Capítulo 4 -
Sistemas de arquivos

Sistemas de arquivos

Motivação:

Todas as aplicações precisam armazenar e recuperar informações.

Problemas na utilização do endereçamento virtual:

1. espaço é pequeno para apps bancárias, reserva de passagens, etc
2. processo termina ⇒ info perdida. Inaceitável para muitos casos: ex: banco de dados onde a info precisa ficar retida por longo tempo
3. diversos processos precisam ter acesso à informação ao mesmo tempo

   ⇒ não é possível se a informação estiver armazenada no espaço de endereçamento de um processo específico

Solução: **tornar a informação independente do processo**

Sistemas de arquivos

Condições essenciais para armazenamento de informações por um longo prazo:

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

2) A informação deve sobreviver ao término do processo que está usando a mesma.

3) Múltiplos processos devem ser capazes de acessar a informação simultaneamente.

Sistemas de arquivos

Pense em um disco como uma sequência linear de blocos de tamanho fixo que dão suporte a 2 operações:

- **ler bloco k**
- **escrever no bloco k**

As questões a seguir surgem rapidamente:

1. **Como encontrar a informação?**

2. **Como impedir que um usuário tenha acesso à info de outro?**

3. **Como saber quais os blocos que estão livres?**

Sistemas de arquivos

SO abstrai o conceito de processador ⇒ processo

SO abstrai o conceito de memória física ⇒ espaço de endereçamento virtual

Nova abstração: ***arquivo***

Def1:

"Arquivos são unidades lógicas de informação criadas por processos"

Def2:

"Um arquivo é um mecanismo de abstração que oferece meios de armazenar informações no disco e de lê-las posteriormente."

Objetivo:

Arquivos são usados para modelar o disco (assim como um espaço de endereçamento modela uma memória RAM)

Sistemas de arquivos

Características:

- ✓ **informação é persistente:** não é afetada pela criação ou término de um processo. Só desaparece por remoção explícita do proprietário;

- ✓ **são gerenciados pelo SO:** modo como são estruturados, nome, acesso, uso, proteção e implementação;

- ✓ **detalhes de implementação só interessam ao projetista**

Sistema de arquivos:

Parte do sistema operacional que gerencia os arquivos

Nomeação de arquivos

- Oferece meios de armazenar e ler informações do disco;
- Isola o usuário de detalhes sobre como e onde as informações estão armazenadas;
- Todos permitem nomes de, pelo menos, 1 a 8 caracteres. Muitos Sistemas de Arquivos permitem nomes de até 255 caracteres;
- Alguns distinguem letras maiúsculas de minúsculas (UNIX vs DOS);
- Exemplo de sistemas de arquivos: NTFS (Windows NT, XP, Vista);

Nomeação de arquivos

- extensão do arquivo (.c, .pas, etc). Pode dar (ou não) alguma informação sobre o arquivo:

  Ex: DOS – extensão de 1 a 3 caracteres
  UNIX – pode ter 2 ou mais extensões: *.tar.gz*

- no UNIX, extensões podem ter significado *.c* (para o compilador, não para o SO) ou não *.txt*;

- no Windows as extensões indicam qual programa possui aquela extensão.

Ex: dois cliques em um arquivo *.doc* inicia a execução do Word (ou similar)

Nomeação de arquivos

Extensão	Significado
.bak	Cópia de segurança
.c	Código-fonte de programa em C
.gif	Imagem no formato Graphical Interchange Format
.hlp	Arquivo de ajuda
.html	Documento em HTML
.jpg	Imagem codificada segundo padrões JPEG
.mp3	Música codificada no formato MPEG (camada 3)
.mpg	Filme codificado no padrão MPEG
.o	Arquivo objeto (gerado por compilador, ainda não ligado)
.pdf	Arquivo no formato PDF (Portable Document File)
.ps	Arquivo PostScript
.tex	Entrada para o programa de formatação TEX
.txt	Arquivo de texto
.zip	Arquivo compactado

Extensões de arquivo mais comuns e seu significado

Estrutura de arquivos

Os arquivos podem ser estruturados de várias formas:

1º tipo - sequência desestruturada de bytes

O significado é imposto pelos programas no nível de usuário.

