuário está listado, o acesso é permitido. Senão, acontece uma violação de proteção e o acesso não é liberado.

- A estrutura pode ter um **tamanho bastante extenso** considerando que um arquivo pode ter seu acesso compartilhado por diversos usuários
  - A pesquisa sequencial na lista pode causar **overhead**

PROTEÇÃO – GRUPO DE USUÁRIOS

- Associa **cada usuário** a um **grupo de usuários** que **compartilham arquivos e diretórios**

- Existem três níveis de proteção:
  - **owner** (dono) – usuário que criou o arquivo
  - **group** (grupo) – conjunto de usuários que compartilham o arquivo e precisam do mesmo acesso
  - **all** (todos) – todos os outros usuários do sistema

- Necessário associar o tipo do acesso (leitura, escrita, execução e eliminação) aos três níveis de proteção.

PROTEÇÃO – GRUPO DE USUÁRIOS

- **Exemplo:**
  - Sara esta escrevendo um novo livro com ajuda de João, Pedro e Rafael.
  - O arquivo do livro é mantido em um arquivo livro.txt, com a seguinte proteção:
    - Sara pode fazer qualquer operação no arquivo
    - João, Pedro e Rafael: podem ler e escrever (não podem deletar o arquivo)
    - Todos os outros usuários podem ler, mas não podem escrever no arquivo.

  - Grupo: João, Pedro e Rafael (nome do grupo deve ser associado ao arquivo e os direitos de acesso devem ser configurados de acordo com a autorização)

PROTEÇÃO – SENHA DE ACESSO

- Associar uma **senha** a cada **arquivo** para **limitar o acesso** a determinados arquivos/diretórios

- **Limitações:**
  - número de senhas para o usuário lembrar pode se tornar muito grande
  - se apenas uma senha for utilizada, pouca segurança
  - desvantagem de compartilhamento, pois além do dono, todos os demais usuários precisam conhecer a senha de acesso.

IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS

- **Arquivos são armazenados em discos** (memória secundária)
  - Dados em disco podem ser reescritos
  - Um disco pode acessar diretamente qualquer bloco de informação que está contido nele
  - Discos podem ser divididos em uma ou mais partições, com sistemas de arquivos independentes

- **Tópicos que serão abordados:**
  - Como os arquivos e diretórios são armazenados
  - Como o espaço de memória (blocos) são alocados
  - Questões de eficiência e possíveis problemas

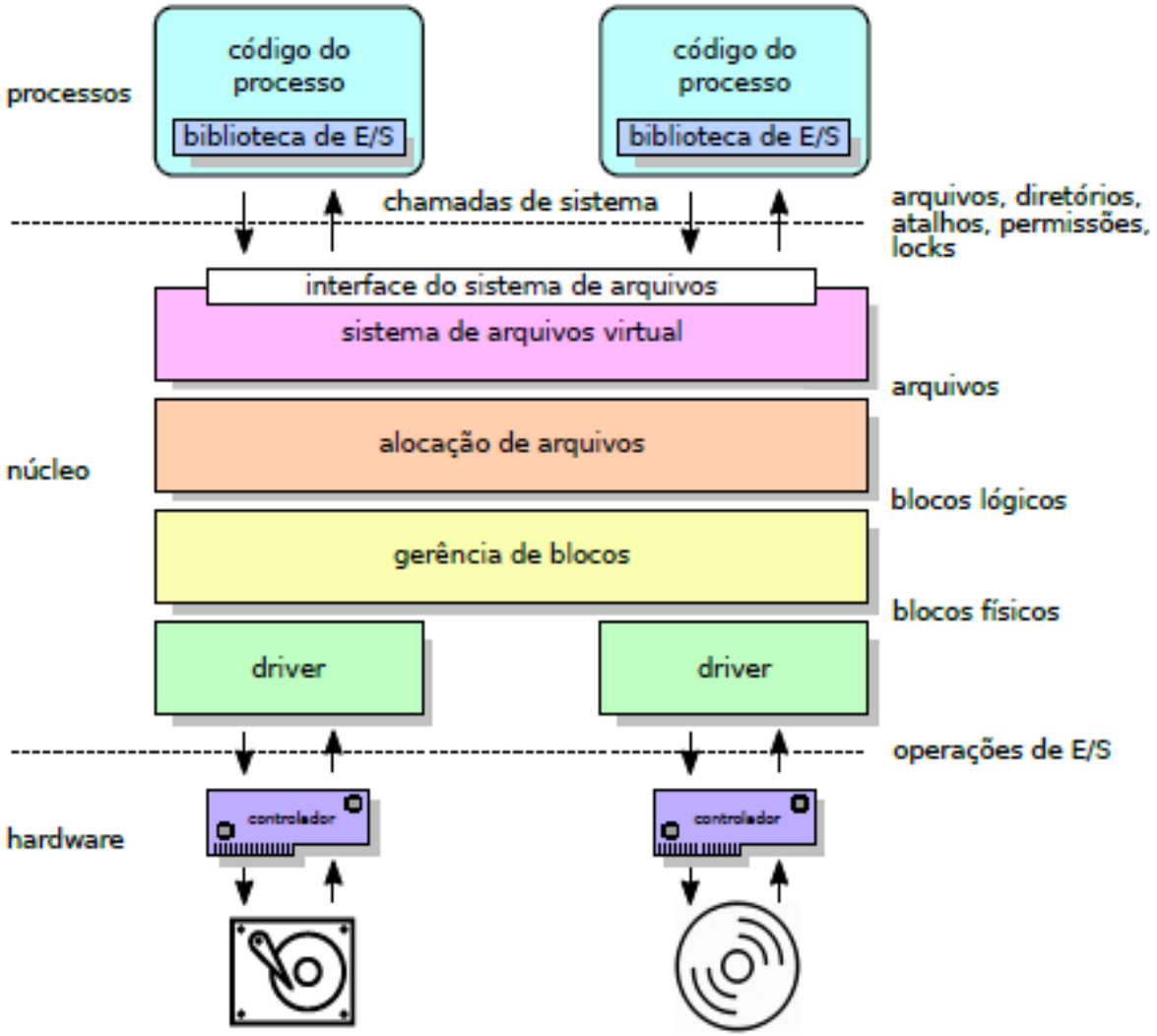
IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS

- Na memória secundária, um **arquivo** consiste de uma coleção de **blocos**
  - aumenta a eficiência de transferência de dados (E/S)
  - os discos são dispositivos orientados a blocos, as operações de leitura e escrita de dados são sempre feitas com blocos de dados, e nunca com bytes individuais;

- O SO é responsável pela **alocação de blocos** para os arquivos:
  - alocar memória secundária para os arquivos
  - gerenciar o espaço disponível para alocação

IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS

- Camadas da implementação da gerência de arquivos



IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS CONCEITOS

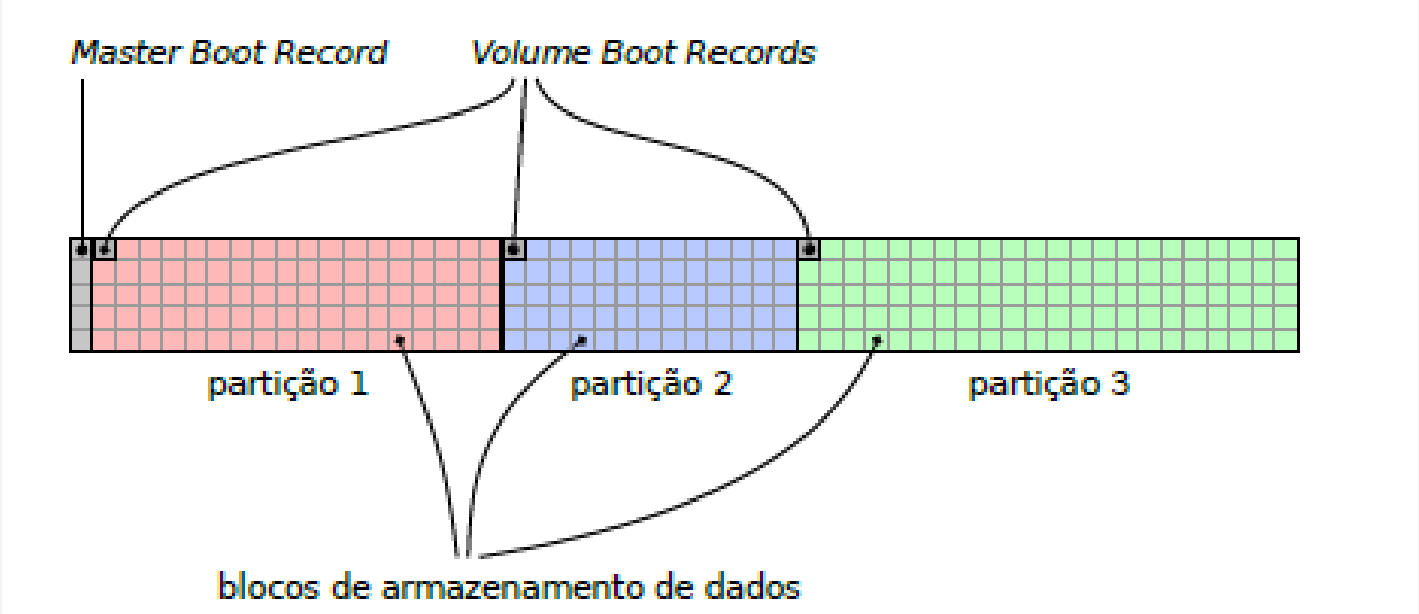
- **Bloco –** cada arquivo é visto como uma sequência de blocos que deve ser armazenada nos dispositivos de memória
- **Segmento –** conjunto contíguo de blocos alocados
- **Espaços de armazenamento –** discos rígidos (HD), discos óticos (CD-ROM, etc.), discos de estado sólido (baseados em memória flash, como pendrives USB, SSD)

- **Um disco é visto pelo SO como um grande vetor de blocos de dados de tamanho fixo, numerados sequencialmente.**
- Operações de leitura e escrita são feitas bloco a bloco

IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS CONCEITOS

- O espaço de armazenamento de cada dispositivo é dividido em:
  - **pequena área de configuração reservada,** no início do disco
    - tabela de partições (informações sobre o particionamento do dispositivo como número do bloco inicial, quantidade de blocos e outras informações)
    - pequeno código executável usado no processo de inicialização do SO (boot)
  - **uma ou mais partições,** que são espaços independentes
    - no início de cada partição há blocos reservados usados para a descrição do conteúdo daquela partição e para armazenar o código de lançamento do SO, se aquela for uma partição inicializável (bootable partition)
    - restante dos blocos para armazenamento de arquivos

IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS CONCEITOS



IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS CONCEITOS

- **Volume –** espaço de armazenamento de dados, do ponto de vista do SO.
- Cada volume corresponde a uma partição

- **Antes de ser usado, cada volume ou partição, deve ser formatado**, ou seja, preenchido com as estruturas de dados necessárias para armazenar arquivos, diretórios, atalhos e outras entradas.

- Cada volume pode ser formatado de forma independente e receber um sistema de arquivos distinto dos demais volumes.

IMPLEMENTAÇÃO DE SISTEMAS DE ARQUIVOS CONCEITOS

- **Montagem de volumes**
  - Para que o SO possa acessar os arquivos armazenados em um volume, ele deve ler os dados presentes em seu bloco de inicialização, que descrevem o tipo de sistema de arquivos do volume, e **criar as estruturas em memória que representam esse volume dentro do núcleo do SO**
  - Também deve definir um identificador para o volume, de forma que os processos possam acessar seus arquivos
  - Frequente em mídias removíveis
  - Cada volume normalmente é identificado por uma letra (“C”, “D”, etc.)

