UFFS
Ciência da Computação

Sistemas Operacionais

Prof. Marco Aurélio Spohn

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

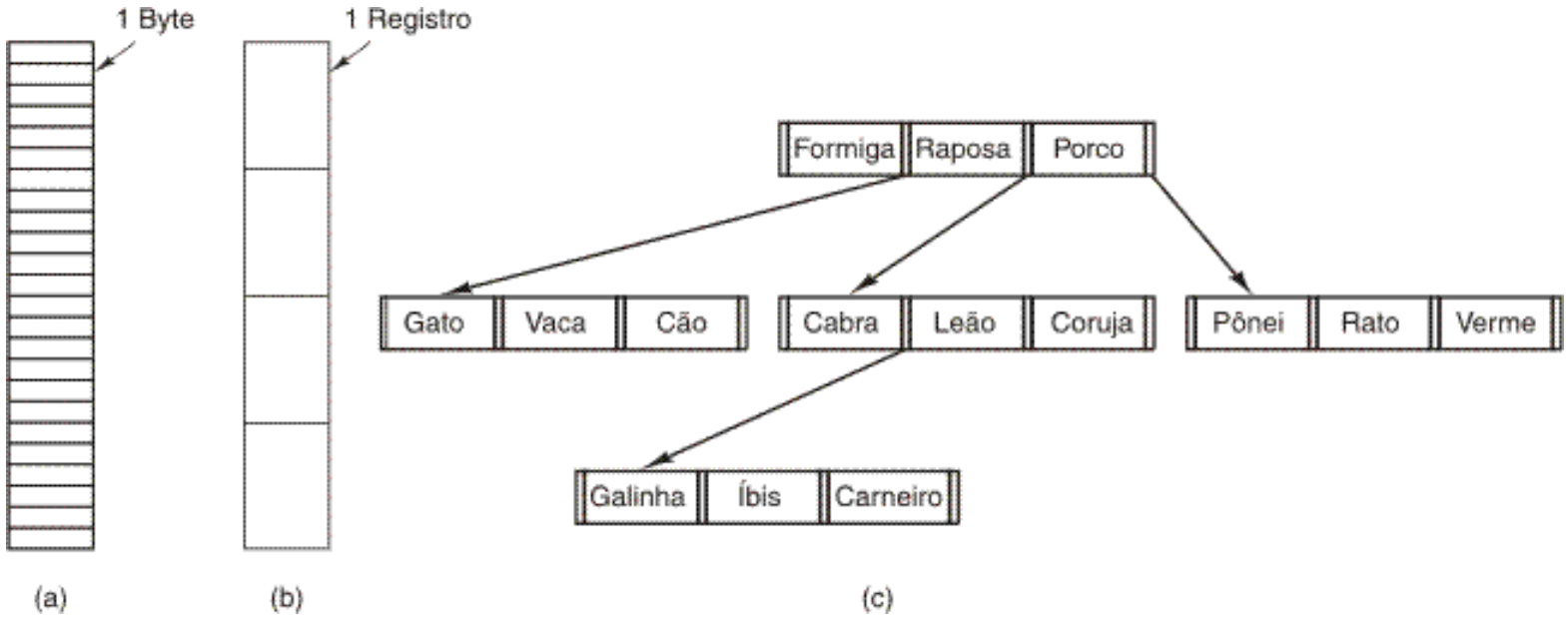
1. Deve ser possível armazenar uma quantidade muito grande de informação
2. A informação deve sobreviver ao término do processo que a usa
3. Múltiplos processos devem ser capazes de acessar a informação concorrentemente

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

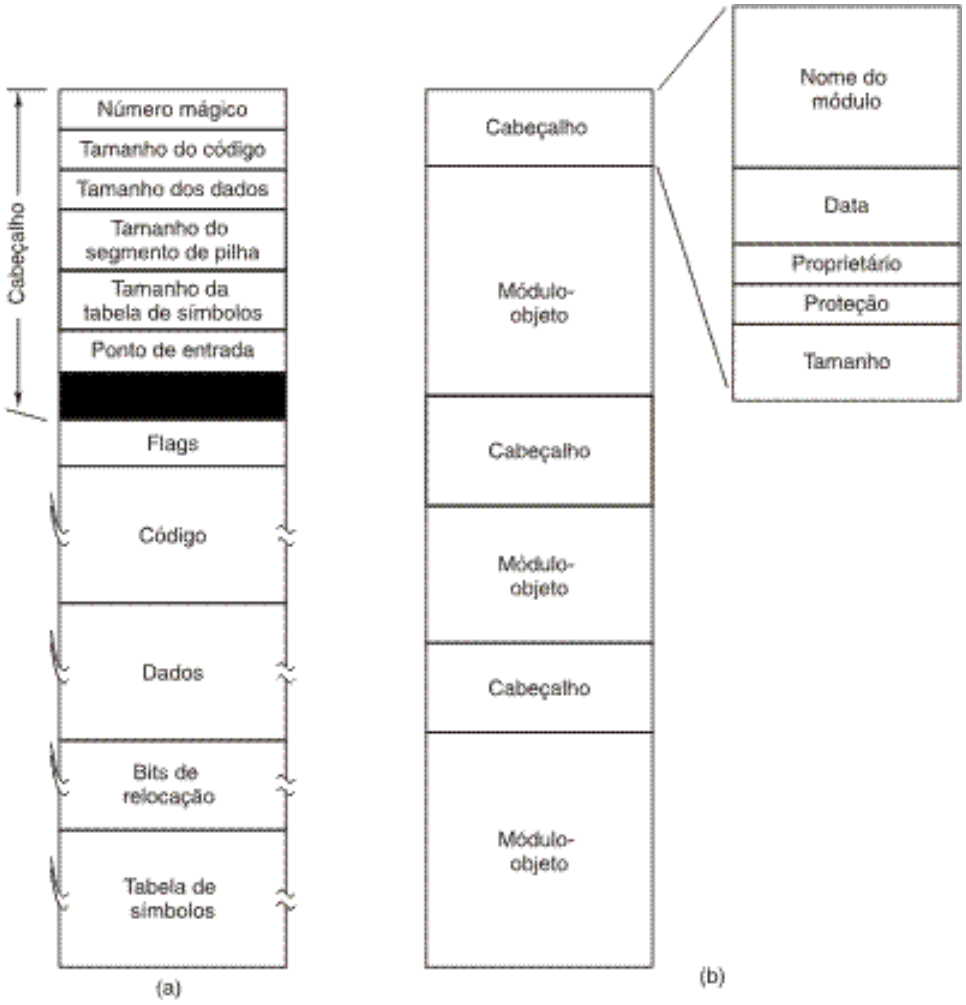
- Extensões típicas de arquivos

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

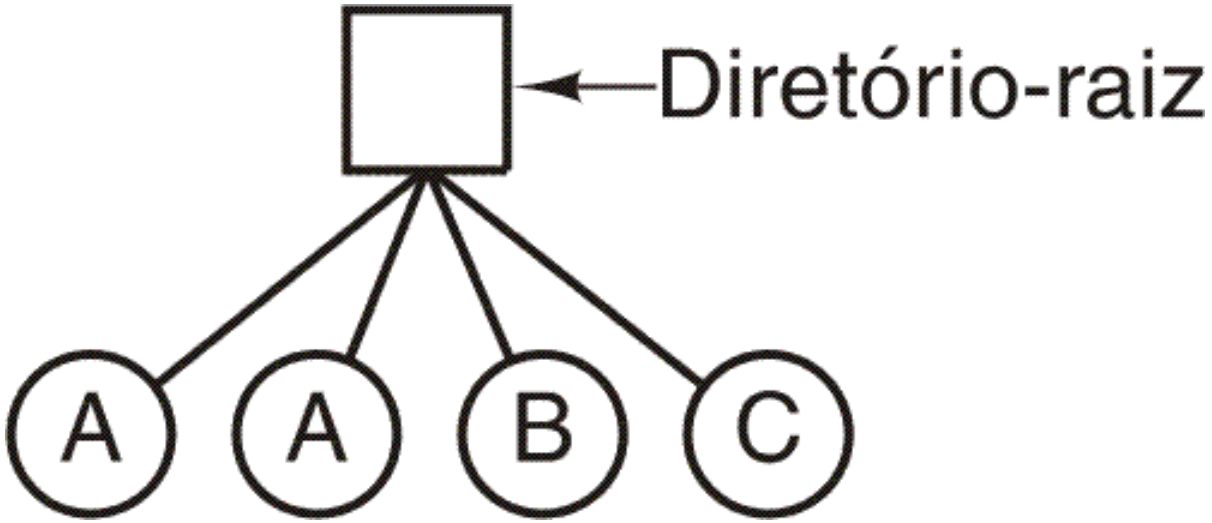
1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write

1. Append
2. Seek
3. Get attributes
4. Set Attributes
5. Rename

Exemplo de um Programa com Chamadas ao Sistema para Arquivos

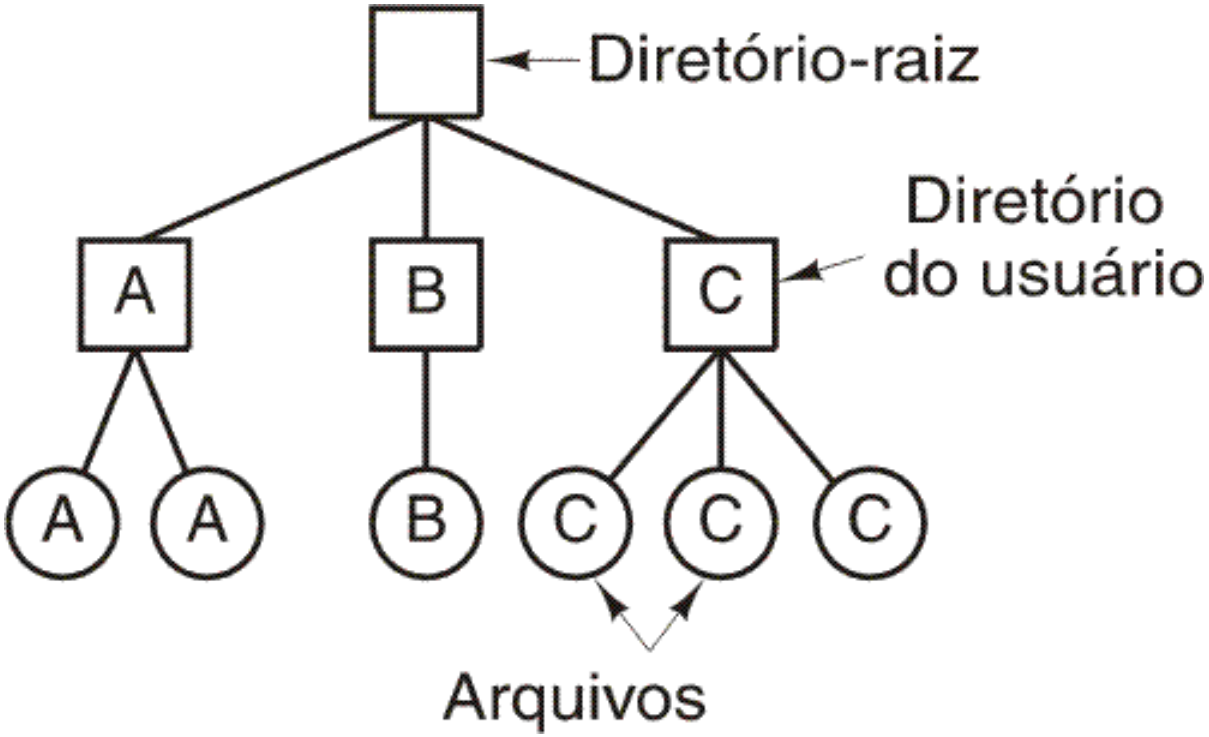
```
/* Programa que copia arquivos. Verificação e relato de erros é mínimo.*/

#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 */
}
```

Diretórios
Sistemas de Diretório em Nível Único



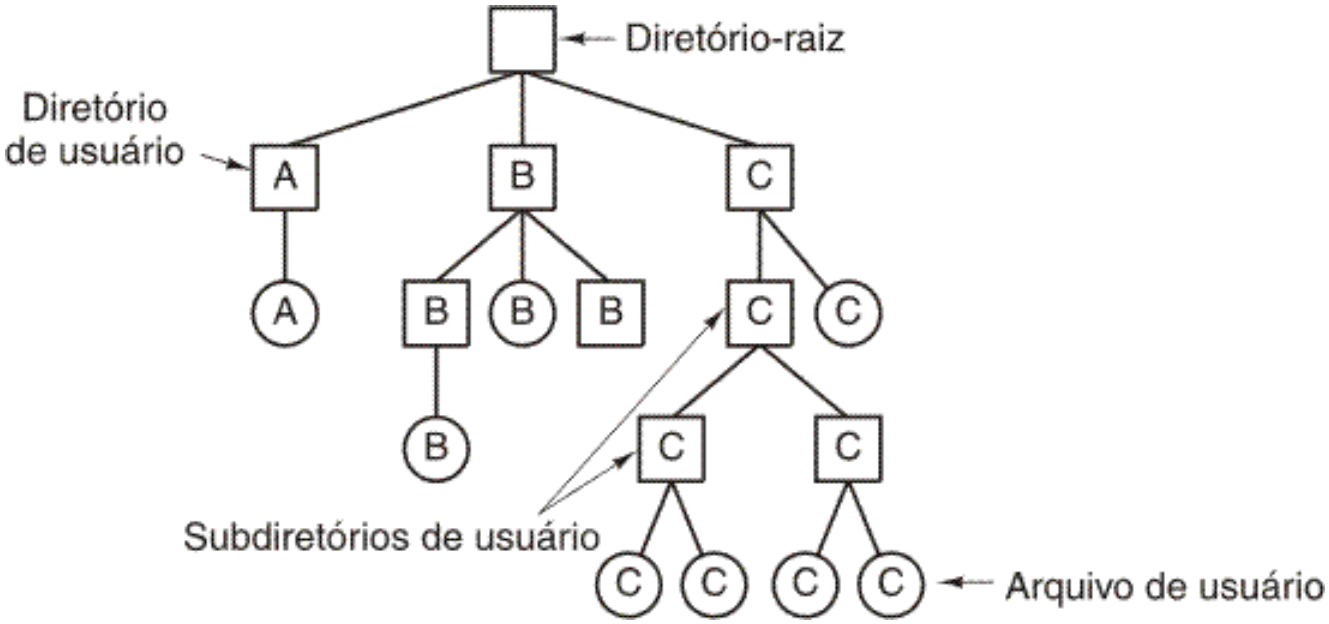
- Um sistema de diretório de nível único
  - contém 4 arquivos
  - propriedades de 3 pessoas diferentes, A, B, e C

