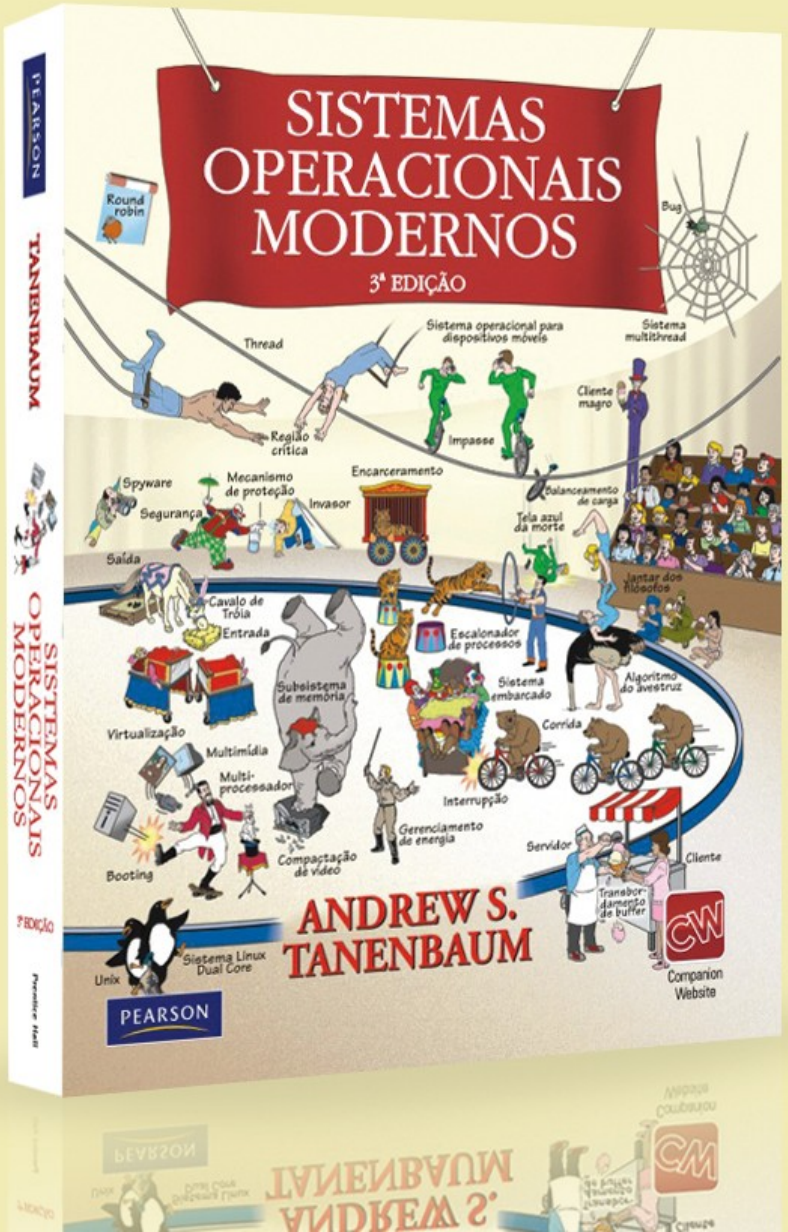
Sistemas Operacionais I

L. F. Friedrich

Sistema de Arquivos : Interface

Sistemas operacionais modernos
Terceira edição
ANDREW S. TANENBAUM

Capítulo 4
Sistemas de arquivos



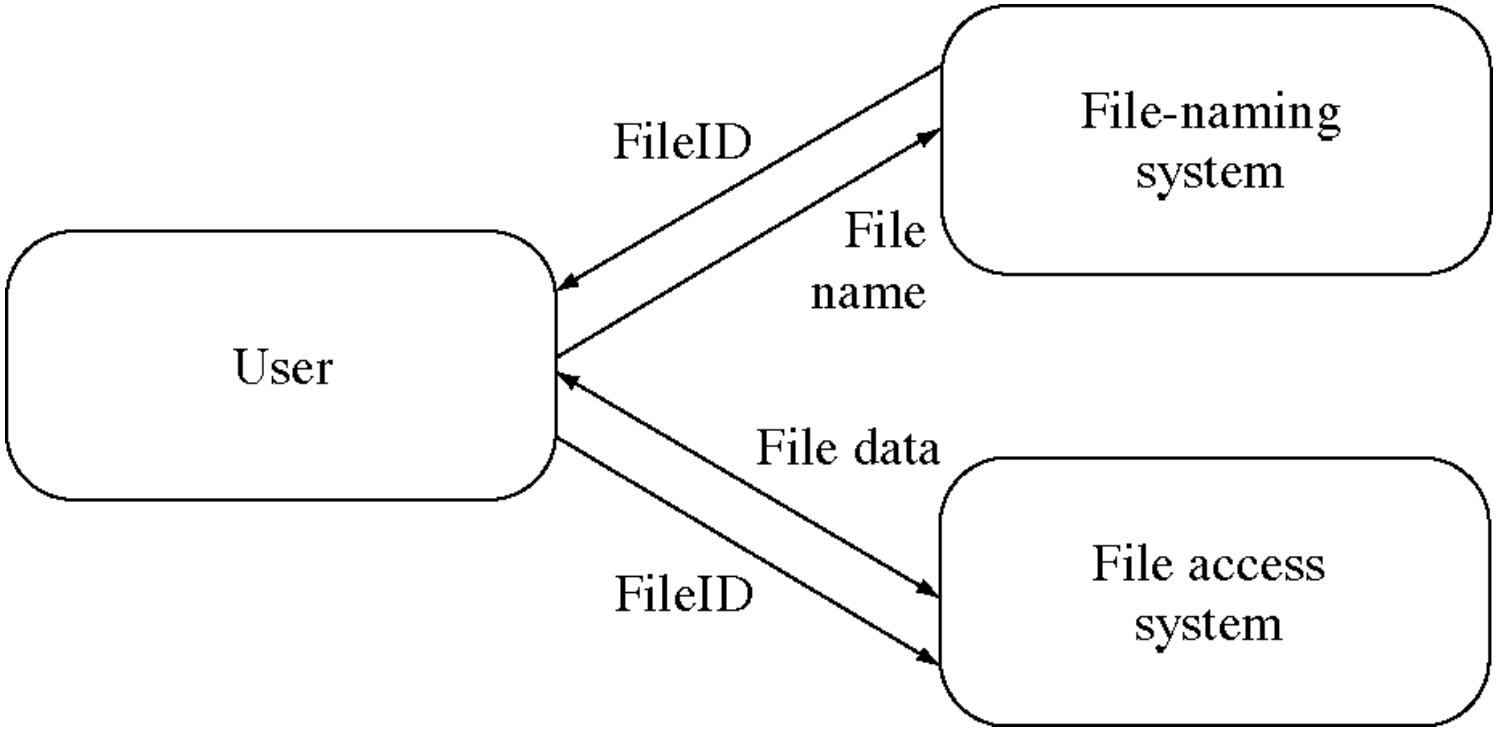
Apresentação

- Porque arquivos são necessários?
  - Armazenar informação na memória é bom porque memória é rápida. Entretanto, memória é limitada e desaparece depois que o processo termina. Além disso, vários processos acessam a informação.
  - Arquivo oferece uma forma de armazenar quantidade grande de informação e a longo prazo.
    - É persistente e sobrevive depois do término do processo.
  - Arquivo oferece uma interface comum para a manipulação transparente de dados em memória secundária.
  - Arquivo é também um objeto compartilhado por processos que o acessam concorrentemente.
    - Mecanismos de acesso
  - Arquivos são suportados atraves de um **Sistema de Arquivos**

Arquivos

- Sistema de Arquivos
  - Responsável pelo gerenciamento, estrutura, nomes, acesso, uso, proteção e implementação.
  - Abstração de um dispositivo de armazenamento, como por exemplo: disco, como se fosse:
    - Uma coleção de dados (arquivos), e
