BC1518 - Sistemas Operacionais

Aula 10: Sistemas de Arquivos

Material parcialmente baseado nos slides do Prof. José Artur Quilici Gonzalez

Tópicos desta aula

- Conceito de Arquivo
- Métodos de Acesso
- Estrutura de Diretórios
- Compartilhamento de Arquivos
- Alocação de espaço
- Gerenciamento de espaço livre

Fundamentos

- Processos durante a execução armazenam e recuperam informações da memória principal (em seu espaço de endereçamento)
  - Estão limitados ao tamanho do espaço de endereçamento
    - Muito pequeno para grandes sistemas (bancos, sistemas corporativos)
  - As informações são perdidas quando o processo termina ou se houver interrupção de energia

- Dispositivos de armazenamento de longo prazo:
  - Discos magnéticos (sequências lineares de blocos de tamanho fixo)
  - Fitas magnéticas
  - Discos óticos

Fundamentos

- Operações de leitura/escrita em disco são inconvenientes:
  - Como encontrar uma informação?
  - Quais blocos estão livres?
  - Como impedir que um usuário acesse informações de outros usuários?
- O SO oferece uma abstração: Arquivo
  - Oferece meios para armazenar informações no disco e para ler essas informações
  - Escondendo do usuário como e onde as informações estão armazenadas
  - E como é o funcionamento do disco

- Arquivo: é uma coleção (nomeada) de informações relacionadas, em armazenamento secundário
  - Unidade básica de armazenamento para armazenar e recuperar dados

Fundamentos

- Sistema de Arquivos: parte do SO que faz o gerenciamento de arquivos
  - Trata da estruturação, nomeação, acesso, manipulação, compartilhamento, proteção de arquivos
  - Oferece uma interface de programação conveniente para armazenamento em disco

- Nomeação de arquivos
  - Quando um arquivo é criado (por um processo), este recebe um nome
  - O nome é utilizado para que outros processos possam ler/escrever no arquivo

Conceito de arquivo

➢Um arquivo é uma coleção de dados armazenada em um dispositivo físico não-volátil, com um nome ou outra referência que permita sua localização posterior

➢Diferentes tipos de informações podem ser armzenadas

➢Em geral, os arquivos representam **dados, programas**

- Dados
  - numéricos
  - alfabéticos
  - alfanuméricos
  - binários
- Programas (fonte, objeto, executáveis)

➢Do ponto de vista de um usuário, os dados não podem ser escritos no armazenamento secundário a menos que estejam dentro de um arquivo → unidade básica de armazenamento

Atributos de arquivos

➢Em geral os sistemas operacionais associam algumas informações para cada arquivo, os atributos, que podem variar de um sistema para outro

- **Nome:** sequência de caracteres que identifica o arquivo (prog.c, relatório.txt, etc.)
- **Identificador:** identifica o arquivo dentro do sistema de arquivos
- **Tipo:** indica o formato dos dados em um arquivo, como texto, imagem, áudio, etc. Em geral, parte do nome do arquivo é usada para identificar o tipo (extensão): “.doc”, “.jpg”, “.mp3”, etc.
- **Posição:** ponteiro para a localização do arquivo no dispositivo de armazenamento
- **Tamanho:** tamanho atual do arquivo
- **Proteção:** controla quem pode ler, escrever ou executar
- **Hora**, **data e identificação do usuário:** dados para proteção, segurança, e monitoramento de uso
- Informações sobre arquivos são mantidas na **estrutura de diretório**, que é mantida em disco

Exemplos de atributos de arquivos

Atributo	Significado
Proteção	Quem tem acesso ao arquivo e de que modo
Senha	Necessidade de senha para acesso ao arquivo
Criador	ID do criador do arquivo
Proprietário	Proprietário atual
Flag de somente leitura	0 para leitura/escrita; 1 para somente leitura
Flag de oculto	0 para normal; 1 para não exibir o arquivo
Flag de sistema	0 para arquivos normais; 1 para arquivos do sistema
Flag de arquivamento	0 para arquivos com backup; 1 para arquivos sem backup
Flag de ASCII/binário	0 para arquivos ASCII; 1 para arquivos binários
Flag de acesso aleatório	0 para acesso somente sequencial; 1 para acesso aleatório
Flag de temporário	0 para normal; 1 para apagar o arquivo ao sair do processo
Flag de travamento	0 para destravados; diferente de 0 para travados
Tamanho do registro	Número de bytes em um registro
Posição da chave	Posição da chave em cada registro
Tamanho do campo-chave	Número de bytes no campo-chave
Momento de criação	Data e hora de criação do arquivo
Momento do último acesso	Data e hora do último acesso do arquivo
Momento da última alteração	Data e hora da última modificação do arquivo
Tamanho atual	Número de bytes no arquivo
Tamanho máximo	Número máximo de bytes no arquivo

Proteção do arquivo

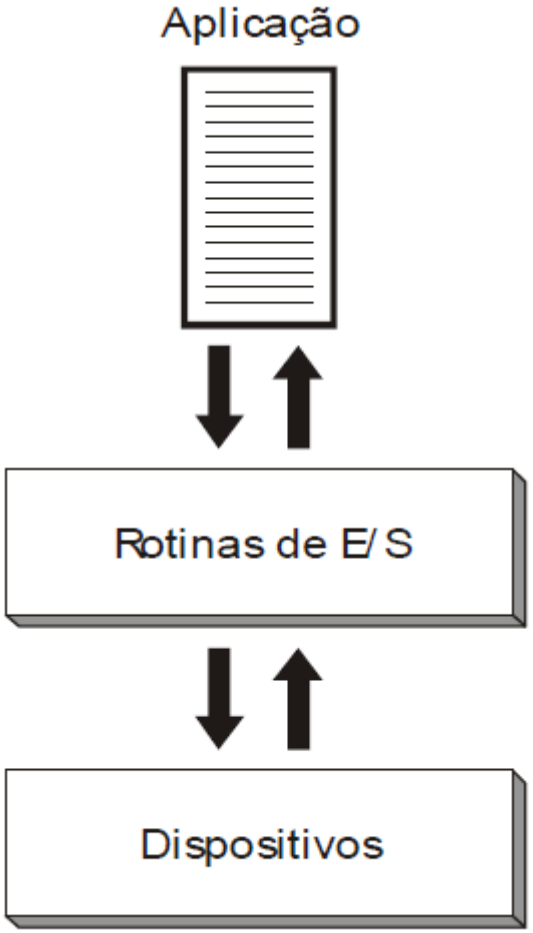
Flags são bits que controlam c

Usados em arquivos cujos reg

Informações sobre data e hora

Alguns atributos possíveis de arquivos.

Operações com arquivos



- O Sistema de Arquivos disponibiliza um conjunto de rotinas que permite à aplicações realizarem operações de E/S

- Essas operações são executadas através de chamadas de sistema (criar, ler, escrever, remover arquivos, etc.)

Operações com arquivos

- O SO fornece chamadas de sistema para realizar operações com os arquivos:
  - Criar: alocar espaço no dispositivo de armazenamento e definir seus atributos (nome, localização, permissões, etc.), criando-se uma entrada no diretório
  - Abrir: antes de usar um arquivo, o processo deve fazer abertura do arquivo; o sistema localiza o seu conteúdo no dispositivo de armazenamento e cria uma referência na memória
  - Ler: para a leitura, na chamada de sistema é preciso especificar o nome do arquivo e onde (na memória) o próximo bloco do arquivo deve ser colocado (permite transferir um bloco do arquivo no dispositivo de armazenamento para a área de memória especificada)
  - Escrever: para a escrita, na chamada de sistema é especificado o nome do arquivo e as informações a serem escritas no arquivo (transfere dados da memória para o arquivo no dispositivo de armazenamento)

Operações com arquivos (cont.)

- Fechar: o processo avisa o sistema quando termina de usar o arquivo, a fim de liberar espaço na memória
- Remover: quando o arquivo não é mais necessário este é removido do dispositivo de armazenamento, liberando o espaço ocupado

Tabela de arquivos abertos

- A maioria das operações de arquivo envolve a pesquisa no diretório a entrada associada ao arquivo (para obter a sua localização no dispositivo físico)
- O SO mantém uma tabela de arquivos abertos contendo informações de todos os arquivos abertos, na memória
  - Quando um arquivo é aberto, é criada uma entrada na tabela
  - Quando uma operação no arquivo é requisitada, a tabela é acessada, não requerendo a pesquisa no diretório
- Várias informações estão associadas a um arquivo aberto:
  - Ponteiro de Arquivo: ponteiro para localização da última leitura/escrita realizada pelo processo
  - Contador de arquivo aberto: contabiliza o número de vezes que o arquivo foi aberto por processos e permite remover a entrada do arquivo da tabela quando o contador chega a 0 (todos os processos fecharam o arquivo)
  - Localização em disco do arquivo: as informações para localizar o arquivo no disco são mantidas na memória, evitando a leitura do disco para cada operação de arquivo

Tipos de arquivos

- Arquivos regulares:
  - Arquivos ASCII: constituído por linhas de texto (cada linha termina com um caractere especial); podem ser visualizados e impressos como são e podem ser editados em qualquer editor de texto
  - Ex.: códigos-fonte de programas, páginas HTML, arquivos de configuração, etc.
  - Arquivo binário: qualquer arquivo que não é ASCII, possui uma estrutura interna conhecida pelos programas que os usam