Vantagem: ***máxima flexibilidade***

Ex: Unix e Windows

2º tipo - sequência de registros (de tamanho fixo) com alguma estrutura interna

Usado em sistemas antigos (cartões perfurados com 80 caracteres ou impressoras de linha – 132 caracteres)

Estrutura de arquivos

3º tipo - árvore de registros

- ✓ registros não possuem, necessariamente, o mesmo tamanho
- ✓ cada registro tem uma ***chave***
- ✓ árvore ordenada pela ***chave***: facilita a busca por uma chave e a inserção de novos registros

Ex: Usados em computadores de grande porte para processamento de dados comerciais

⇒ **facilita a busca por uma chave específica**

Estrutura de arquivos

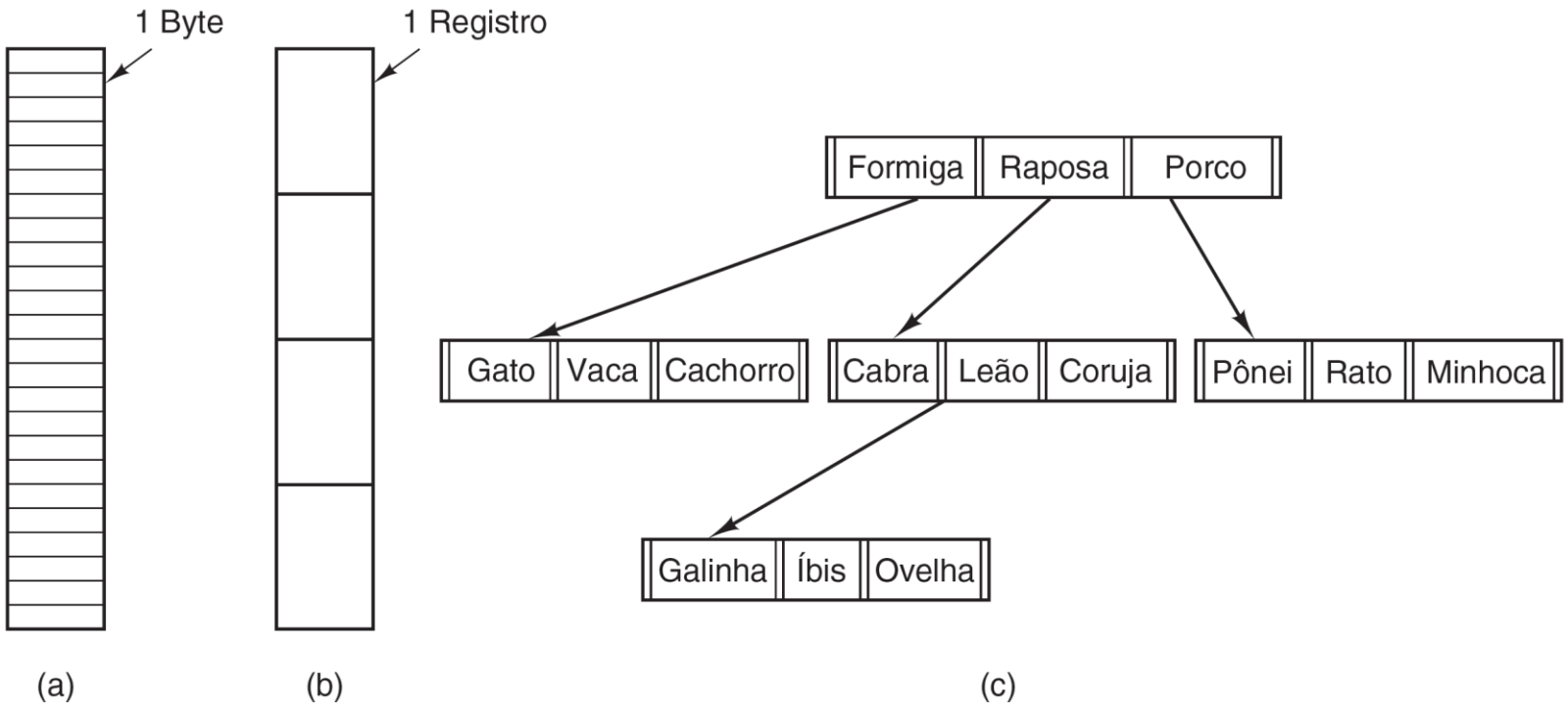

Figura 4.1 Três tipos de arquivos. (a) Sequência de bytes. (b) Sequência de registros. (c) Árvore.