    - Uma coleção de informações de controle (diretório)
  - A interação com os dispositivos de armazenamento é feita através de serviços (funcionalidades) que são oferecidas pelos tratadores de dispositivos(drivers)
    - O dispositivo é visto como um array de blocos
  - É a parte (estrutura) mais visível de um SO
  - Alguns exemplos:
    - FAT, NTFS, EXT2, MINIX, ISO9660, etc

Duas principais partes do SA



- Visão de um SA
  - Interface de Sistemas de Arquivos
  - Implementação de Sistemas de Arquivos

Arquivos: nomeação

- Um arquivo é um mecanismo de abstração
  - Oferece meios de armazenar e recuperar informações
  - Sem revelar detalhes de como e onde estão armazenadas
- Importante em um mecanismo de abstração: como os objetos são gerenciados e nomeados
- O sistema de arquivos define um espaço de nomes
  - Conjunto de regras e convenções para identificar simbolicamente um arquivo
- Espaço de nomes varia de sistema para sistema
  - Case sensitive, extensão, tamanho máximo
  - Linux:
    - Case sensitive, sem extensão, 255 char

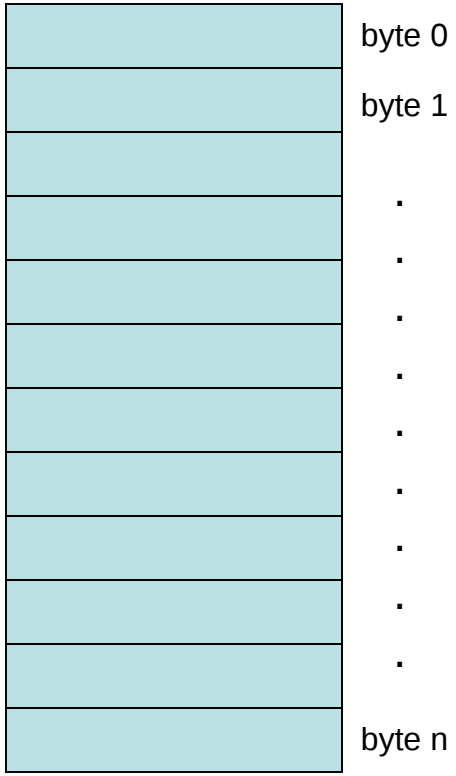
Nomeação de arquivos

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

Tabela 4.1 Algumas extensões comuns de arquivos.

