Capítulo 6

Sistemas de Arquivos

6.1 Arquivos
6.2 Diretórios
6.3 Implementação do sistema de arquivos
6.4 Exemplos de sistemas de arquivos

Armazenamento da Informação a Longo Prazo

1. Deve ser possível armazenar uma quantidade muito grande de informação

2. A informação deve sobreviver ao término do processo que a usa

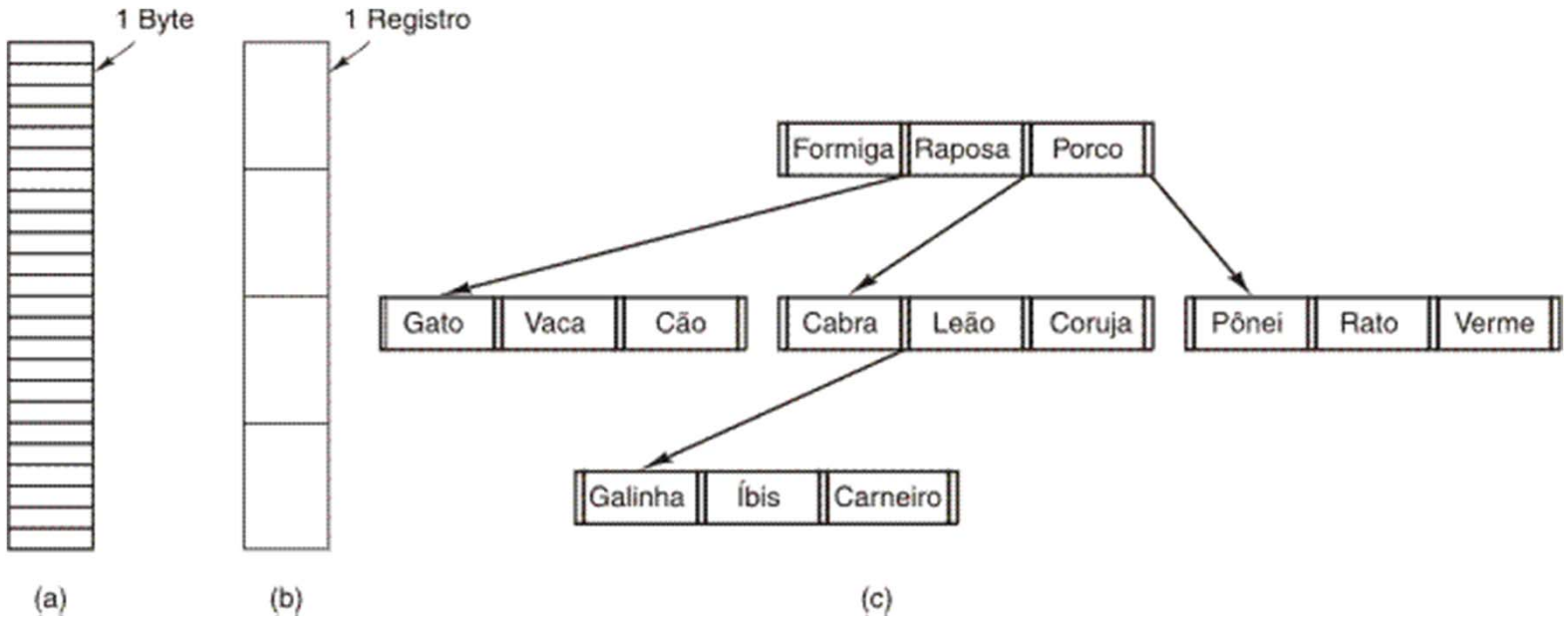
3. Múltiplos processos devem ser capazes de acessar a informação concorrentemente

Nomeação 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

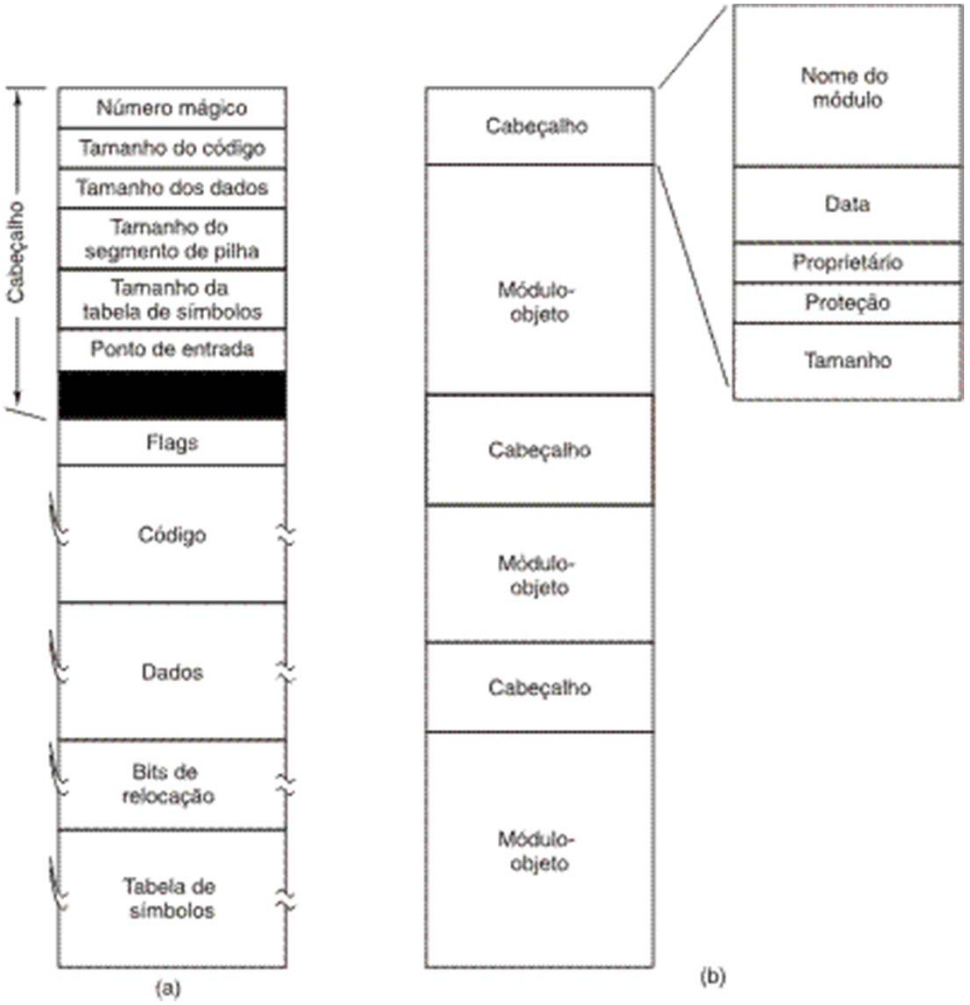
Extensões típicas de arquivos

Estrutura de Arquivos



- Três tipos de arquivos
  a) seqüência de bytes
  b) seqüência de registros
  c) árvore

Tipos de Arquivos



(a) Um arquivo executável (b) Um repositório *(archive)*

Acesso aos Arquivos

- Acesso sequencial
  - lê todos os bytes/registros desde o início
  - não pode saltar ou ler fora de seqüência
  - conveniente quando o meio era a fita magnética
- Acesso aleatório
  - bytes/registros lidos em qualquer ordem
  - essencial para sistemas de bases de dados
  - ler pode ser …
    - mover marcador de arquivo (seek), e então ler ou …
    - ler e então mover marcador de arquivo