- Diretórios: arquivos do sistema que mantém a estrutura do sistema de arquivos
- Arquivos especiais de caracteres: relacionados a E/S e usados para modelar dispositivos de E/S como terminais, impressoras e redes
- Arquivos especiais de blocos: são usados para modelar discos

UFABC

Tipos de arquivos

tipo de arquivo	extensão normal	função
executável	exe, com, bin ou nenhuma	programa em linguagem de máquina pronto para ser executado
objeto	obj, o	programa compilado, em linguagem de máquina, não vinculado
Código-fonte	c, cc, Java, pas, asm, a	Código-fonte nas diversas linguagens
lote	bat, sh	comandos ao interpretador de comandos
texto	txt, doc	dados textuais, documentos
processador de textos	wp, tex, rtf, doc	diversos formatos de processador de textos
biblioteca	lib, a, so, dll	bibliotecas de rotinas para programadores
impressão ou exibição	ps, pdf, jpg	arquivo ASCII ou binário em um formato para impressão ou exibição
arquivamento	arc, zip, tar	arquivos relacionados, agrupados em um arquivo, às vezes compactado, para arquivamento ou armazenamento
multimídia	mpeg, mov, rm	arquivo binário contendo informações de áudio e/ou vídeo

Estrutura de arquivos

➢A estrutura do arquivo define como os dados estão organizados internamente

Depende basicamente de seu propósito (ex.: arquivo texto ≠ arquivo executável)

➢Os arquivos podem ser estruturados de 3 maneiras:

Sequência (não estruturada) de bytes: não há uma estrutura lógica imposta pelo SO

- A aplicação/usuário deve definir a estrutura
- Ao SO não importa o conteúdo do arquivo, é visto apenas como uma sequência de bytes
- Principal vantagem: flexibilidade
- Todas as versões do Unix, MS-DOS eWindows usam essa estrutura

Sequência de registros: o arquivo é organizado em uma sequência de registros de tamanho fixo, cada um com uma estrutura interna

- Uma operação de leitura retorna um registro e uma operação de escrita sobrepõe ou concatena um registro
- Era muito comum nos computadores de grande porte (arquivos com registros de 80 caracteres (refletir os cartões perfurados) ou 132

Estrutura de arquivos

- Árvore (organização indexada): o arquivo é estruturado em uma árvore de registros (podem não ter o mesmo tamanho), cada um possui um campo-chave em uma posição fixa do registro
- A árvore é ordenada pelo campo-chave, a busca é feita pela chave (não precisa conhecer a posição do registro)
- Usado em computadores de grande porte

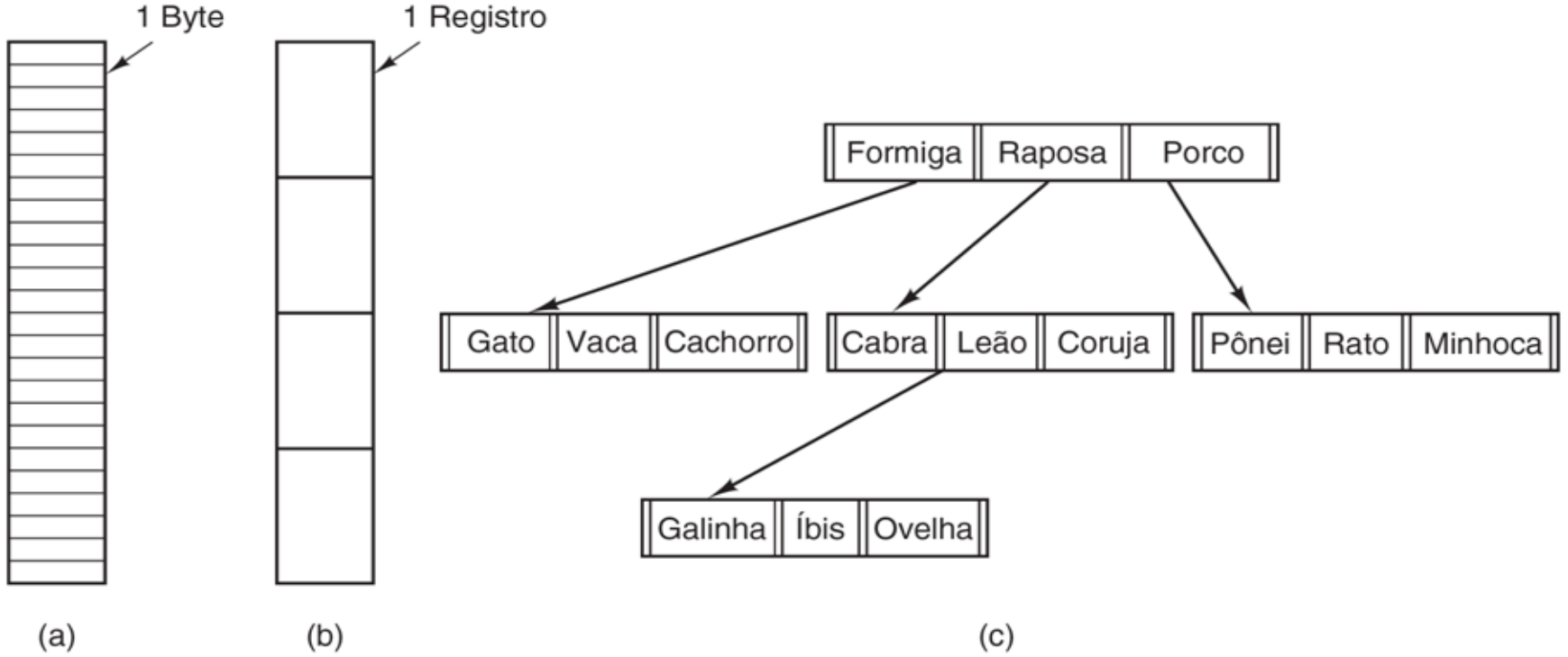
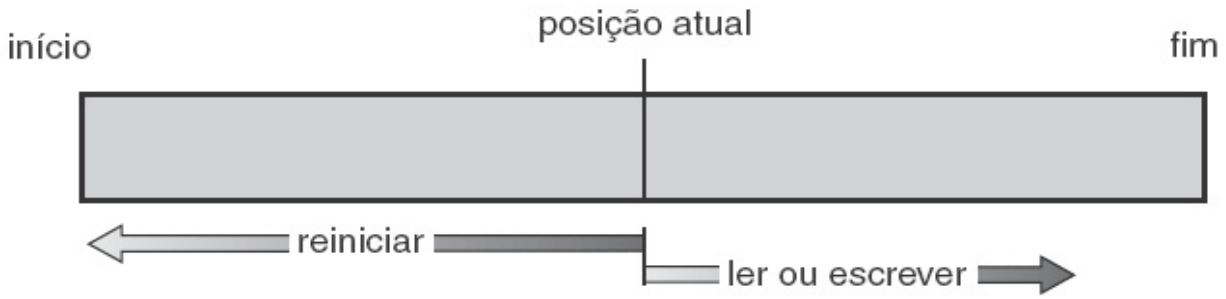


Figura 4.1 Três tipos de arquivos. (a) Sequência de bytes. (b) Sequência de registros. (c) Árvore.

Métodos de acesso

➢Acesso Sequencial

- É o método de acesso mais simples, as informações no arquivo são processadas em ordem, um registro após o outro
- É também o mais comum, em geral os editores e compiladores acessam os arquivos dessa maneira
- Operações:
  - read next (lê a próxima porção e avança o ponteiro de arquivo)
  - write next (acrescenta no final do arquivo e avança o ponteiro)
  - reset (retorna ao início)



Métodos de acesso

➢**Acesso Direto (ou Relativo):** o arquivo é visto como uma sequência numerada de blocos ou registros lógicos de tamanho fixo

- Não existem restrições sobre a ordem de leitura ou escrita
- Muito utilizado para o acesso imediato a uma grande quantidade de informações
- Ex.: banco de dados
- Operações:

read *n*

write *n*

ou

position to *n*

read next

write next

n = número relativo do bloco

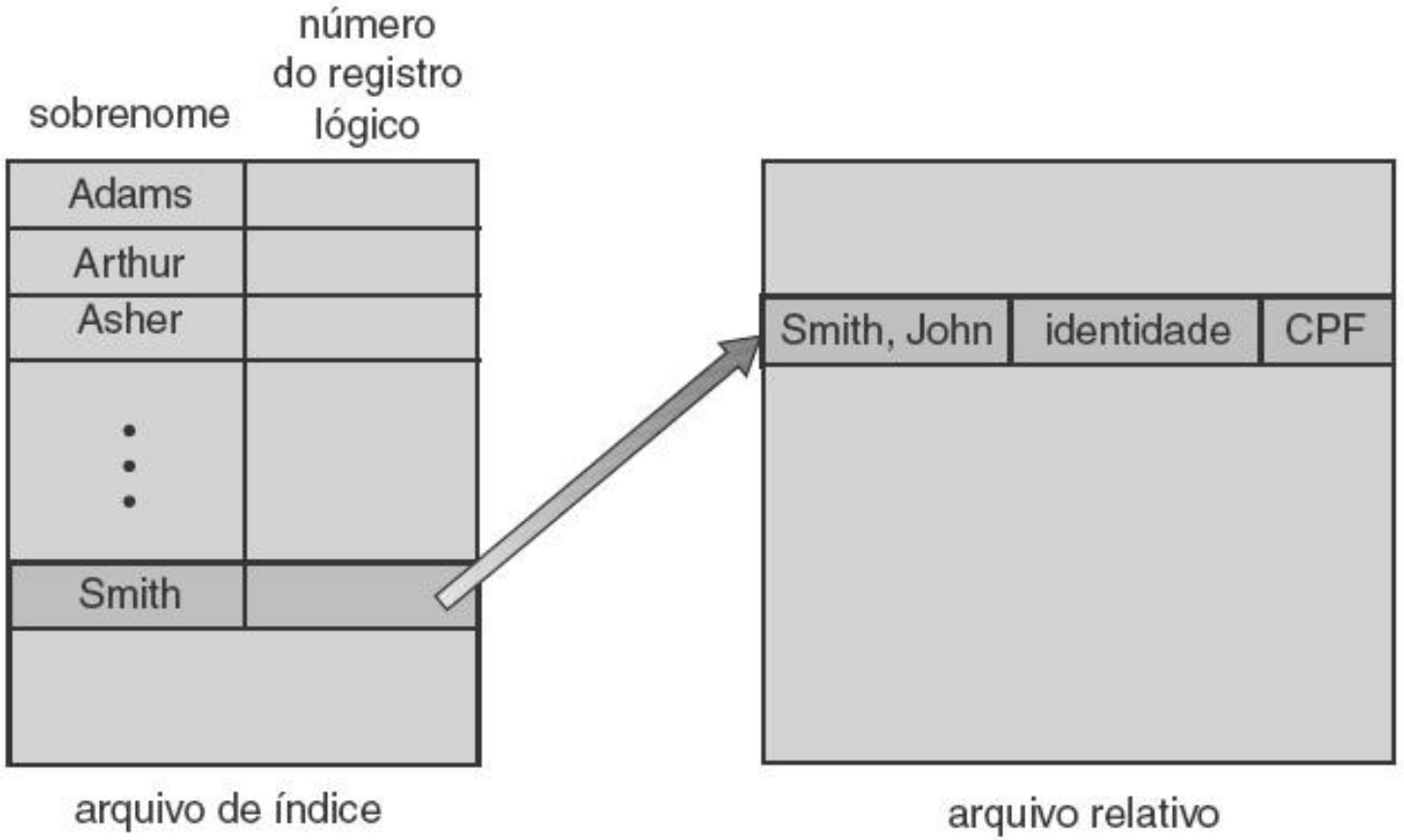
- O número relativo do bloco é um índice relativo ao início do arquivo

Simulação de acesso sequencial num arquivo de acesso direto

- Alguns sistemas só permitem o acesso sequencial, outros só permitem o acesso direto
- Pode-se simular o acesso sequencial em um arquivo de acesso direto mantendo uma variável que define a posição atual (*pos*)

acesso seqüencial	implementação para acesso direto
Reset	*pos = 0;*
read next	*read pos;* *pos = pos+1;*
write next	*write pa;* *pos = pos+1;*

Outros métodos de acesso: exemplo de arquivo índice e arquivo relativo



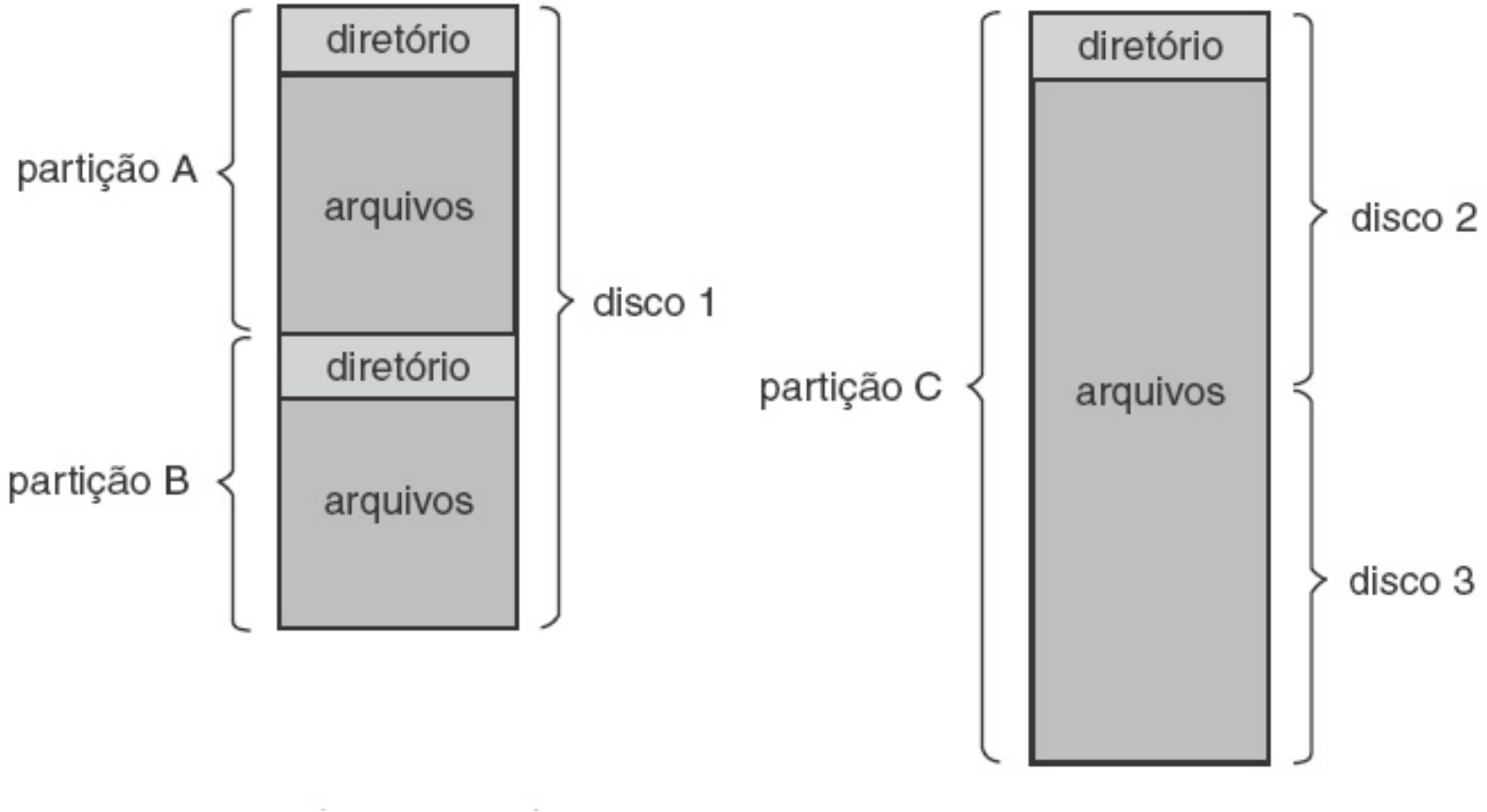
Para cada arquivo manter um índice com ponteiros para os diversos blocos

Para encontrar um **registro** no arquivo, pesquisamos o **índice** e utilizamos o **ponteiro** para fazer acesso ao arquivo diretamente e encontrar o registro

Operações realizadas num diretório

- **Pesquisar arquivos** – para encontrar arquivos cujos nomes correspondam a um determinado padrão
- **Criar arquivo** – criar e adicionar o arquivo ao diretório
- **Excluir arquivo** – para liberar espaço
- **Listar um diretório** – listar os arquivos em um diretório
- **Renomear um arquivo** – quando o conteúdo ou uso mudar
- **Percorrer o sistema de arquivo** – para acessar todos os diretórios e todos os arquivos dentro de uma estrutura de diretório

Uma organização típica de Sistema de Arquivos



Alguns sistemas utilizam partições para fornecer áreas separadas em um mesmo disco, enquanto outros permitem partições maiores que o disco

