Estrutura de Arquivos

Prof. Tiago Eugenio de Melo, MSc

 tiago@comunidadesol.org

 www.tiagodemelo.info

Sumário

- Introdução
  - Hierarquia de dados
  - Características físicas do disco
  - Armazenamento
  - Leitura e gravação
  - Estrutura de arquivos
  - Acesso aos dados
- Sistemas de arquivo
- Entrada e saída de arquivos

Introdução

- Conceitos
  - Os arquivos são utilizados para retenção a longo prazo de grandes quantidades de dados, mesmo depois de terminar o programa que criou os dados.
  - O que é persistência de dados?
    - É a existência dos dados por mais tempo que a duração da execução do programa.

Hierarquia de dados

- Em última instância, o computador processa todos os itens de dados como combinações de zeros e uns.
- Qual é a razão para isto?
  - É mais simples e econômico construir dispositivos eletrônicos que podem assumir dois estados estáveis.
- O menor item de dados que um computador pode assumir é o valor 0 ou 1.

Hierarquia de dados

- Os programadores não trabalham com bits, mas sim com caracteres.
- Assim como os caracteres são compostos por bits, os campos são compostos de caracteres ou bytes.
- O que é campo?
  - É um grupo de caracteres ou bytes que possui um significado.

Hierarquia de dados

- A reunião de vários campos relacionados forma um registro.
- Os dados processados por computadores formam uma hierarquia de dados em que os itens tornam-se maiores e mais complexos, em termos de estrutura.
- Bits, bytes, caracteres, campos, registros etc.

Hierarquia de dados

- E o que vem a ser arquivo?
  - Arquivo é um grupo de registros relacionados.
- Como fazer a busca num arquivo de maneira mais rápida?
  - É através de um campo de cada registro escolhido como chave de registro.
- Há várias formas de organizar os registros num arquivo.

Hierarquia de dados

- O grupo de arquivos relacionados se chama banco de dados.
- A coleção de programas projetados para criar e gerenciar os bancos de dados se chama sistema de gerenciamento de banco de dados (SGBD).

Características físicas do disco

- Um disco rígido é composto por uma pilha de pratos que armazenam os dados em ambas as faces, exceto os pratos inferior e superior.
- Cada prato é composto de trilhas.
- Cada trilha é composta de setores.
- As trilhas que ficam uma acima da outra compõem um cilindro.

Características físicas do disco

- Hierarquia

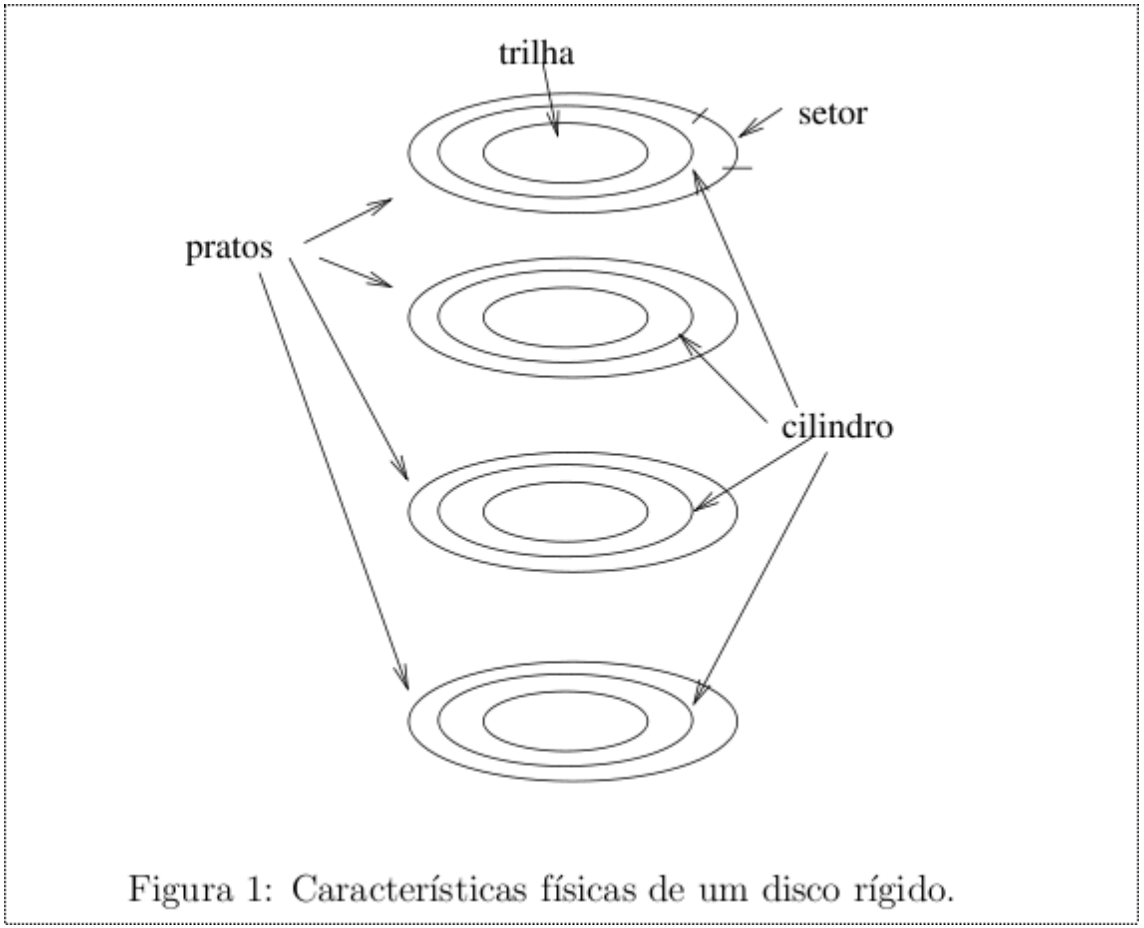
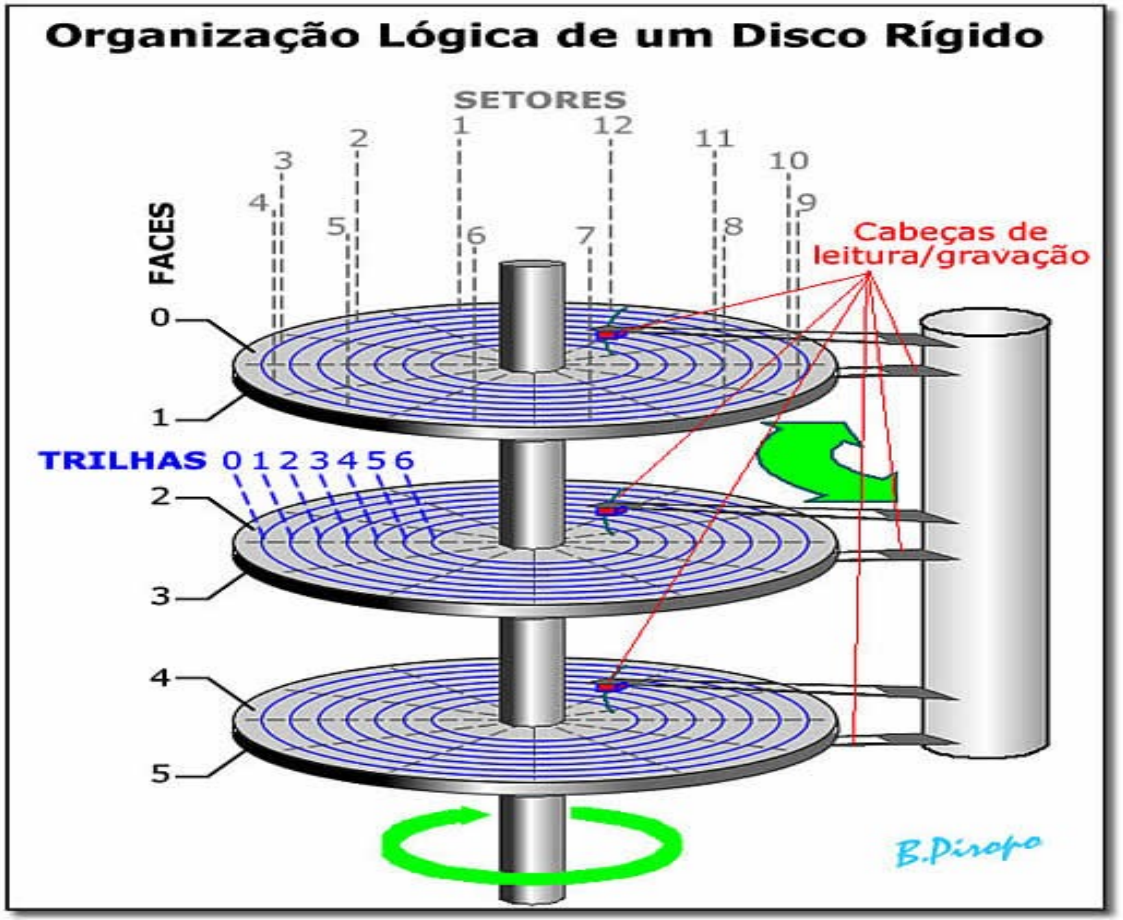


Figura 1: Características físicas de um disco rígido.