Atributos de 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

Possíveis atributos de arquivos

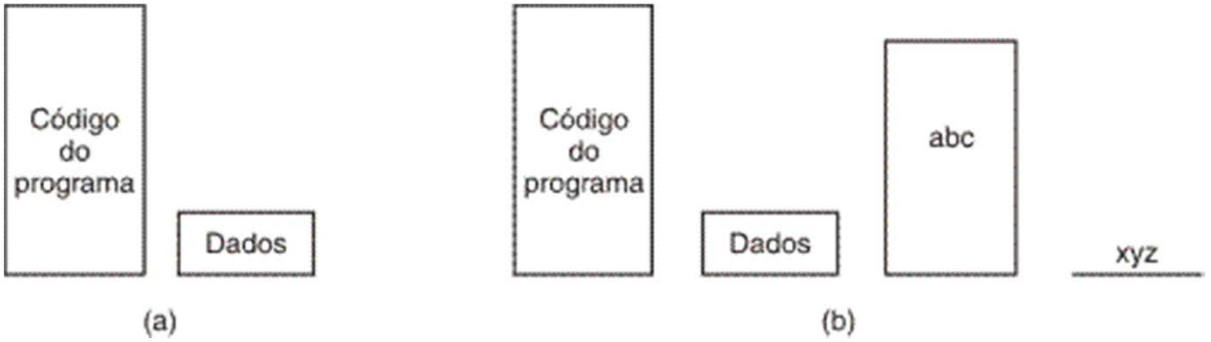
Operações com Arquivos

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

Exemplo de um Programa com Chamadas ao Sistema para Arquivos

```
/* Programa que copia arquivos. Verificação e relato de erros é mínimo.*/

#include <sys/types.h>                  /* inclui os arquivos de cabeçalho necessários*/
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /*  protótipo ANS */

#define BUF_SIZE 4096                   /* usa um tamanho de buffer de 4096 bytes*/
#define OUTPUT_MODE 0700                /* bits de proteção para o arquivo de saída*/

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);             /* erro de sintaxe se argc não for 3 */

    /* Abre o arquivo de entrada e cria o arquivo de saída */
    in_fd = open(argv[1], O_RDONLY);    /* abre o arquivo de origem */
    if (in_fd < 0) exit(2);             /* se não puder ser aberto, saia*/
    out_fd = creat(argv[2], OUTPUT_MODE);  /* cria o arquivo de destino*/
    if (out_fd < 0) exit(3);            /* se não puder ser criado, saia*/

    /* Laço de cópia */
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE); /* lê um bloco de dados */
    if (rd_count <= 0) break;           /* se fim de arquivo ou erro, sai do laço */
        wt_count = write(out_fd, buffer, rd_count); /* escreve dados */
        if (wt_count <= 0) exit(4);     /* wt_count <= 0 é um erro */
    }

    /* Fecha os arquivos */
    close(in_fd);
    close(out_fd);
    if (rd_count == 0)                  /* nenhum erro na última leitura*/
        exit(0);
    else
        exit(5);                        /* erro na última leitura*/
}
```

Arquivos Mapeados em Memória



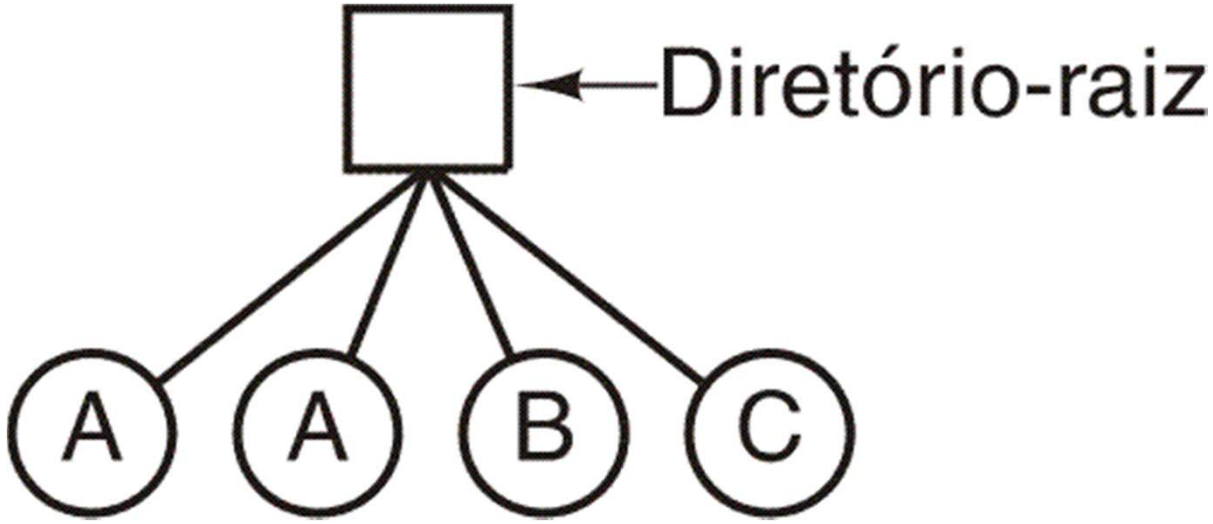
(a) Um processo segmentado antes de mapear arquivos em seu espaço de endereçamento