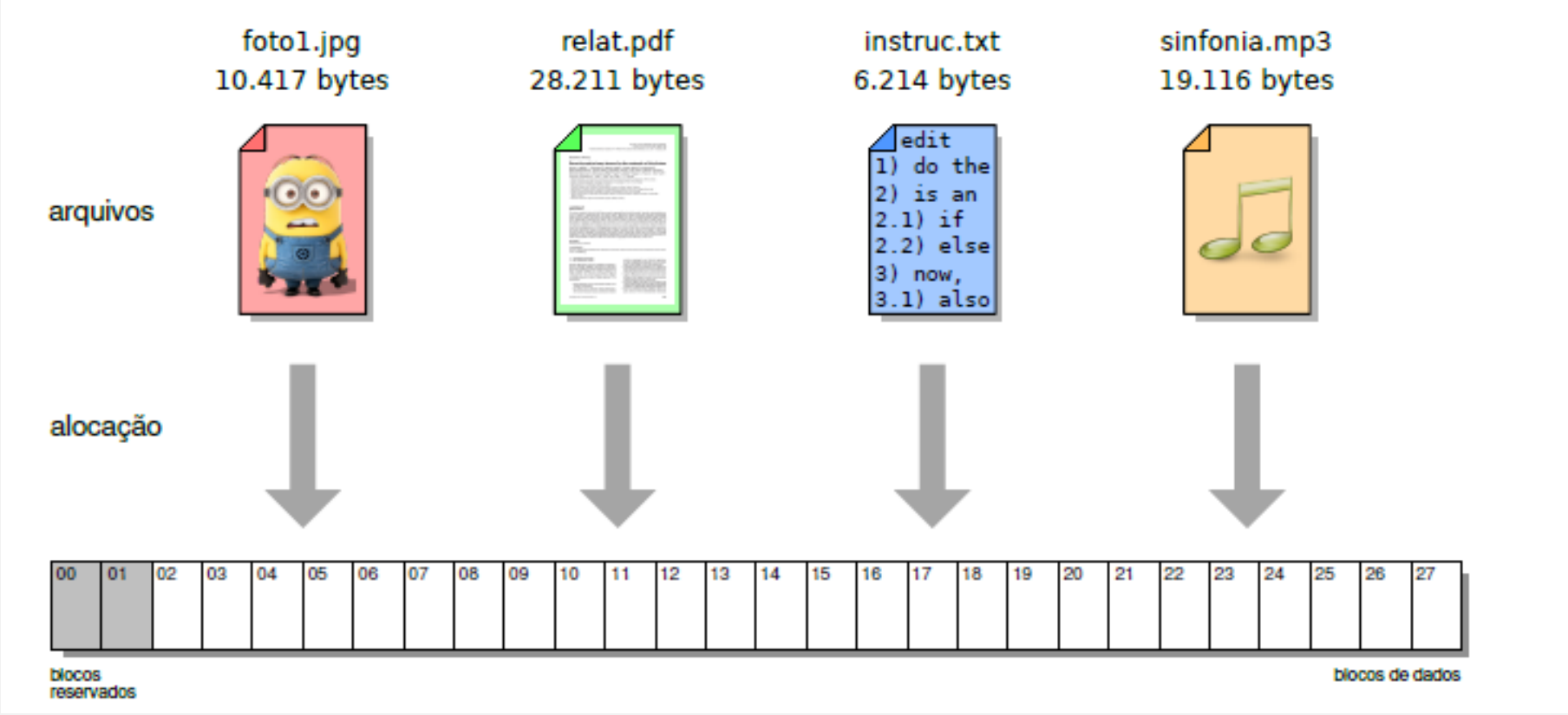
ALOCAÇÃO DE ARQUIVOS

- **Problema da alocação de arquivos:** alocar o conteúdo e os metadados dos arquivos dentro dos blocos de memória

- O conteúdo do arquivo deve estar alocado nestes blocos de forma a permitir um acesso rápido, flexível e confiável.

- Por isso, a forma de alocação dos arquivos nos blocos do disco tem um impacto importante sobre o desempenho e a robustez do sistema de arquivos

ALOCAÇÃO DE ARQUIVOS



MÉTODOS DE ALOCAÇÃO DE ARQUIVOS

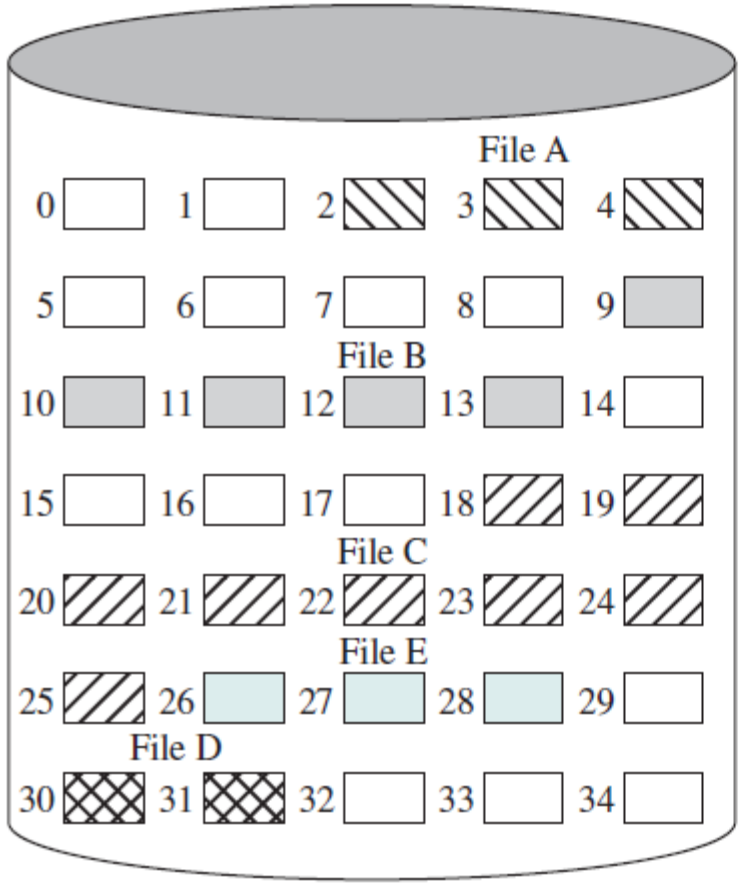
- **Alocação contígua**
- **Alocação encadeada**
- **Alocação indexada**

- O problema principal é como alocar espaço para arquivos de forma que o espaço de armazenamento é utilizado **eficientemente** e arquivos podem ser acessados de **forma rápida**.

- **Velocidade** (de acesso) **x robustez** (frente a erros) **x flexibilidade**

ALOCAÇÃO CONTÍGUA

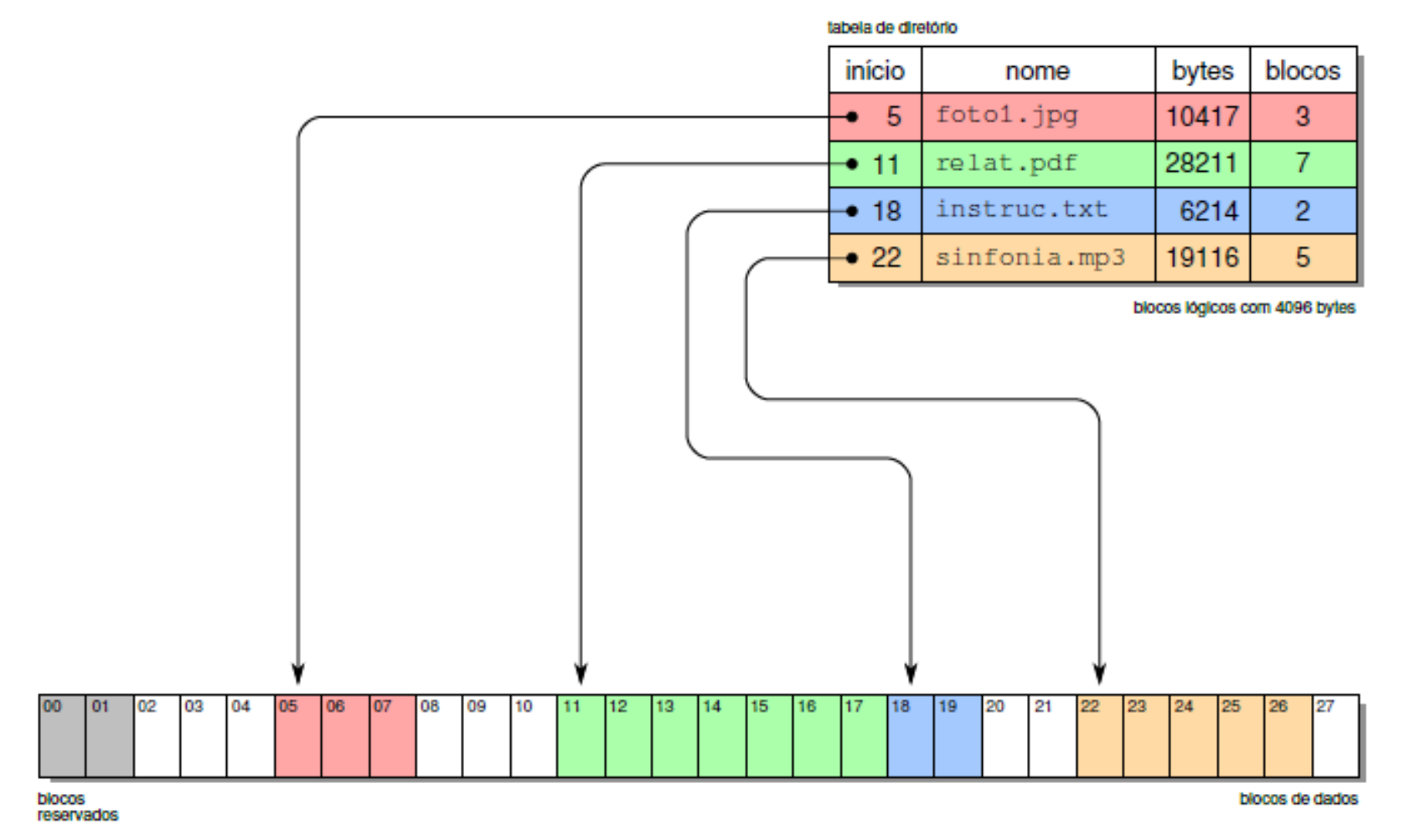
- Método mais simples
- **Ideia**: Armazenar arquivos de forma contínua no disco
- Localização do conteúdo do arquivo no disco é definida pelo endereço do primeiro bloco.