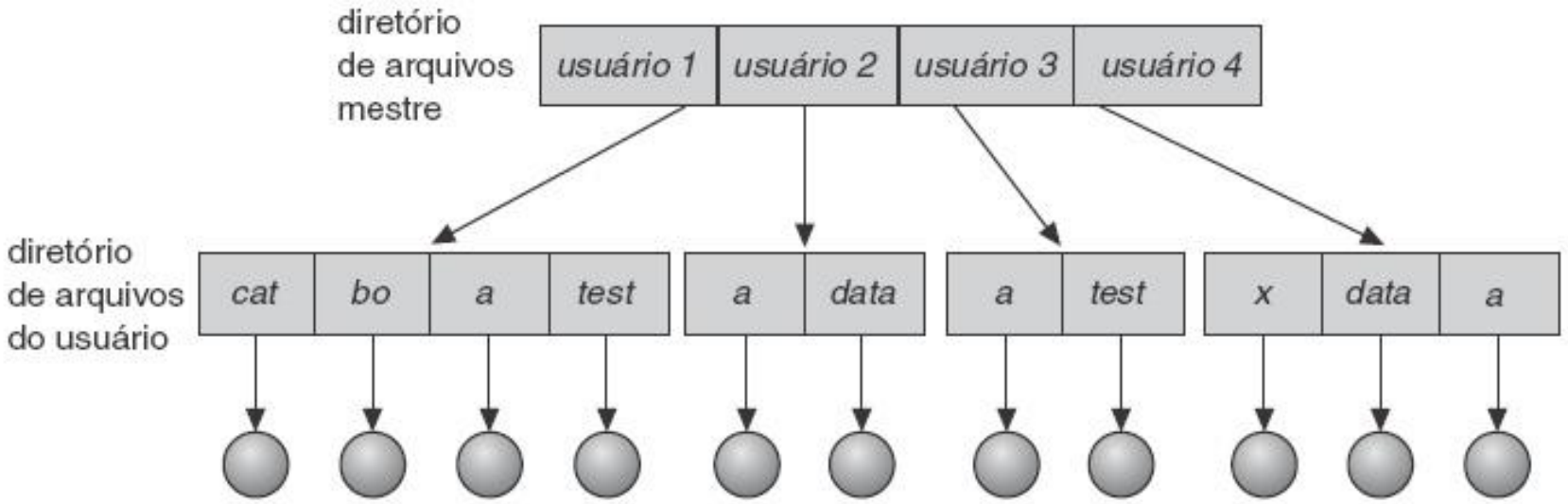
Organização de diretório

- Organizar um diretório (logicamente) para obter:
  - **Eficiência** – localizando uma arquivo rapidamente
  - **Nomenclatura** – conveniente para os usuários
    - Dois usuários podem ter o mesmo nome para diferentes arquivos
    - O mesmo arquivo pode ter diferentes nomes
  - **Agrupamento** – agrupamento lógico de arquivos pelas propriedades (ex., todos os programas Java, músicas, jogos, …)

- Estruturas de diretórios:
  - Diretório em um nível
  - Diretório em dois níveis
  - Diretório estruturado em árvore
  - Diretório em grafo acíclico

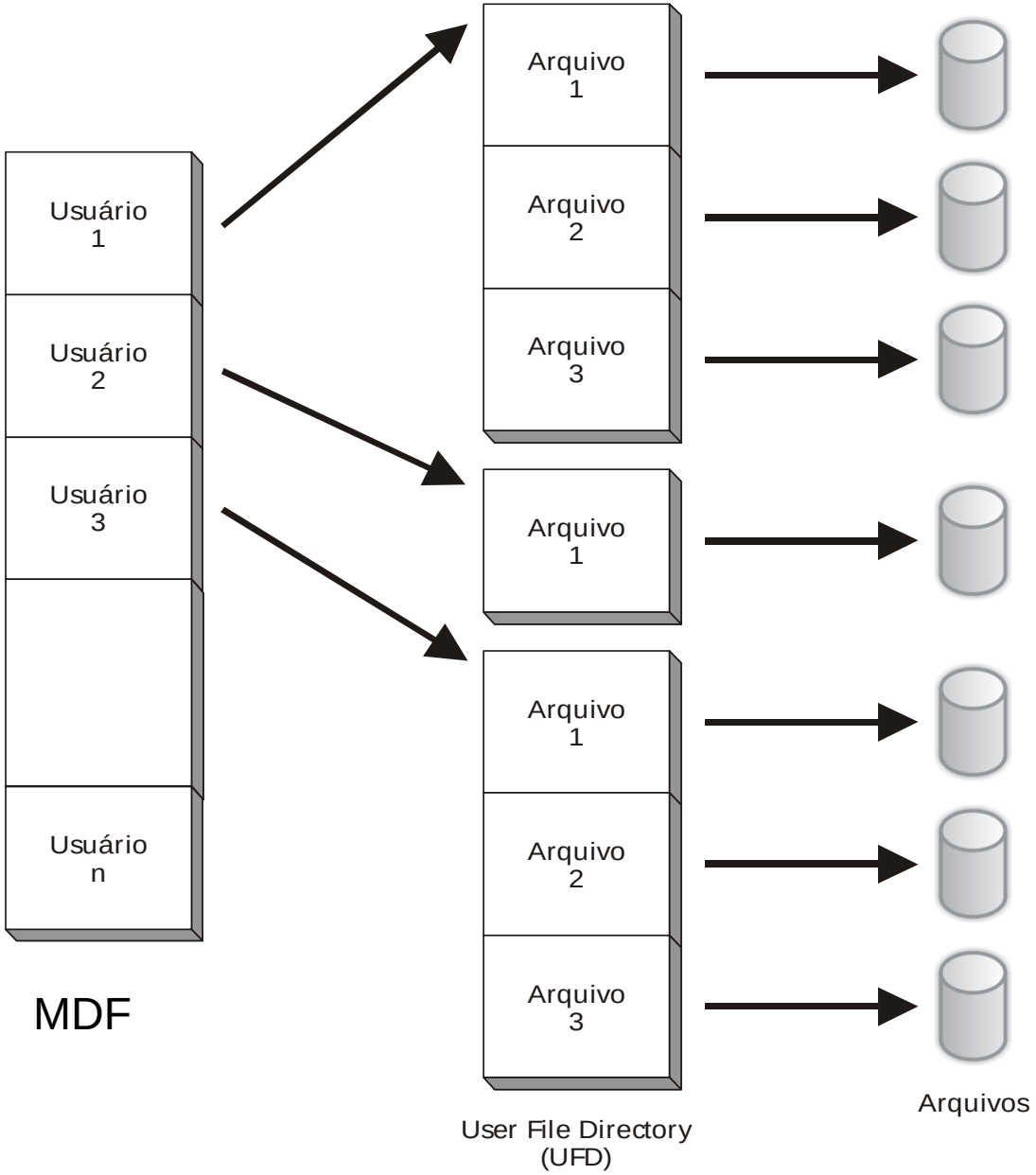
Diretório de dois níveis

- Um diretório separado para cada usuário



- Nome de usuário e de arquivo definem um **nome de caminho**
- Um arquivo pode ter o mesmo nome para vários usuários
- Busca eficiente
- Sem capacidade de agrupamento (não permite compartilhamento)

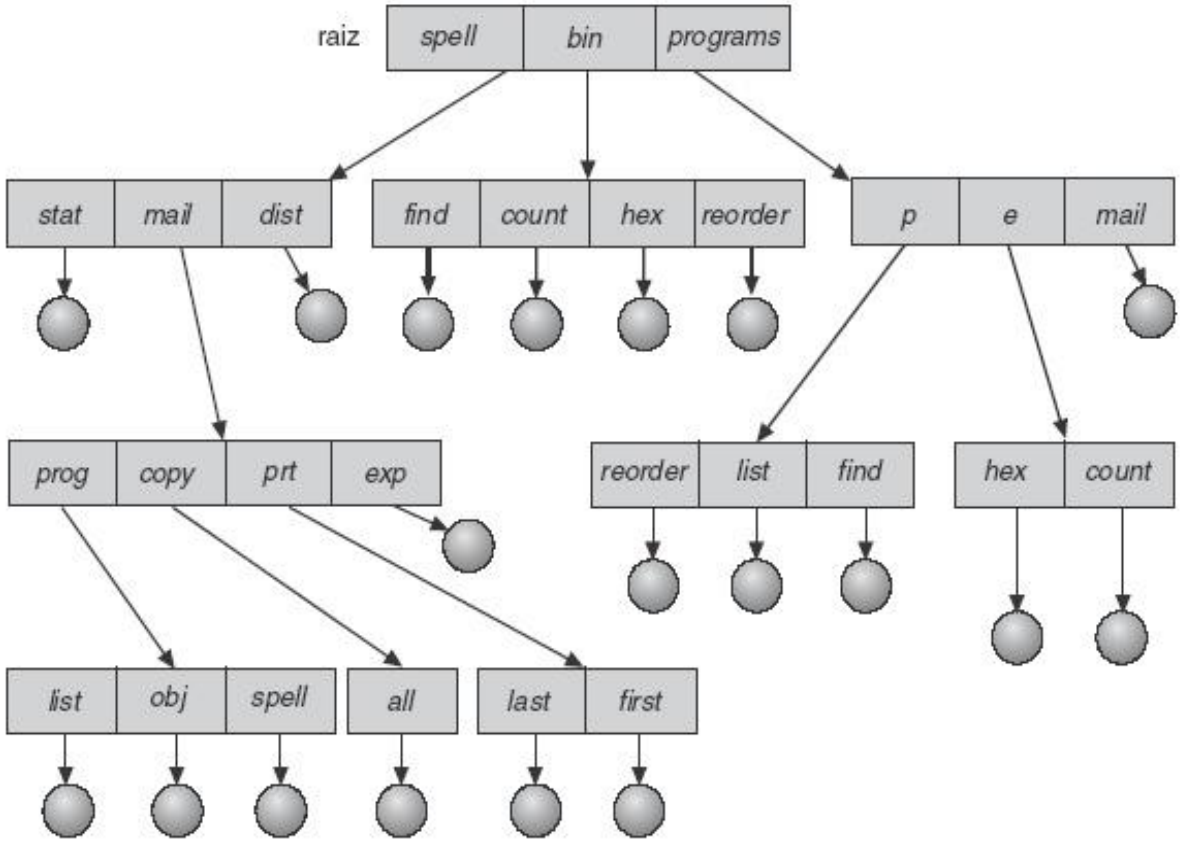
Diretório de dois níveis



- UFD (*User File Directory*): estrutura de diretório particular para cada usuário