Características físicas do disco

- Hierarquia



Armazenamento

- Os arquivos são armazenados por cilindro, pois o tempo de mover o braço de leitura de um cilindro para outro é a parte mais cara do tempo de acesso aos dados (seek time).
- O sistema operacional divide o disco em blocos de capacidade fixa de armazenamento.
- Nesta caso, para que o número de blocos por trilha seja constante, as trilhas mais próximas do centro do prato são mais densas.

Armazenamento

- Exercício:
  - Suponha que a capacidade de um bloco é 512 bytes, o número de blocos por trilha é 40, o número de trilhas por cilindro é 11, e o número de cilindros é 1000. Quantos cilindros são necessários para armazenar um arquivo com 10.000 registros de 256 bytes cada?

Armazenamento

- Resposta
  - 1 cilindro = 11 trilhas x 40 blocos x 2 registros = 880 registros por cilindro.
  - Então, o número de cilindros necessários é 10.000/880 ou aproximadamente 12 cilindros.

Leitura e gravação

- Uma operação de leitura recupera vários blocos de uma só vez e mantém esses blocos em uma área na memória principal.
- No processo de escrita, ocorre a gravação de arquivos contendo várias páginas.
- O usuário não tem o controle sobre esses processos.

Estruturas de arquivos

- O conceito de registro é lógico e não físico, a integridade dos campos e a integridade dos registros podem ser perdidas quando o arquivo é gravado em disco, impossibilitando a recuperação dos dados.

Estruturas de arquivos

- Organização dos campos
  - Forçar que cada campo ocupe um tamanho fixo em bytes.
  - Reservar um certo número de bytes antes de cada campo para indicar o seu comprimento.
  - Inserir um delimitador separando os campos.
  - Utilizar uma palavra chave para identificar o conteúdo de cada campo.

Estruturas de arquivos

- Organização dos registros
  - Registros de tamanho fixo
    - Campos de tamanho fixo.
      - Não é necessário se preocupar com os delimitadores.
      - Desperdício de muito memória nos campos dos registros (fragmentação).
    - Campos de tamanho variável.
      - O número de campos pode ser variável no registro com delimitadores entre campos.
      - Não teremos desperdício de campos.

Estruturas de arquivos

- Organização dos registros
  - Registros de tamanho variável
    - Os campos podem ser de tamanhos variáveis separados por delimitadores, porém precisamos identificar os registros também.

Acesso aos dados

- Acesso sequencial
  - A forma mais simples de acesso é o sequencial.
- Acesso direto
  - Alternativa para o acesso sequencial.
  - O acesso direto a um dado registro em disco requer conhecer o endereço do registro no arquivo de dados

Referências

(1) Falcão, Alexandre X. ***Estrutura de arquivos***. http://www.ic.unicamp.br/~afalcao/mc326/notas_aula.pdf

(2) Deitel, H. M. ***Java – Como Programar***. 4 ed. Bookman, 2002.

Exercícios

1. Explique como está organizado um disco rígido?
2. O conceito de registro é lógico ou físico? Explique a sua resposta.
3. Como os registros podem ser organizados?
4. Comente uma vantagem e uma desvantagem no uso de arquivos com acesso sequencial.

Sistemas de arquivos

Um sistema de arquivos é um conjunto de tipos abstratos de dados que são implementados para armazenar, organizar hierarquicamente, manipular, navegar, acessar e recuperar dados.

Sistemas de arquivos

- Fazendo uma analogia, o sistema de arquivos assemelha-se à organização de uma biblioteca.
- O bibliotecário (sistema operacional) organiza os livros (arquivos) por assuntos (diretórios).
- A organização deve ser de tal forma que a busca (métodos de busca) seja eficiente.
- O armazenamento é feito de forma que se tenha uma economia de espaço na prateleira (dispositivos de armazenamento secundário).

Sistemas de arquivos

- Extensões comuns de arquivos

Extensão	Significado
file.bak	Arquivo de cópia de segurança
file.c	Programa fonte em C
file.gif	Imagem no formato de intercâmbio gráfico da Compuserve (graphical interchange format)
file.hlp	Arquivo de auxílio
file.html	Documento da World Wide Web em Linguagem de Marcação de Hipertexto (*hypertext markup language* — HTML)
file.jpg	Imagem codificada com o padrão JPEG
file.mp3	Música codificada no formato de áudio MPEG — camada 3
file.mpg	Filme codificado com o padrão MPEG
file.o	Arquivo-objeto (saída do compilador, ainda não ligado)
file.pdf	Arquivo no formato portátil de documentos (portable document format — PDF)
file.ps	Arquivo no formato PostScript
file.tex	Entrada para o programa de formatação TEX
file.txt	Arquivo de textos
file.zip	Arquivo comprimido

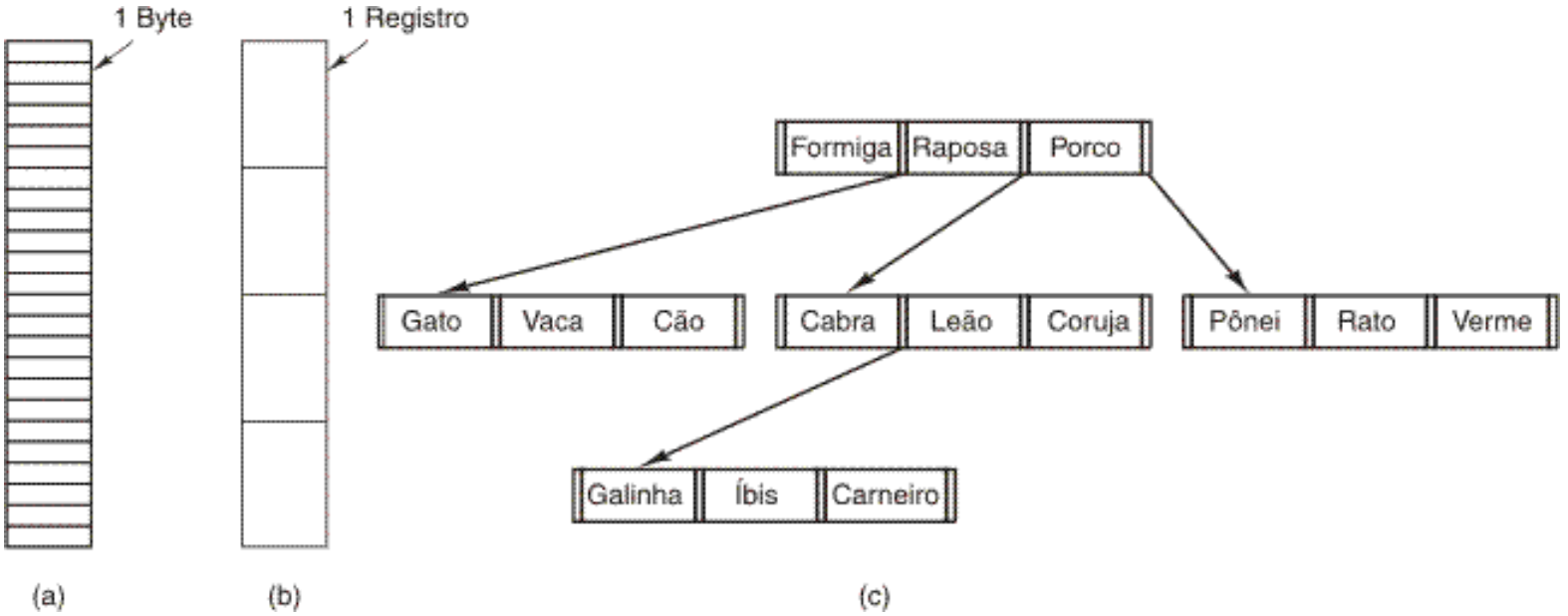
Sistemas de arquivos

- Tipos de arquivos

  (a) seqüência de bytes.

  (b) seqüência de registros.

  (c) árvore.



Sistemas de arquivos

- Acesso aos arquivos
  - Sequencial
    - Lê todos os bytes/registros desde o início.
    - Não pode saltar ou ler fora da seqüência.
    - Conveniente quando o meio era a fita magnética.
  - Aleatório
    - Bytes/registros lidos em qualquer ordem.
    - Essencial para sistemas de bancos de dados.