Arquivo: estrutura

- Como o conteúdo do arquivo é estruturado?
  - Usual: uma seqüência de bytes.
  - Alternativas: linhas, registros, etc.
  - Vantagem : maxima flexibilidade.
    - Programas de usuário podem fazer qualquer coisa no arquivo.



Estrutura de arquivos

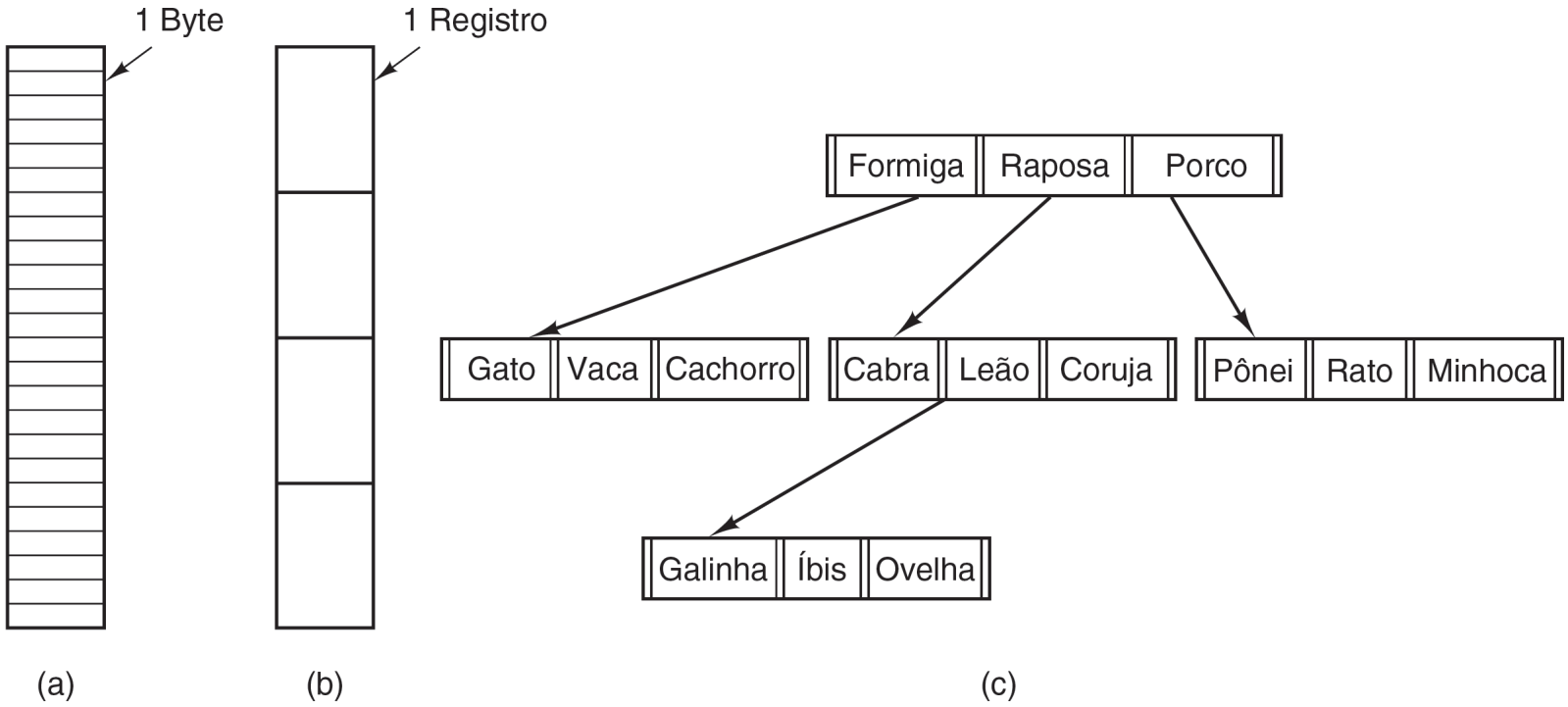


Figura 4.1 Três tipos de arquivos. (a) Sequência de bytes. (b) Sequência de registros. (c) Árvore.