(b) Processo depois do mapeamento

arquivo *abc* existente em um segmento

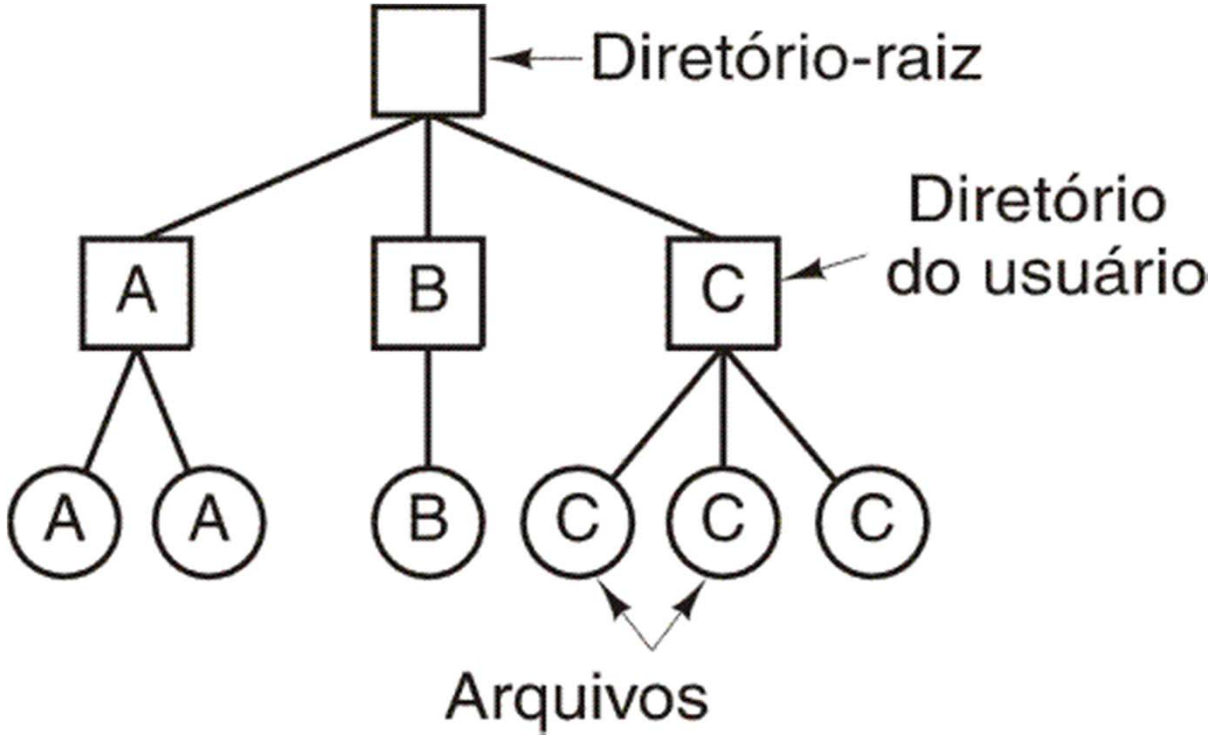
criando novo segmento para *xyz*

Diretórios
Sistemas de Diretório em Nível Único



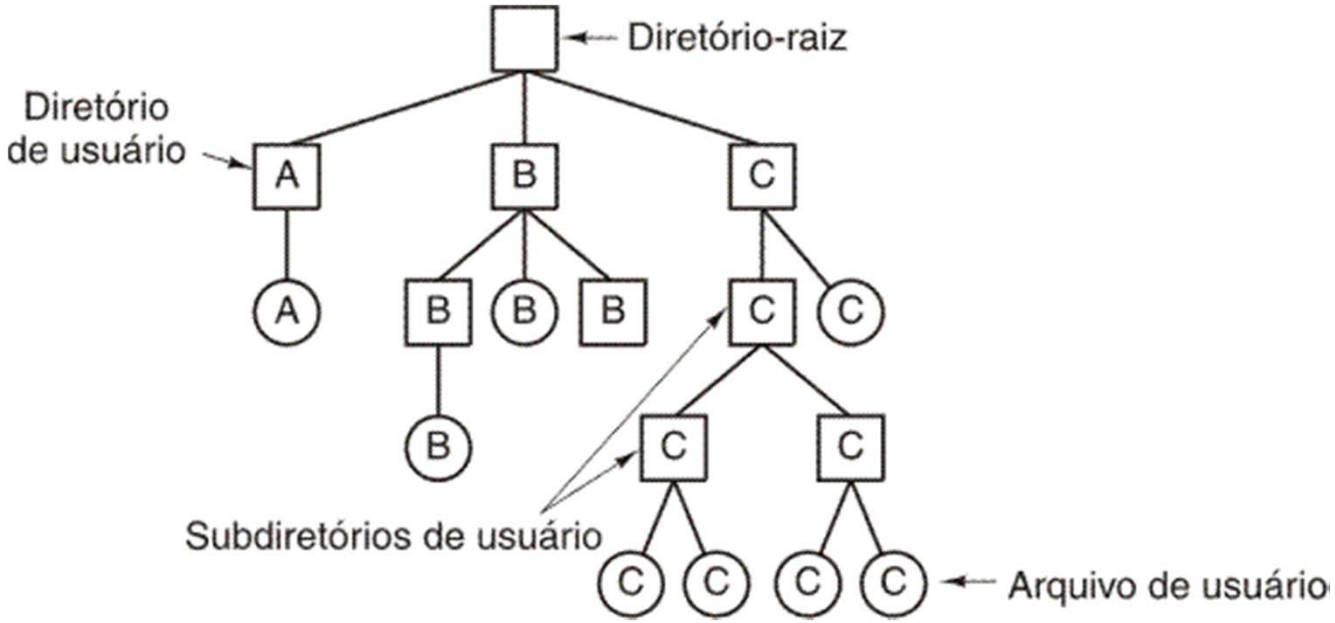
- Um sistema de diretório de nível único
  - contém 4 arquivos
  - propriedades de 3 pessoas diferentes, A, B, e C

Sistemas de Diretórios em Dois Níveis



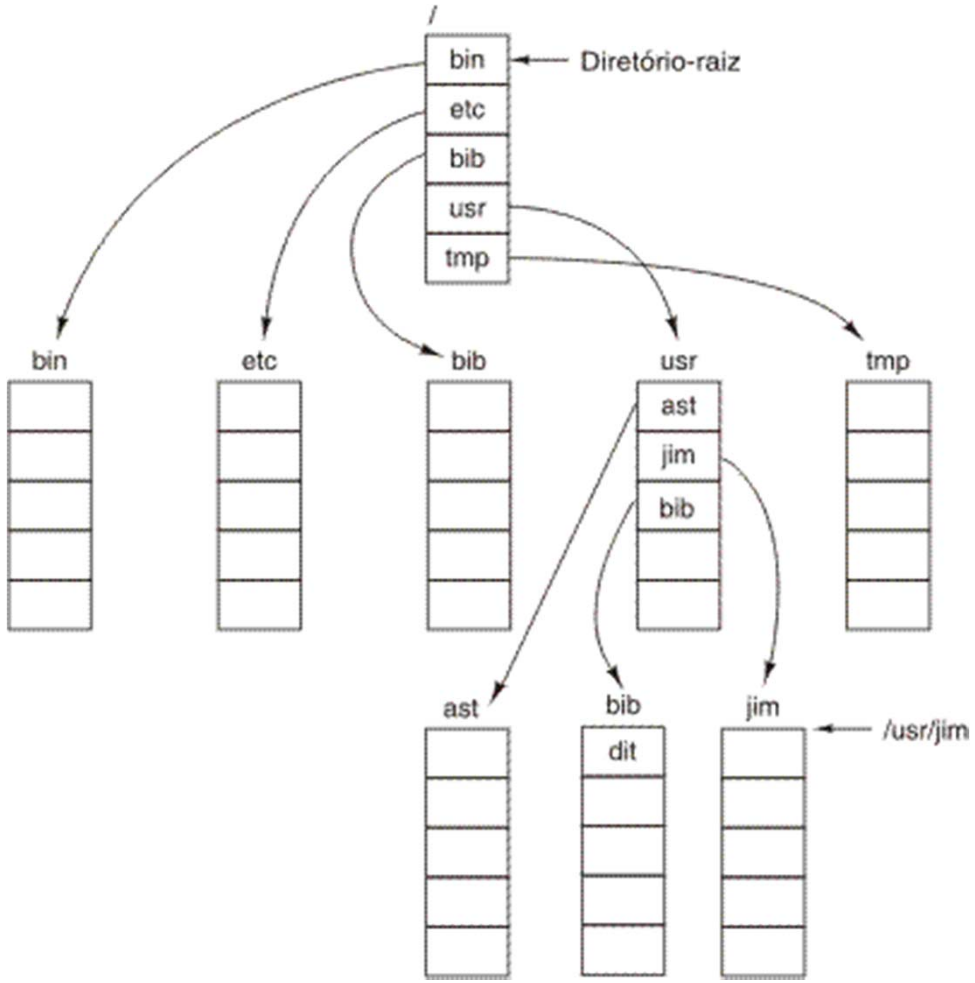
As letras indicam os donos dos diretórios e arquivos

Sistemas de Diretórios Hierárquicos



Um sistema de diretório hierárquico

Nomes de Caminhos



Uma árvore de diretórios UNIX

Operações com Diretórios

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

Operações com Arquivos e Diretórios em JAVA

```java
package arquivos;

import java.awt.Desktop;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Date;
import java.text.SimpleDateFormat;

public class OperacoesArquivos {

    public static void criaDirArq(){
        /*
        Verifica a existência do Diretório e do Arquivo e,
        se necessário, cria.
        */
        String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
        File diretorio = new File(newDir);
        if (!diretorio.exists()){
            diretorio.mkdir();
        }
        String newArq = "arquivo.txt";
        File arquivo = new File(diretorio,newArq);
        if (!arquivo.exists()){
            try {
                arquivo.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
```

```java
public static void propriedadesArquivos(){
    //Modificando as propriedades dos arquivos

    String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
    File diretorio = new File(newDir);
    String newArq = "arquivo.txt";
    File arquivo = new File(diretorio,newArq);

    arquivo.setReadable(false);
    arquivo.setReadable(true, true);
    arquivo.setReadOnly();
    arquivo.setWritable(false);
    arquivo.setWritable(true, true);
    boolean oculto = arquivo.isHidden();
    if (oculto){
        System.out.println("Arquivo oculto");
    } else {
        System.out.println("Arquivo não oculto");
    }
    boolean executavel = arquivo.canExecute();
    if (executavel){
        System.out.println("Arquivo executável");
    } else {
        System.out.println("Arquivo não executável");
    }
    boolean legivel = arquivo.canRead();
    if (legivel){
        System.out.println("Arquivo permite leitura");
    } else {
        System.out.println("Arquivo não permite leitura");
    }
    boolean permiteEscrita = arquivo.canWrite();
    if (permiteEscrita){
        System.out.println("Arquivo permite escrita");
    } else {
        System.out.println("Arquivo não permite escrita");
    }
    long tamanho = arquivo.length();
    System.out.println(tamanho + " bytes");
    long ultimaModificacao = arquivo.lastModified();
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm");
    Date resultdate = new Date(ultimaModificacao);
    System.out.println(sdf.format(resultdate));

}
```