Tipos de arquivos

Arquivos regulares:

contêm informação do usuário

Diretórios:

arquivos que mantêm a estrutura do sistema de arquivos

Arquivos especiais de caracteres:

usados para modelar dispositivos de E/S (terminais, impressoras e redes)

Arquivos especiais de blocos:

usados para modelar discos

Tipos de arquivos

Arquivos regulares: ASCII ou BINÁRIOS

ASCII: *linhas de texto terminadas por CR ou LF*

Vantagens:

1. são mostrados e impressos como são e editados com qualquer editor de texto.

2. fácil conectar a saída de um programa com a entrada do outro se ambos usam entrada e saída ASCII. Ex: pipeline do *SHELL*

Tipos de arquivos

Binário: possuem uma estrutura interna reconhecida pelos programas que os usam

Ex1: *executável do Unix*

composto de 5 partes: *cabeçalho, texto, dados, bits de realocação e tabela de símbolos*

Cabeçalho:

- Nº mágico: identifica o arquivo como executável
- Tamanho das várias partes do arquivo
- Endereço onde deve se iniciar a execução (ponto de entrada)
- Bits de sinalização

Tipos de arquivos

Texto e dados do programa:

- ✓ Carregados na memória
- ✓ Realocados usando os bits de realocação

Tabela de Símbolos: usada para depuração

Ex2: ARCHIVE (repositório do UNIX)

- ✓ Coleção de procedimentos de bilioteca compilados, mas não ligados
- ✓ Cada procedimento tem um cabeçalho com: nome, data de criação, proprietário, código de proteção e tamanho (tudo em binário)