Sistemas de Diretórios em Dois Níveis



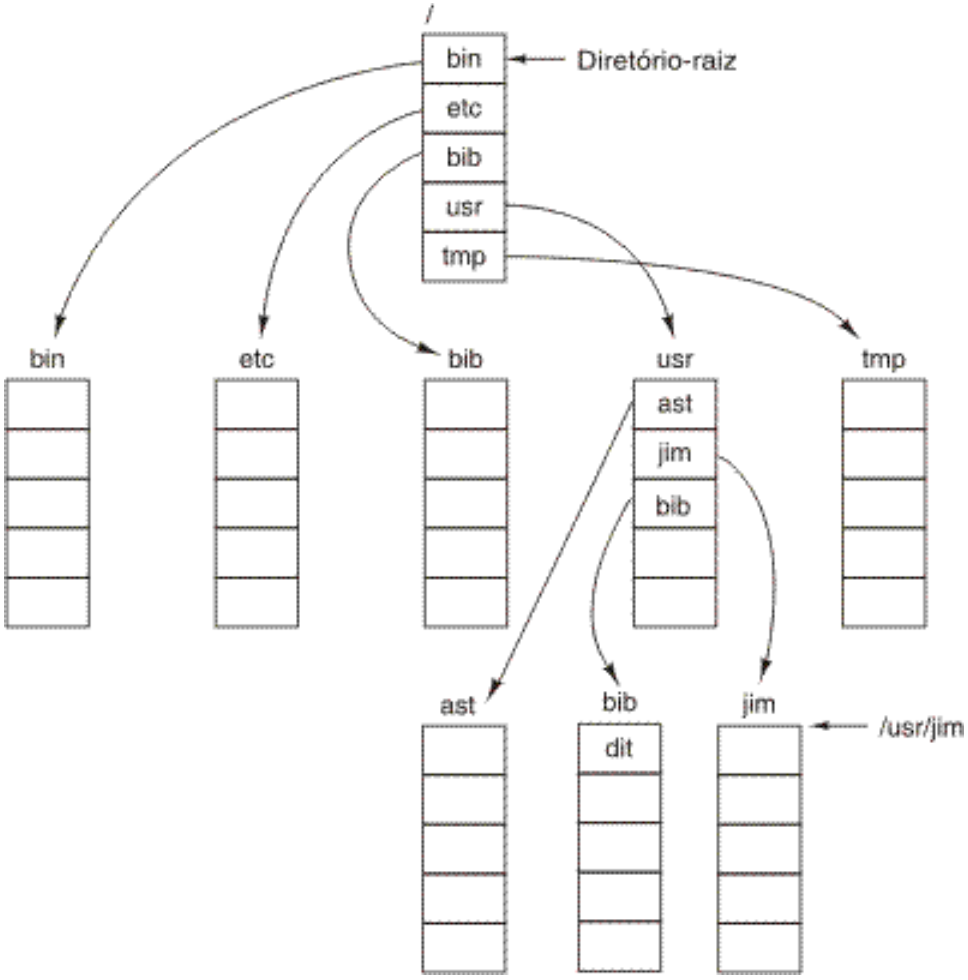
- As letras indicam os donos dos diretórios e arquivos

Sistemas de Diretórios Hierárquicos



- Um sistema de diretório hierárquico

Nomes de Caminhos



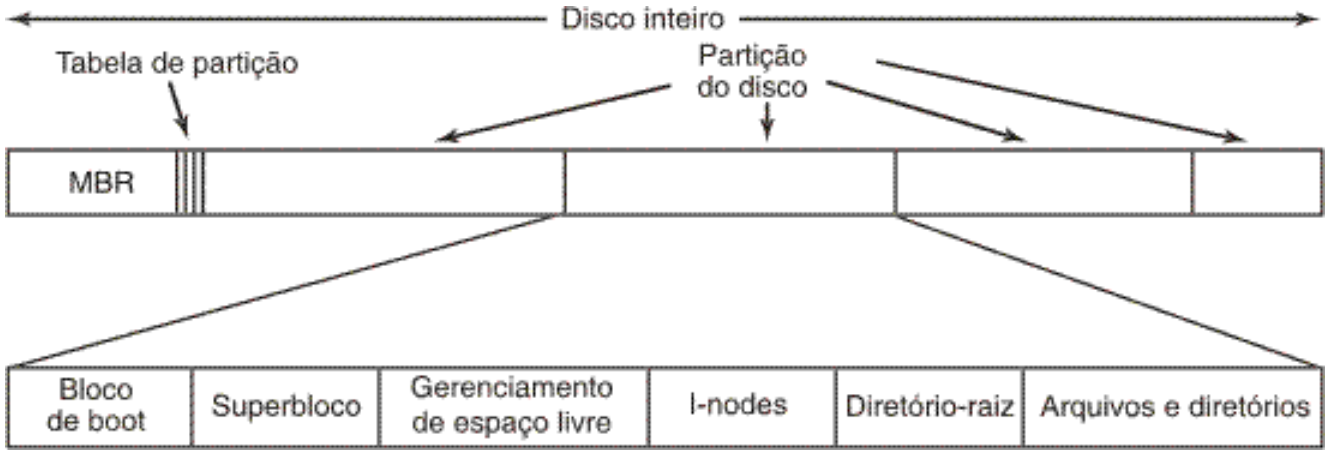
- Uma árvore de diretórios UNIX

Operações com Diretórios

1. Create
2. Delete
3. Opendir
4. Closedir

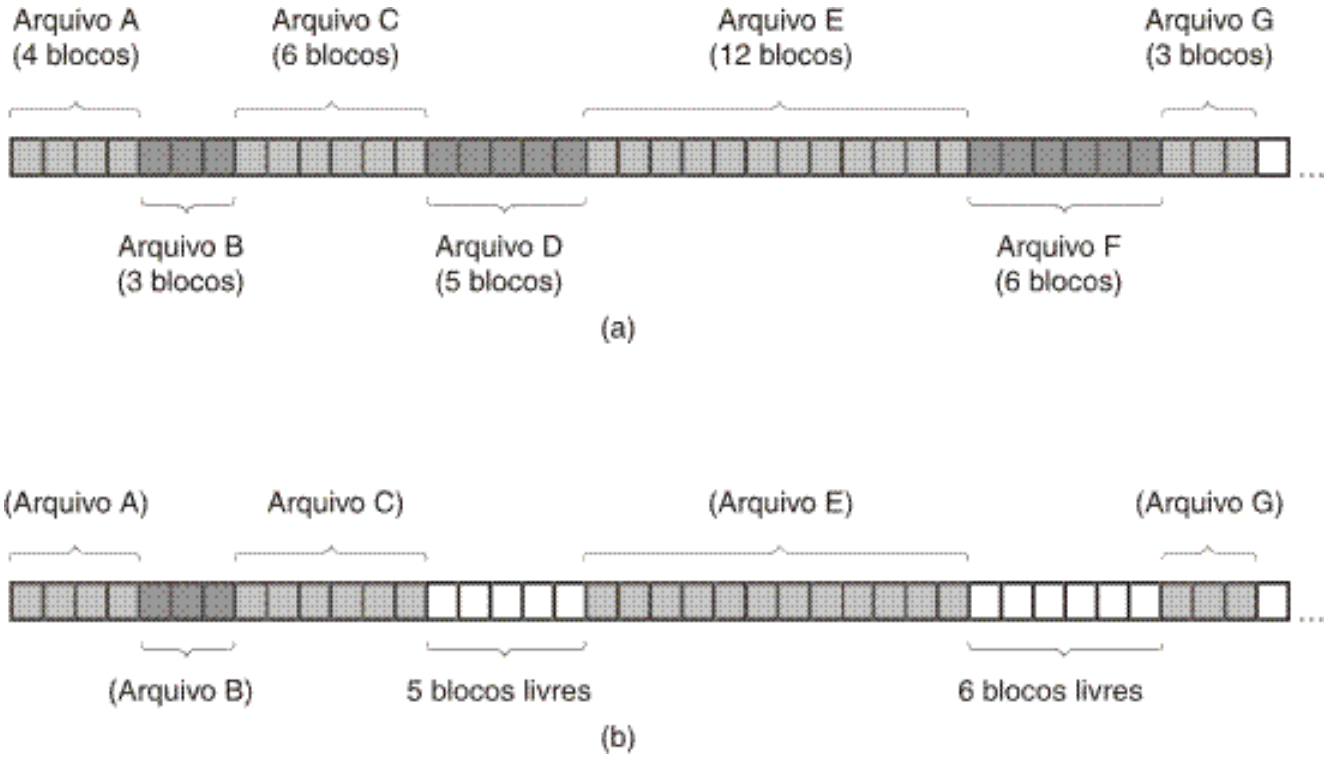
1. Read dir
2. Rename
3. Link
4. Unlink

Implementação do Sistema de Arquivos



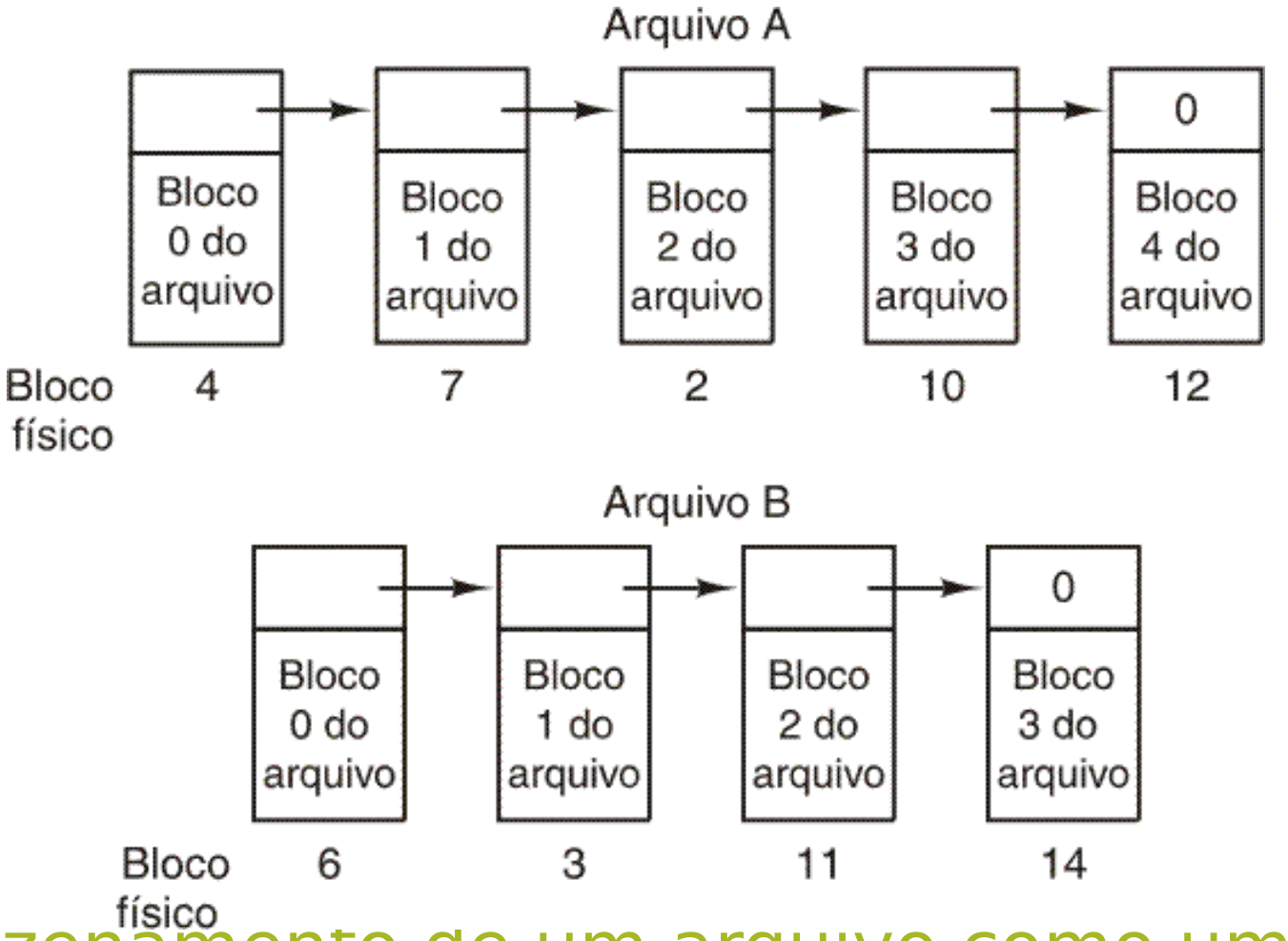
- Um possível *layout* de sistema de arquivo (superbloco contém informações sobre o sistema de arquivos: número mágico para identificação sist. arq., número de blocos, etc).