Arquivo: tipos

- Sistemas Operacionais dão suporte a vários tipos de arquivos, alguns exemplos:
- Regular
  - Contêm informação do usuário
- Diretório
  - Estrutura do sistema de arquivos
- Especial (Device file)
  - Relacionados a E/S

Arquivo Regular

- Arquivos regulares são em geral, ASCII ou Binario
  - ASCII armazena apenas (linhas) texto.
    - vantagem?
  - Binario não é ASCII.
    - Não é printável, tem uma estrutura interna conhecida pelos seus usuários
  - Em UNIX/Linux, o comando file pode distinguir o tipo do arquivo.

Distingindo Arquivos Binarios

- Todo SO deve reconhecer pelo menos um tipo de arquivo: ?
- Um padrão especial é armazenado no início de cada arquivo binário.
  - Conhecido como magic number.
  - Ex., arquivos GIF sempre iniciam com o string “GIF87a” ou “GIF89a”.
  - Ex., arquivos PDF sempre iniciam com “%PDF-”.

- O comando “file” conhece todos os magic numbers de todos tipos de arquivos.
  - Todos os magic numbers estão listados em “/usr/share/file/magic”.
  - “file” não verifica a extensão do nome do arquivo.

  - Exemplo TOPS-20 (DECsystem 20): make
    - Verificava qual a data e horário de criação de qualquer arquivo a executar (comparava com fonte)-extensão obrigatória-

Tipos de arquivo

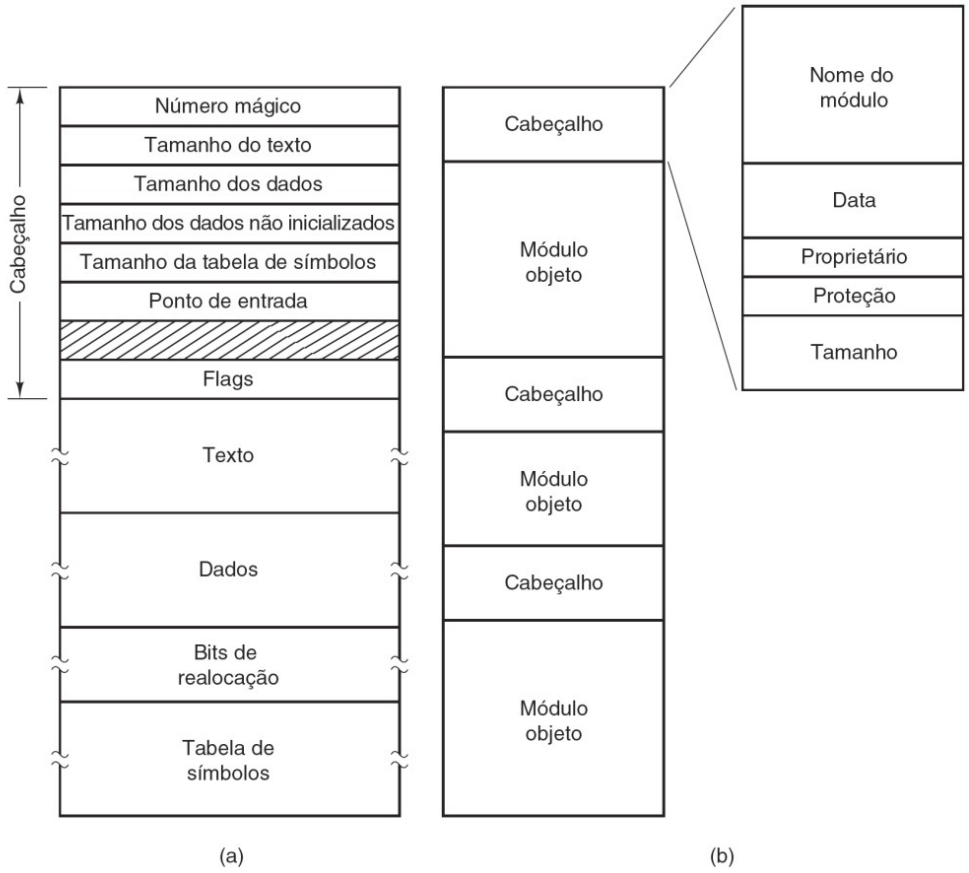
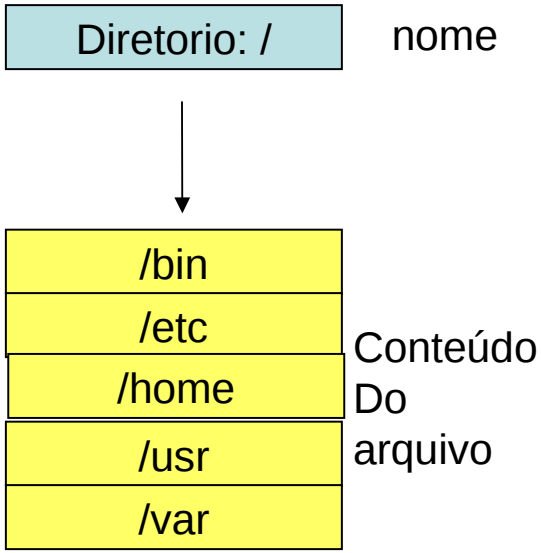


Figura 4.2 (a) Um arquivo executável. (b) Um arquivo.