Operações com Arquivos e Diretórios em JAVA

```
85      public static void listaDir(){
86
87          /*
88          Lista o conteúdo de um diretório diferenciando
89          arquivos e diretorios
90           */
91          String docDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
92          File documentos = new File(docDir);
93          for (File files : documentos.listFiles()){
94              if (files.isDirectory()){
95                  System.out.println(files + " é um diretório");
96              } else {
97                  if (files.isFile()){
98                      System.out.println(files + " é um arquivo");
99                  }
100             }
101         }
102     }
103
104     public static void deletaArqDir(){
105         // Deletar arquivos e diretórios
106         String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
107         File diretorio = new File(newDir);
108         if (!diretorio.exists()){
109             diretorio.mkdir();
110         }
111         int numArquivos = diretorio.listFiles().length;
112         if (numArquivos != 0){
113             for (int i = 0 ; i < numArquivos ; i++){
114                 String arqDeletado = diretorio.listFiles()[i].getName();
115                 System.out.println(arqDeletado);
116                 File deletado = new File (diretorio,arqDeletado);
117                 System.out.println(deletado.delete());
118             }
119         }
120         if (diretorio.list().length == 0){
121             String nomeDir = diretorio.getName();
122             boolean deletado = diretorio.delete();
123             if (deletado){
124                 System.out.println("Diretório "+nomeDir+" excluído");
125             }
126
127         }
128     }
```

```
174     public static void escreveArquivo(){
175         // Escreve no arquivo
176         try {
177             String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
178             File diretorio = new File(newDir);
179             String newArq = "arquivo.txt";
180             File arquivo = new File(diretorio,newArq);
181             if (!arquivo.exists()){
182                 try {
183                     arquivo.createNewFile();
184                 } catch (IOException e) {
185                     e.printStackTrace();
186                 }
187             }
188
189             FileWriter fileWriter = new FileWriter(arquivo);
190             PrintWriter printWriter = new PrintWriter(fileWriter);
191             printWriter.println("Qualquer coisa");
192             printWriter.println("de teste");
193             printWriter.close();
194             fileWriter.close();
195         } catch (IOException e) {
196             e.printStackTrace();
197         }
198
199     }
```

Operações com Arquivos e Diretórios em JAVA

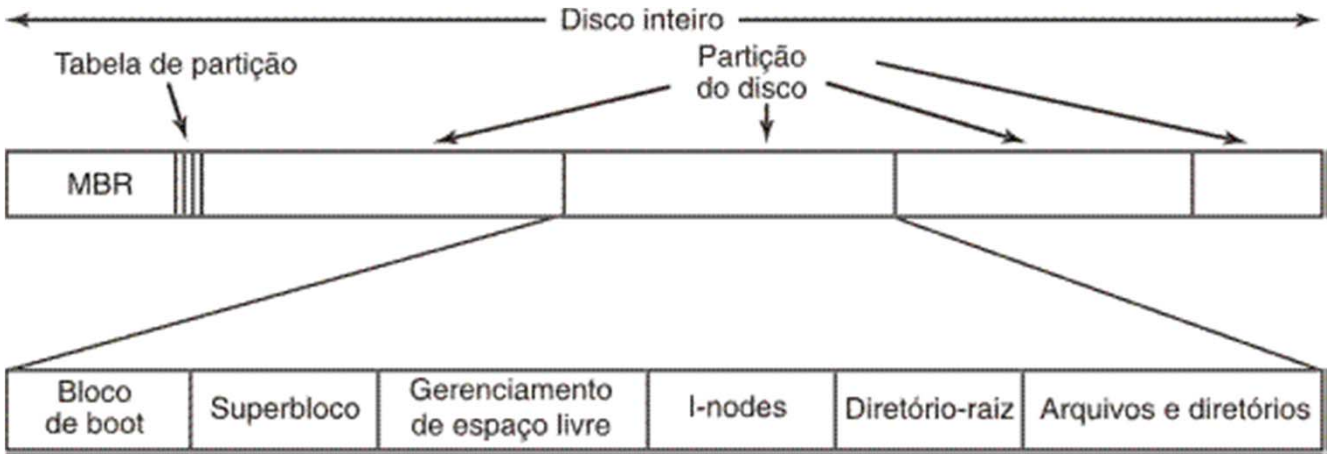
```
201     public static void leArquivo(){
202         String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
203         File diretorio = new File(newDir);
204         String newArquivo = "arquivo.txt";
205         File arq = new File(diretorio,newArquivo);
206         if (!arq.exists()){
207             try {
208                 arq.createNewFile();
209             } catch (IOException e) {
210                 e.printStackTrace();
211             }
212         }
213         try {
214             StringBuffer stringBuffer = new StringBuffer();
215             FileInputStream fileInputStream = new FileInputStream(arq);
216             InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
217             BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
218             String line = bufferedReader.readLine();
219             while (line != null){
220                 stringBuffer.append(line+"\n");
221                 line = bufferedReader.readLine();
222             }
223             fileInputStream.close();
224             inputStreamReader.close();
225             bufferedReader.close();
226             System.out.println(stringBuffer.toString());
227
228         } catch (FileNotFoundException e) {
229             e.printStackTrace();
230         } catch (IOException e) {
231             e.printStackTrace();
232         }
233
234     }
```

Operações com Arquivos e Diretórios em JAVA

```
130     public static void usaDesktop(){
131         //Usando opções de Desktop
132
133         String newDir = "C:\\Users\\Leandro\\Downloads\\arquivos";
134         File diretorio = new File(newDir);
135         String newArq = "arquivo.txt";
136         File arquivo = new File(diretorio,newArq);
137
138
139         Desktop desktop = Desktop.getDesktop();
140         boolean abrir = desktop.isSupported(Desktop.Action.OPEN);
141         if (abrir){
142             System.out.println("O sistema permite associar o arquivo a um editor e abrí-lo");
143         } else {
144             System.out.println("O sistema não permite associar o arquivo a um editor e abrí-lo");
145         }
146         boolean editar = desktop.isSupported(Desktop.Action.EDIT);
147         if (editar){
148             System.out.println("O sistema permite associar o arquivo a um editor e editá-lo");
149         } else {
150             System.out.println("O sistema não permite associar o arquivo a um editor e editá-lo");
151         }
152         boolean imprimir = desktop.isSupported(Desktop.Action.PRINT);
153         if (imprimir){
154             System.out.println("O sistema permite imprimir");
155         } else {
156             System.out.println("O sistema não permite imprimir");
157         }
158         try {
159             if (abrir) {
160                 desktop.open(arquivo);
161             }
162             if (imprimir) {
163                 desktop.print(arquivo);
164             }
165             if (editar){
166                 desktop.edit(arquivo);
167             }
168         } catch (IOException e) {
169             e.printStackTrace();
170         }
171
172     }
```

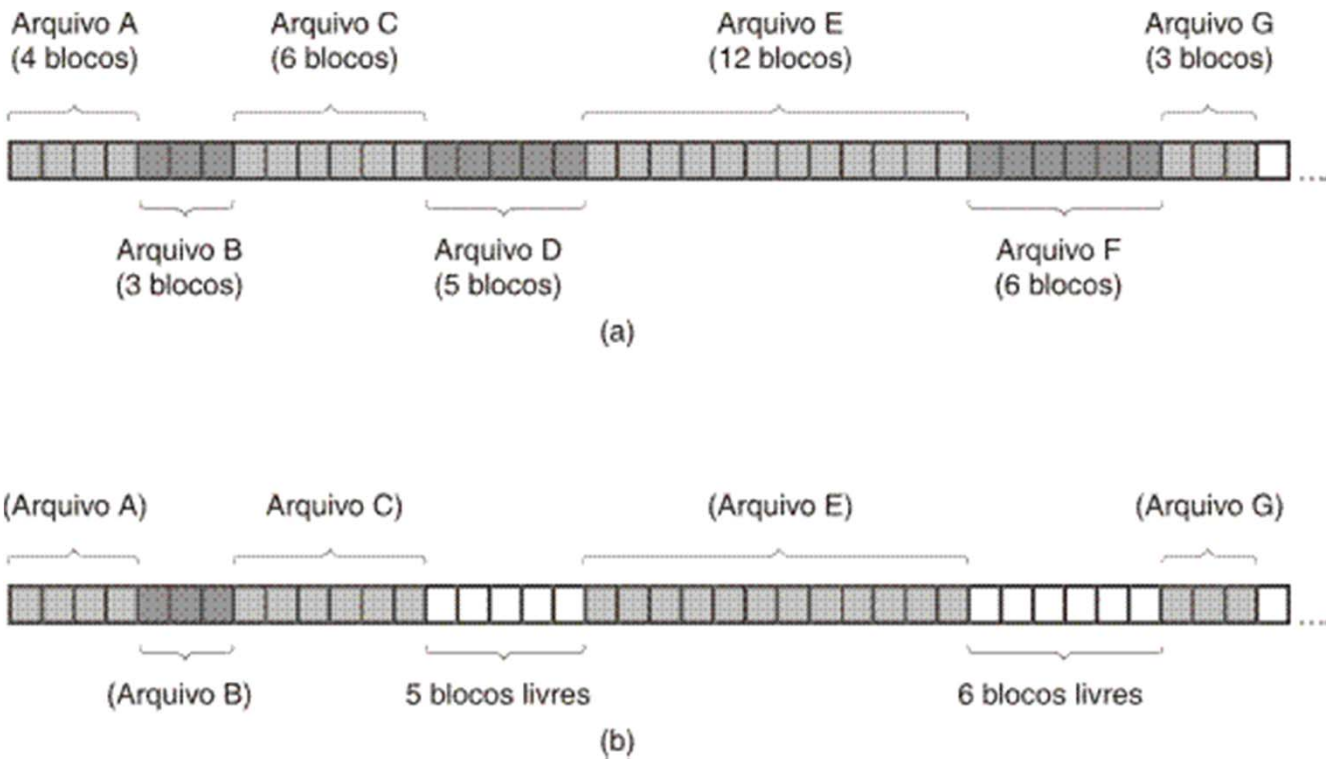
```
236     public static void main(String[] args) {
237         criaDirArq();
238         deletaArqDir();
239         escreveArquivo();
240         leArquivo();
241         listaDir();
242         propriedadesArquivos();
243         usaDesktop();
244     }
245
246 }
247
```