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)

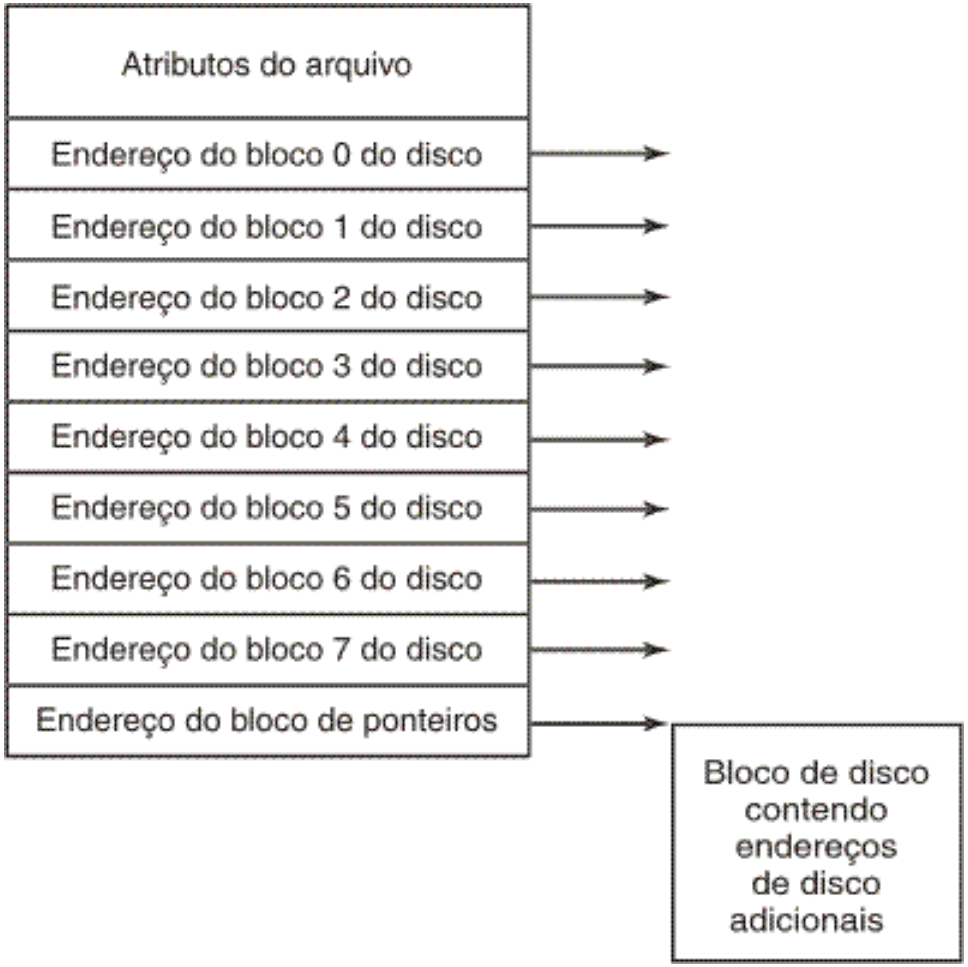


- Armazenamento de um arquivo como uma lista encadeada de blocos de disco (*i.e.*, encadeamento nos próprios blocos)

Implementação de Arquivos (3)

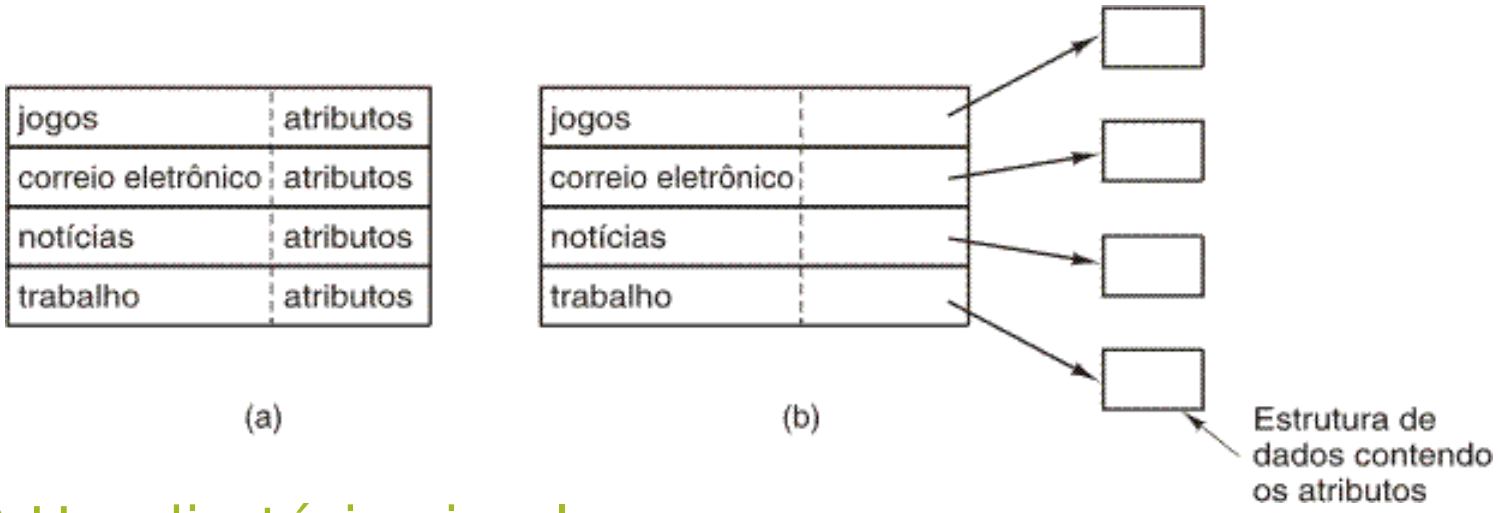
- Alocação por lista encadeada usando uma tabela de alocação de arquivos (mesmo exemplo anterior): deve ser mantido na memória (problema=espaço)

Implementação de Arquivos (4)



- Um exemplo de *i-node*
- vantagem=somente entradas correspondentes aos arquivos abertos precisam ser mantidas na memória

Implementação de Diretórios (1)



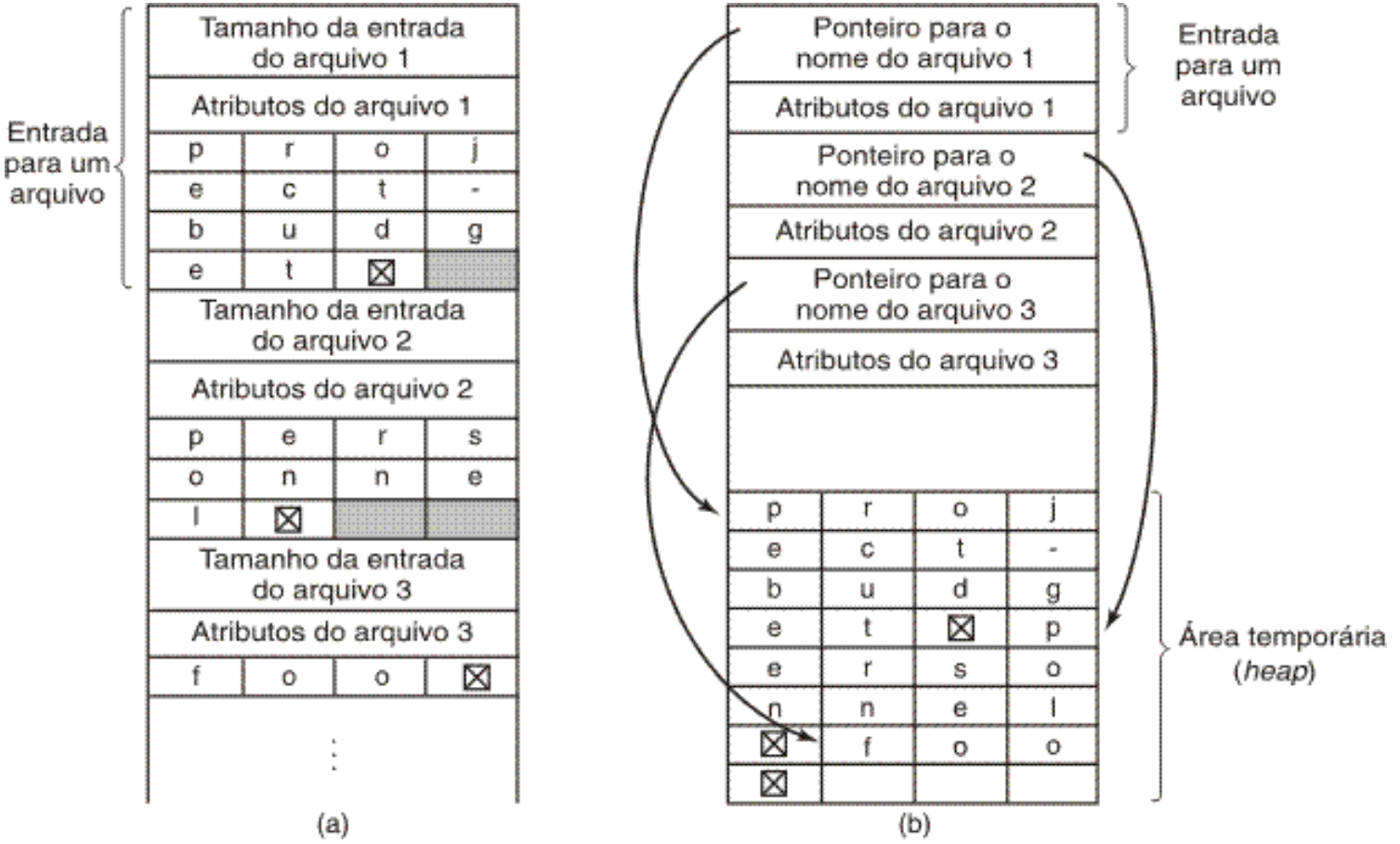
- (a) Um diretório simples

entradas de tamanho fixo

endereços de disco e atributos na entrada de diretório

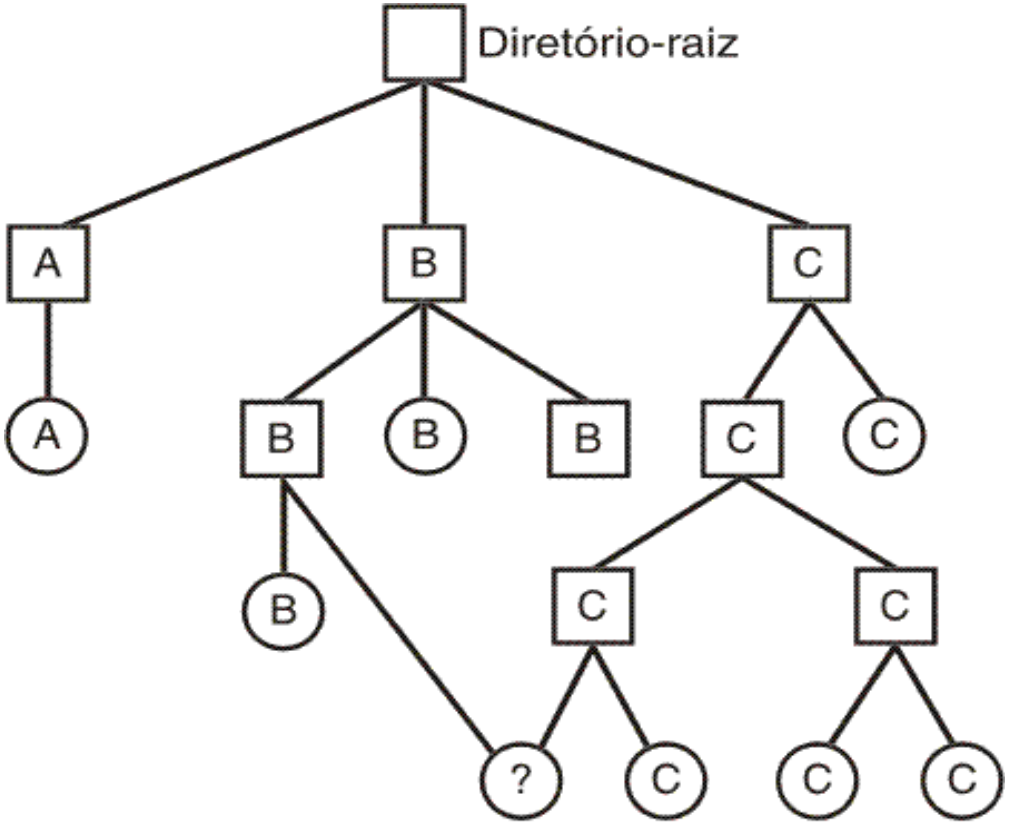
- (b) Diretório no qual cada entrada se refere apenas a um i-node

Implementação de Diretórios (2)