Arquivo Diretório

- Tem uma estrutura conhecida do SA que contêm as entradas de um determinado diretório:



Arquivo Dispositivo

- No Unix/Linux, a maioria dos dispositivos são representados como arquivos.
  - /dev/hda é o 1o. IDE hard disk.
  - /dev/sda é o 1o. SATA ou SCSI hard disk.
  - /dev/fd0 é o floppy drive.

```
[root@homepc /]# ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 2007-09-24 17:11 /dev/sda
```

  - Qual é o efeito de
    - $cat qualquerarq > /dev/fd0 ?

Arquivo Dispositivo

- Unix/Linux criam alguns pseudo-dispositivos por questões de conveniencia.
  - /dev/null: ou o buraco negro. Qualquer dado escrito neste dispositivo será descartado.
    - Ex.: cat test.txt > /dev/null; depois, cat /dev/null.

Arquivos: Acesso

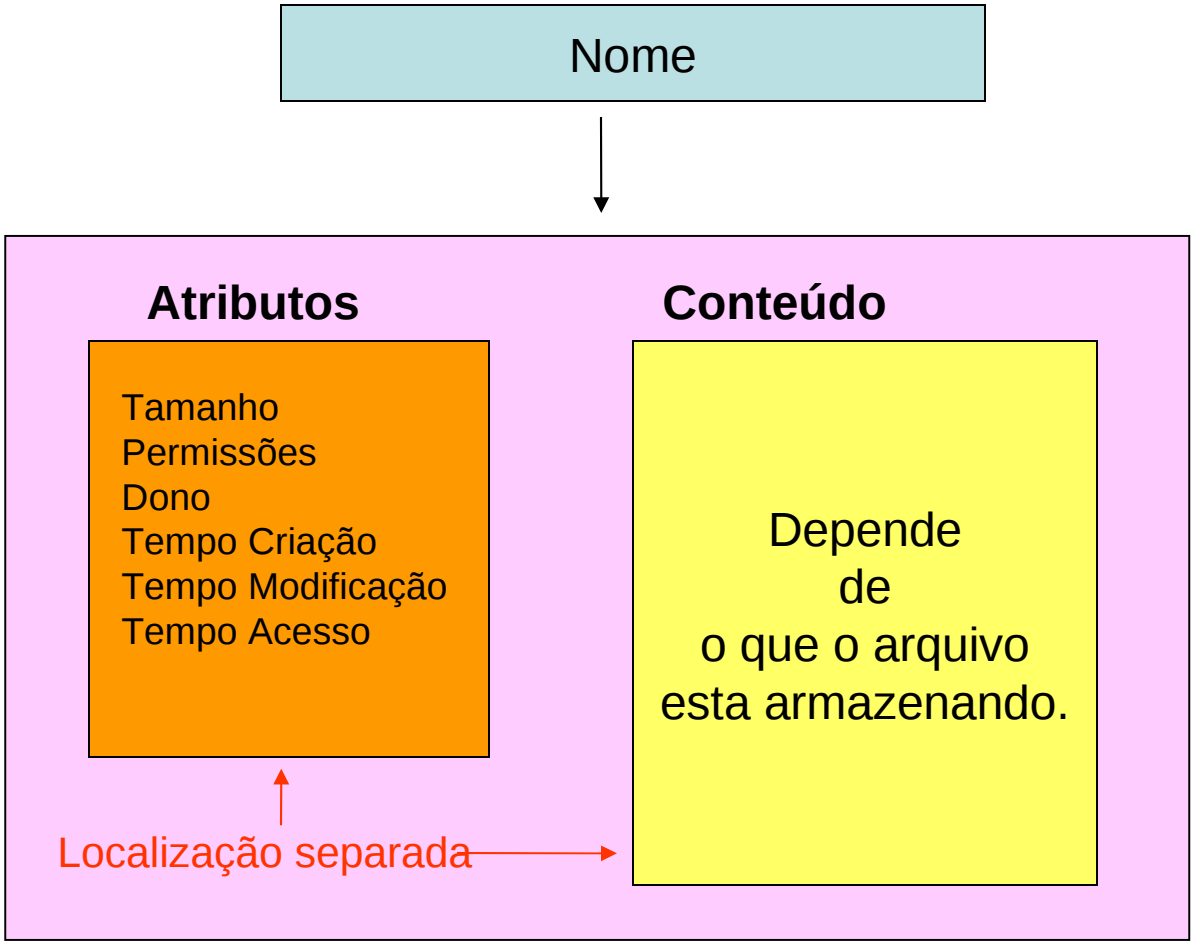
- Acesso Sequencial
  - Apenas encontrado em sistemas antigos.
  - Bytes ou registros devem ser lidos em ordem, do início. Sem operação “seek” (rewind permitido)
  - Como uma fita.