Tipos de arquivos

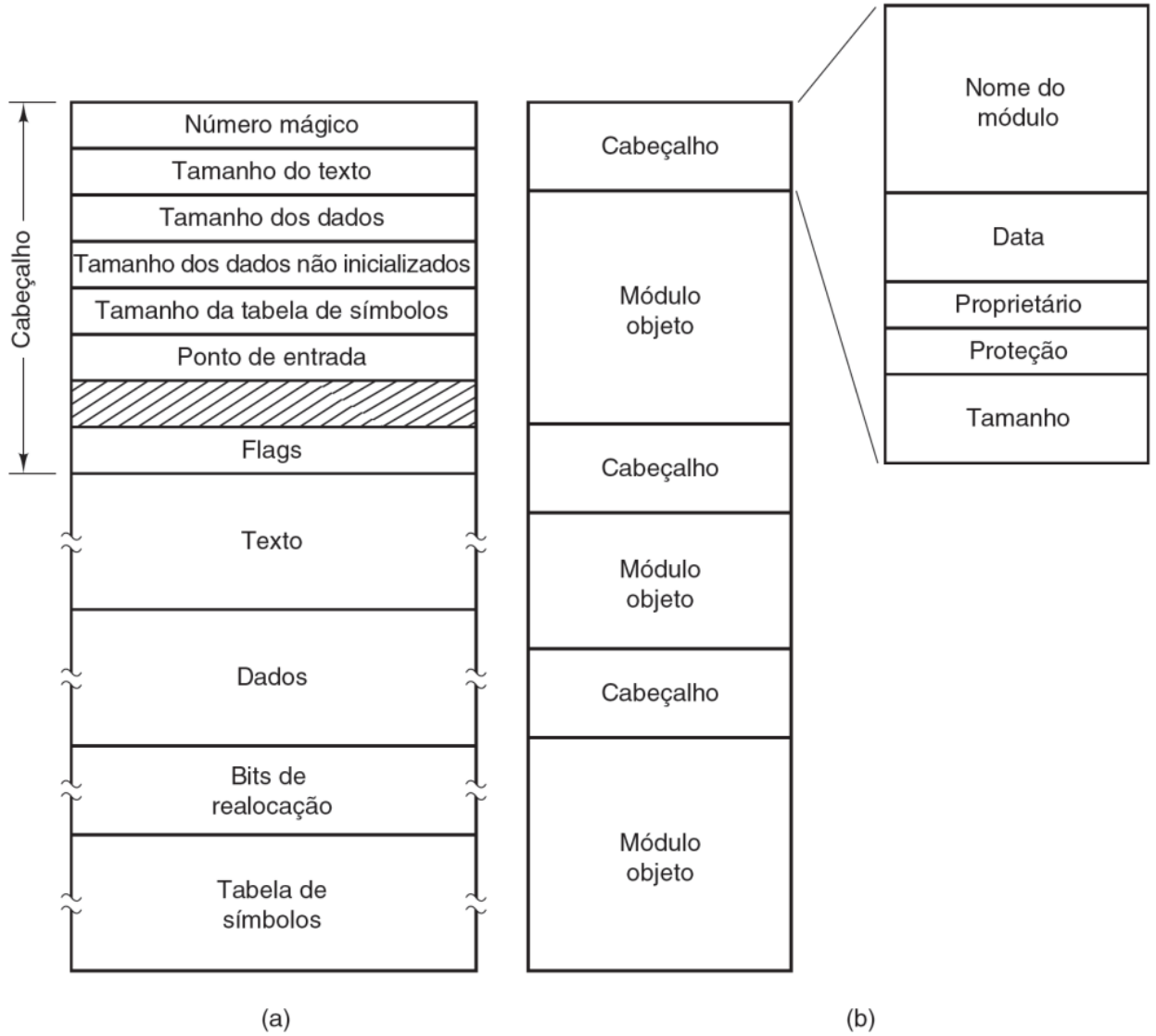

Figura 4.2 (a) Um arquivo executável. (b) Um arquivo.

Acesso aos arquivos

- **arquivos de acesso sequencial**: lê todos os bytes/registros a partir do início, mas nunca saltando e lendo fora de ordem.

  Ex: fita magnética

- **arquivos de acesso aleatório**: bytes e registros são lidos em qualquer ordem. Usados nos discos, pode-se ter acesso aos registros pela ***chave*** ao invés de pela posição. **Essencial em sistemas de banco de dados.**

  Ex: reserva de um voo em uma cia aérea. Deseja-se acesso direto ao registro do voo.

Início da leitura de dados: dois métodos

READ – indica a posição do arquivo na qual se inicia a leitura

SEEK – estabelece uma posição dentro do arquivo.

Leitura é feita a partir de então: Unix e Windows

Atributos de arquivos

Conjunto de informações que todos os sistemas operacionais associam aos arquivos, além de seu nome e de seus dados:

Ex: data e horário em que foi criado/modificado

Linhas de 1 a 4 – proteção – quem pode acessar o arquivo;

Linhas de 5 a 12 – flags diversos; Ex: 8 – cópia de segurança;

11 – (arqs. temporários) - permite remoção automática do arquivo quando um processo termina;

Linhas de 13-15 – informações necessárias para encontrar uma chave;

Linhas 16 – 18: momentos de criação, último acesso e alteração

Exemplo de utilidade: ***recompilar um arquivo se fonte foi modificado depois da criação do arquivo objeto***

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

Tabela 4.2 Alguns atributos possíveis de arquivos.

Operações com arquivos

Chamadas de sistemas mais comuns relacionadas aos arquivos:

- Create (criar)
- Delete (apagar)
- Open (abrir)
- Close (fechar)
- Read (ler)
- Write (escrever)
- Append (anexar)
- Seek (posicionar)
- Get Attributes (obter atributos)
- Set Attributes (configurar atributos)
- Rename (renomear)

Operações com arquivos

- Create – arquivo é criado sem dados; definir alguns atributos
- Delete – remove o arquivo e libera o espaço ocupado pelo mesmo
- Open – SO busca e coloca na memória os atributos e a lista de endereços do arquivo no disco a fim de tornar mais rápido o acesso em chamadas posteriores
- Close – fecha o arquivo, libera espaço na tabela interna (pois os seus atributos e endereços não são mais necessários), forçando a escrita do último bloco do arquivo
- Read – bytes são lidos a partir da posição atual; quantidade de dados deve ser especificada, assim como um buffer para colocá-los

Operações com arquivos

- Write – dados são escritos no arquivo na posição atual.