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

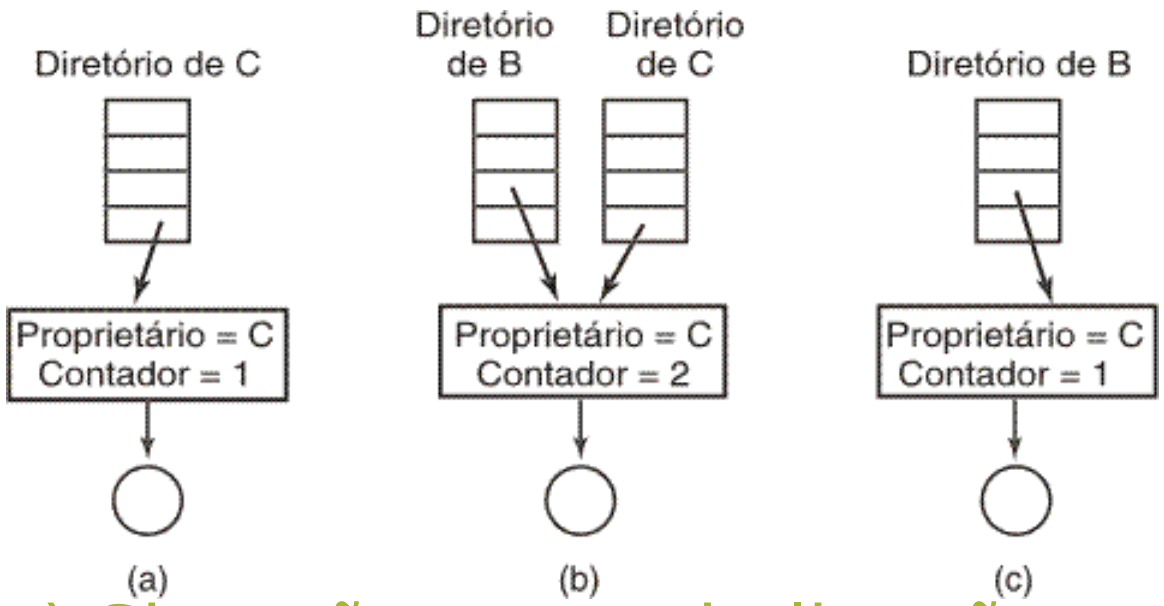
Arquivos Compartilhados (1)



Arquivo compartilhado

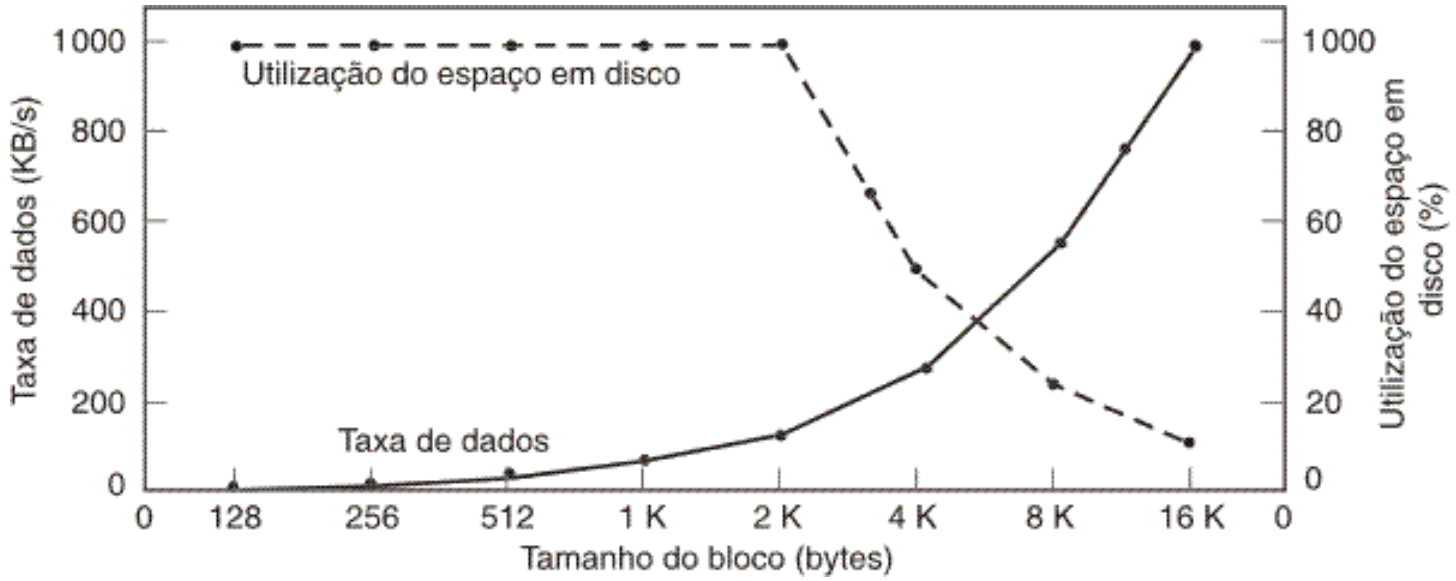
- **Sistema de arquivo contendo um arquivo compartilhado (*soft link*: não é um *link* para o arquivo propriamente dito, mas sim para um nome de arquivo; *hard link*: *link* para um arquivo de fato; com *hard links*, referências são atualizadas caso o arquivo alvo mude de localização; com *soft links*, referências podem ficar inválidas com mudanças.)**

Arquivos Compartilhados (2)



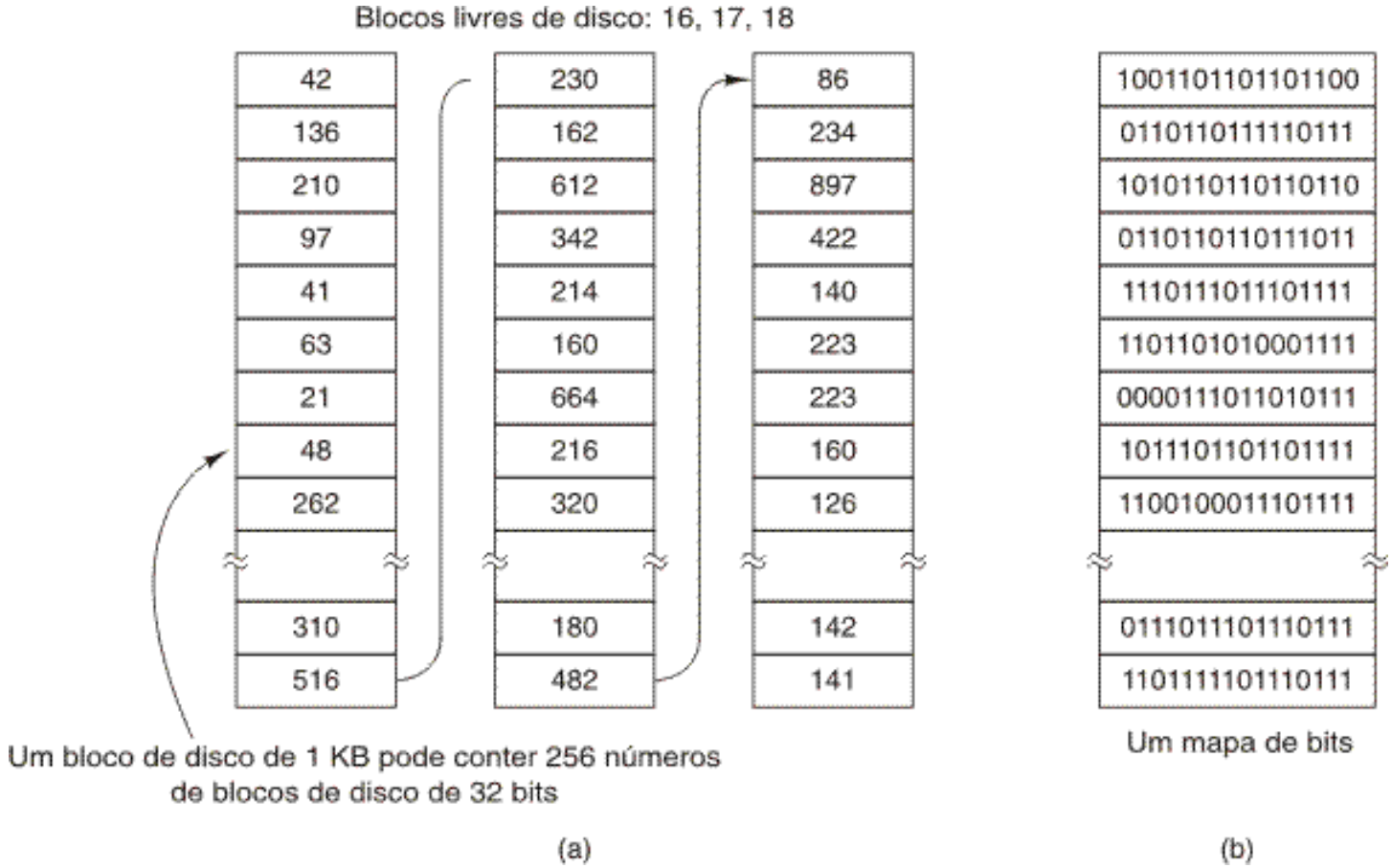
- (a) Situação antes da ligação
- (b) Depois de a ligação ser criada
- (c) Depois de o proprietário original remover o arquivo: como ainda há uma ligação, arquivo permanece no disco (e contabilizado na quota do usuário proprietário :(

Gerenciamento do Espaço em Disco (1)



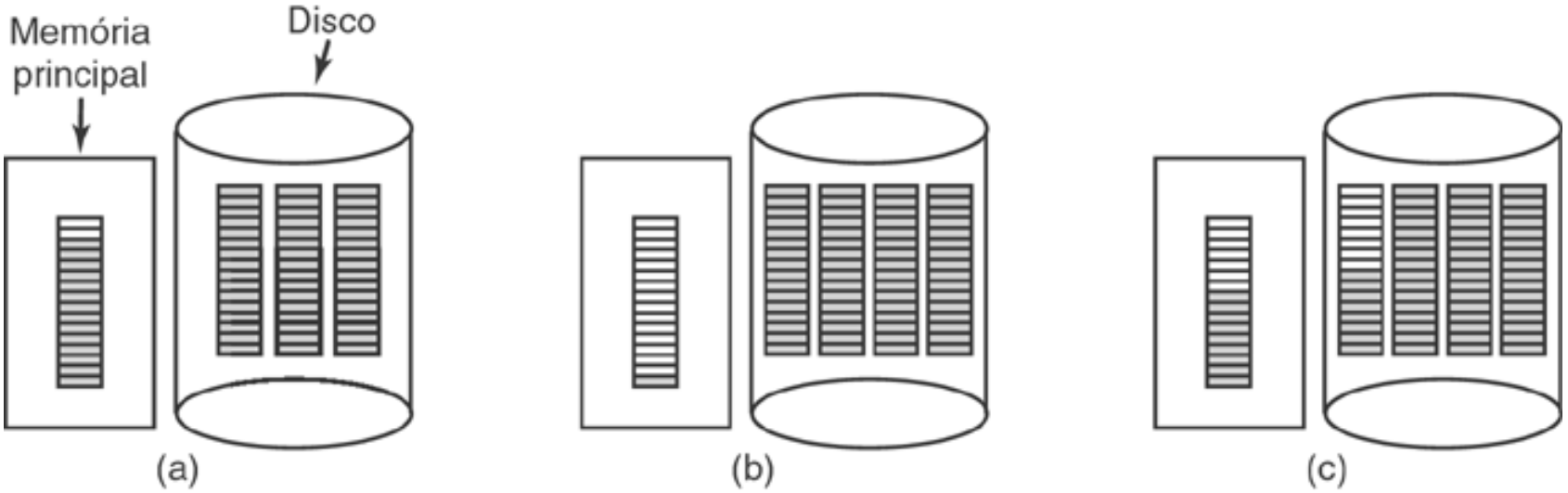
- A curva contínua (escala no lado esquerdo) mostra a taxa de dados de um disco
- A linha tracejada (escala no lado direito) mostra a eficiência de ocupação do disco
- **Todos os arquivos são de 2KB**

Gerenciamento do Espaço em Disco (2)



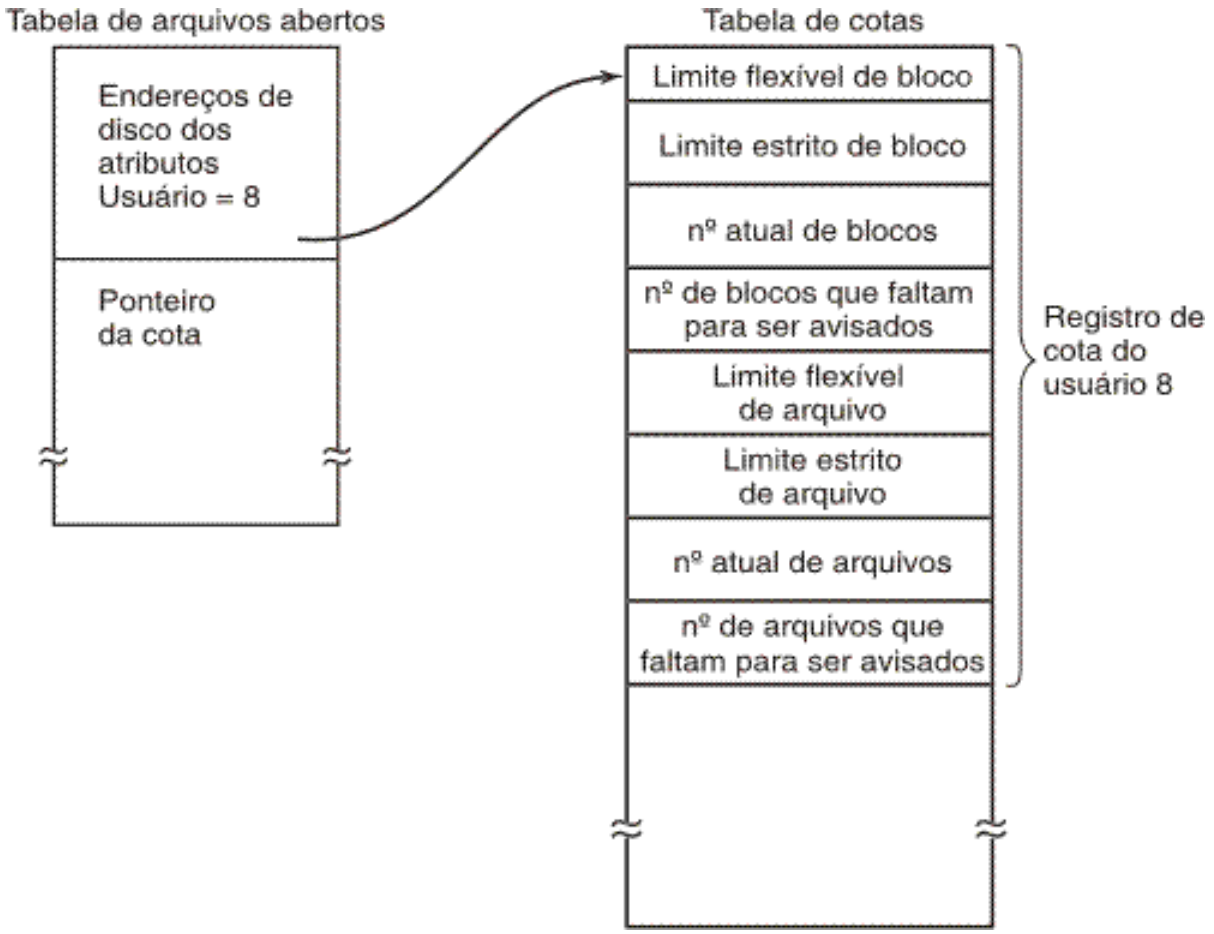
a)Armazenamento da lista de blocos livres em uma lista encadeada
b)Um mapa de bits