Sistemas de arquivos

- Atributos comuns dos arquivos

Atributo	Significado
Proteção	Quem pode ter acesso ao arquivo e de que maneira
Senha	Senha necessária para ter acesso ao arquivo
Criador	ID da pessoa que criou o arquivo
Proprietário	Atual proprietário
Flag de apenas para leitura	0 para leitura/escrita; 1 se apenas para leitura
Flag de oculto	0 para normal; 1 para não exibir nas listagens
Flag de sistema	0 para arquivos normais; 1 para arquivos do sistema
Flag de repositório (*archive*)	0 se foi feita cópia de segurança; 1 se precisar fazer cópia de segurança
Flag ASCII/binário	0 para arquivo ASCII; 1 para arquivo binário
Flag de acesso aleatório	0 se apenas para acesso seqüencial; 1 para acesso aleatório
Flag de temporário	0 para normal; 1 para remover o arquivo na saída do processo
Flag de impedimento	0 para desimpedido; diferente de zero para impedido
Tamanho do registro	Número de bytes em um registro
Posição da chave	Deslocamento da chave dentro de cada registro
Tamanho da chave	Número de bytes no campo-chave
Momento da criação	Data e horário em que o arquivo foi criado
Momento do último acesso	Data e horário do último acesso ao arquivo
Momento da última mudança	Data e horário da última mudança ocorrida no arquivo
Tamanho atual	Número de bytes no arquivo
Tamanho máximo	Número de bytes que o arquivo pode vir a ter

Sistemas de arquivos

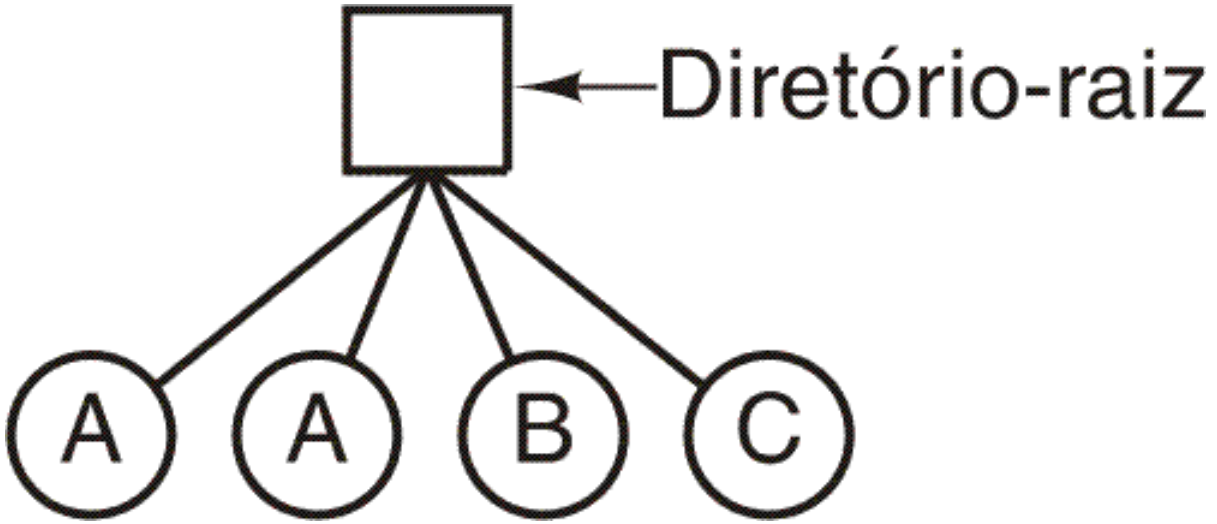
- Operações comuns com arquivos
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write
  - Append
  - Seek
  - Get attributes
  - Set Attributes
  - Rename

Sistemas de arquivos

- Diretórios
  - São utilizados para organizar os arquivos. Normalmente, os sistemas de arquivos criam uma estrutura chamada de diretórios.
  - Na estrutura hierárquica, cada diretório possui um caminho.
  - Cada diretório pode possuir somente arquivos, somente diretórios, ou uma combinação de ambos.

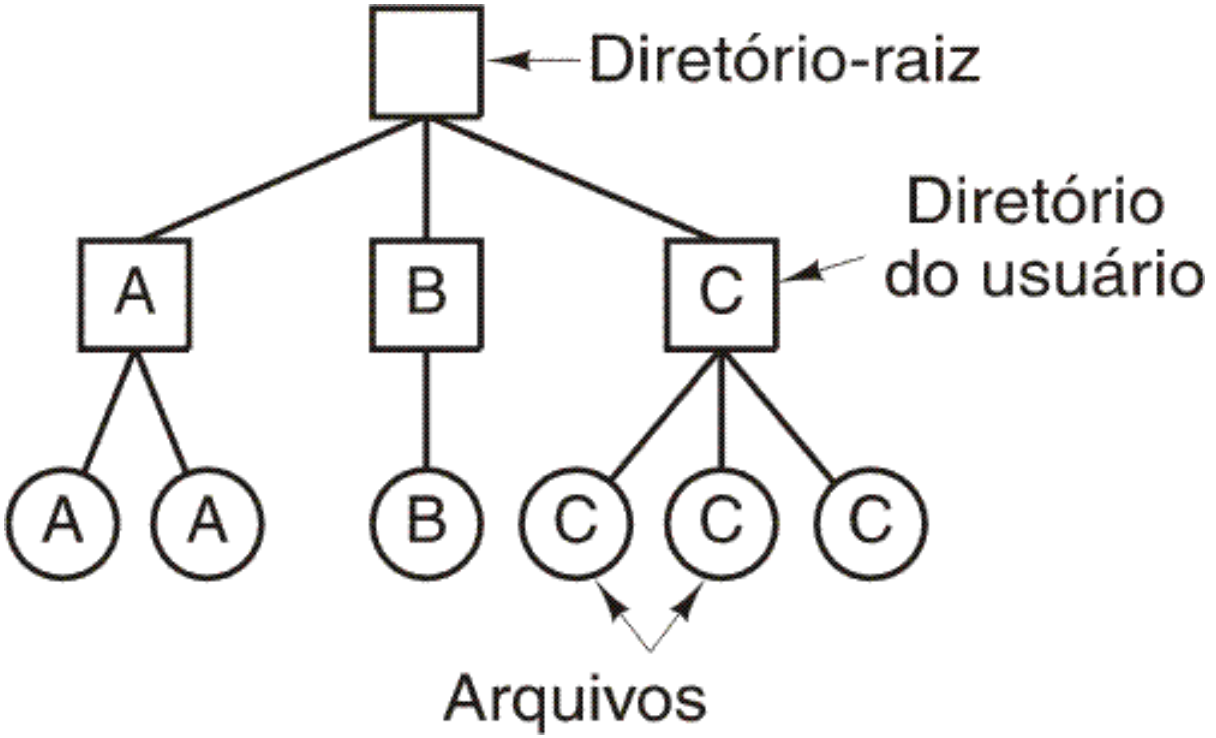
Sistemas de arquivos

- Diretórios
  - Único nível
    - Contém 4 arquivos.
    - Propriedades de 3 pessoas diferentes: A, B e C.



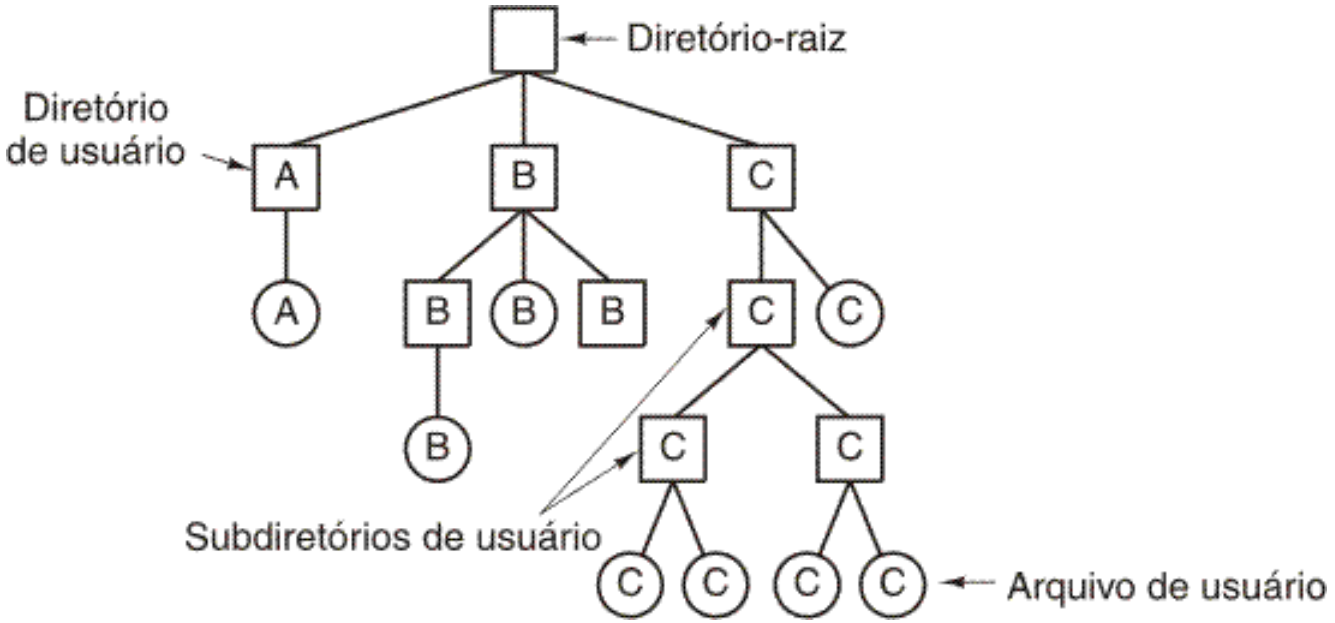
Sistemas de arquivos

- Diretórios
  - Dois níveis
    - As letras indicam os donos dos diretórios e arquivos.