File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

ALOCAÇÃO CONTÍGUA



ALOCAÇÃO CONTÍGUA

- **Vantagens:**
  - Simplicidade: Somente o endereço do primeiro bloco e o número de blocos no arquivo são necessários
  - Desempenho para o acesso ao arquivo: Todos os arquivos podem ser lidos em um único acesso sequencial. O acesso aleatório também é rápido, pois a posição de cada byte do arquivo pode ser facilmente calculada a partir da posição do bloco inicial. Blocos próximos.
  - Robustez: Boa, pois caso um bloco do disco apresente defeito e não permita a leitura dos dados, apenas o conteúdo daquele bloco é perdido (conteúdo antes e depois pode ser acessado normalmente)

ALOCAÇÃO CONTÍGUA

- **Desvantagens:**
  - Baixa flexibilidade, pois o tamanho máximo de cada arquivo precisar ser conhecido no momento de sua criação.
  - Fragmentação externa, a medida que os arquivos são criados e destruídos, as áreas livres do disco vão sendo divididas em pequenas áreas isoladas (os fragmentos), que diminuem a capacidade de alocação de arquivos maiores
    - Técnicas de alocação best/first/next fit também podem ser utilizadas para atenuar este problema
    - Compactação – alto custo (operação lenta e arquivos não podem ser utilizados)
  - O reuso de espaço necessita de atualização da lista de espaços livres
  - Se não reusar os espaços, em algum momento a compactação deve ocorrer

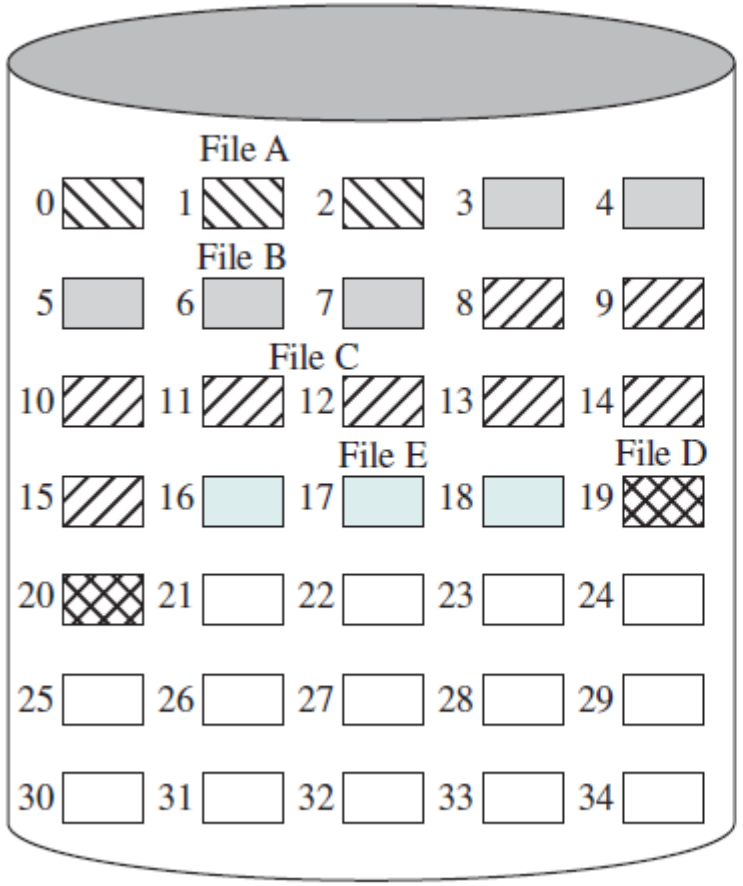
ALOCAÇÃO CONTÍGUA

- **Conclusão**
  - A baixa flexibilidade desta estratégia e a possibilidade de fragmentação externa limitam muito seu uso em SOs de propósito geral, nos quais arquivos são constantemente criados, modificados e destruídos.

  - Pode encontrar uso em situações específicas, nas quais os arquivos não sejam modificados constantemente e seja necessário rapidez nos acessos sequenciais e aleatórios aos dados
    - Discos multimídia (CD-ROM é um exemplo)

ALOCAÇÃO CONTÍGUA

- Exemplo anterior após compactação:



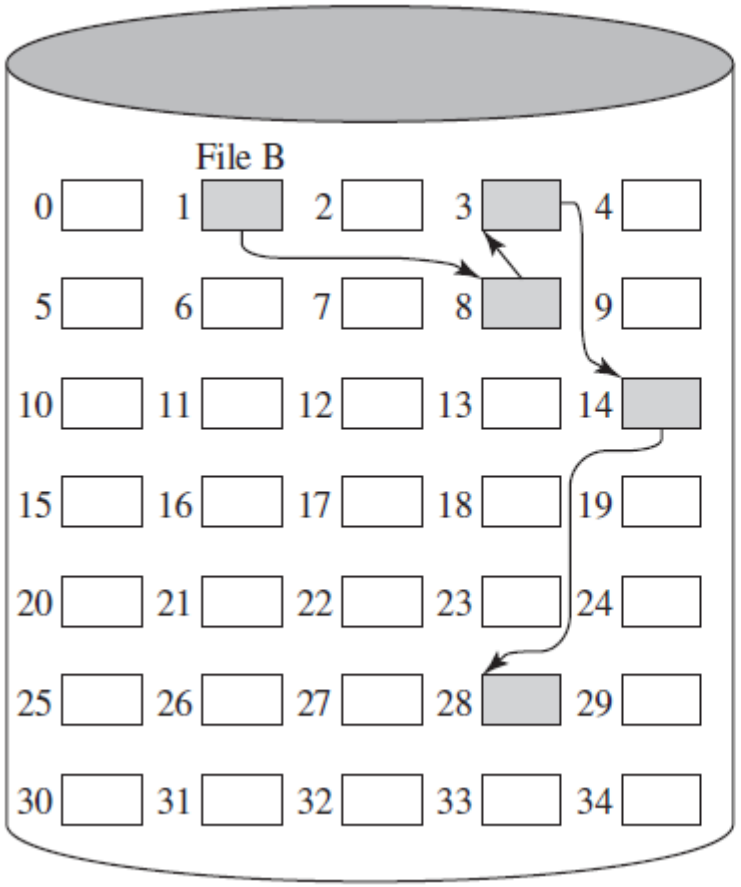
File allocation table

File name	Start block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

ALOCAÇÃO ENCADEADA

- Um arquivo pode ser organizado como um **conjunto de blocos ligados logicamente no disco**, independente da sua localização física