Gerenciamento do Espaço em Disco (3)



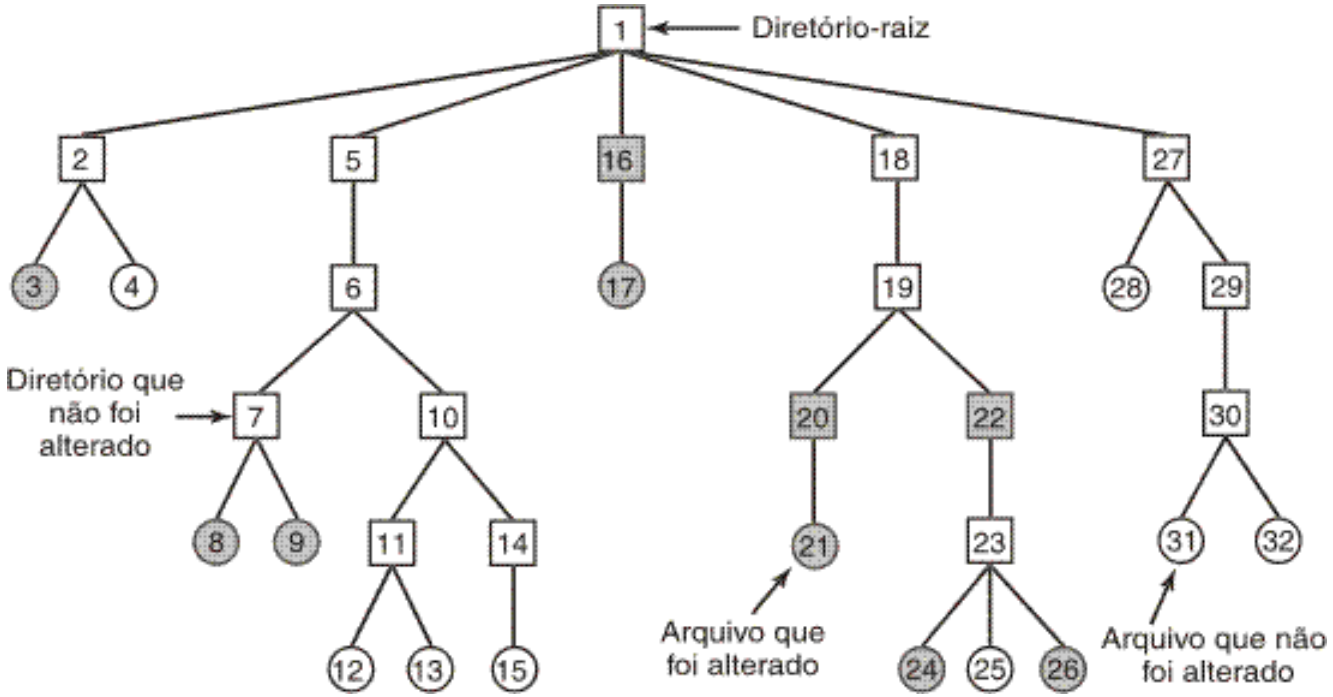
- (a) Um bloco em RAM quase cheio de ponteiros para blocos livres de disco: com três blocos de ponteiros em disco
- (b) Resultado da liberação de um arquivo de três blocos: um novo bloco de livres é escrito no disco: criação/remoção de arquivos temporários pequenos resulta em muitas escritas/leituras de blocos de livres em disco
- (c) Estratégia alternativa para tratar 3 blocos livres: manter um bloco de livres semi-cheio em RAM e outro em disco.
  - entradas sombreadas são ponteiros para blocos livres de disco

Gerenciamento do Espaço em Disco (4)



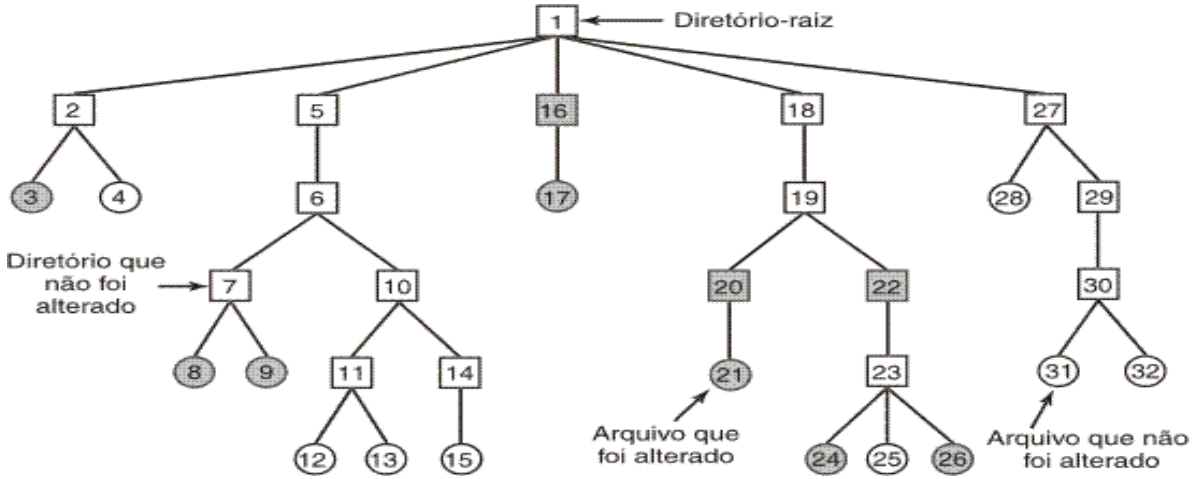
- Cotas para controlar o uso do disco por usuário

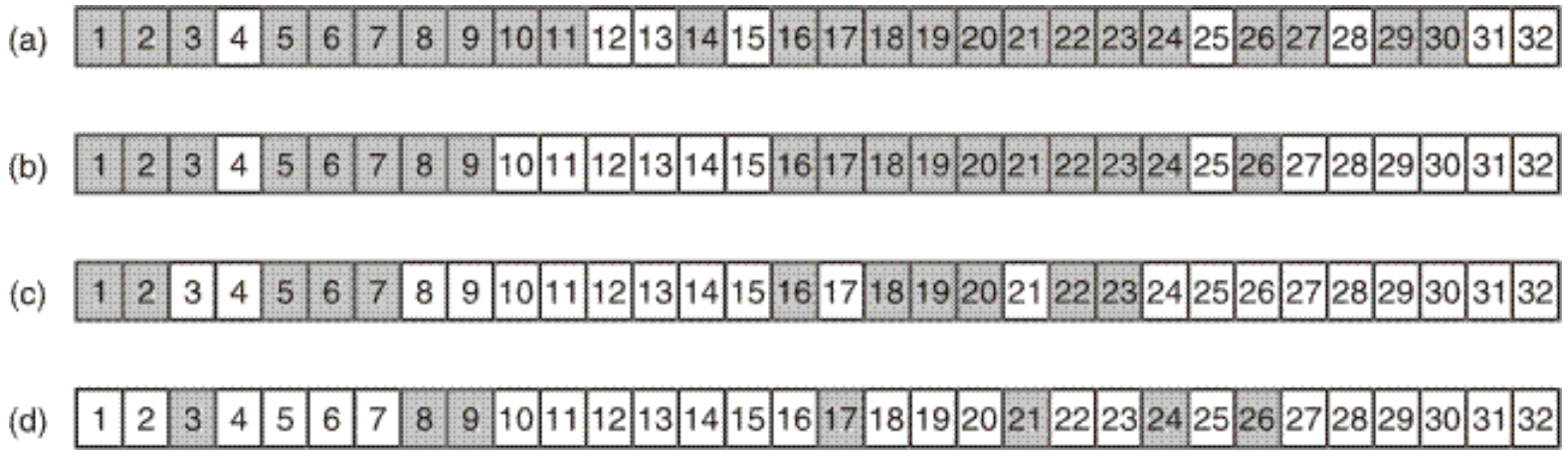
Confiabilidade do Sistema de Arquivos (cópia de segurança) (1)



- Um sistema de arquivos a ser copiado
  - Os quadrados são diretórios e os círculos são arquivos
  - Os itens sombreados foram modificados desde a última cópia
  - Cada diretório e arquivo rotulado por seu número de i-node

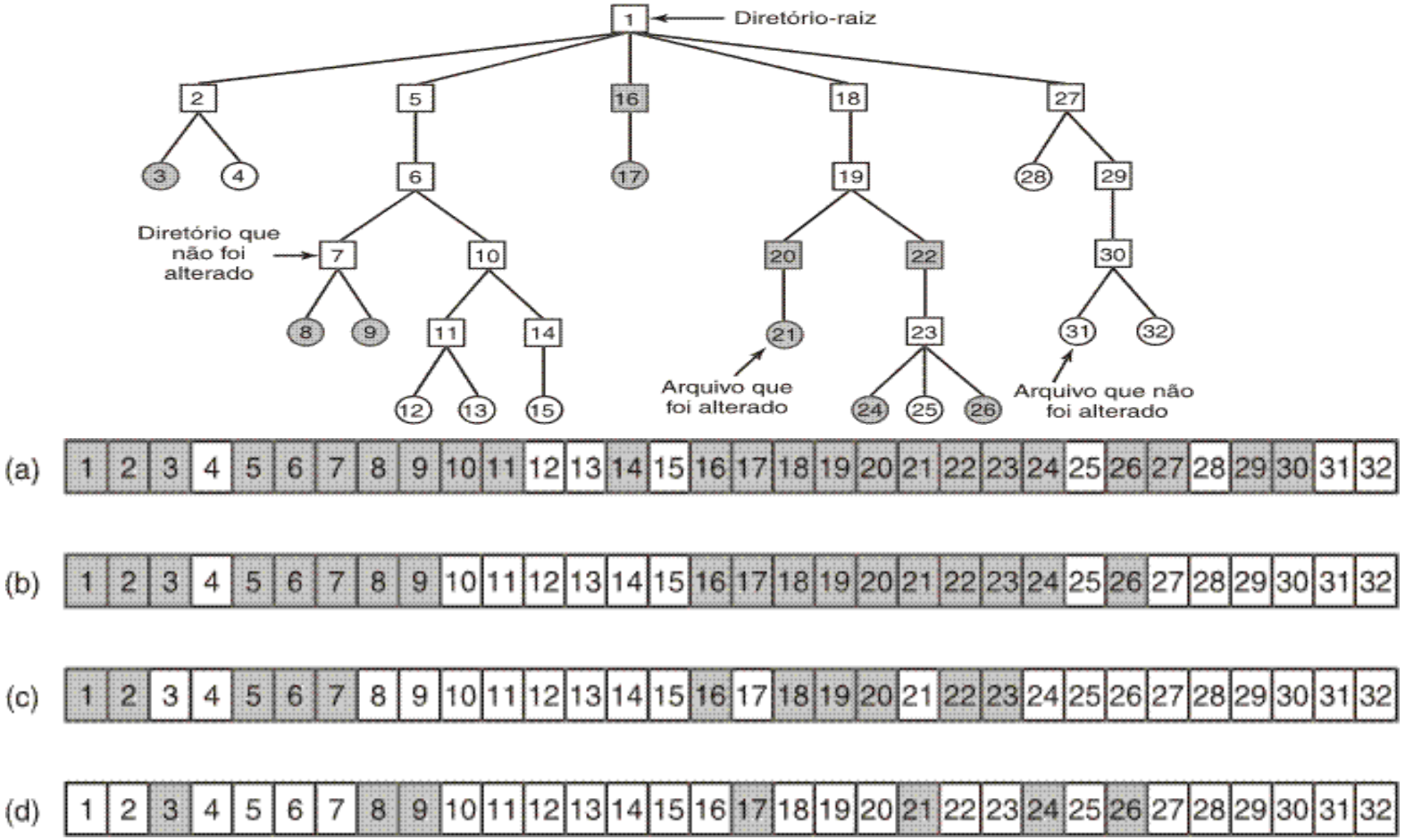
Confiabilidade do Sistema de Arquivos (2): Mapas de bits usados pelo algoritmo de cópia lógica (ou *dump* lógico)