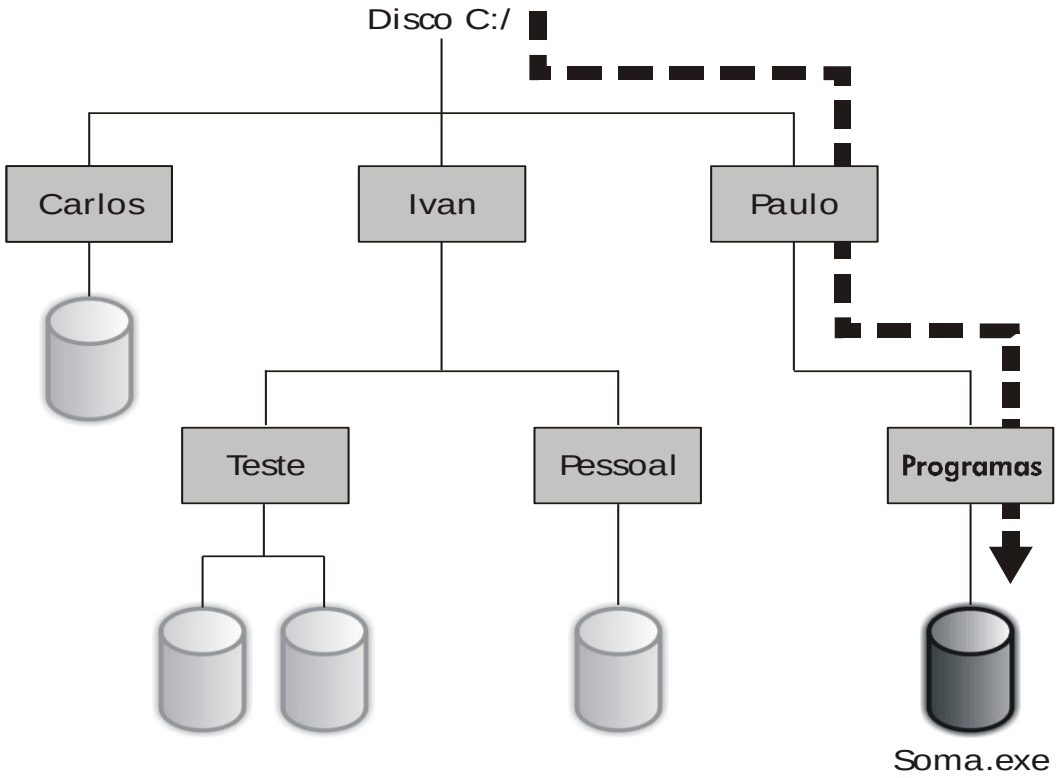
- MDF (*Master File Directory*): diretório que contém os diretórios individuais dos usuários, indexado pelo nome do usuário, cada entrada aponta para o diretório pessoal

Diretórios estruturados em árvore



- É a estrutura de diretório mais comum
- Usuários podem criar diversos diretórios e agrupar arquivos de modo conveniente
- A árvore possui um diretório raiz e cada arquivo do sistema possui um nome de caminho exclusivo
- Um diretório ou subdiretório contém um conjunto de arquivos ou diretórios
- Um diretório é um tipo de arquivo, tratado de forma especial
- Um bit em cada entrada de diretório identifica a entrada como arquivo (0) ou diretório (1)

Caminho de um arquivo



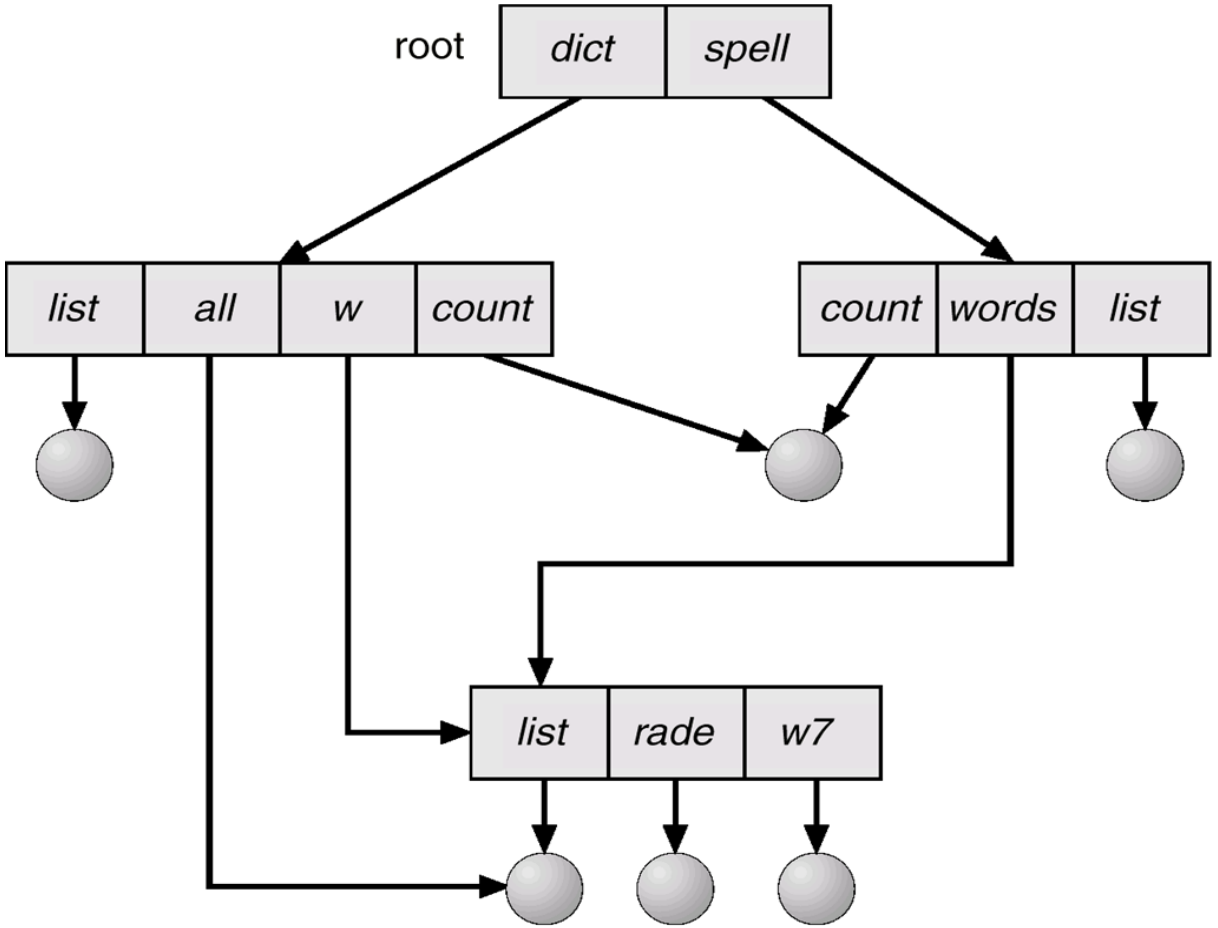
Um arquivo nesta estrutura de árvore pode ser especificado através de um caminho => contém todos os diretórios desde a raiz até o diretório que contém o arquivo

Caminho absoluto e relativo

- Os nomes dos caminhos podem ser de dois tipos:
  - Caminho absoluto (*absolute path name*): é um caminho que começa no diretório raiz até o arquivo especificado
  - Exemplos:
    - UNIX: /usr/john/texto.txt
    - Windows: \usr\john\texto.txt
  - Caminho relativo (*relative path name*): define um caminho a partir do diretório atual
    - O usuário estabelece um diretório como sendo o diretório atual; caminhos não iniciados no diretório raiz são tidos como relativos ao diretório atual
    - Exemplos:
      - Absoluto: cp /usr/john/texto.txt /usr/john/texto2.txt
      - Relativo: se o diretório atual é: /usr/john:
      - cp texto.txt texto2.txt

Diretórios em Grafos Acíclicos

- É uma generalização do esquema de diretório em árvore
  - Permite o compartilhamento de arquivos, o que não é possível na estrutura de árvore
  - O compartilhamento permite que mesmo arquivo esteja em dois diretórios distintos



Alocação de espaço em disco

Como alocar espaço aos arquivos de modo que o espaço em disco seja utilizado com eficácia e os arquivos sejam acessados rapidamente?