Implementação do Sistema de Arquivos



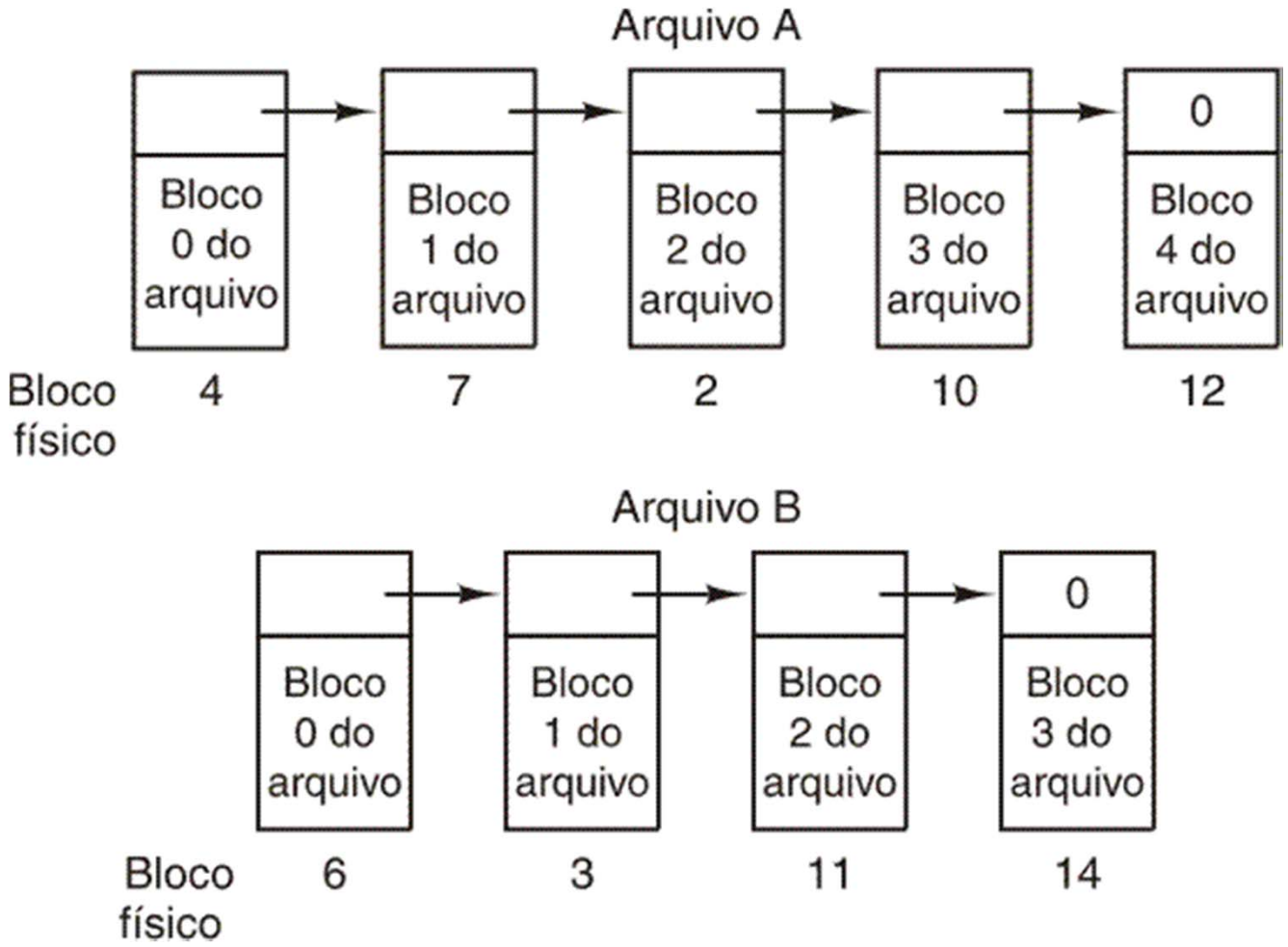
Um possível layout de sistema de arquivo

Implementação de Arquivos (1)



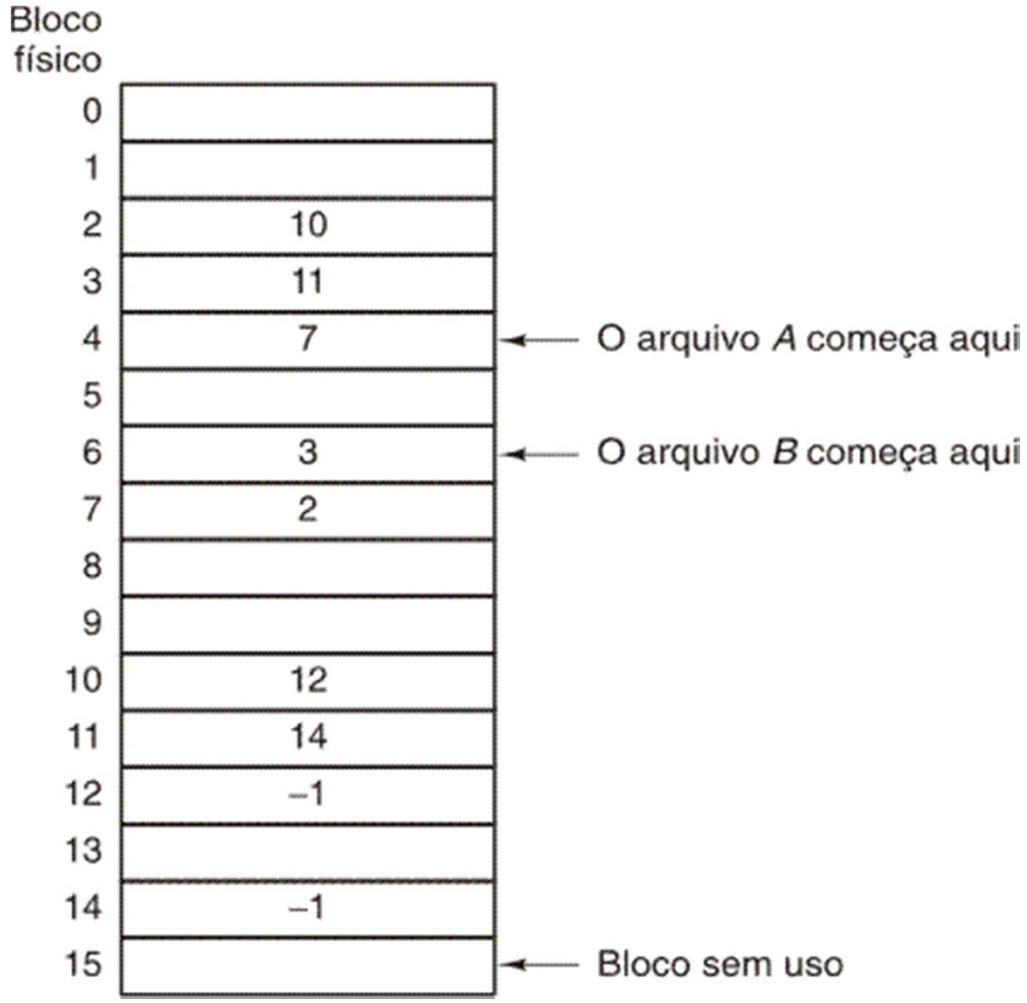
(a) Alocação contígua do espaço em disco para 7 arquivos
(b) Estado do disco depois dos arquivos *D* e *E* terem sido removidos

Implementação de Arquivos (2)



Armazenamento de um arquivo como uma lista encadeada de blocos de disco

Implementação de Arquivos (3)



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

Implementação de Arquivos (4)

Um arquivo é descrito por um i-node. O i-node é uma estrutura de dados com tamanho padrão de 128 bytes(o tamanho é definido na formatação).