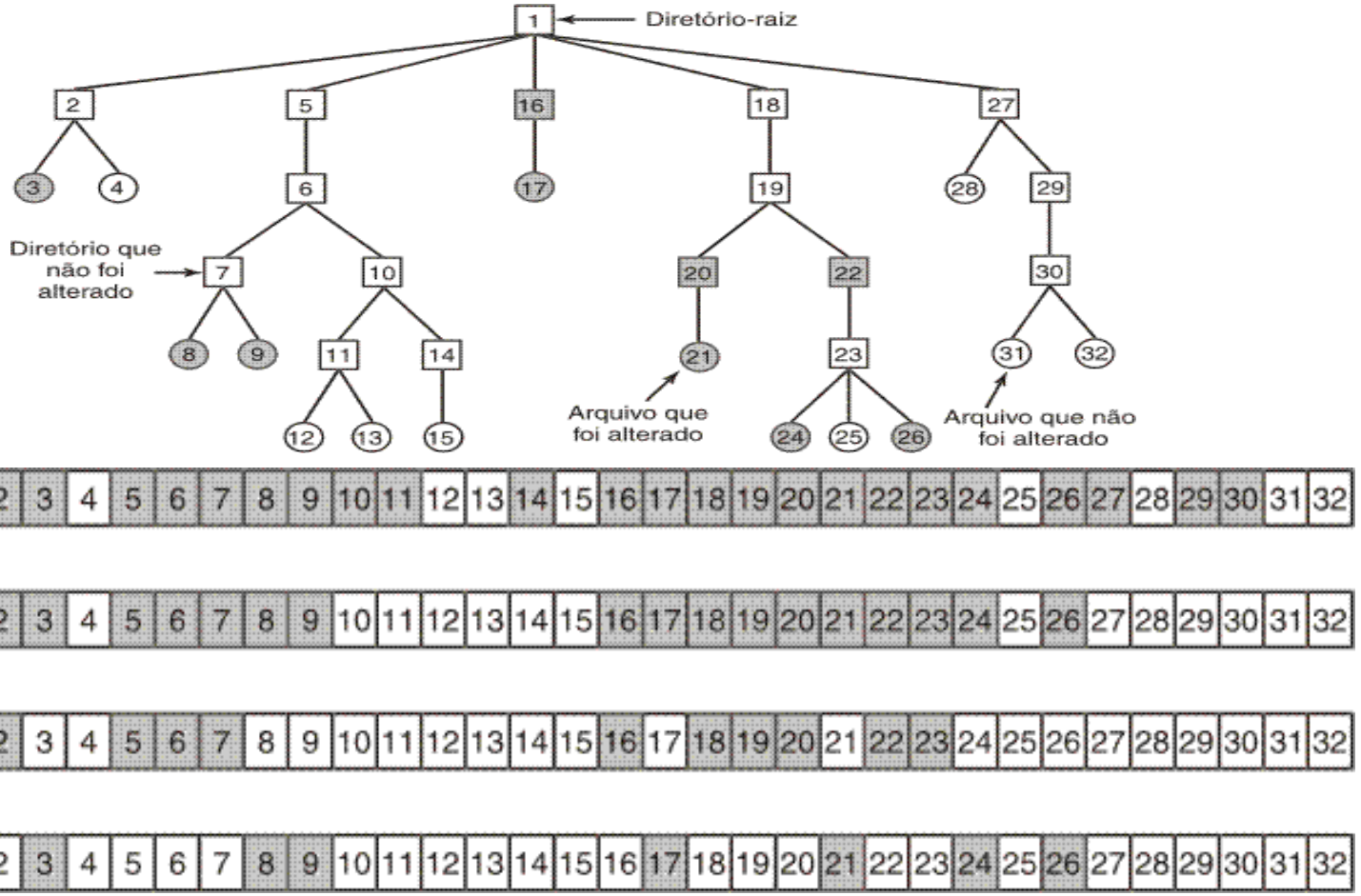
- (a) Fase 1: para cada arquivo modificado, seu i-node é marcado; **diretórios, modificados ou não, são marcados também**!!!

Confiabilidade do Sistema de Arquivos (2): Mapas de bits usados pelo algoritmo de cópia lógica (ou *dump* lógico)



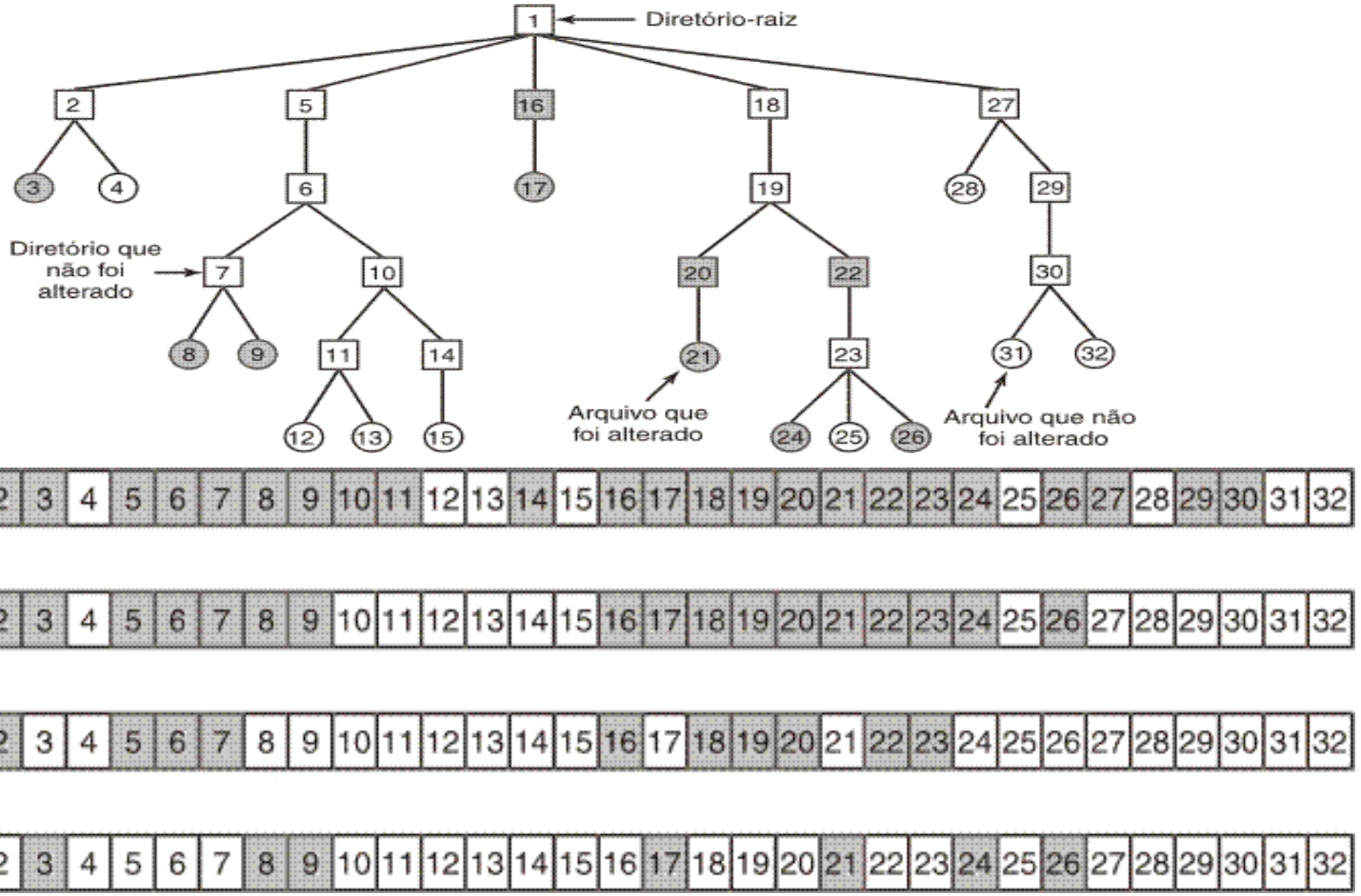
(b) Fase 2: percorre recursivamente a árvore, desmarcando os diretórios que não tenham arquivos ou diretórios modificados dentro dele ou sob ele.

Confiabilidade do Sistema de Arquivos (2): Mapas de bits usados pelo algoritmo de cópia lógica (ou *dump* lógico)



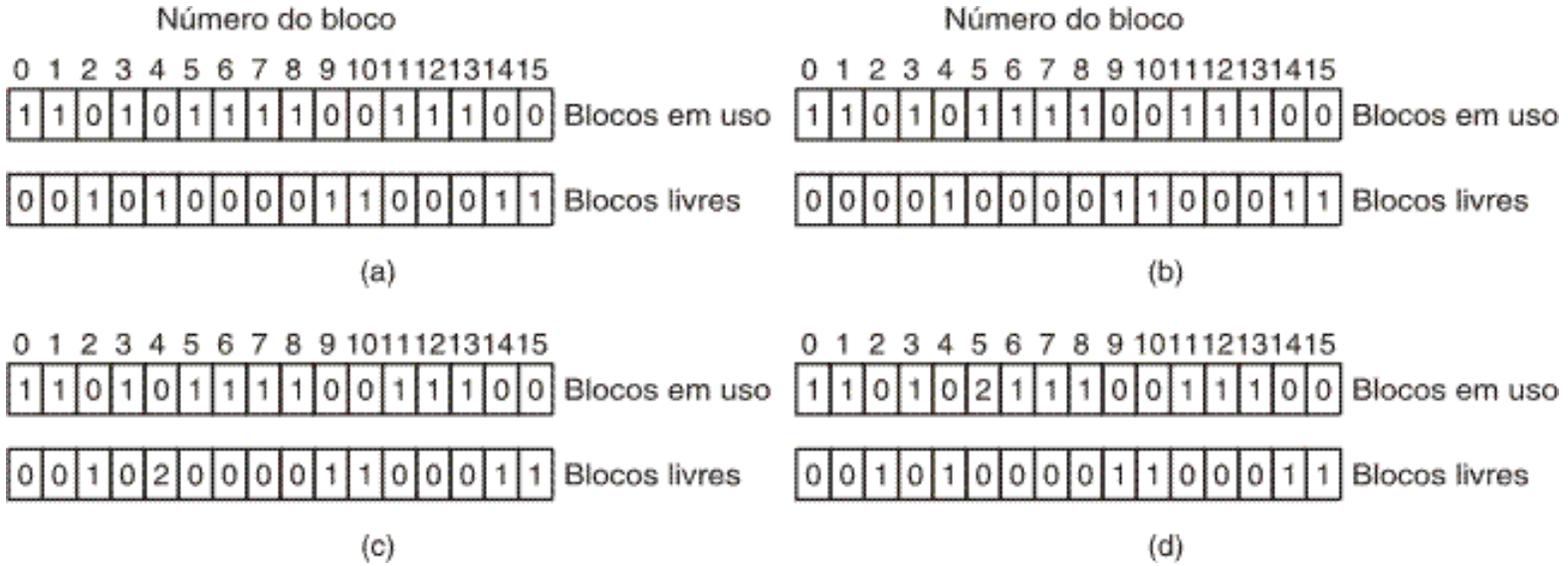
- (c) Fase 3: copia primeiro os diretórios marcados (figura destaca apenas os diretórios marcados).

Confiabilidade do Sistema de Arquivos (2): Mapas de bits usados pelo algoritmo de cópia lógica (ou *dump* lógico)



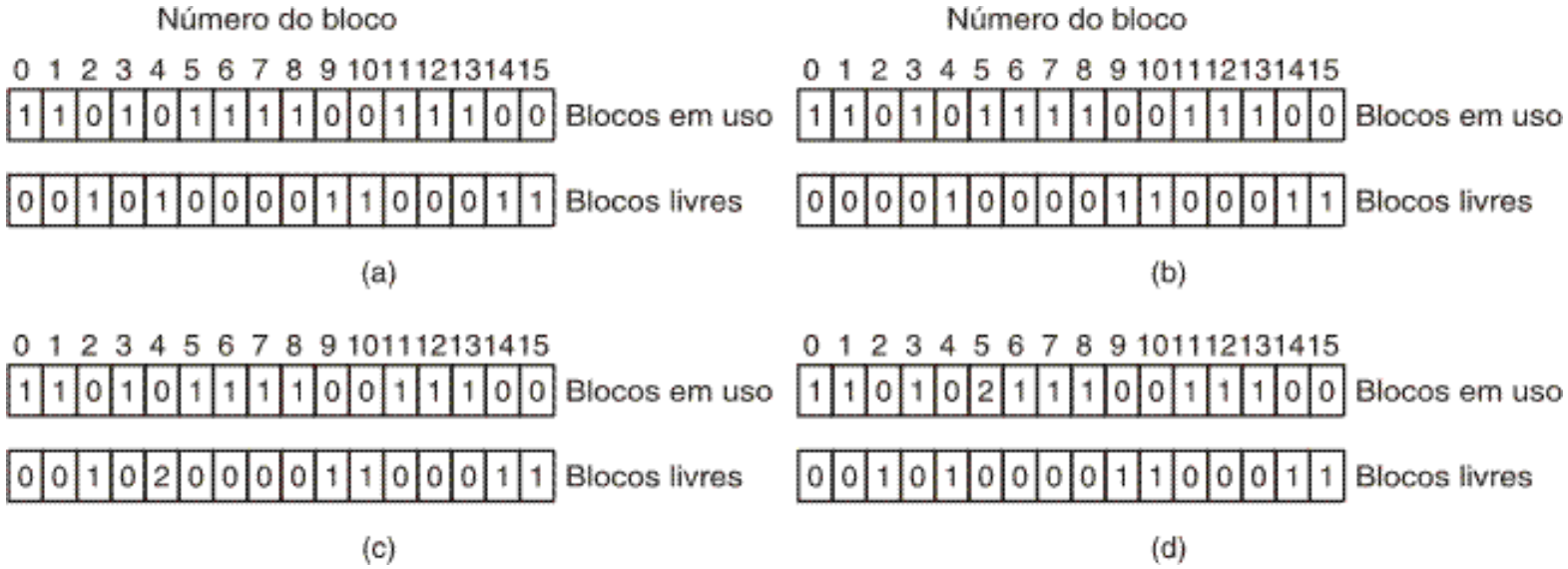
- (d) Fase 4: copia os arquivos marcados, terminando o processo de cópia.