  Se pos atual é o fim do arquivo ⇒ tamanho aumenta.

  Se pos atual é o meio do arquivo
  ⇒ **dados sobrescritos são perdidos!**

- Append – adiciona dados somente ao final do arquivo

- Seek – para arquivos de acesso aleatório. Reposiciona o ponteiro do arquivo para um local específico. Dados podem ser lidos ou escritos a partir daquela posição

Operações com arquivos

- Get attributes – necessário para alguns programas.

  Ex: ***make*** verifica os momentos de alteração para realizar o mínimo de compilações necessárias

- Set attributes – serve para o usuário alterar alguns atributos do arquivo. Ex: proteção

- Rename – serve para o usuário alterar o nome de um arquivo existente

Sistemas de diretórios – nível único

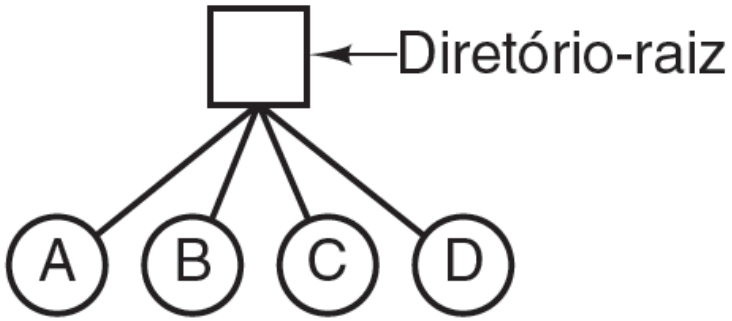

Figura 4.4 Um sistema de diretórios em nível único contendo quatro arquivos.

Sistemas de diretórios – nível único

Vantagens:

- Simplicidade
- Rapidez na localização do arquivo

Ex: sistemas embarcados simples (telefones, câmeras digitais e players de música)