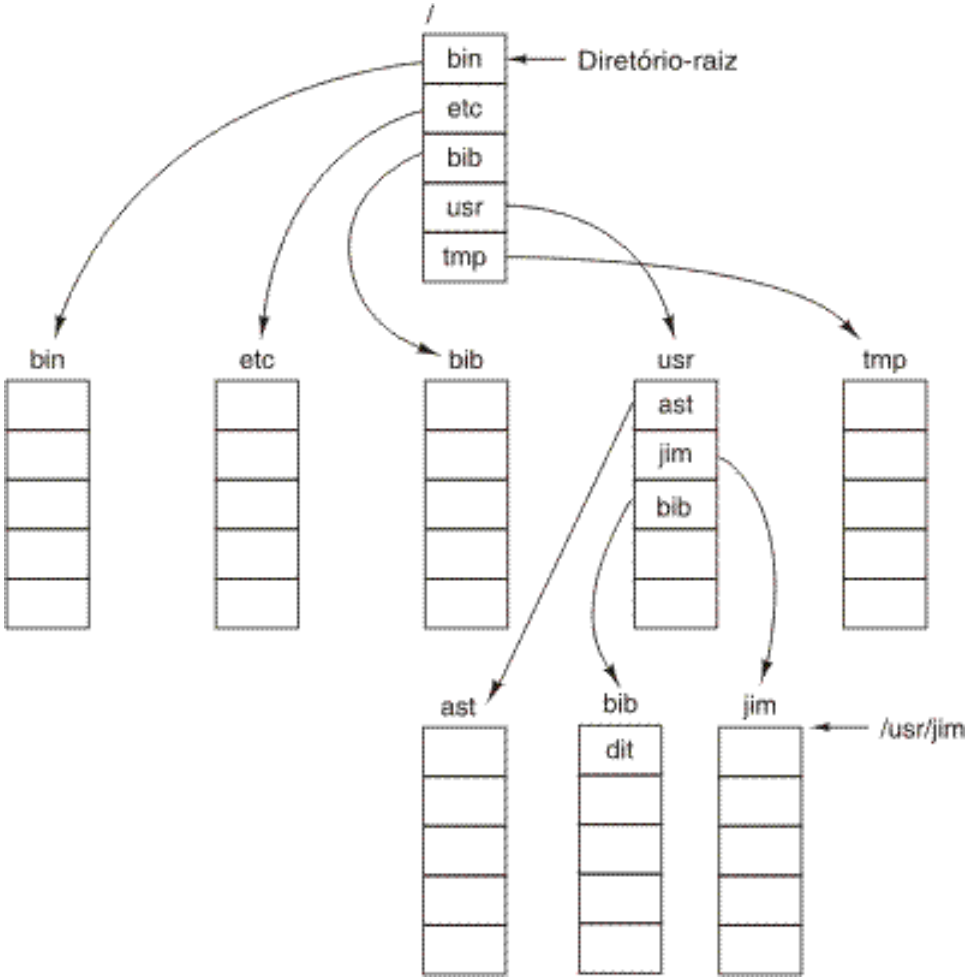
Sistemas de arquivos

- Diretórios
  - Múltiplos níveis
    - Um sistema de diretório hierárquico



Sistemas de arquivos

- Uma árvore de diretórios UNIX

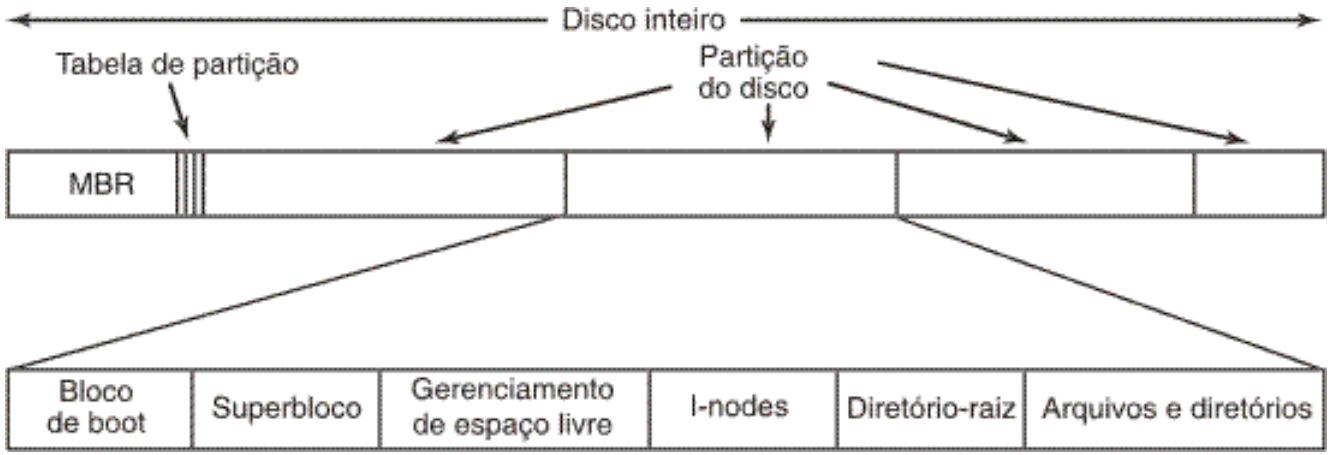


Sistemas de arquivos

- Operações com diretórios
  - Create
  - Delete
  - Opendir
  - Closedir
  - Readdir
  - Rename
  - Link
  - Unlink