➢Um método de alocação determina **como os blocos do disco são alocados aos arquivos**:

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

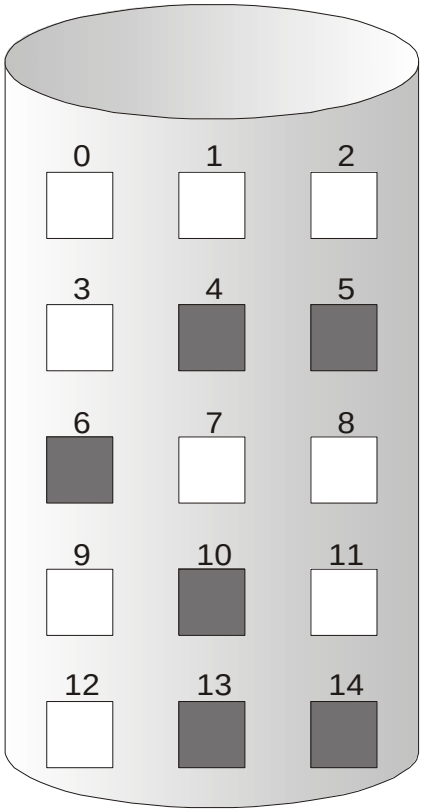
Alocação contígua

- Cada arquivo ocupa um conjunto de **blocos contíguos** no disco
- A alocação contígua é definida pelo endereço inicial e tamanho do arquivo (em unidades de bloco)
  - Se um arquivo possui n blocos de extensão e estiver armazenado no disco a partir do bloco b, ele ocupará os blocos: b, b+1, b+2, …, b+n-1

- É possível o acesso sequencial ou direto ao arquivo
  - Para o acesso direto ao bloco i do arquivo que começa em b, basta acessar diretamente o bloco b+i

- Uma dificuldade é como encontrar espaços livres para alocar um novo arquivo (como encaixar um arquivo de tamanho n a partir de uma lista de espaços livres) → semelhante ao problema da alocação de memória, visto anteriormente
  - Estratégias para selecionar um espaço livre para alocação: *Best-fit*, *Worst-fit* e *First-fit*

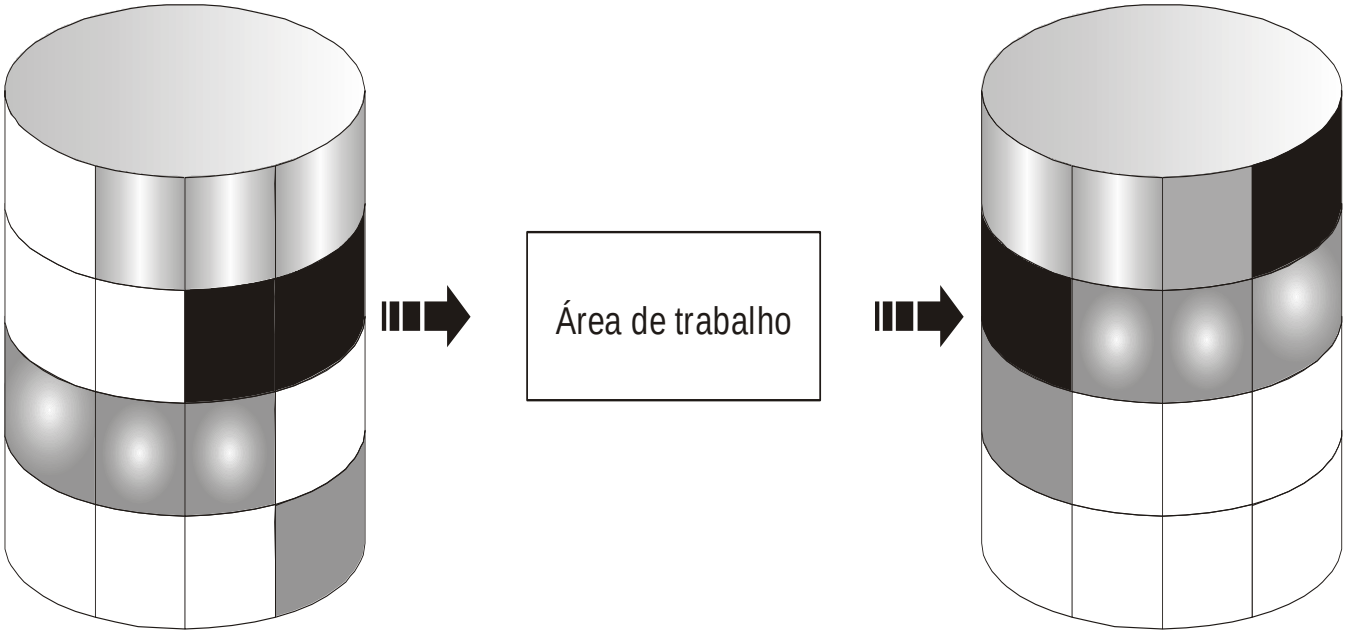
Alocação contígua



Arquivo	Bloco	Extensão
A. TXT	4	3
B. TXT	10	1
C. TXT	13	2

- Cada arquivo ocupa um conjunto de **blocos contíguos** no disco
- Cada entrada na tabela (para cada arquivo) há o endereço do bloco inicial e o tamanho da área alocada para o arquivo (número de blocos)
- Há o problema de **fragmentação externa**
- Um outro problema com este método é que os arquivos não podem crescer, o tamanho máximo do arquivo deve ser conhecido no momento de sua criação

Desfragmentação

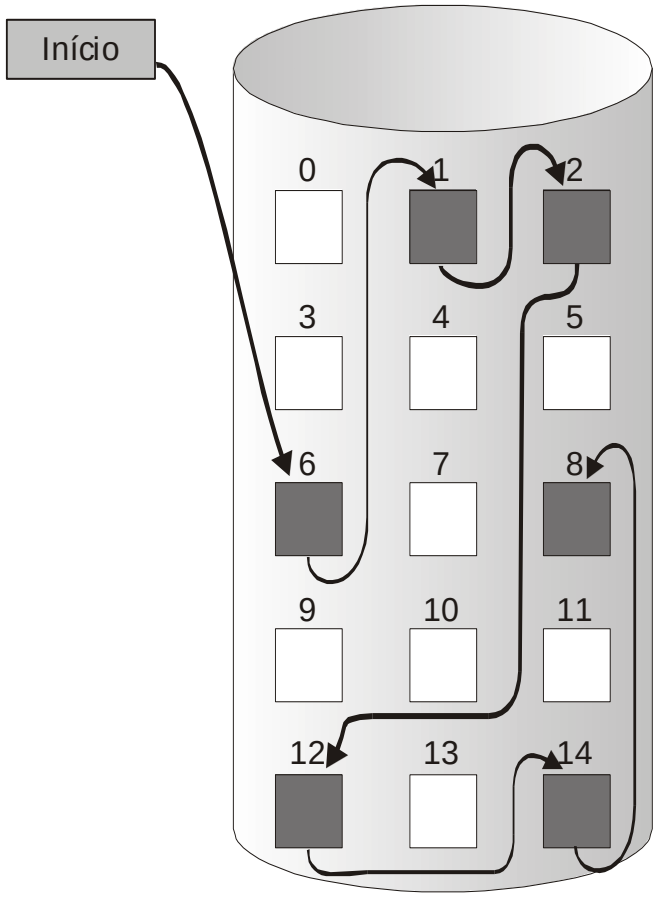


➢Reoganizar os arquivos para que os blocos livres fiquem em um só bloco contínuo

Alocação encadeada

- Um arquivo pode ser organizado como uma lista encadeada de blocos de disco: os blocos podem estar dispersos em qualquer parte do disco
- Cada bloco contém um ponteiro para o próximo bloco
  - Requer espaço adicional para o ponteiro (se cada bloco possui 512 bytes de tamanho e o ponteiro exige 4 bytes, o usuário verá blocos de 508 bytes)
- Para a leitura do arquivo é preciso seguir os ponteiros de um bloco para outro do arquivo
  - É eficiente apenas para acesso sequencial
  - Para encontrar o bloco de posição i, cada acesso a um ponteiro requer uma leitura de disco e busca (blocos podem estar em diferentes posições do disco, requer movimentação da cabeça de leitura do disco)
- É flexível: não é preciso definir o tamanho do arquivo na sua criação, além disso, o arquivo podem crescer ou diminuir de tamanho
- Problema de robustez: se um bloco for corrompido, todos os outros blocos posteriores se tornarão inacessíveis

Alocação encadeada



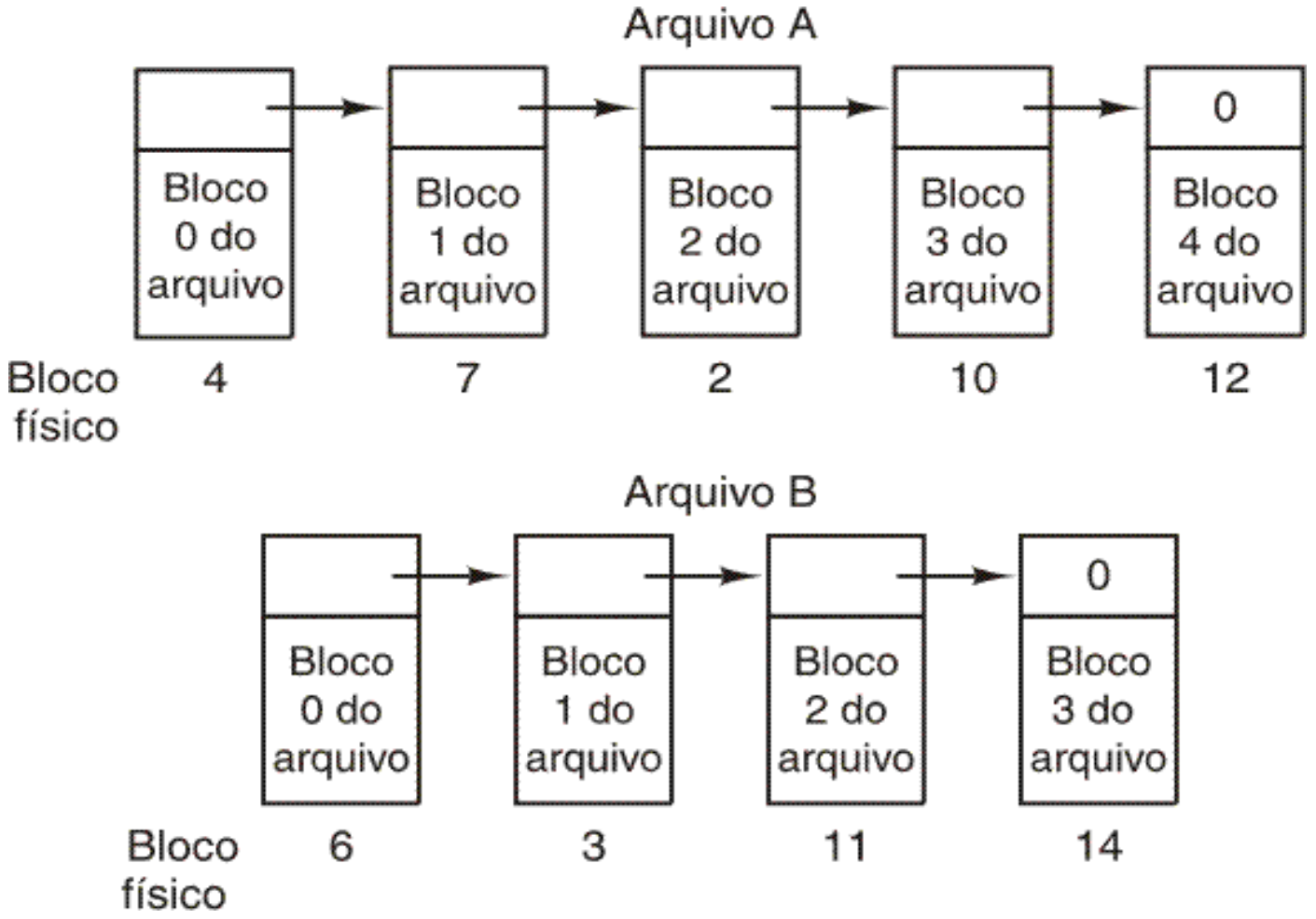
Arquivo	Bloco
A.TXT	6
. . .	. . .
. . .	. . .
. . .	. . .
. . .	. . .