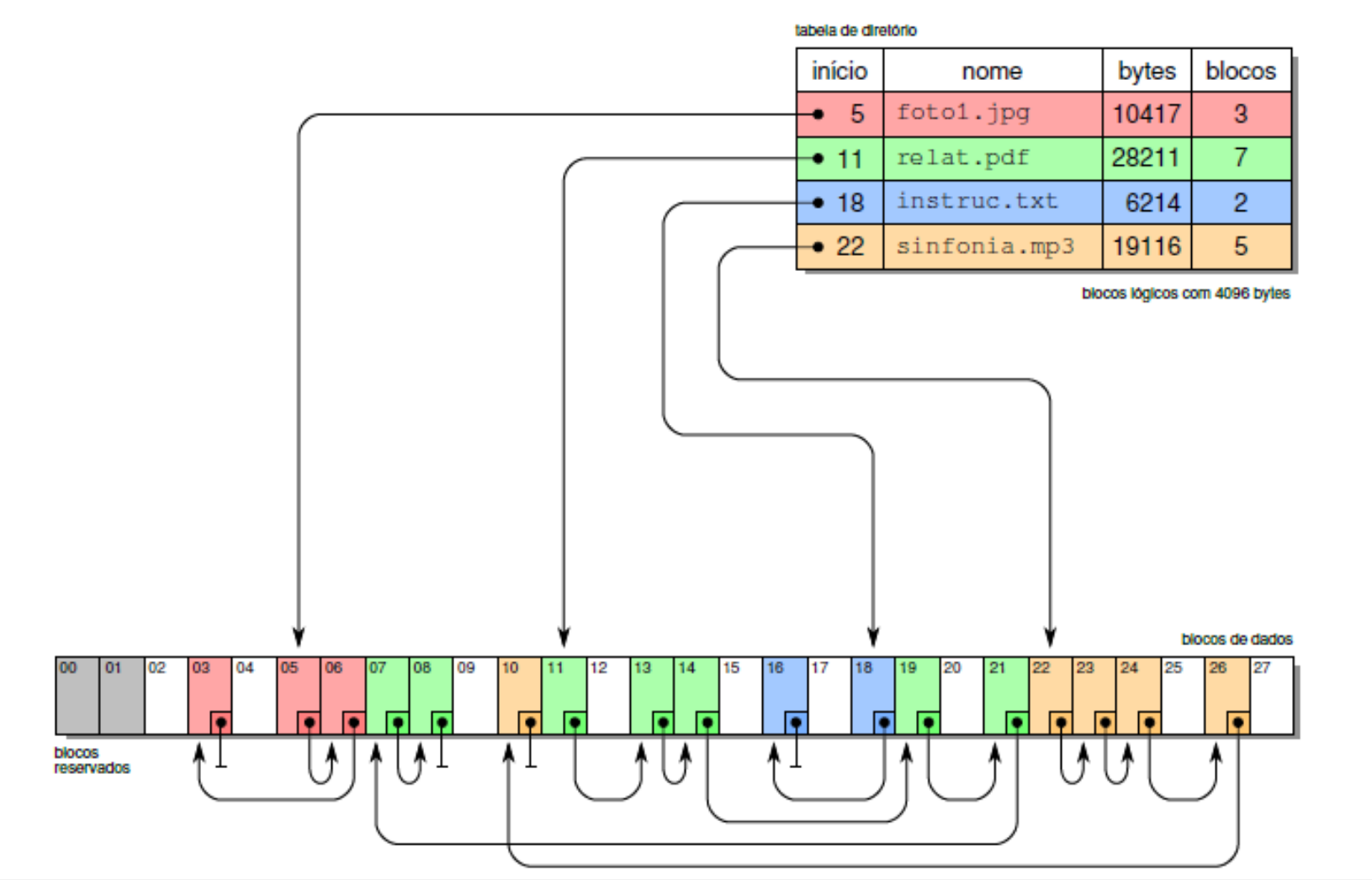
- Cada bloco possui um **ponteiro para o bloco seguinte** do arquivo e assim sucessivamente.



File allocation table

File name	Start block	Length
• • •	• • •	• • •
File B	1	5
• • •	• • •	• • •

ALOCAÇÃO ENCADEADA



ALOCAÇÃO ENCADEADA

- **Vantagens:**
  - Elimina a fragmentação externa – todos os blocos livres do disco podem ser utilizados sem restrições
  - Flexibilidade - permite que arquivos sejam criados sem a necessidade de definir o tamanho final
  - Acesso sequencial simples e rápido, cada bloco do arquivo contém um ponteiro para o próximo bloco. Isso pode ser prejudicado se os blocos estiverem muito espalhados no disco, exigindo muitas movimentações do disco