Sistemas de diretórios hierárquicos

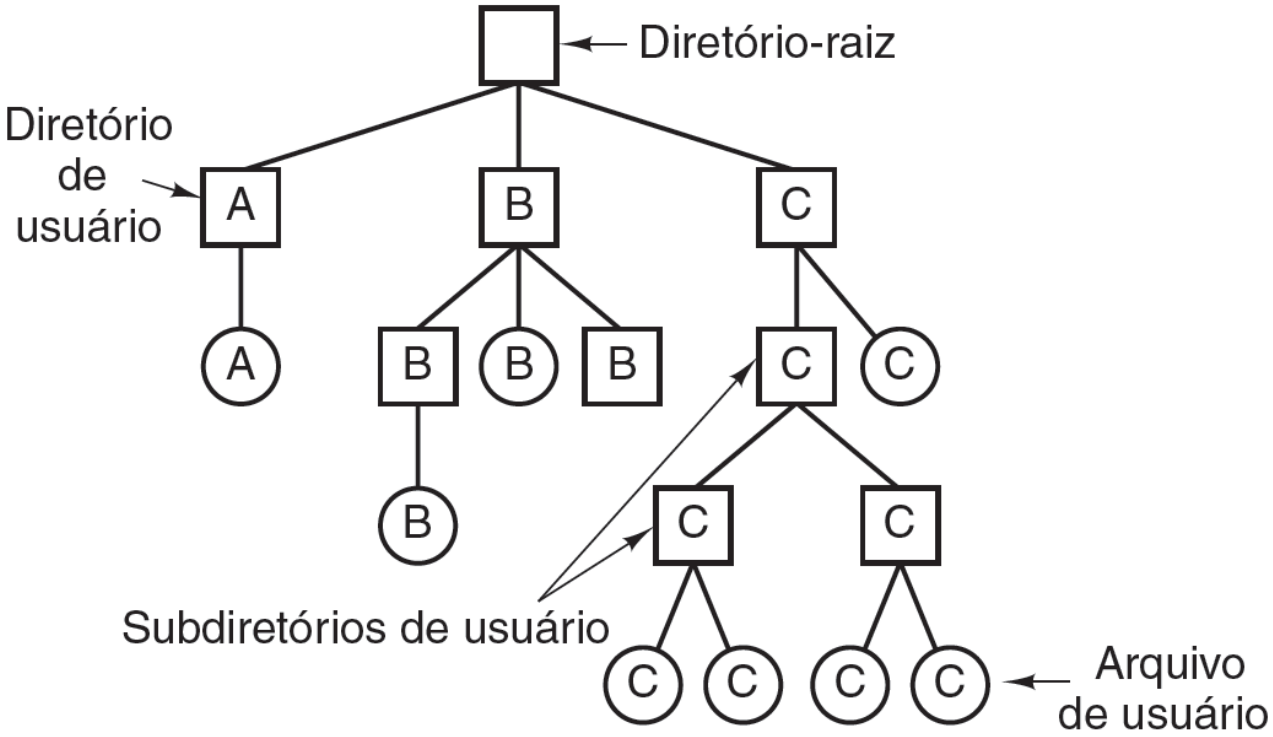

Figura 4.5 Um sistema hierárquico de diretórios.

Sistemas de diretórios hierárquicos

Características:

- Localização rápida entre centenas de milhares de arquivos
- Cada usuário tem quantos diretórios quiser para agrupar seus arquivos
- Cada usuário cria sua própria hierarquia

Ex: quase todos os SOs

Nomes de caminhos

Nome de caminho absoluto:

- caminho entre o diretório-raiz e o arquivo
- sempre iniciam no diretório-raiz e são únicos
- primeiro caracter é o separador
- Ex: */usr/ast/teste (UNIX) ou \usr\ast\teste (Windows)*

Nome de caminho relativo:

- usado com o conceito de ***diretório atual*** (***ou de trabalho***)
- nome do arquivo que não começar com o separador refere-se ao diretório atual

Nomes de caminhos

Nome de caminho relativo:

Ex: se diretório atual for ***ast*** os comandos:

cp /usr/ast/teste /usr/ast/teste.bkp e

cp teste teste.bkp têm o mesmo efeito

Obs:

1. nome de caminho absoluto sempre funcionará para programas que precisem acessar algum arquivo usado pelo programa
2. cada processo tem seu próprio diretório de trabalho
3. rotinas de biblioteca raramente alteram seu diretório de trabalho ou, quando alteram, retornam ao diretório original antes de retornar
   ⇒ causaria problemas a programas que as usam se não fosse assim

Nomes de caminhos

Entradas especiais: ´.´ e ´..´ (diretório atual e seu pai)

úteis para que não seja necessária a troca do diretório atual para seu pai ou para algum de seus filhos

Ex: supondo que o diretório atual seja /usr/ast, o comando:

cp ../lib/dictionary . (UNIX)

Copia o arquivo **dictionary** (que está no diretório /usr/lib) para o diretório atual. O nome se mantém inalterado

Obs: quando o último argumento é um diretório, o arquivo é copiado neste diretório