bloco = | ponteiro |

- Cada entrada no diretório possui um ponteiro para o primeiro bloco em disco do arquivo
- Cada bloco contém um ponteiro para o próximo bloco

- Vantagens:
  - **Não há desperdício de espaço**
  - **Não** apresenta **fragmentação externa**
- Problemas:
  - Baixo desempenho para acesso direto (aleatório)
  - Fragilidade a erros nos blocos
  - Requer espaço adicional em cada bloco para o armazenamento de ponteiros

Alocação encadeada

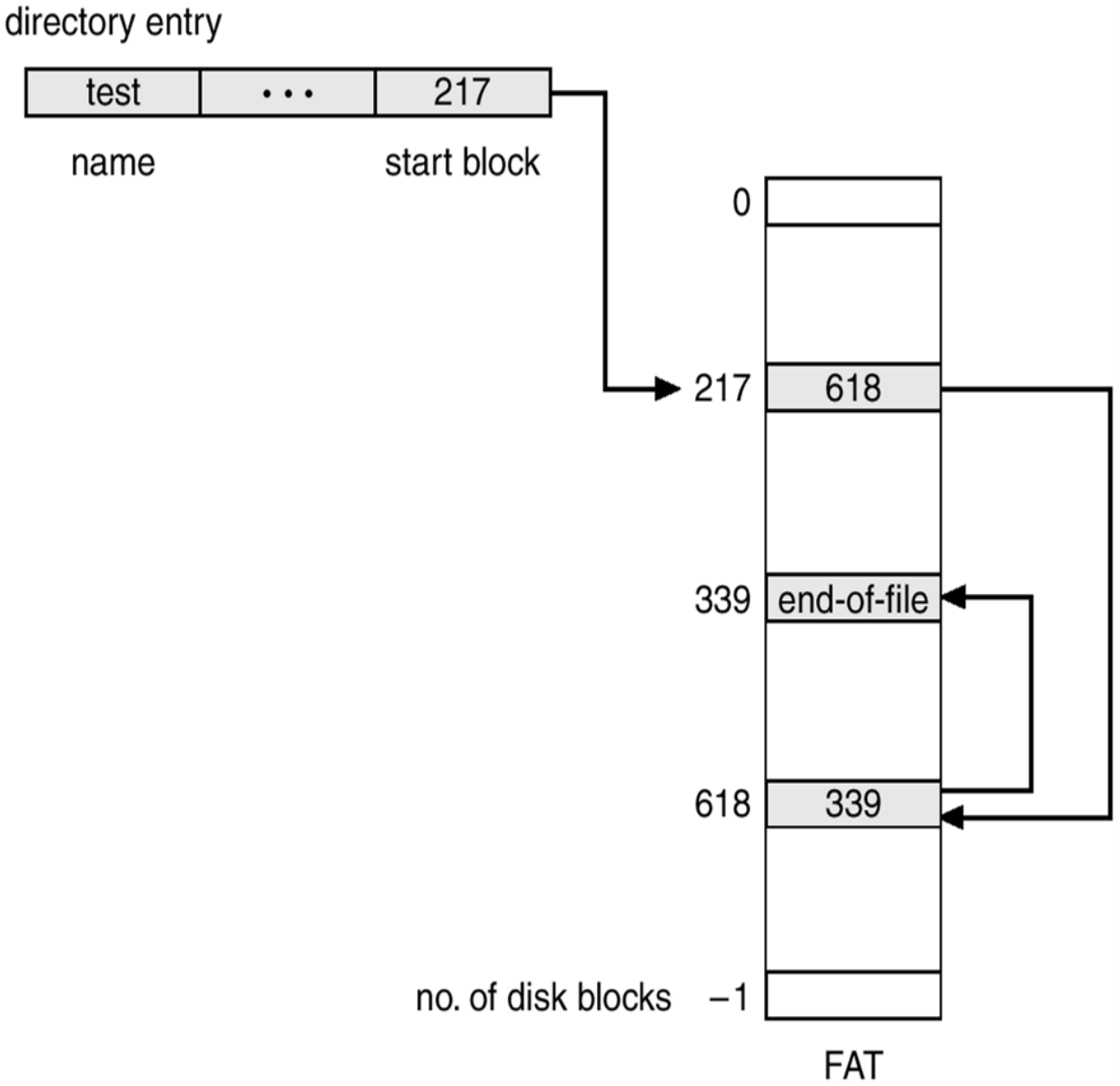


- O acesso sequencial é rápido e fácil (somente seguir os ponteiros)
- Porém, se os blocos estiverem muito espalhados no disco, serão necessários muitos deslocamentos da cabeça de leitura, diminuindo o desempenho de acesso ao disco

Tabela de alocação de arquivos

- Uma variação à alocação encadeada é a **Tabela de Alocação de Arquivos** (ou **File-Allocation Table - FAT**)
  - Alocação de espaço de disco originalmente usada pelos sistemas operacionais MS-DOS e OS/2
  - Os ponteiros dos blocos são mantidos em uma única tabela, armazenada em uma seção do disco, em blocos reservados no início de uma partição
  - A tabela é indexada pelo número do bloco de disco e cada entrada na tabela contém o número do próximo bloco do arquivo
  - A entrada correspondente ao último bloco do arquivo contém um valor especial de fim de arquivo

Exemplo de FAT



Uma entrada no diretório contém o número do primeiro bloco do arquivo na tabela FAT

Cada entrada na tabela contém o número do próximo bloco do arquivo

Neste exemplo, o arquivo possui 3 blocos: 217, 618 e 339

O esquema de alocação FAT pode resultar em um **nº. significativo de buscas do cabeçote,** a menos que a tabela esteja em um *cache*