- Acesso Aleatório
  - Aplicações podem ler bytes em qualquer posição, em qualquer ordem.
  - Onde a leitura começa?
  - Operação “Seek” é suportada.

Arquivos: atributos

- Todo arquivo possui um nome e seus dados.
- Além disso, todos os SO associam outras informações a cada arquivo – como data/horário da última modificação, tamanho. Estes itens extras são chamados de atributos (metadados) do arquivo.
- Os atributos do arquivo são importantes para o SA.
  - Um arq pode estar vazio, mas não sem atributos.
- Os atributos são dependentes do SA.
  - Proteção
  - Flags de controle, determinam controlam características
  - Datas atualização/criação
  - Tamanho

Anatomia de Arquivo



Tudo esta sob o controle do sistema operacional.

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

Tabela 4.2 Alguns atributos possíveis de arquivos.

Como Ler Atributos - Linux

- Use o comando stat.

```
tywong@homepc:~$ stat /
  File: `/'
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 802h/2050d      Inode: 2           Links: 21
Access: (0755/drwxr-xr-x)  Uid: (  0/  root)   Gid: (  0/  root)
Access: 2007-10-07 18:10:02.000000000 +0800
Modify: 2007-07-25 20:22:22.000000000 +0800
Change: 2007-07-25 20:22:22.000000000 +0800
```

```
tywong@homepc:~$ stat /boot/grub/menu.lst
  File: `/boot/grub/menu.lst'
  Size: 4552            Blocks: 16         IO Block: 4096   regular
file
Device: 801h/2049d      Inode: 30669       Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/  root)   Gid: (  0/  root)
Access: 2007-10-06 23:05:58.000000000 +0800
Modify: 2007-10-05 10:00:12.000000000 +0800
Change: 2007-10-05 10:00:12.000000000 +0800
```

Como Ler Atributos - Linux

- Como usar a chamada stat()?
  - Ler man : man stat.

```
int main(int argc, char **argv) {
    struct stat file_stat;
    if( stat(argv[1], &file_stat) == -1)
        exit(1);

    printf("file size of argv[1] = %d\n",
            file_stat.st_size);
    return 0;
}
```

Dispositivo do arquivo
Número do i-node (qual arquivo do dispositivo)
Modo do arquivo (inclui informação de proteção)
Número de ligações para o arquivo
Identificação do proprietário do arquivo
Grupo ao qual pertence o arquivo
Tamanho do arquivo (em bytes)
Hora da criação
Hora do último acesso
Hora da última modificação

Tabela 10.10 Os campos retornados pela chamada de sistema *stat*.

Como Ler Atributos - Linux

- Outro uso interessante da chamada stat() :
  - Verificar quando um nome de arquivo aponta para um arquivo ou um diretorio;

macro

```
int main(int argc, char **argv) {
    struct stat file_stat;
    if( stat(argv[1], &file_stat) == -1)
        exit(1);

    if(S_ISDIR(file_stat.st_mode))
        printf("%s is a directory\n", argv[1]);
    if(S_ISREG(file_stat.st_mode))
        printf("%s is a regular file\n", argv[1]);

    return 0;
}
```

Arquivos: operações

Operações são oferecidas para armazenar e recuperar informações

Algumas chamadas são:

- Create: criar arquivo sem dados, definir atributos
- Delete: remover o arquivo e liberar espaço
- Open: necessário antes de usar, o sistema deve buscar atributos e localização dos dados
- Close: os acessos terminam, dados para o acesso não mais necessários, liberar espaço nas tabelas, finalizar escritas
- Read: leitura de dados a partir da posição atual, quantidade e local especificados

Arquivos: operações

Operações são oferecidas para armazenar e recuperar informações

Algumas chamadas são:

- Write: escrita de dados, a partir da posição atual, quantidade e local especificado (Append)
- Seek: indica posição dos dados (ponteiro do arquivo)
- Get attributes: leitura dos atributos de um arquivo, Ex. make
- Set attributes: altera valores dos atributos, Ex. Modo de proteção
- Rename: altera o nome do arquivo, é necessária?

Chamadas de sistemas de arquivo do Linux

Chamada de sistema	Descrição
fd = creat(nome, modo)	Uma maneira de criar um novo arquivo
fd = open(arquivo, como, ...)	Abre um arquivo para leitura, escrita ou ambos
s = close(fd);	Fecha um arquivo aberto
n = read(fd, buffer, nbytes)	Lê dados de um arquivo para um buffer
n = write(fd, buffer, nbytes)	Escreve dados de um buffer para um arquivo
posição = lseek(fd, deslocamento, de-onde)	Move o ponteiro do arquivo
s = stat(nome, &buf)	Obtém a informação de estado do arquivo
s = fstat(fd, &buf)	Obtém a informação de estado do arquivo
s = pipe(&fd[0])	Cria um pipe
s = fcntl(fd, comando, ...)	Trava de arquivo e outras operações

Tabela 10.9 Algumas chamadas de sistema relacionadas a arquivos. O código de retorno *s* é –1 se ocorrer algum erro; *fd* é um descritor de arquivo e *position* é um offset de arquivo. Os parâmetros são autoexplicativos.