Nomes de caminhos

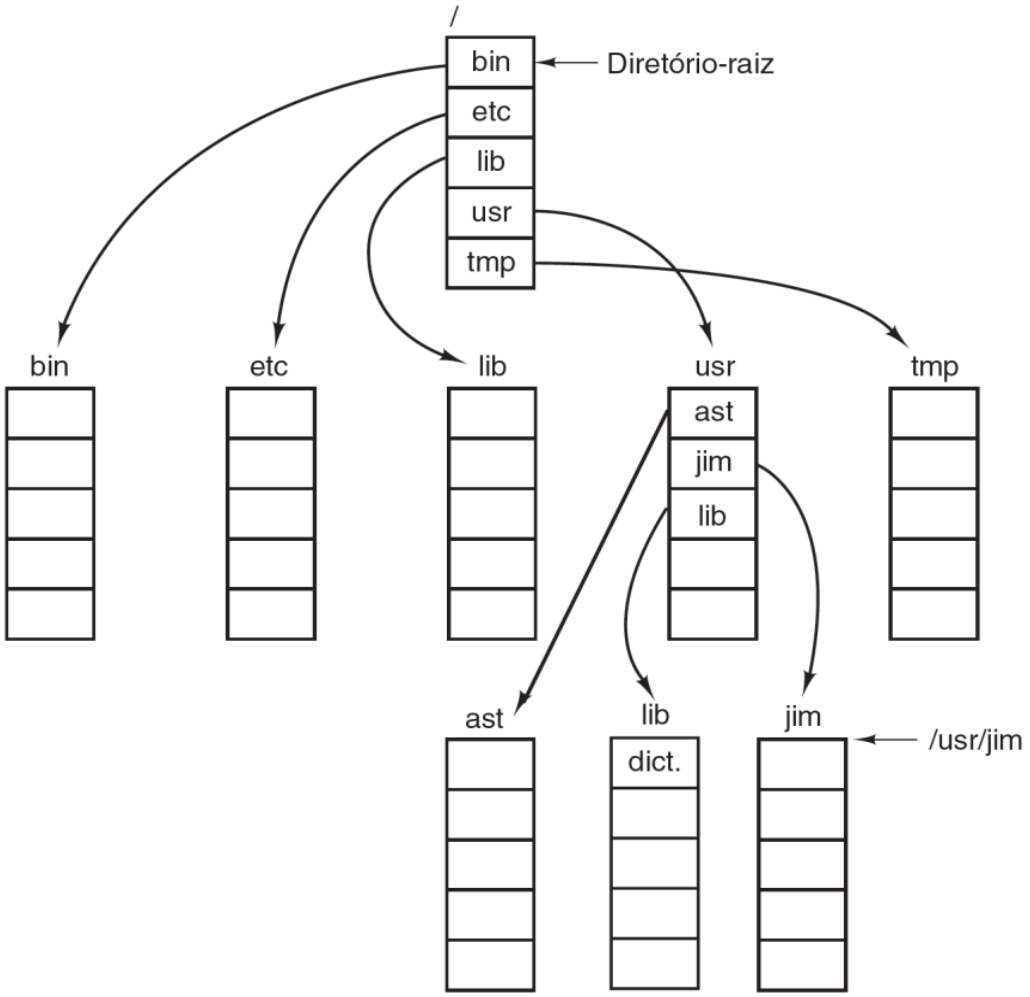

Figura 4.6 Uma árvore de diretórios UNIX.

Operações com diretórios

Chamadas de sistema para gerenciamento de diretórios:

- Create (criar)
- Delete (apagar)
- Opendir (abrir diretório)
- Closedir (fechar diretório)
- Readdir (ler diretório)
- Rename (renomear)
- Link (ligar)
- Unlink

Operações com diretórios

Chamadas de sistema para gerenciamento de diretórios:

- Create (criar) – cria um dir vazio, exceto . e ..
- Delete (apagar) – remove um dir (somente vazio)
- Opendir (abrir diretório) – antes de ser lido (listagem de seus arquivos) um diretório deve ser aberto
- Closedir (fechar diretório) – diretório deve ser fechado para liberar espaço na tabela interna

Operações com diretórios

Chamadas de sistema para gerenciamento de diretórios:

- Readdir (ler diretório) – devolve a entrada de um diretório em um formato padronizado, independentemente da estrutura usada

- Rename (renomear) – renomeia um diretório da mesma forma que é feita para um arquivo comum

Operações com diretórios

Chamadas de sistema para gerenciamento de diretórios:

- Link (ligar) – possibilita a um arquivo aparecer em mais de um diretório. Cria uma ligação entre um arquivo e um nome de caminho (***hard link***) e monitora o número de entradas de diretório contend o arquivo

- Unlink – remove uma entrada de diretório. Somente o nome do caminho especificado é removido. Se o arquivo estiver presente somente em um diretório, ele é removido do sistema de arquivos