A tabela seguinte mostra seus atributos:

Type	Field	Description
_ _u16	i_mode	File type and access rights
_ _u16	i_uid	Owner identifier
_ _u32	i_size	File length in bytes
_ _u32	i_atime	Time of last file access
_ _u32	i_ctime	Time that inode last changed
_ _u32	i_mtime	Time that file contents last changed
_ _u32	i_dtime	Time of file deletion
_ _u16	i_gid	Group identifier
_ _u16	i_links_count	Hard links counter
_ _u32	i_blocks	Number of data blocks of the file
_ _u32	i_flags	File flags
union	osd1	Specific operating system information
_ _u32 [EXT2_N_BLOCKS]	i_block	Pointers to data blocks
_ _u32	i_version	File version (for NFS)
_ _u32	i_file_acl	File access control list
_ _u32	i_dir_acl	Directory access control list
_ _u32	i_faddr	Fragment address
union	osd2	Specific operating system information

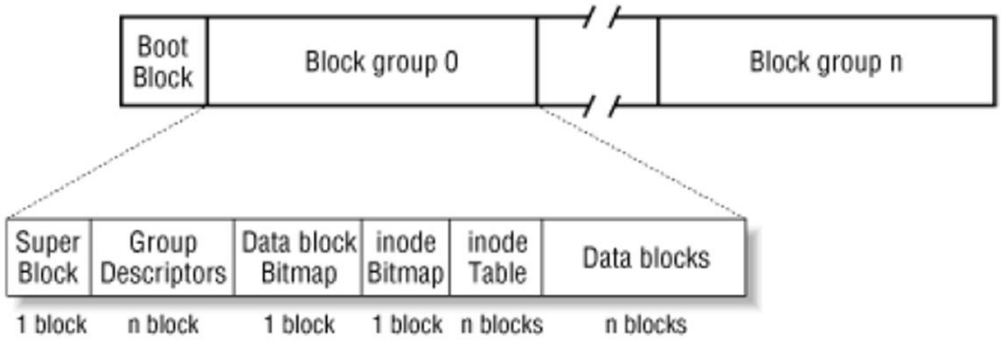
Implementação de Arquivos (4)

- **Grupos de blocos**

Cada partição é dividida em grupos de blocos de mesmo tamanho. Com um tamanho de bloco de 4 KB, um grupo contém 32768 blocos.

Cada grupo de blocos possui uma tabela de descritores que endereçam os mapas de bits dos blocos e dos i-nodes e a tabela de i-nodes. O primeiro grupo contém o superbloco, que possui cópia em alguns outros grupos. No mapa de bits de blocos, cada byte mapeia 8 blocos (um por bit). O bit menos significativo identifica o bloco de menor número. O mapa de bits de i-nodes é análogo.

Os grupos de blocos são representados pela seguinte estrutura:



Implementação de Arquivos (4)

- **Mapa de bits de blocos**

Em cada grupo de blocos, um bloco é alocado para indicar se os restantes estão ocupados ou livres. Quando o bloco possui tamanho 4 KB, é possível mostrar 32768 blocos (4096 * 8). Se está alocado, é marcado com o bit '1', e caso contrário, com o bit '0'.

- **Mapa de i-nodes**

De forma análoga ao mapa de bits de blocos, um bloco é utilizado para indicar a alocação dos i-nodes. Geralmente o bloco não é totalmente utilizado pois o número de i-nodes é menor que o número de blocos.

- **Tabela de i-nodes**

A tabela de i-nodes é formada por blocos consecutivos após os mapas de bits. Cada entrada na tabela é um i-node. Portanto, para um bloco de tamanho 4 KB, é possível conter até 32 i-nodes.

Implementação de Arquivos (4)

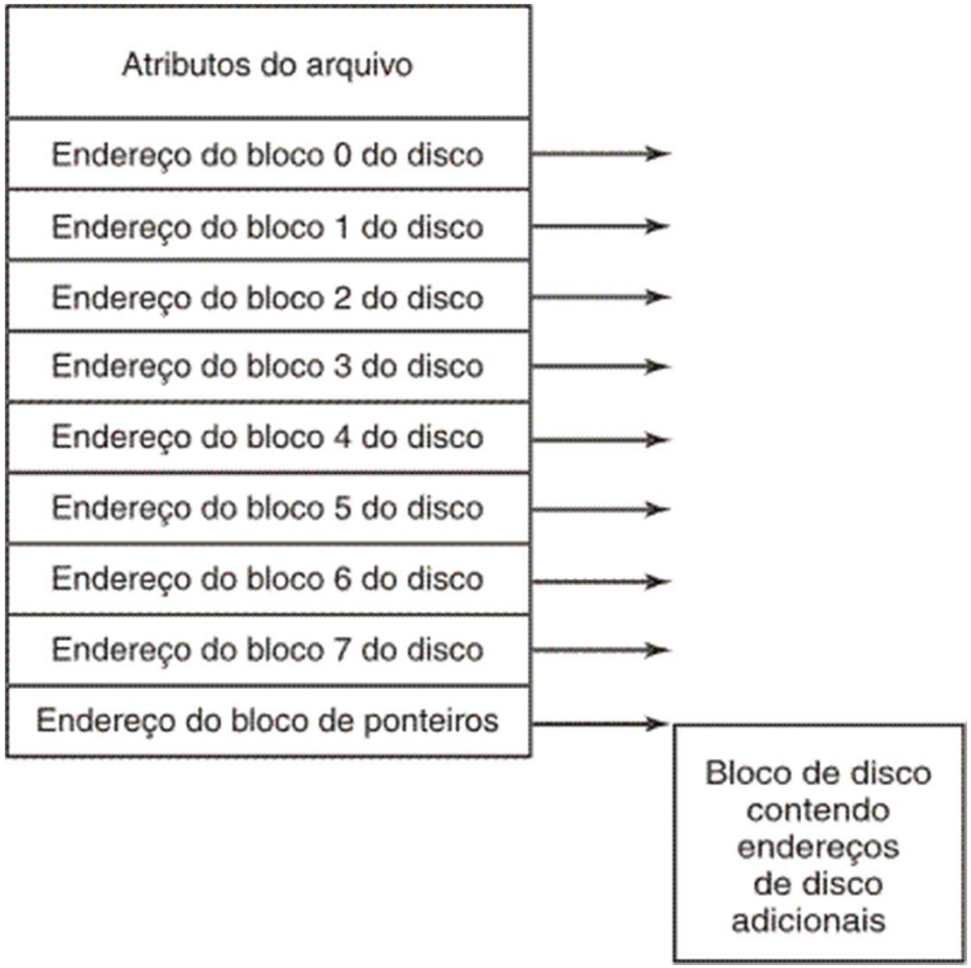
- **Alocação de blocos**

Quando é feita a operação de escrita em um arquivo, o Ext2 tenta sempre que possível alocar blocos de dados no mesmo grupo de blocos que contém o i-node. Este comportamento reduz o movimento da cabeça de leitura-gravação da unidade de disco.