Exemplo de um programa usando chamadas de sistemas 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 */
```

(***Continua***)

Exemplo de um programa usando chamadas de sistemas para arquivos

(*Continuação*)

```
    /* 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 */
}
```

Figura 4.3 Um programa simples para copiar um arquivo.

Diretórios

- Para controlar os arquivos os SA utilizam diretórios
- A maneira mais simples de um sistema de diretórios é ter um diretório contendo todos os arquivos: diretório-raiz
  - Comum nos primórdios

    (ex. CDC6600 - http://en.wikipedia.org/wiki/CDC_6600)
  - Projeto simples, usado em sistemas embarcados simples

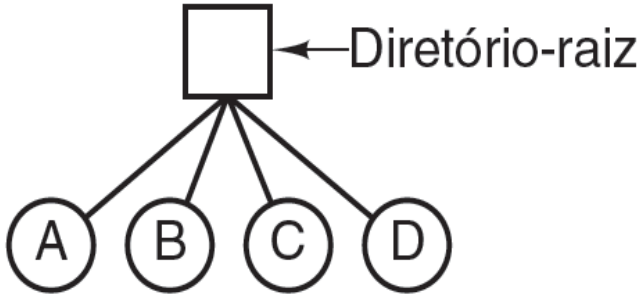


Figura 4.4 Um sistema de diretórios em nível único contendo quatro arquivos.

Diretórios

- Usuários com muitos milhares de arquivos, necessita facilidade de agrupamento
- Definição de uma hierarquia geral (arvore de diretórios) onde os usuários podem agrupar seus arquivos, podendo ter sua própria hierarquia

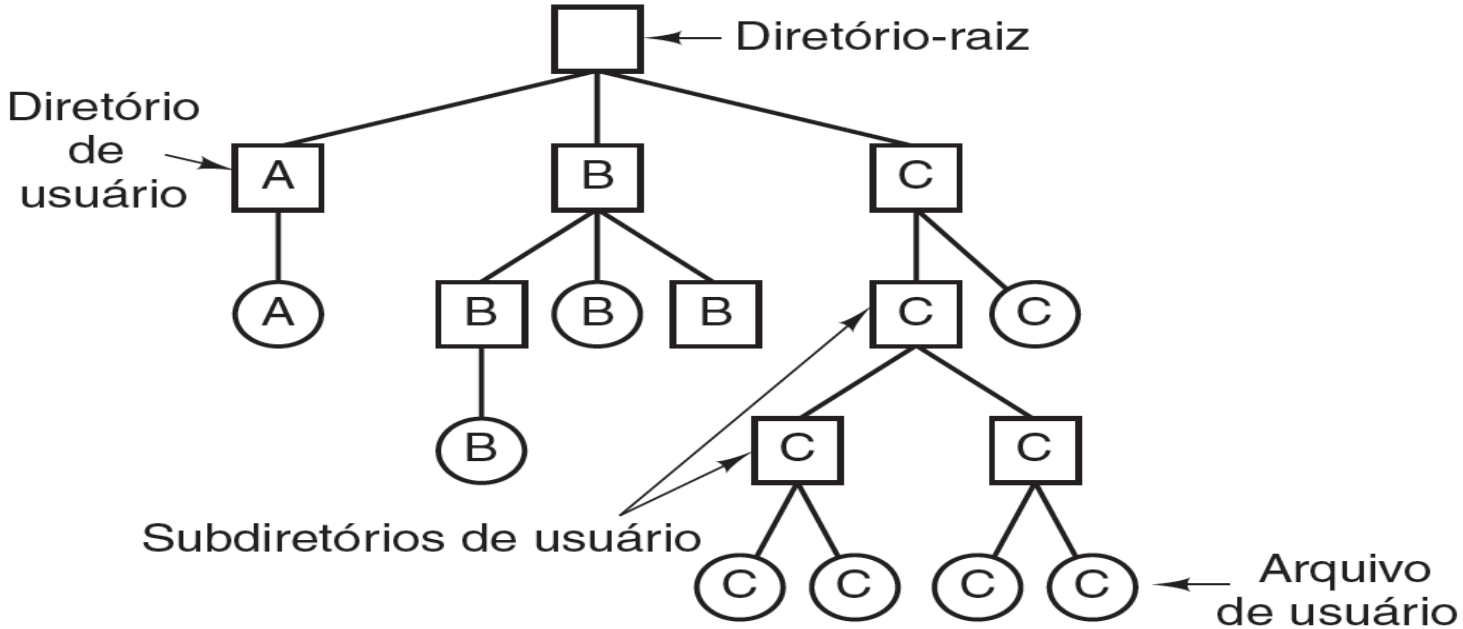


Figura 4.5 Um sistema hierárquico de diretórios.

Diretórios: nomes de caminho

- É preciso especificar o nome dos arquivos
  - Caminho absoluto,
  - caminho relativo,
  - diretório de trabalho
- Sistemas de diretório hierárquico suportam entradas especiais
  - '.' e
  - '..'

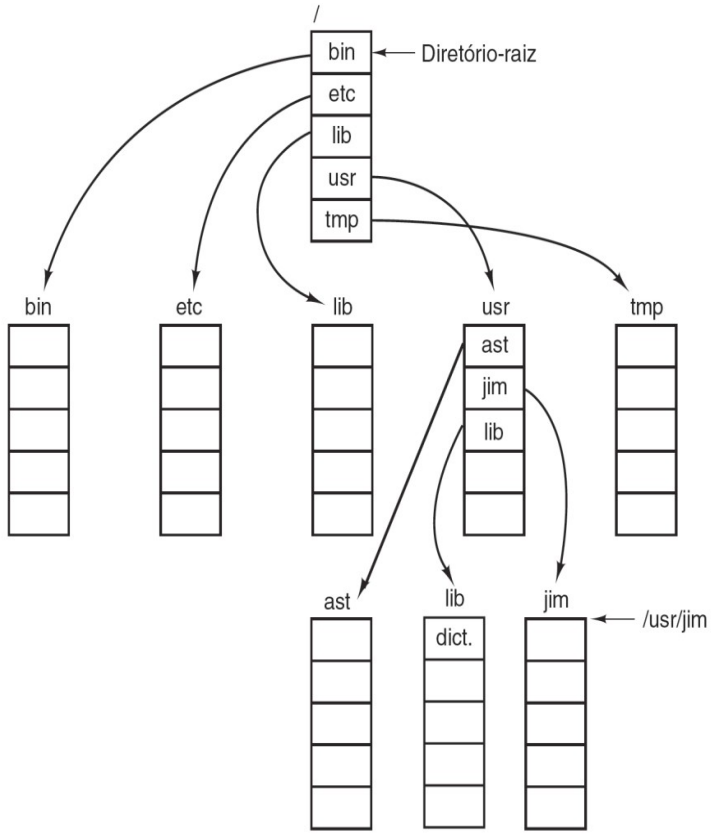


Figura 4.6 Uma árvore de diretórios UNIX.

O sistema de arquivos do Linux

Diretório	Conteúdos
bin	Programas binários (executáveis)
dev	Arquivos especiais para dispositivos de E/S
etc	Arquivos diversos do sistema
lib	Bibliotecas
usr	Diretórios de usuários

Tabela 10.8 Alguns diretórios importantes encontrados na maioria dos sistemas Linux.

Diretórios: operações

Operações são oferecidas para gerenciar diretórios variam de sistema para sistema.