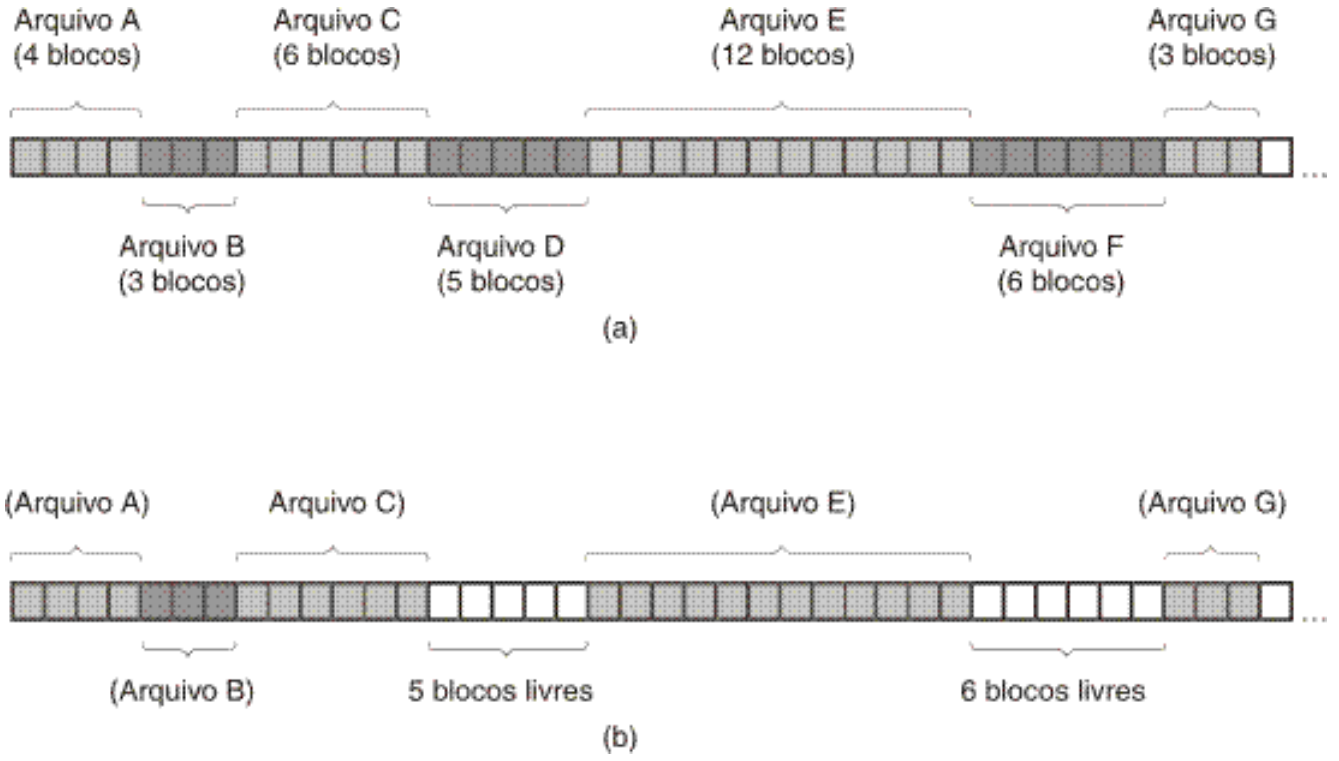
Sistemas de arquivos

- Um possível layout de sistema de arquivo



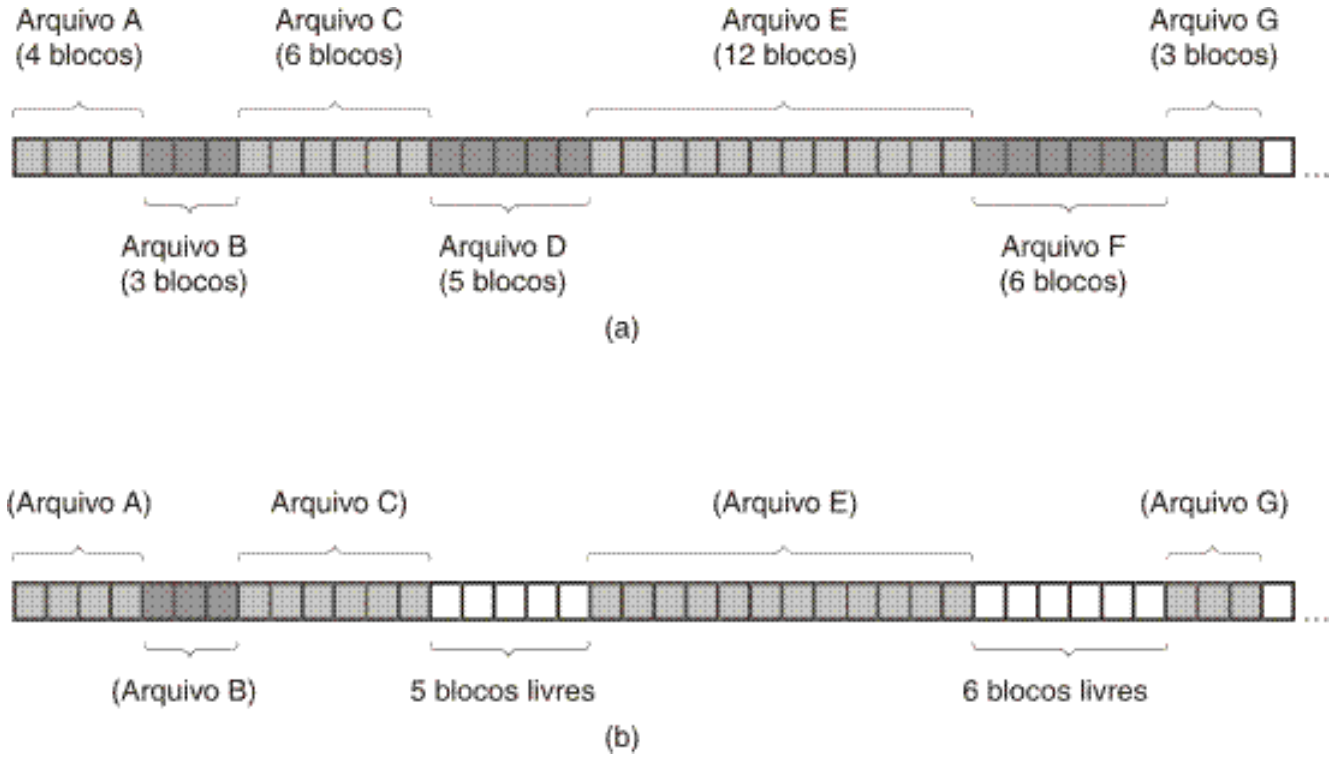
Sistemas de arquivos

- Implementação de arquivos
  - Alocação contígua do espaço em disco para 7 arquivos.



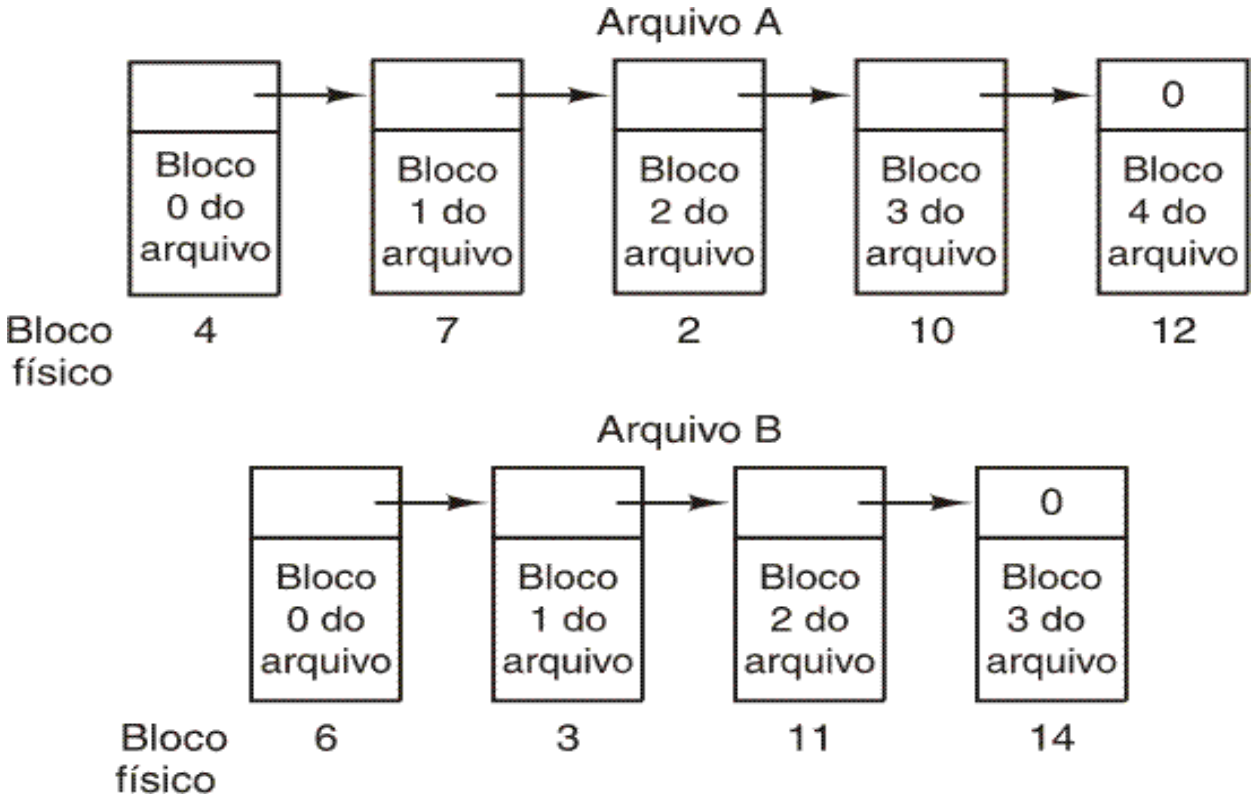
Sistemas de arquivos

- Implementação de arquivos
  - Estado do disco após os arquivos D e E terem sido removidos.



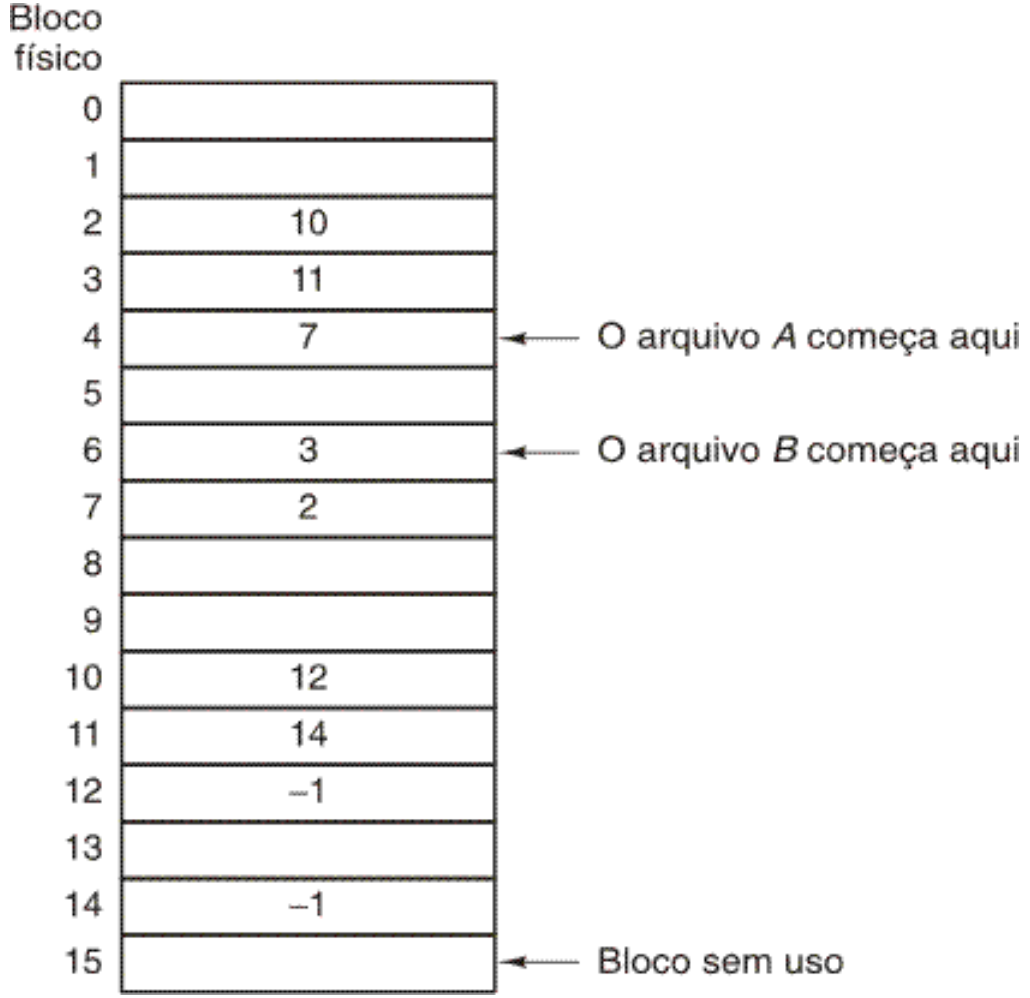
Sistemas de arquivos

- Implementação de arquivos
  - Armazenamento de um arquivo como uma lista encadeada de blocos de disco.