Em um sistema de arquivos podemos encontrar dois tipos de fragmentação: A fragmentação interna, que ocorre quando o tamanho do arquivo não é múltiplo do tamanho do bloco, ocorrendo uma perda de espaço no último bloco; fragmentação externa, que ocorre quando o arquivo possui blocos alocados não contiguamente, prejudicando seu desempenho.

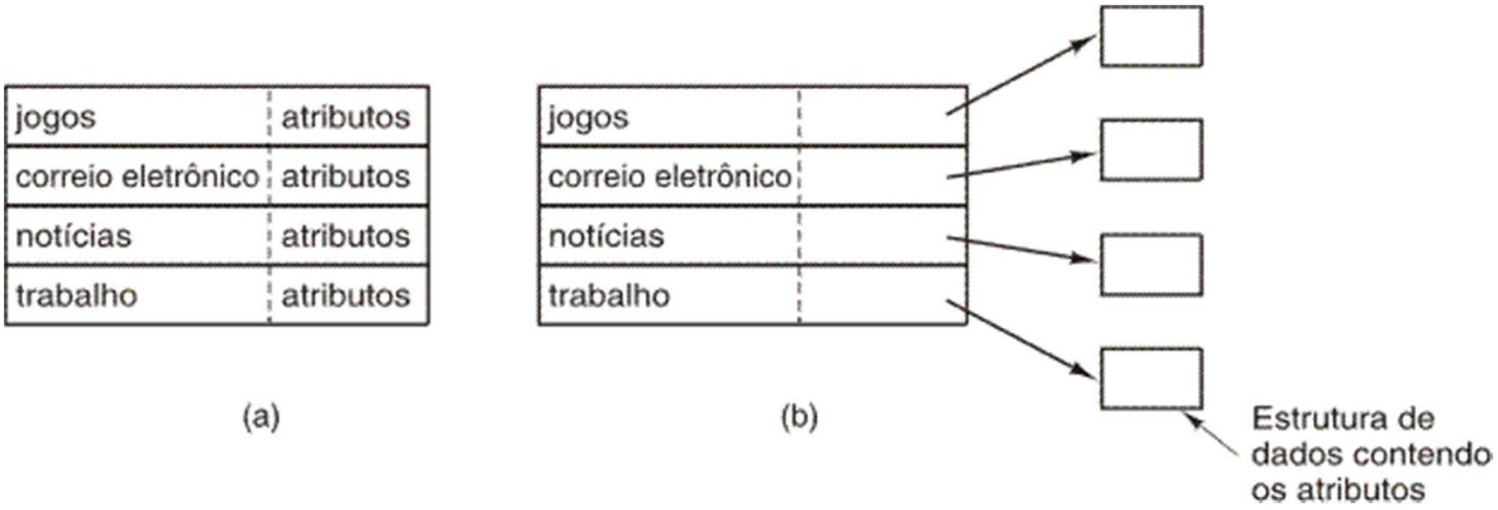
Para a fragmentação interna, a solução é diminuir o tamanho padrão do bloco.

Implementação de Arquivos (4)



Um exemplo de i-node

Implementação de Diretórios (1)