Confiabilidade do Sistema de Arquivos (3)



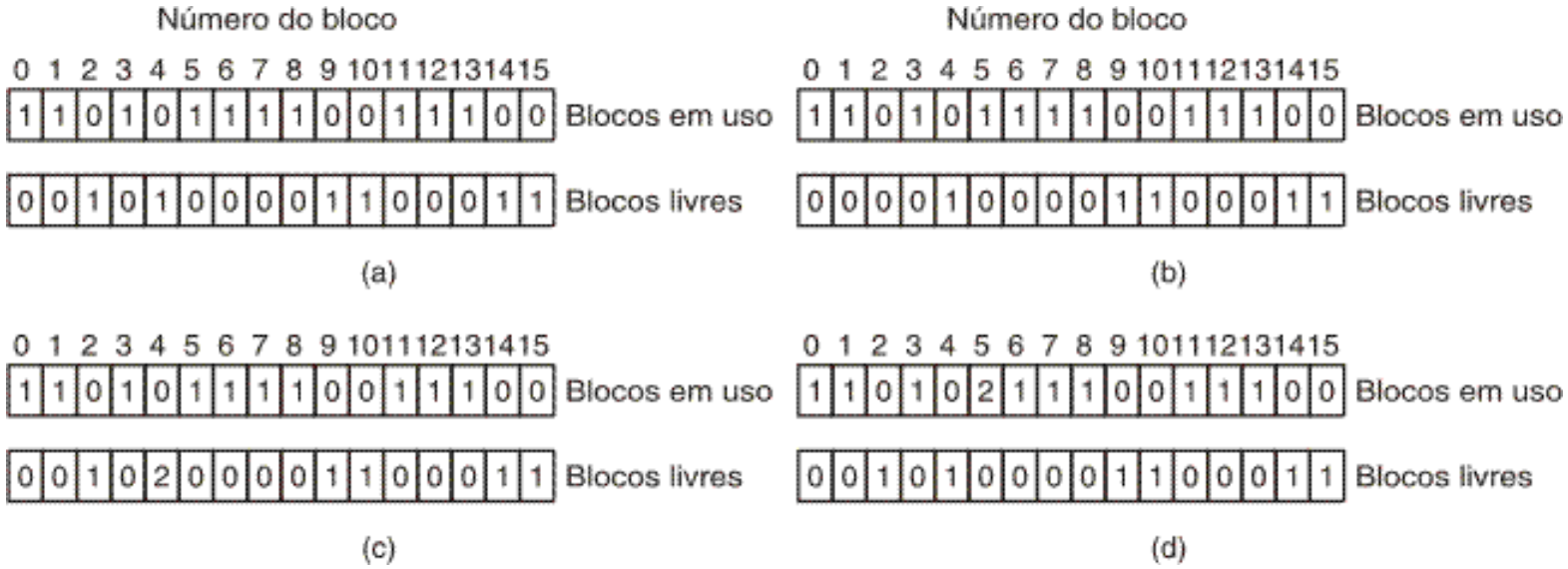
- **Estados do sistema de arquivos**
  a) Consistente
  b) Inconsistente: após um ***crash***, bloco desaparecido (2). Solução: apenas adicioná-lo à lista de livres.

Confiabilidade do Sistema de Arquivos (3)



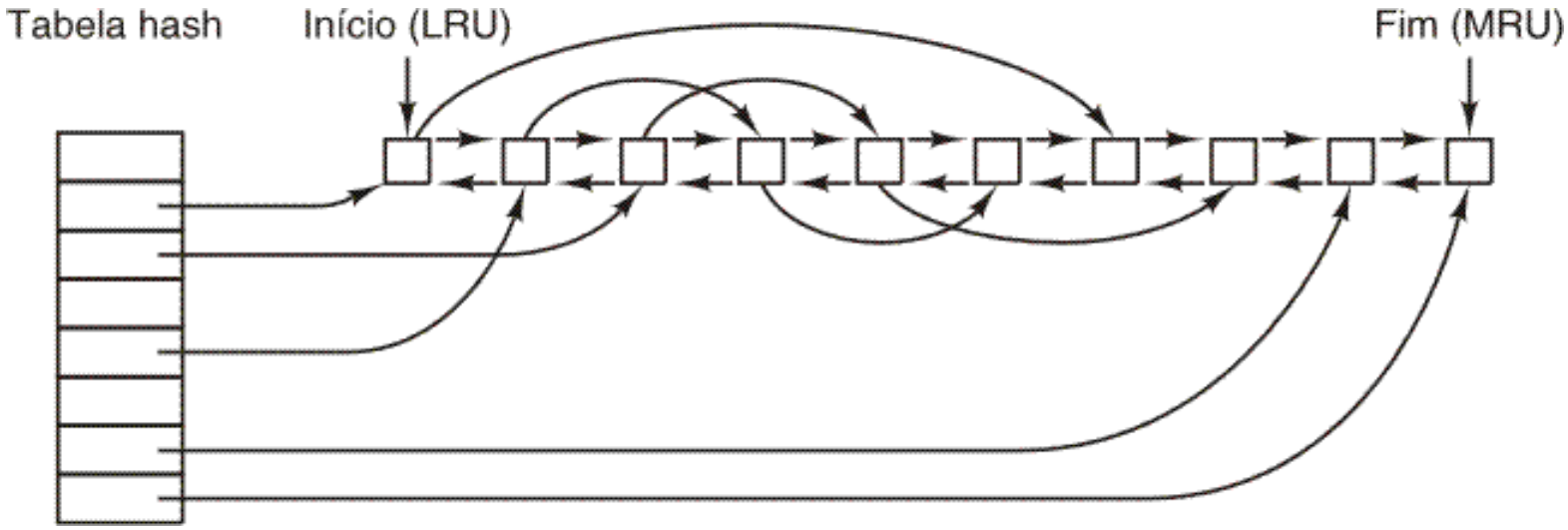
- **Estados do sistema de arquivos**
- (c) inconsistente: bloco duplicado na lista de livres (naturalmente que isso ocorre somente quando é uma lista e não um *bitmap*!). Solução também simples: reconstruir a lista de blocos livres.

Confiabilidade do Sistema de Arquivos (3)



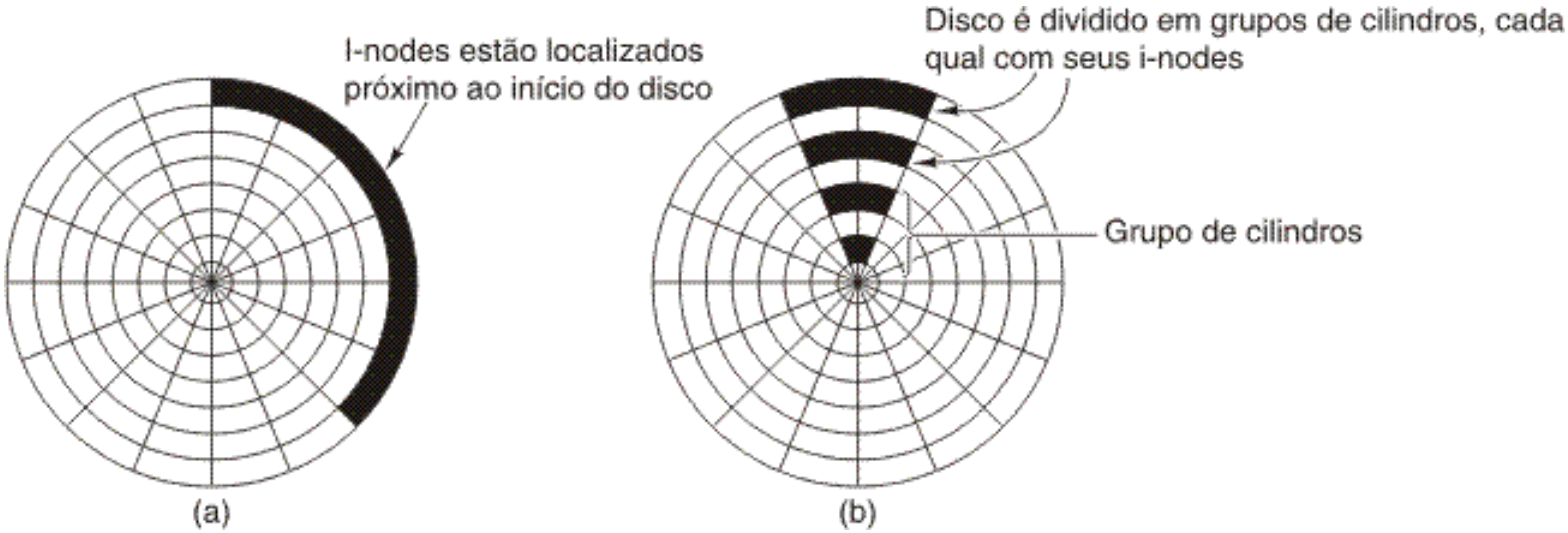
- (d) inconsistente: bloco (5) é reportado como pertencendo/alocado a dois arquivos distintos. Solução: alocar um novo bloco, copiar o conteúdo para ele e fazer um dos arquivos apontar para o novo bloco. O sistema de arquivo fica consistente, mas certamente algum dos arquivos terá problemas!

Desempenho do Sistema de Arquivos (1): *cache de blocos*



- Lista duplamente encadeada refletindo a ordem de uso. Tabela *hash* agiliza a localização de cada bloco (colisões também são tratadas junto à lista).
- As estruturas de dados da *cache* de *buffer*: estrutura que mantém controle de quais blocos de disco estão em memória (*buffer*). Similar ao problema de substituição de páginas de memória.
  - ***Write-through caches***: blocos modificados são escritos imediatamente em disco!

Desempenho do Sistema de Arquivos (2)



- *i-nodes* colocados no início do disco: distância média entre *i-node* e blocos do arquivo é aproximadamente metade do número de cilindros (*long seeks*!!!)
- Disco dividido em grupos de cilindros
  - cada qual com seus próprios blocos e *i-nodes*

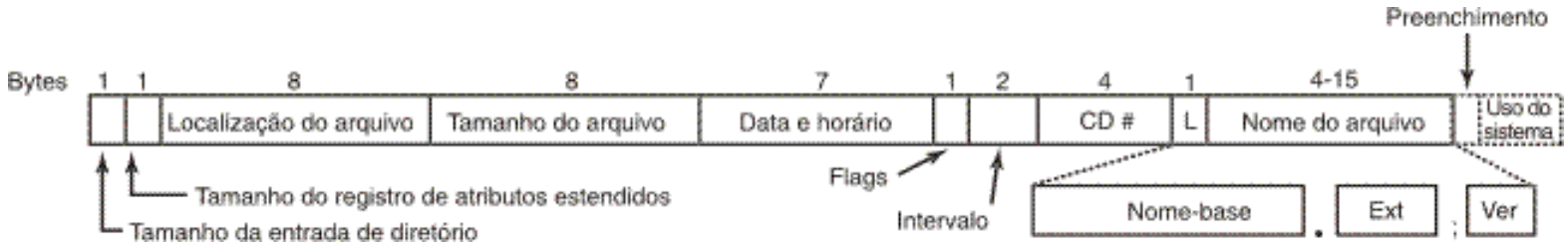
Sistemas de Arquivos *Log*-Estruturados

- Com CPUs mais rápidas, memórias maiores
  - *caches* de disco também podem ser maiores
  - número maior de requisições de leitura pode ser atendido pelo *cache*
  - assim, a maioria dos acessos a disco serão para escrita

Sistemas de Arquivos Log-Estruturados

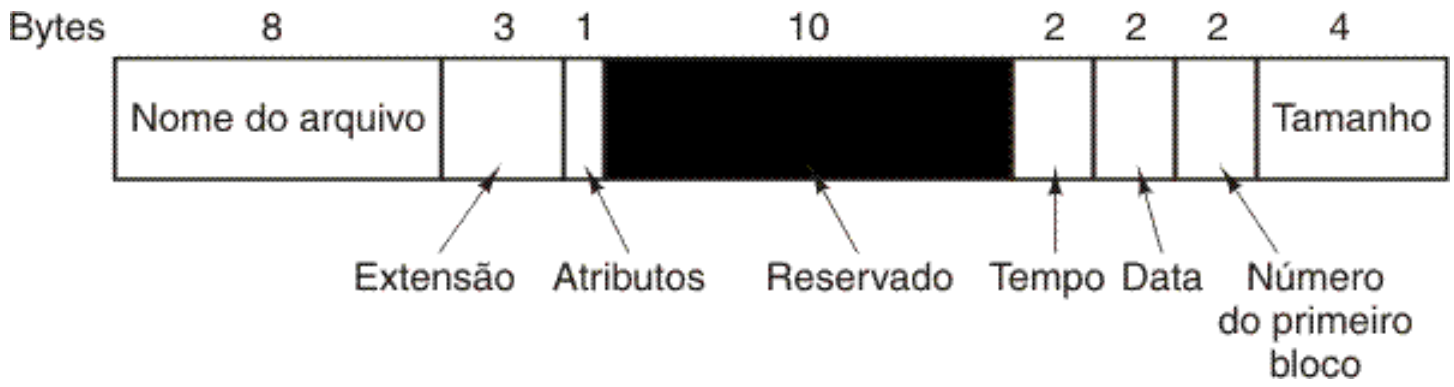
- A estratégia *Log-structure File System* (LFS) estrutura o disco todo como um *log*
  - inicialmente todas as escritas são armazenadas na memória
  - periodicamente todas escritas pendentes são escritas no fim do *log* em disco em um único segmento: assim, o *log* pode conter *i-nodes* e blocos de arquivos; portanto, não há mais um lugar fixo para os *i-nodes* (estão espalhados pelo disco)
  - Como todas as escritas pendentes são em um único segmento, pode-se gravar utilizando a eficiência máxima do disco.
  - quando um arquivo é aberto, localiza *i-node* e encontra os blocos