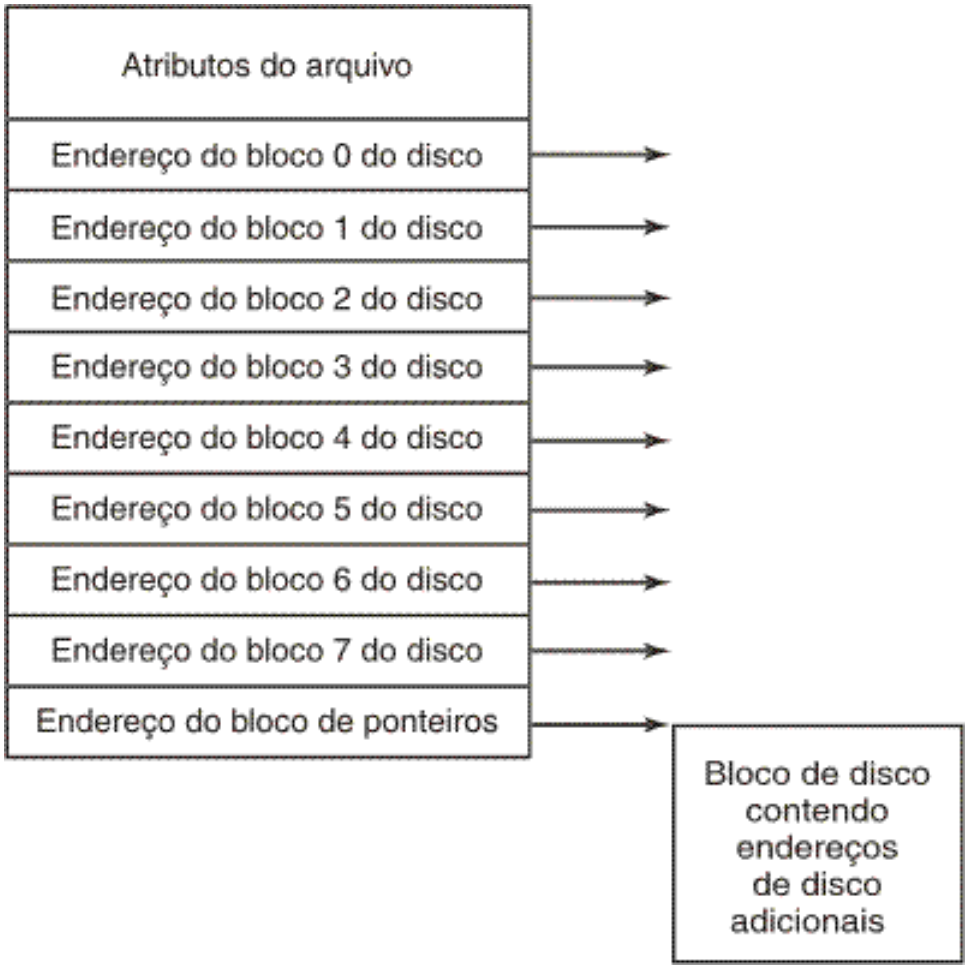
Sistemas de arquivos

- Alocação por lista encadeada usando uma tabela de alocação de arquivos em RAM.



Sistemas de arquivos

- Um exemplo de i-node

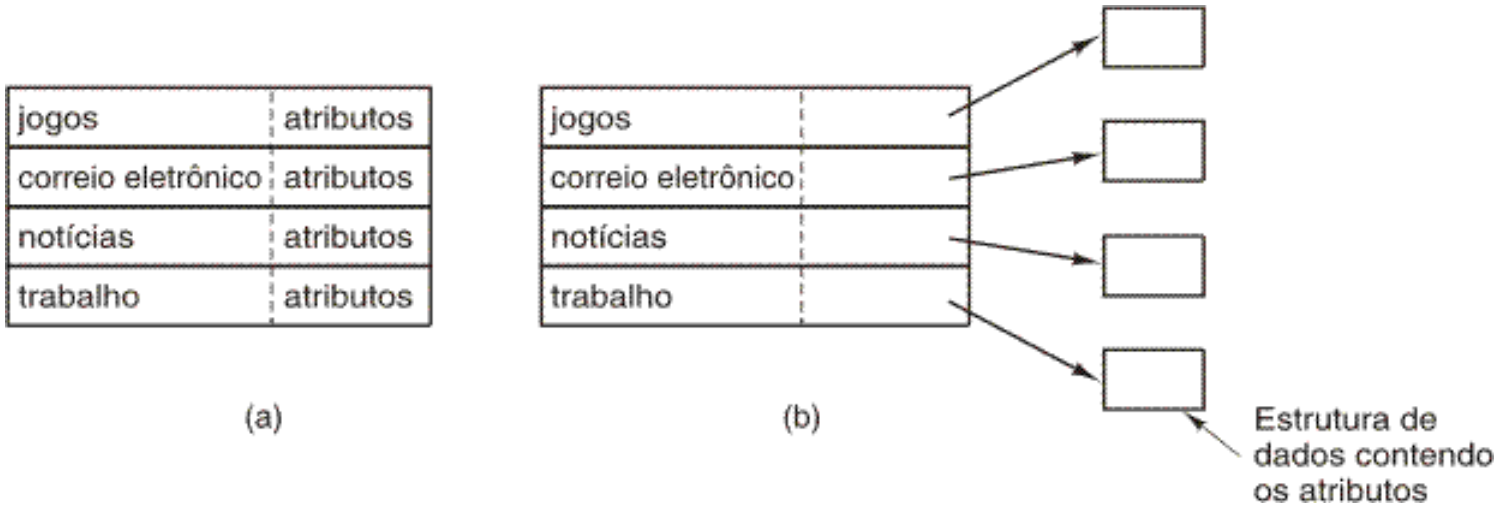


Sistemas de arquivos

- Implementação de diretórios

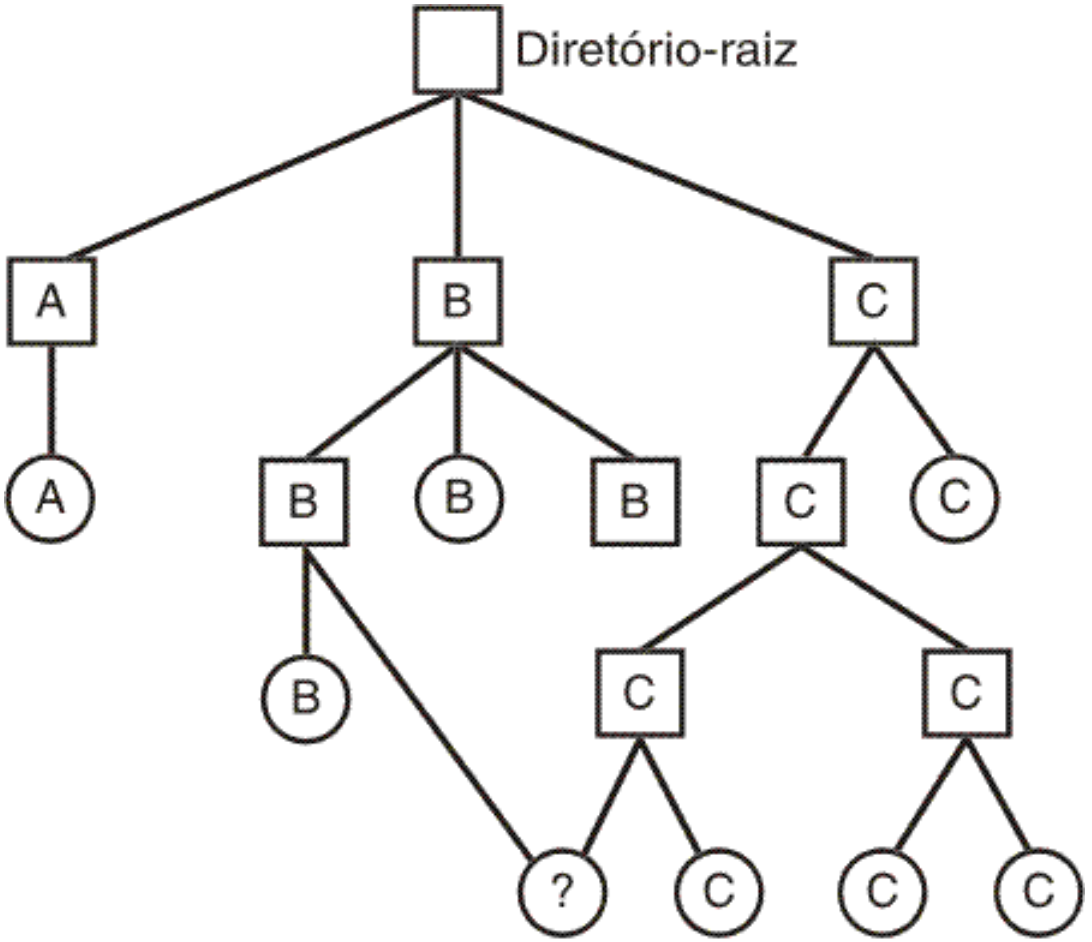
(a) Um diretório simples. Entradas de tamanho fixo e endereços de disco e atributos nas entradas de diretório.

(b) Diretório no qual cada entrada se refere apenas a um i-node.