ALOCAÇÃO ENCADEADA

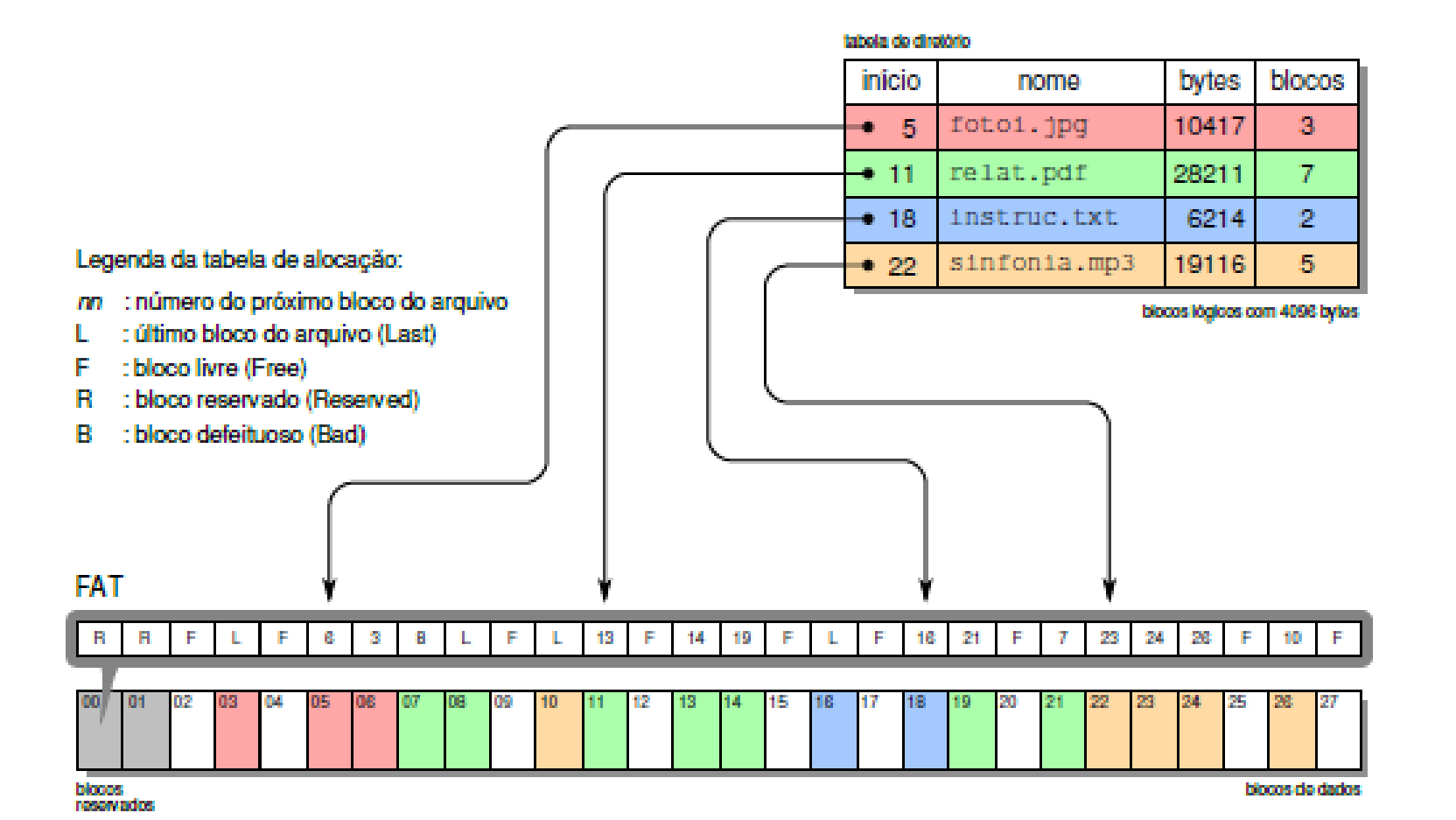
- **Desvantagens**:
  - Baixa robustez - dependência dos ponteiros de blocos; caso um bloco do arquivo seja corrompido ou se torne defeituoso, todos os blocos posteriores também ficarão inacessíveis
  - Acesso aleatório – caso necessitar acessar um bloco n, os n – 1 blocos anteriores terão de ser lidos em sequência, para poder encontrar os ponteiros que levam ao bloco desejado

ALOCAÇÃO ENCADEADA - FAT

- Os principais problemas da alocação encadeada simples são o baixo desempenho nos acessos aleatórios e a relativa fragilidade em relação a erros nos blocos do disco.
- Ambos os problemas provêm do fato de que os ponteiros dos blocos são armazenados nos próprios blocos, junto dos dados do arquivo.

- **FAT –** ponteiros são retirados dos blocos de dados e armazenados na Tabela de alocação de arquivos
  - cada entrada da tabela corresponde a um bloco do disco e contém um ponteiro
  - pode indicar blocos livres, reservados ou defeituosos