Exemplos de Sistemas de Arquivos Sistemas de Arquivos para CD-ROM



- A entrada de diretório ISO 9660

O Sistema de Arquivos MS-DOS (1)



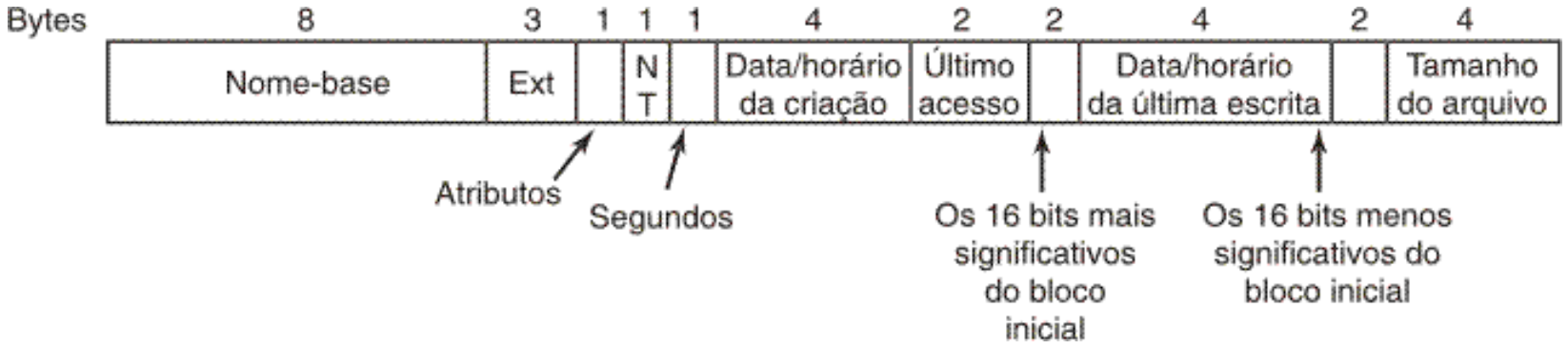
- A entrada de diretório do MS-DOS

O Sistema de Arquivos MS-DOS (2)

Tamanho do bloco	FAT-12	FAT-16	FAT-32
0,5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 BK		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

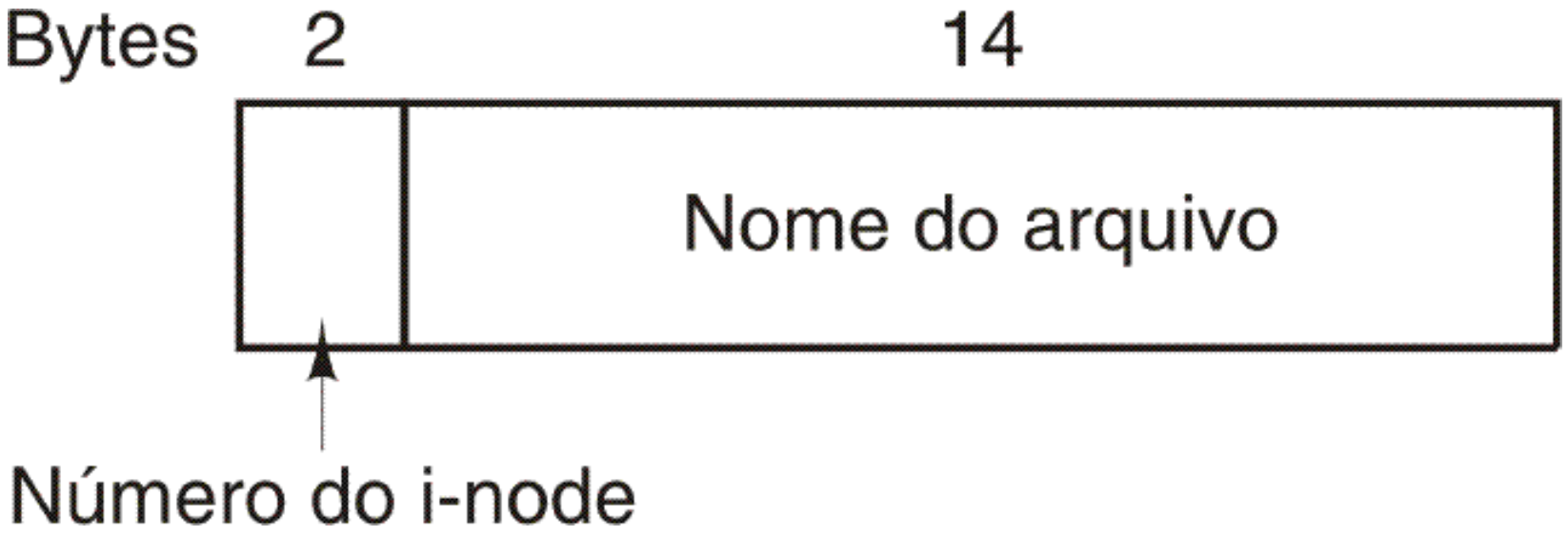
- Tamanho máximo de partição para diferentes tamanhos de bloco
- As caixas vazias representam combinações proibidas
- OBS.: No FAT-32 as partições são manipuladas como um múltiplo de 512 bytes (2^9 x 2^32 = 2 Tbytes).

O Sistema de Arquivos do Windows 98 (1)



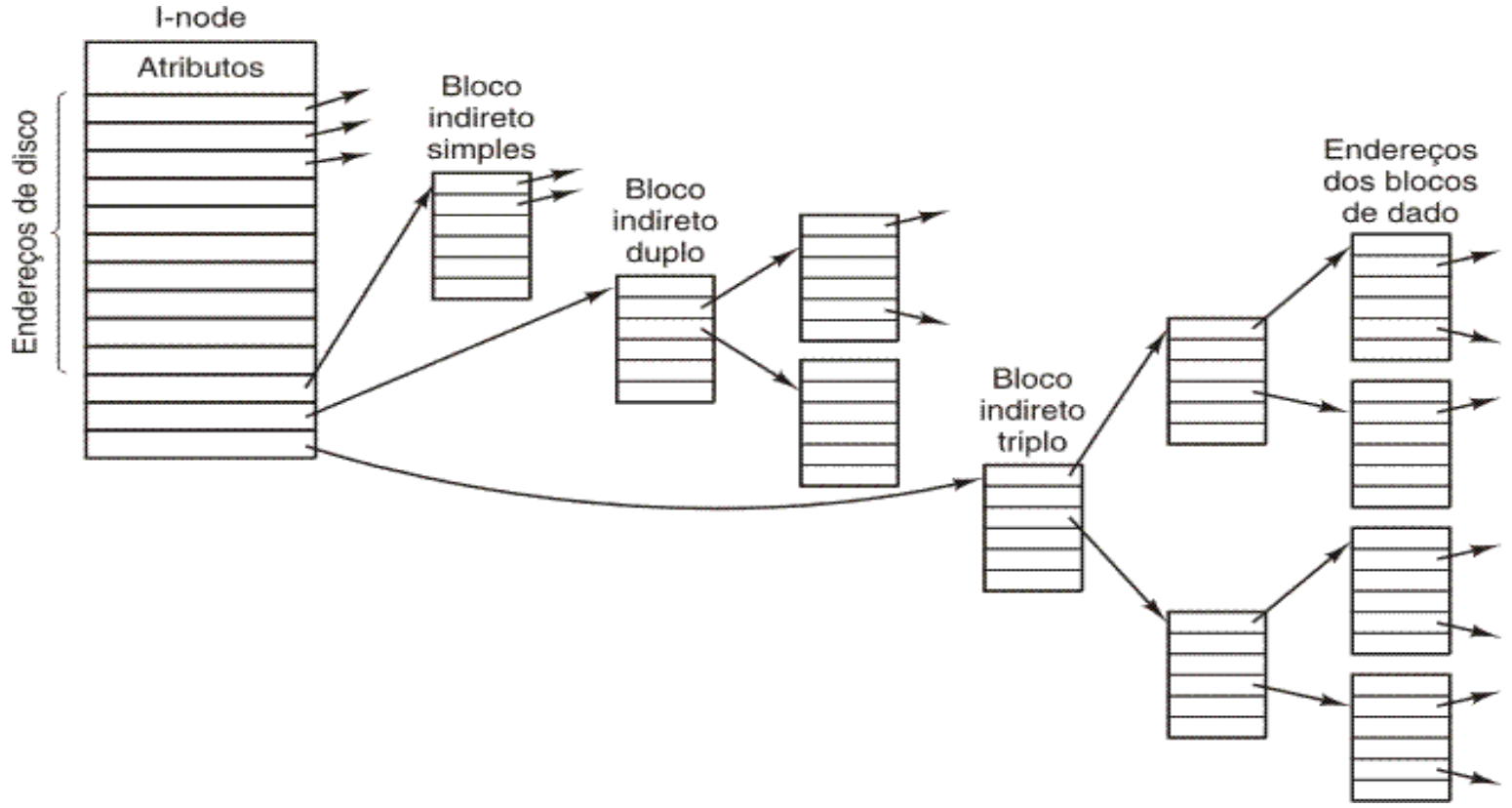
- **A entrada de diretório estendida do MS-DOS usada no Windows 98 (aproveitando os 10 bytes reservados): entre outras modificações, utiliza FAT-32, empregando-se 32 bits (distribuídos nas entradas: 16 bits mais significativos e 16 bits menos significativos).**

O Sistema de Arquivos do UNIX V7 (1)



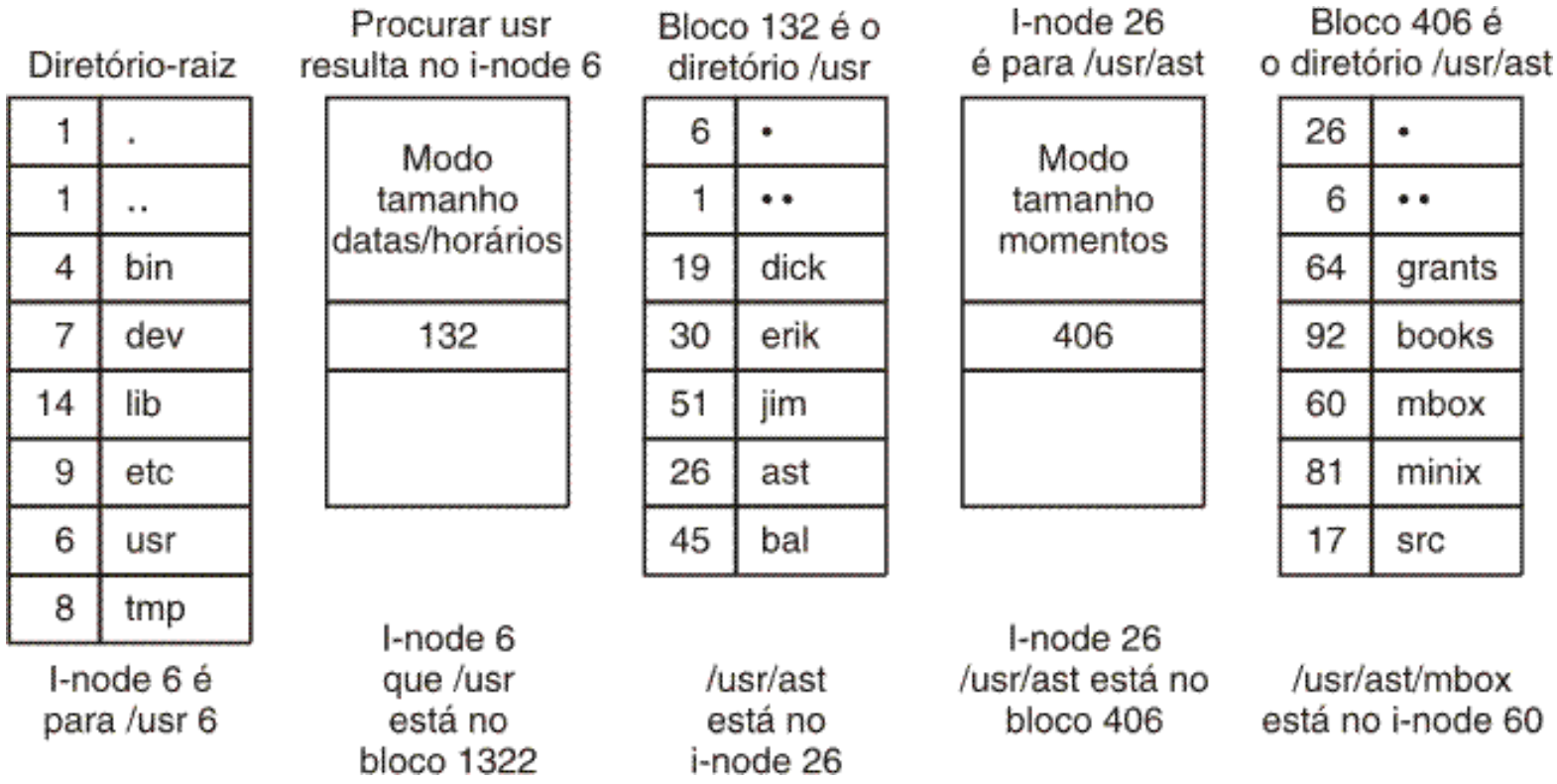
- Uma entrada de diretório do UNIX V7

O Sistema de Arquivos do UNIX V7 (2)



- Um i-node UNIX: caso necessário, pode-se estender o tamanho do arquivo utilizando indireção (blocos extras para armazenar índices de blocos).

O Sistema de Arquivos do UNIX V7 (3)



- Os passos para localizar */usr/ast/mbox*