Sistemas de arquivos

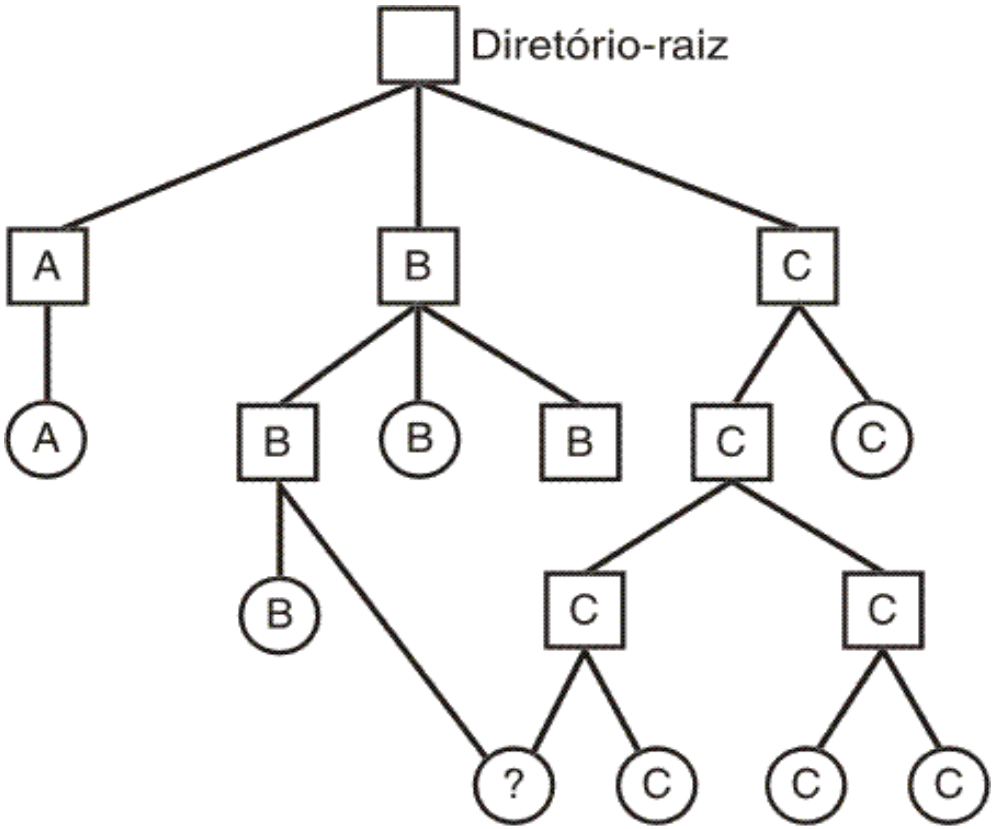
- Arquivos compartilhados



Arquivo compartilhado

Sistemas de arquivos

- Sistema de arquivo contendo um arquivo compartilhado



Arquivo compartilhado

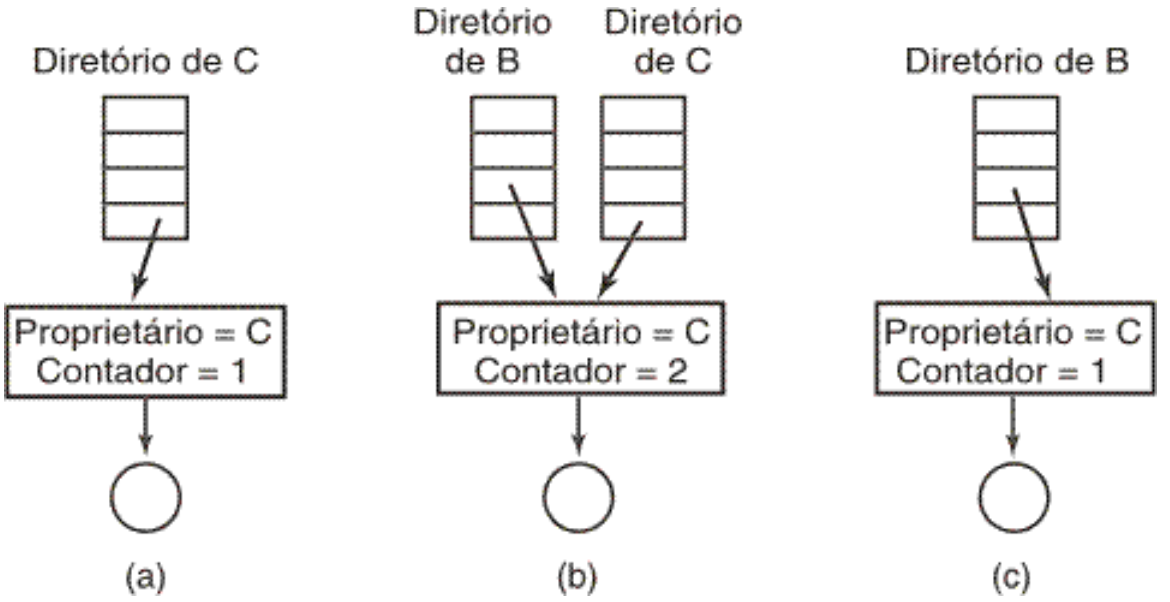
Sistemas de arquivos

- Arquivos compartilhados

  (a) Situação antes da ligação.

  (b) Depois da ligação ser criada.

  (c) Depois de o proprietário original remover o arquivo.



Exercícios

- Qual é a importância da extensão no nome de arquivo?
- Explique a operação de append num arquivo.
- O que acontece quando uma operação de link é executado num diretório? Dê um exemplo.
- O que é um i-node?
- Explique como é possível implementar um arquivo sendo compartilhado por mais de um diretório.

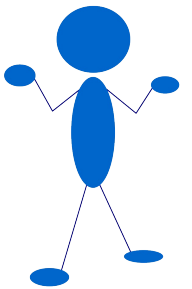
Entrada e saída de arquivos

- Os dados dos programas passam por três etapas:

- Sendo assim, a entrada e a saída de dados são partes fundamentais na construção de um programa.

Entrada e saída de arquivos

- Razões para uso de arquivos:
  - Menor tempo de teste de um programa.
  - Menor preocupação com tratamento de erros de entrada (dados inválidos, incompletos etc).
  - Processamento de séries de conjuntos de dados.
  - Processamento de grandes quantidades de dados.



Entrada e saída de arquivos

- Os arquivos podem ser abertos em dois modos: leitura e gravação.
- Um programa C com entrada e saída em arquivos é composto por duas estruturas FILE:
  - O arquivo de entrada (input) é aberto no modo leitura (read) e dele serão lidas as informações que o programa irá utilizar.
  - O arquivo de saída (output) é aberto para gravação (write) e nele serão gravados os resultados obtidos pelo programa.

Entrada e saída de arquivos

- A linguagem C não possui nenhum comando de entrada/saída.
- Todas as operações de E/S ocorrem mediante chamadas a funções da biblioteca C padrão.
- Embora tal característica possa, a primeira vista, parecer uma desvantagem, na verdade ela permite que o sistema de arquivos de C seja extremamente poderoso e flexível.

Entrada e saída de arquivos

- O sistema de E/S de C é único porque os dados podem ser transferidos na sua representação binária interna ou em formato texto, legível aos humanos.
- Isto torna fácil criar arquivos que satisfaçam qualquer necessidade.

Entrada e saída de arquivos

- O padrão C ANSI define um conjunto completo de funções de E/S que pode ser utilizado para ler e escrever qualquer tipo de dado.

Entrada e saída de arquivos

- O antigo padrão C UNIX contém dois sistemas distintos de rotinas que realizam operações de E/S:
  - O primeiro método assemelha-se vagamente ao definido pelo padrão C ANSI e é denominado sistema de arquivo com buffer.
  - O segundo é o sistema de arquivo tipo UNIX (não formatado ou sem buffer) definido apenas sobre o antigo padrão UNIX.

Entrada e saída de arquivos

- O padrão ANSI não define o sistema sem buffer porque os sistemas são amplamente redundantes.

E/S em C versus C++

- C++ suporta todo o sistema de arquivos definido pelo C ANSI.
- Entretanto, C++ também define seu próprio sistema de E/S, orientado a objetos, que inclui tanto funções, quanto operadores de E/S.
- O sistema de E/S C++ duplica por completo a funcionalidade do sistema de E/S C ANSI.

Streams

- O sistema de arquivos de C é projetado para trabalhar com uma ampla variedade de dispositivos, incluindo:
  - Terminais.
  - Acionadores de disco.
  - Acionadores de fita.

Streams

- Embora cada um daqueles dispositivos seja muito diferente, o sistema de arquivos com buffer transforma-os em um dispositivo lógico chamado de stream.

Streams

- Todas as streams comportam-se de forma semelhante.
- Pelo fato de as streams serem totalmente independentes do dispositivo, a mesma função pode escrever em um arquivo em disco ou em algum outro dispositivo, como o console.
- Existem dois tipos de streams: texto e binária.

Streams

- Um stream de texto é uma seqüencia de caracteres.
- O padrão ANSI permite, mas não exige, que um stream de texto seja organizado em linhas terminadas por um caractere de nova linha.
- Porém, o caractere de nova linha é opcional na última linha e é determinado pela implementação.