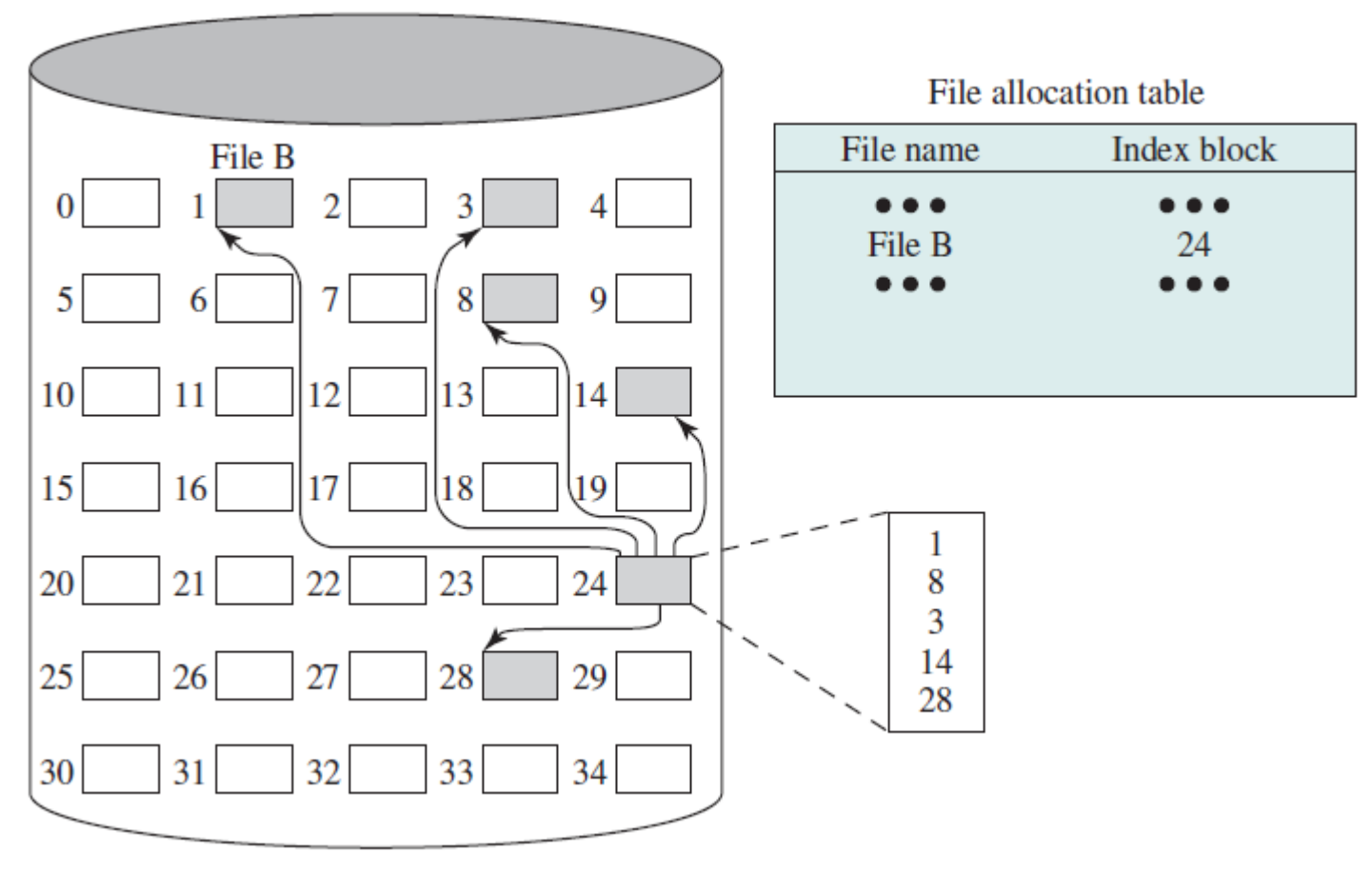
ALOCAÇÃO ENCADEADA - FAT



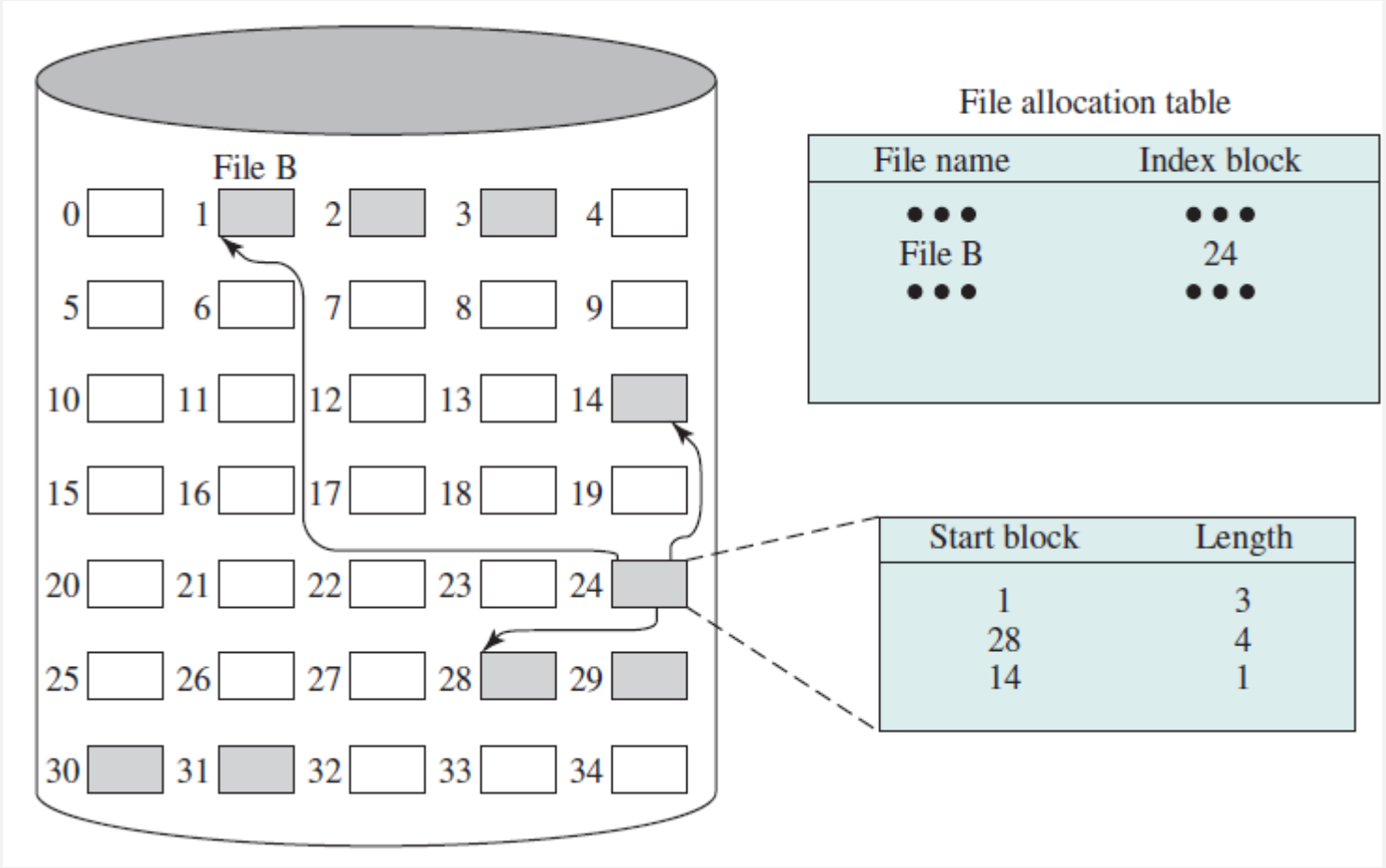
ALOCAÇÃO INDEXADA

- File allocation table contém um **índice de blocos do arquivo**
- O índice contém uma **entrada** para cada **segmento** alocada para o arquivo
- A alocação indexada soluciona o problema da alocação encadeada referente ao **acesso direto aos blocos** dos arquivos pois mantém **os ponteiros de todos os blocos do arquivo em uma única estrutura** denominada **bloco de índice**

ALOCAÇÃO INDEXADA



ALOCAÇÃO INDEXADA – COM BLOCOS DE TAMANHO VARIÁVEL



ALOCAÇÃO INDEXADA

- Cada entrada desse índice corresponde a um bloco do arquivo e aponta para a posição desse bloco no disco

- O índice de blocos de cada arquivo é mantido no disco em uma estrutura denominada nó de índice (i-node)
- **O i-node de cada arquivo contém o índice de seus blocos, e os principais atributos.**

- Tabela de i-nodes é mantida em uma área reservada do disco, separada dos blocos de arquivos.