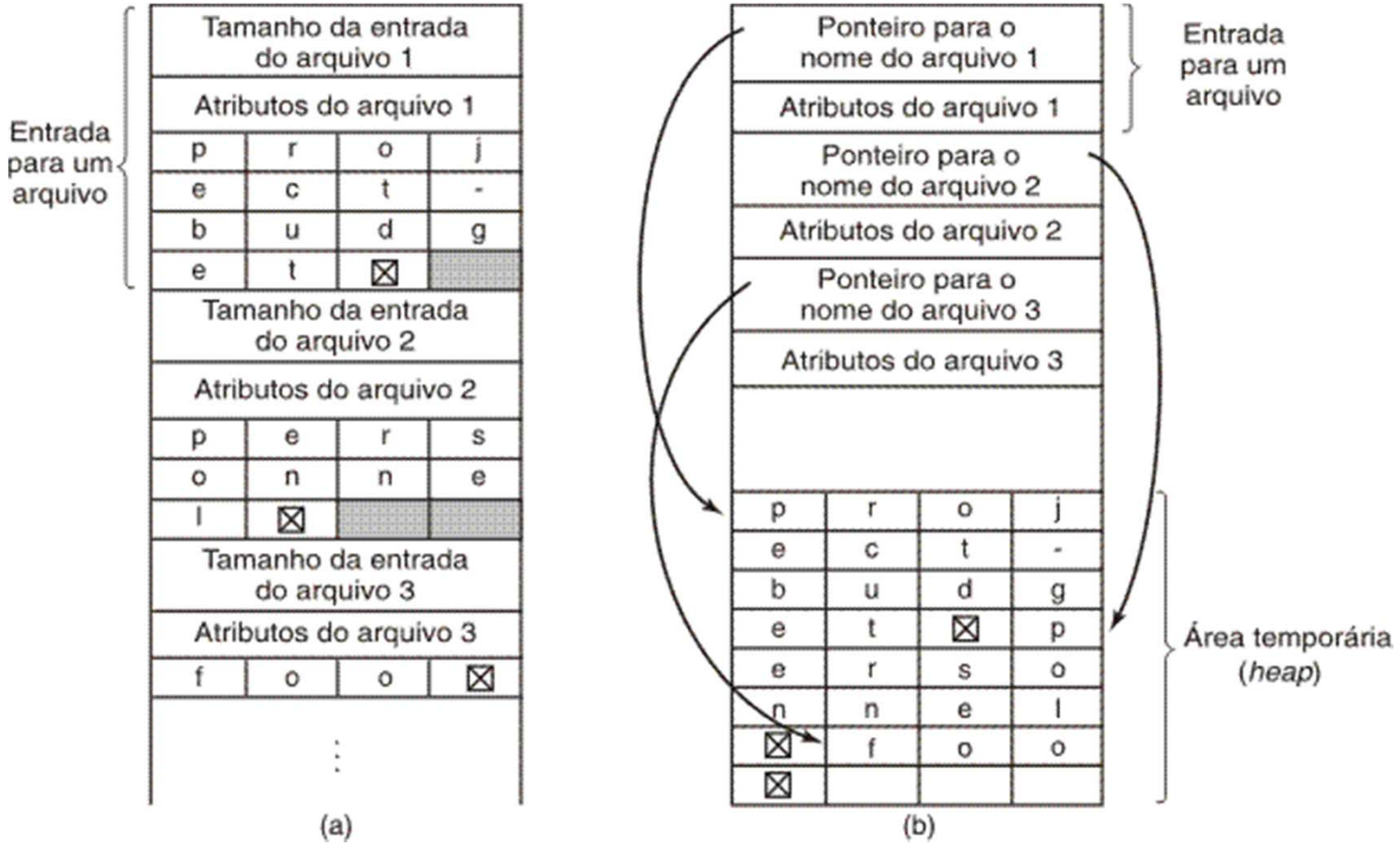
(a) Um diretório simples

entradas de tamanho fixo

endereços de disco e atributos na entrada de diretório

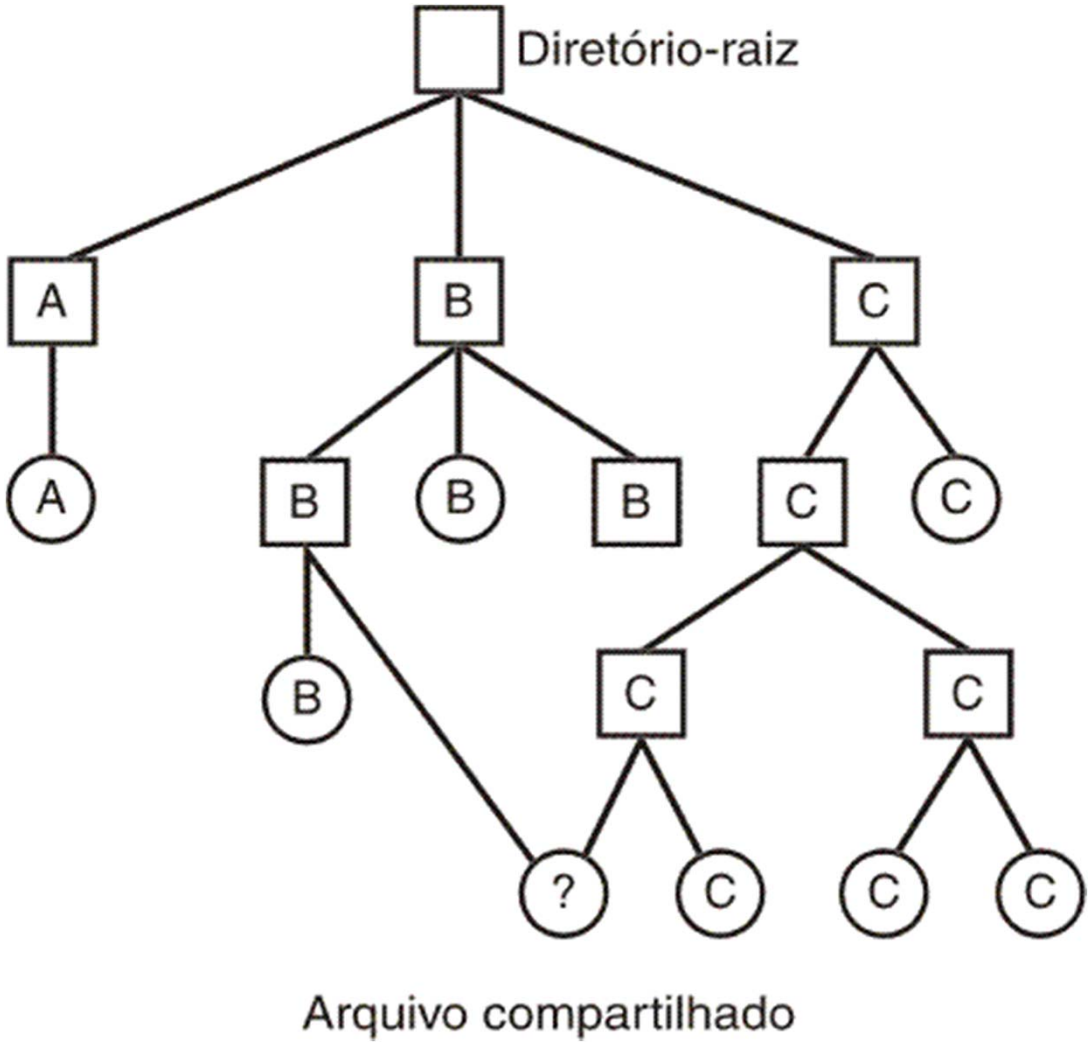
(b) Diretório no qual cada entrada se refere apenas a um i-node

Implementação de Diretórios (2)



- Duas formas de tratar nomes longos de arquivos em um diretório
  - (a) Em linha
  - (b) Em uma área temporária *(heap)*

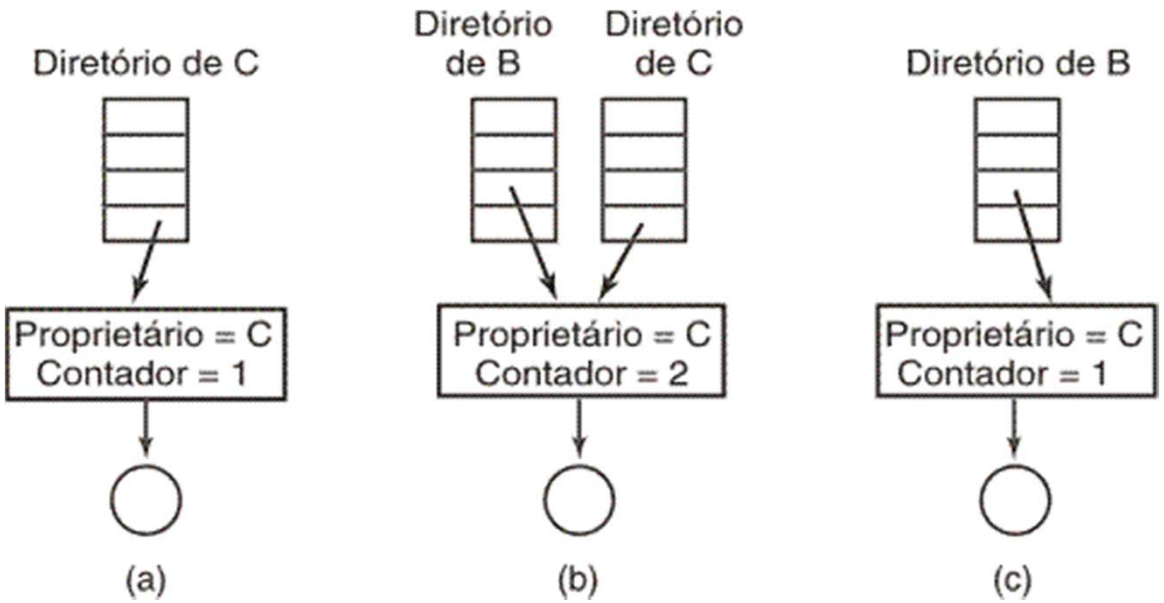
Arquivos Compartilhados (1)



Arquivo compartilhado

Sistema de arquivo contendo um arquivo compartilhado

Arquivos Compartilhados (2)

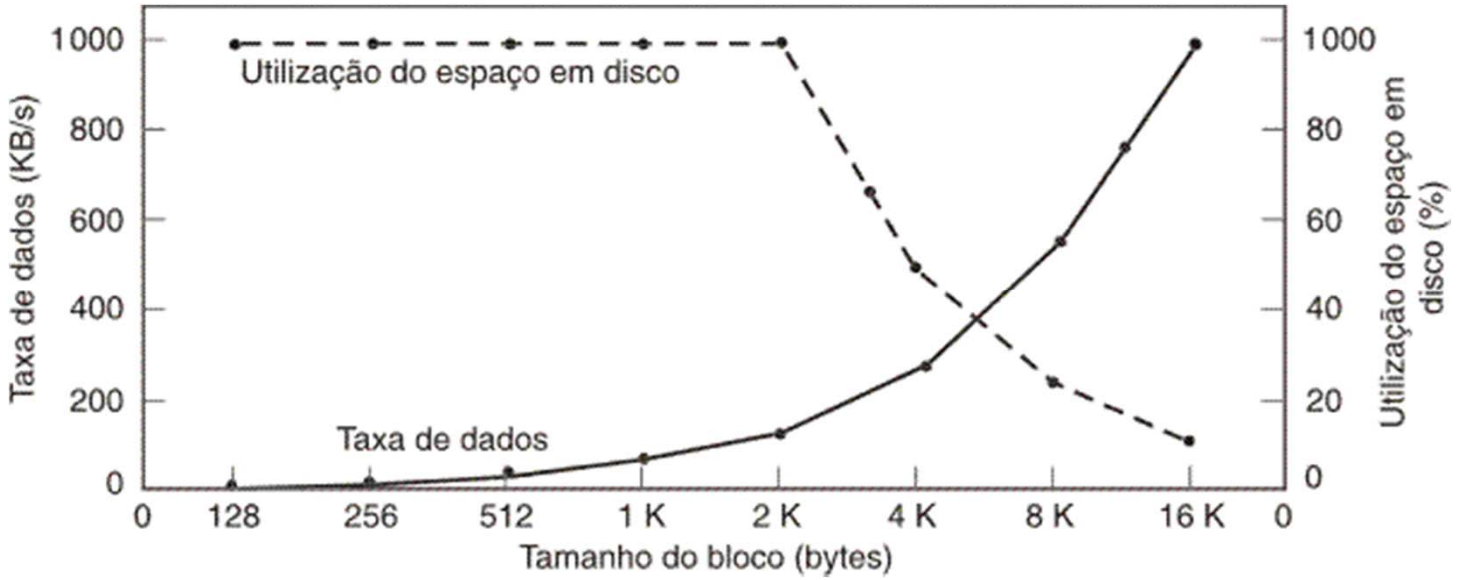


(a) Situação antes da ligação

(b) Depois de a ligação ser criada

(c) Depois de o proprietário original remover o arquivo

Gerenciamento do Espaço em Disco (1)



- A curva contínua (escala no lado esquerdo) mostra a taxa de dados de um disco
- A linha tracejada (escala no lado direito) mostra a eficiência de ocupação do disco
- Todos os arquivos são de 2KB