Streams

- Uma stream binária é uma sequência de bytes com uma correspondência de um para um com aqueles encontrados no dispositivo externo.
- O número de bytes escritos (ou lidos) é o mesmo que o encontrado no dispositivo externo.

Manipulação de arquivos em C

- Em C, um arquivo pode ser qualquer coisa, desde um arquivo em disco até um terminal ou uma impressora.
- Associa-se um stream com um arquivo específico, realizando uma operação de abertura.
- Uma vez o arquivo aberto, informações podem ser trocadas entre ele e o seu programa.

Manipulação de arquivos em C

- Nem todos os arquivos apresentam os mesmo recursos.
- Por exemplo, um arquivo em disco pode suportar acesso aleatório, enquanto um teclado não pode.
- Isso revela um ponto importante sobre o sistema de E/S de C: todas as streams são iguais, mas não todos os arquivos.

Manipulação de arquivos em C

- Todos os arquivos são fechados automaticamente quando o programa termina normalmente em:
  - main ( ) retornando ao sistema operacional.
  - uma chamada a exit ( ).

Manipulação de arquivos em C

- Cada stream associada a um arquivo tem uma estrutura de controle de arquivo do tipo FILE.
- Essa estrutura é definida no cabeçalho *stdio.h*.
- A maioria das funções começa com a letra f.

Manipulação de arquivos em C

- Um ponteiro de arquivo é um ponteiro para informações que definem várias coisas sobre o arquivo: nome, status e a posição atual do arquivo.
- Um ponteiro de arquivo é uma variável ponteiro do tipo FILE.

Manipulação de arquivos em C

- Para ler ou escrever arquivos, seu programa precisa usar ponteiros de arquivos. Para obter uma variável ponteiro de arquivo, use o comando:
  - FILE *fp;

Manipulação de arquivos em C

- A função *fopen ( )* abre uma stream para uso e associa um arquivo a ela.
- Ela retorna o ponteiro de arquivo associado a este arquivo.
- Sintaxe:
  - FILE fopen (const char* nomearq, const char* modo*);

Manipulação de arquivos em C

- Para abrir um arquivo no modo leitura:
  - FILE *entrada = fopen (“dados.dat”, “r”);
- Para abrir um arquivo no modo escrita:
  - FILE *saída = fopen (“dados.dat”, “w”);
- Em caso de erro, a função fopen retorna NULL.
- O arquivo de saída é sempre criado e, caso ele já exista, seu conteúdo anterior é descartado.

Manipulação de arquivos em C

- Uma maneira de tratar o erro seria escrever assim:

```
FILE     *fp;
if (( fp = fopen (“dados.dat”, “w”)) == NULL)
    {
        printf (“Não foi possível criar o arquivo \n”);
        exit(1);
    }
```

Manipulação de arquivos em C

- Opções de abertura de arquivos:

Modo	Significado
r	Abre um arquivo-texto para leitura
w	Cria um arquivo-texto para escrita
a	Anexa a um arquivo-texto
rb	Abre um arquivo binário para leitura
wb	Cria um arquivo binário para escrita
ab	Anexa um arquivo binário
r+	Abre um arquivo-texto para leitura/escrita
w+	Cria um arquivo-texto para leitura/escrita
a+	Anexa ou cria um arquivo-texto para leitura/escrita
r+b	Abre um arquivo binário para leitura/escrita
w+b	Cria um arquivo binário para leitura/escrita
a+b	Anexa um arquivo binário para leitura/escrita

Manipulação de arquivos em C

- Se você deseja adicionar ao final do arquivo, deve usar o modo “a”.
- A função *fclose ( )* fecha uma stream que foi aberta por meio de uma chamada a *fopen ( )*.
- Ela escreve qualquer dado que ainda permanece no buffer de disco no arquivo e, então, fecha normalmente o arquivo em nível de sistema operacional.

Manipulação de arquivos em C

- A função *fclose ( )* também libera o bloco de controle de arquivo associado à stream, deixando-o disponível para reutilização.
- Em muitos casos, há um limite do sistema operacional para o número de arquivos abertos simultaneamente, assim, você deve fechar um arquivo antes de abrir outro.

Manipulação de arquivos em C

- A função *fclose ( )* tem a seguinte sintaxe:
  - int fclose (FILE *fp);
- Onde fp é o ponteiro de arquivo devolvido pela chamada a *fopen( )*.
- Um valor de retorno zero significa uma operação de fechamento bem sucedida.
- Qualquer outro valor indica um erro.

Manipulação de arquivos em C

- A função padrão *ferror ( )* pode ser utilizada para determinar e informar qualquer problema.
- Geralmente, *fclose ( )* falhará quando um disco tiver sido retirado prematuramente do acionador ou não houver mais espaço em disco.

Manipulação de arquivos em C

- A entrada de dados é feita através da função *fscanf*:

```
int n1; double n2; char str[20];

fscanf(entrada, "%d",  &n1);
fscanf(entrada, "%lf", &n2);
fscanf(entrada, "%s",  str);
```

Manipulação de arquivos em C

- O primeiro argumento é o nome do arquivo a ser lido.
- O segundo argumento é uma string contendo o formato das variáveis que estão nos argumentos seguintes (não se esqueça do & antes de cada variável, com exceção das strings).
- O terceiro argumento é a variável a ser escrita no arquivo.

Manipulação de arquivos em C

- A saída de dados é feita através da função fprintf (o & não é utilizado):

```
fprintf(saida, "%d", 3);
fprintf(saida, "%lf", 5.3);
fprintf(saida, "%s %d", "Nota:", 7);
fprintf(saida, "Algoritmos");
```

- A quebra de linha é feita através de "\n":

```
fprintf(saida, "%d\n", 2);
fprintf(saida, "\n%lf", -2.7);
fprintf(saida, "\n%s\n", "TEXTO");
fprintf(saida, "\nNota:\n%d\n", 8);
```

Manipulação de arquivos em C

- Os símbolos utilizados em fscanf e fprintf são:
  - %d um número inteiro decimal.
  - &lf um double.
  - %s uma string.
  - \n uma quebra de linha.
- Existem diversos outros símbolos.
- Se o fim do arquivo for alcançado, fscanf retorna EOF.

Manipulação de arquivos em C

- Quando um arquivo não for mais necessário, é preciso fechá-lo, ou seja, liberar os recursos utilizados por ele.

```
fclose(entrada);
fclose(saida);
```

- Para isto, utiliza-se a função fclose.
- Assim, os arquivos de entrada e saída são fechados, e a memória ocupada por eles é liberada para que possa ser usada novamente.

Manipulação de arquivos em C

- Outra opção para manipular arquivos de entrada e saída em C/C++ é utilizar a função freopen.

```
freopen("entrada.dat", "r", stdin);
freopen("saída.dat", "w", stdout);
```

Manipulação de arquivos em C

- Os exemplos anteriores estão redirecionando a entrada e saída padrão, respectivamente, para o “entrada.dat” e “saída.dat”.
- Desta forma, podem-se utilizar as funções scanf e printf para acessar ou imprimir dados no arquivo.

Manipulação de arquivos em C

- Em C++, podemos utilizar a E/S de arquivos com maior facilidade do que em C. Para isto, utilizam-se as classes ifstream e ofstream de <fstream>.
- Para lermos do arquivo de entrada, precisamos criar uma instância de ifstream:

```
ifstream entrada("teste.in");
```

- Essa instância de ifstream só tem permissão para leitura de “teste.in”.

Manipulação de arquivos em C

- Para podermos gravar num arquivo, criamos instâncias de ofstream:

```
ofstream saida("teste.out");
```

- Essa instância de ofstream tem permissão para gravar em “teste.out”.
- O arquivo de saída é sempre criado, e caso ele já exista, seu conteúdo anterior é descartado.

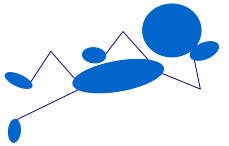
Manipulação de arquivos em C

- Quando o arquivo não for mais necessário, é preciso fechá-lo, ou seja, liberar os recursos utilizados por ele. Para isto, utiliza-se a função *close ( )*:

```
entrada.close();
saida.close();
```

- Assim, os arquivos de entrada e saída são fechados, e a memória ocupada por eles é liberada para que possa ser usada novamente pelo sistema.

Exercícios



- O que você entender por stream?
- Qual é a biblioteca em C utilizada para manipulação de arquivos?
- Comente duas razões para se utilizar arquivos.
- Qual é o papel dos ponteiros na manipulação de arquivos em C?
- Quais são os erros mais comuns ao se tentar gravar num arquivo?
- Qual é a importância de se fechar o arquivo após utilizá-lo?
- O que significa EOF?

Referências

(1) Tanenbaum, Andrew S. ***Sistemas operacionais modernos***. *2 ed.* Prentice-Hall, 2003.

(2) Guimarães, Kátia S. & Salgado, Liliane R. B. Algoritmos e estruturas de dados. Disponível em: <http://moreno.cin.ufpe.br/~if672/2007.2> Acesso em 24/02/2008.

(3) How C Programming Works. Disponível em: < http://computer.howstuffworks.com/c.htm>. Acesso em 24/02/2008.

(4) Oliveira, Lucas F. **Manipulação de arquivos**. Disponível em: <lferrarioliveira.googlepages.com/Prog1_Aula_2.pdf>. Acesso em 24/02/2008.