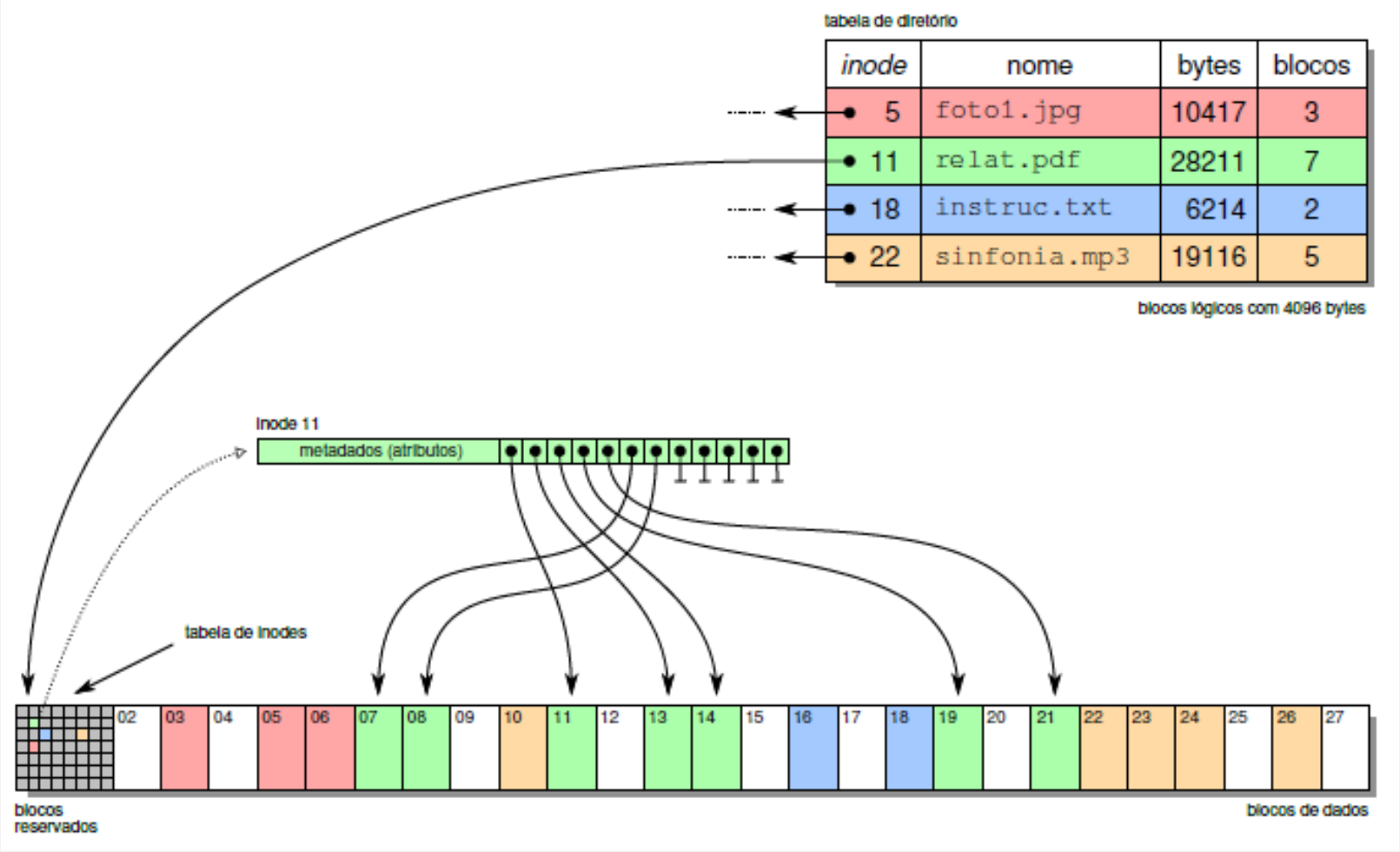
ALOCAÇÃO INDEXADA

- **Vantagens**
- Velocidade, tanto para acessos sequenciais quanto para acessos aleatórios a blocos (devido aos índices de ponteiros dos blocos presentes nos *i-nodes*)
- Robustez, defeitos em blocos de dados não afetam os demais blocos
- Flexibilidade, não apresenta fragmentação externa e permite o uso de todas as áreas livres do disco para armazenar dados

ALOCAÇÃO INDEXADA

- **Desvantagens**
- Defeitos nos metados *(i-nodes* ou os blocos de ponteiros), podem danificar grandes extensões do arquivo
  - Muitos sistemas implementam redundância de *i-nodes* e metadados para melhorar a robustez

ALOCAÇÃO INDEXADA



COMPARAÇÃO FORMAS DE ALOCAÇÃO DE ARQUIVOS

Estratégia	Contígua	Encadeada	FAT	Indexada
Rapidez (acesso sequencial)	Alta, os blocos do arquivo estão sempre em sequência no disco.	Alta, se os blocos do arquivo estiverem próximos no disco.	Alta, se os blocos do arquivo estiverem próximos no disco.	Alta, se os blocos do arquivo estiverem próximos no disco.
Rapidez (acesso aleatório)	Alta, as posições dos blocos podem ser calculadas sem acessar o disco.	Baixa, é necessário ler todos os blocos a partir do início do arquivo até encontrar o bloco desejado.	Alta, se os blocos do arquivo estiverem próximos no disco.	Alta, se os blocos do arquivo estiverem próximos no disco.
Robustez	Alta, blocos com erro não impedem o acesso aos demais blocos do arquivo.	Baixa: erro em um bloco leva à perda dos dados daquele bloco e de todos os blocos subsequentes do arquivo.	Alta, desde que não ocorram erros na tabela de alocação.	Alta, desde que não ocorram erros no *i-node* nem nos blocos de ponteiros.

COMPARAÇÃO FORMAS DE ALOCAÇÃO DE ARQUIVOS

Estratégia	Contígua	Encadeada	FAT	Indexada
Flexibilidade	Baixa, o tamanho máximo dos arquivos deve ser conhecido a priori; nem sempre é possível aumentar o tamanho de um arquivo existente.	Alta, arquivos podem ser criados em qualquer local do disco, sem risco de fragmentação externa.	Alta, arquivos podem ser criados em qualquer local do disco, sem risco de fragmentação externa.	Alta, arquivos podem ser criados em qualquer local do disco, sem risco de fragmentação externa.
Limites	O tamanho de um arquivo é limitado ao tamanho do disco.	O número de bits do ponteiro limita o número de blocos endereçáveis e o tamanho do arquivo.	O número de bits do ponteiro limita o número de blocos endereçáveis e o tamanho do arquivo.	Número de ponteiros no *i-node* limita o tamanho do arquivo; tamanho da tabela de *i-nodes* limita número de arquivos.

GERENCIAMENTO DO ESPAÇO LIVRE

- O SO possui uma **estrutura de dados** que armazena informações que possibilitam ao sistema de arquivos **gerenciar as áreas ou blocos livres**

- Nessa estrutura, geralmente uma **lista ou tabela**, é possível **identificar blocos livres** que poderão ser alocados por um novo arquivo.

- Quando um arquivo é eliminado, todos os **seus blocos são liberados** para a estrutura de espaços livres.

GERENCIAMENTO DO ESPAÇO LIVRE

- Técnicas de implementação da estrutura de espaços livres:
  - **Mapa de bits**
  - **Tabela de grupos de blocos livres**
  - **Lista de blocos livres**

MAPA DE BITS

- Utiliza um **vetor** contendo **um bit para cada bloco** em disco:
  - **Bit 0** → bloco livre
  - **Bit 1** → bloco em uso

- **Vantagens:**
  - facilidade de encontrar um ou um grupo contínuo de blocos livres
  - simples de implementar e é compacto

TABELA DE GRUPOS DE BLOCOS LIVRES

- Tabela contendo a **localização e o tamanho** de um conjunto de blocos livres contíguos no disco

- Cada entrada da tabela contém o **número do bloco inicial** e o **número de blocos no grupo**, de forma similar à alocação contígua de arquivos.

LISTA DE BLOCOS LIVRES

- Cada bloco livre contém um ponteiro para o próximo bloco livre do disco
  - estratégia similar à alocação encadeada de arquivos

- **Problema**: exige um acesso a disco para cada bloco livre requisitado
- Possível solução: armazenar em cada bloco livre um vetor de ponteiros para outros blocos livres (obtenção de um grande número de blocos livres a cada acesso a disco).

BIBLIOGRAFIA

- Tanenbaum, A. S. **Sistemas Operacionais Modernos.** Pearson Prentice Hall. 3rd Ed., 2009.
- Silberschatz, A; Galvin, P. B.; Gagne G.; **Fundamentos de Sistemas Operacionais.** LTC. 9th Ed., 2015.
- Stallings, W.; **Operating Systems: Internals and Design Principles**. Prentice Hall. 5th Ed., 2005.
- Oliveira, Rômulo, S. et al. **Sistemas Operacionais** - V11 - UFRGS. Disponível em: Minha Biblioteca, Grupo A, 2010.