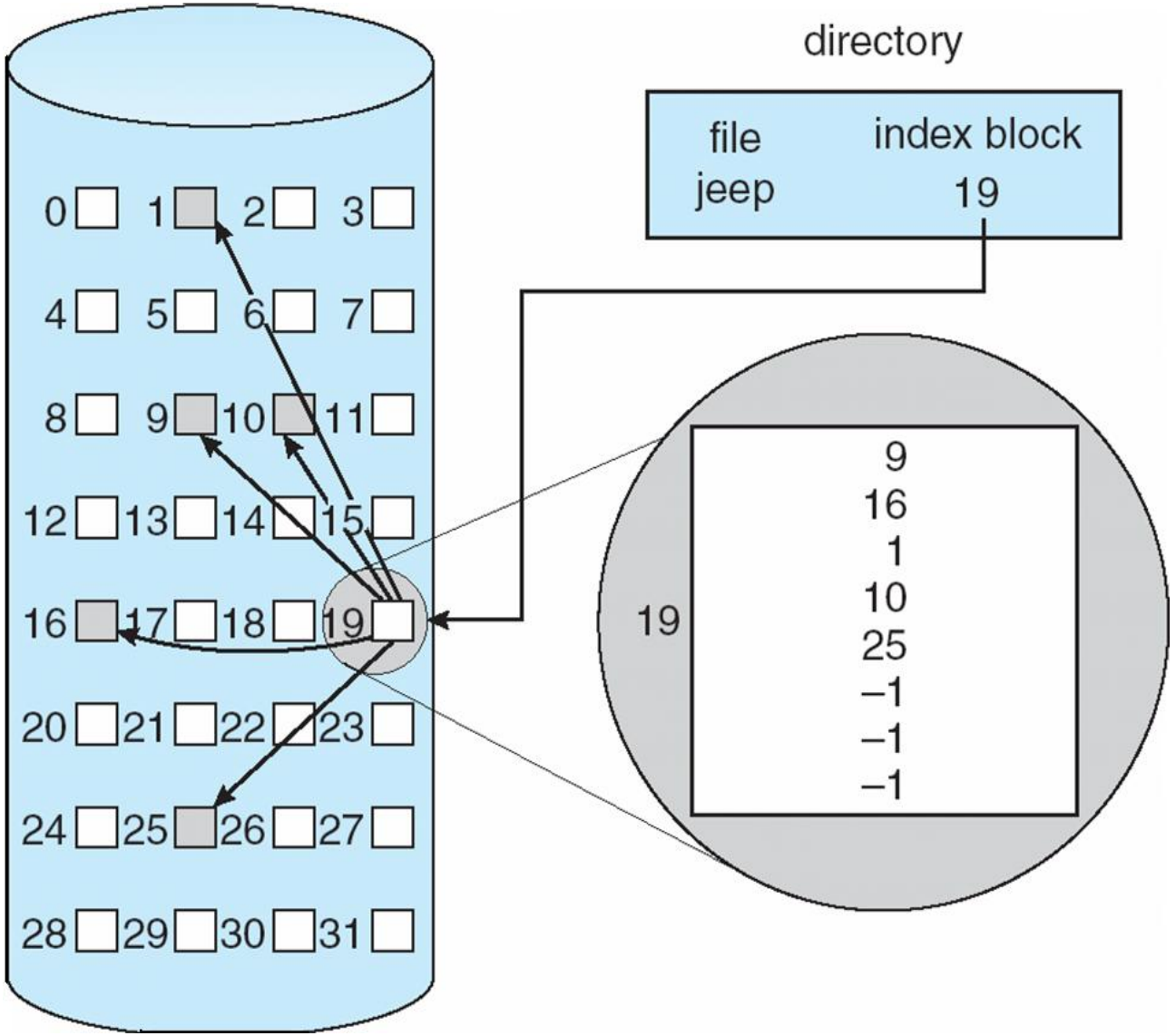
O tempo de acesso aleatório é melhorado, pois o local de qualquer bloco pode ser encontrado na FAT

O cabeçote do disco deve se movimentar até o início da

Alocação indexada

- Soluciona o problema do acesso sequencial ao disco da alocação encadeada: um bloco de índices para cada arquivo
- Reúne todos os ponteiros em um só lugar: o **bloco de índices**
  - Cada arquivo possui um bloco de índices, que é um array de endereços de blocos do disco
  - A entrada i do bloco de índices aponta para o bloco i do arquivo
- Vantagem: possibilita acesso aleatório sem sofrer do problema de fragmentação externa
- Problema: requer espaço para o armazenamento do bloco de índices
  - Com a alocação encadeada, perde-se o espaço de um ponteiro para cada bloco
  - Com a alocação indexada, perde-se o espaço de um bloco inteiro (para o bloco de índices), mesmo que tenha apenas um ponteiro para um bloco de disco (um arquivo pequeno)
  - Questão: o bloco de índices não pode ser muito pequeno, senão

Alocação indexada



- O bloco de índices é um array de endereços de blocos de disco
- A i-ésima entrada do bloco de índices aponta para o i-ésimo bloco do arquivo

Gerenciamento de espaço livre

➢Ao excluir arquivos formam-se espaços vazios que precisam ser reutilizados para armazenar novos arquivos

➢Para gerenciar o espaço livre no disco, o sistema mantém uma **lista de espaço livre**

- Contém todos os blocos de disco livres
- Quando um novo arquivo é criado, a lista é pesquisada em busca do espaço requerido
- O espaço é alocado para o arquivo e é removido da lista de espaço livre
- Quando o arquivo é removido, o espaço é acrescentado à lista de espaço livre

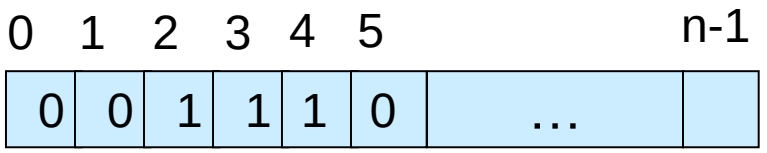
➢ Algumas estratégias para implementar a lista de espaço livre: vetor de bits ou lista encadeada

Vetor de bits (ou mapa de bits)

- Cada bloco é representado por 1 bit

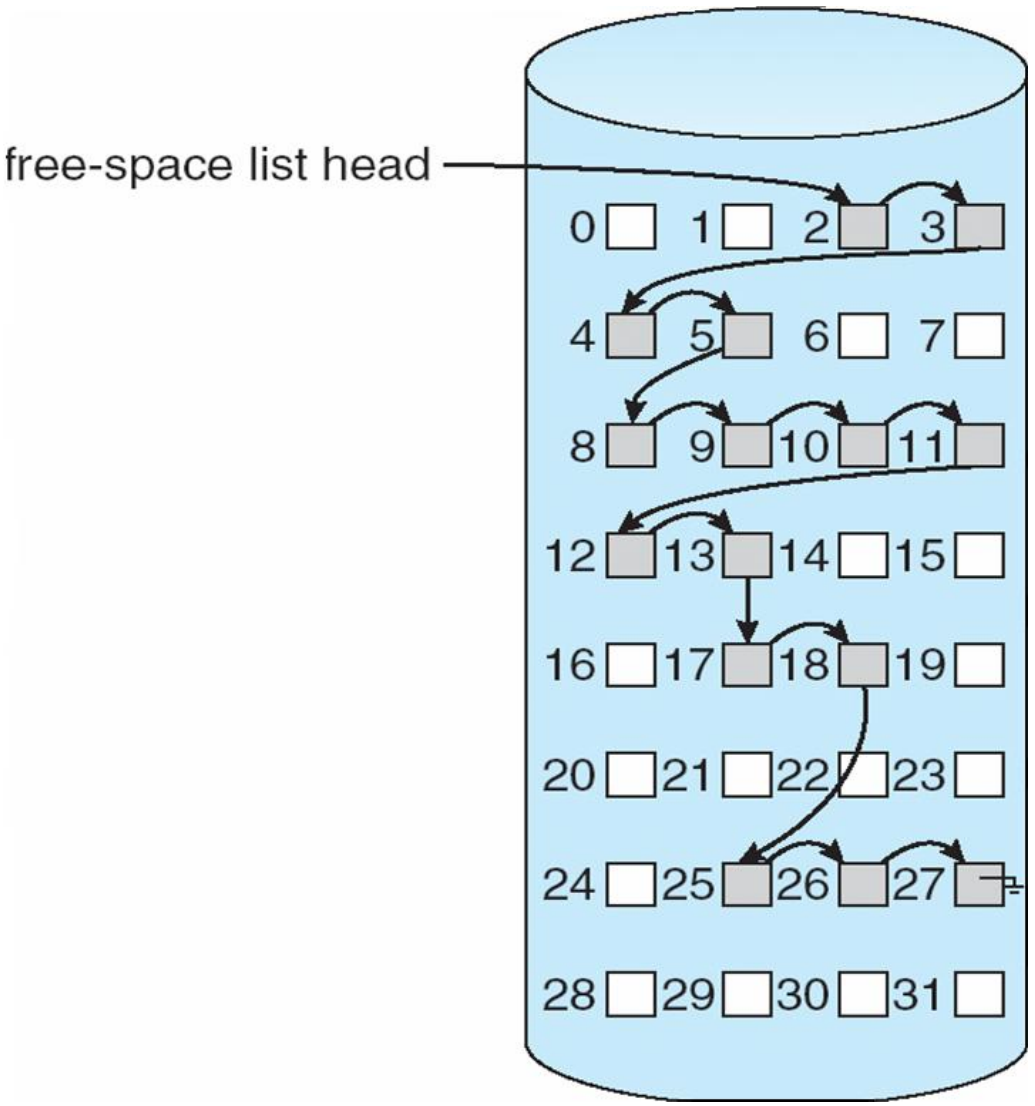
bit[i] = 1 => bloco[i] livre

0 => bloco[i] ocupado



- **Vantagem**: fácil de encontrar o primeiro bloco livre ou *n* blocos livres consecutivos
- **Desvantagem**: ineficiente, a menos que todo vetor seja mantido na memória principal (possível para discos menores)
  - Por exemplo, um disco de 40 GB com blocos de 1 KB requer mais de 5 MB para armazenar o mapa de bits

Lista encadeada



- Outra abordagem para o gerenciamento do espaço livre é encadear todos os blocos de disco livres
  - Mantém um ponteiro para o primeiro bloco livre, esse primeiro bloco livre contém um ponto para o próximo bloco livre e assim sucessivamente
    - Não há desperdício de espaço
    - Mas para atravessar a lista é preciso ler cada bloco o que requer tempo de E/S substancial
    - Em geral, o sistema precisa de um bloco livre para atribuir a um arquivo e utiliza o primeiro elemento da lista

Referências

- [Silberschatz] SILBERCHATZ, A., GALVIN, P. B. e GAGNE, G. **Sistemas Operacionais com Java**. 7ª ed., Rio de Janeiro: Elsevier, 2008.
- [Tanenbaum] TANENBAUM, A. **Sistemas Operacionais Modernos**. 3ª ed. São Paulo: Prentice Hall, 2009.
- [MACHADO] MACHADO, F. B. e MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 4ª ed., Rio de Janeiro: LTC, 2007.