Algumas chamadas são (UNIX):

- Create: criar um diretório, vazio, exceto (. , ..)
- Delete: remover um diretório, só vazio (. , .. é vazio)
- Opendir: permite a leitura, antes de ler o conteúdo de um diretório é preciso abrir
- Closedir: os acessos terminam,liberar espaço nas tabelas
- Readdir: leitura da próxima entrada de um diretório aberto, em formato padronizado (read)
- Rename: mudar nome

Chamadas de sistemas de arquivo do Linux

Chamada de sistema	Descrição
s = mkdir(caminho, modo)	Cria um novo diretório
s = rmdir(caminho)	Remove um diretório
s = link(caminho velho, caminho novo)	Cria uma ligação para um arquivo
s = unlink(caminho)	Remove a ligação para um arquivo
s = chdir(caminho)	Troca o diretório atual
dir = opendir(caminho)	Abre um diretório para leitura
s = closedir(dir)	Fecha um diretório
entradir = readdir(dir)	Lê uma entrada do diretório
rewinddir(dir)	Rebobina um diretório de modo que ele possa ser lido novamente

Tabela 10.11 Algumas chamadas de sistema relacionadas a diretórios. O código de retorno *s* é –1 se ocorrer algum erro; *dir* identifica uma cadeia de diretórios e *entradir* é uma entrada de diretório. Os parâmetros são autoexplicativos.

Diretórios: operações

Operações são oferecidas para gerenciar diretórios variam de sistema para sistema.

Algumas chamadas são (UNIX):

- Link: ligar um arquivo,a ligação possibilita um arquivo aparecer em mais de um diretório. A chamada especifica <nome> <caminho>. Este tipo de ligação, onde o contador de link é incrementado, é chamado de **ligação estrita** (hard link)
- Unlink: remover uma entrada de diretório, se o arquivo estiver presente em apenas 1 diretório o mesmo é removido do SA. Se estiver em vários apenas o nome do caminho especificado será removido. No UNIX remoção de arquivos é feita com esta chamada. Chamada especifica <nome> .

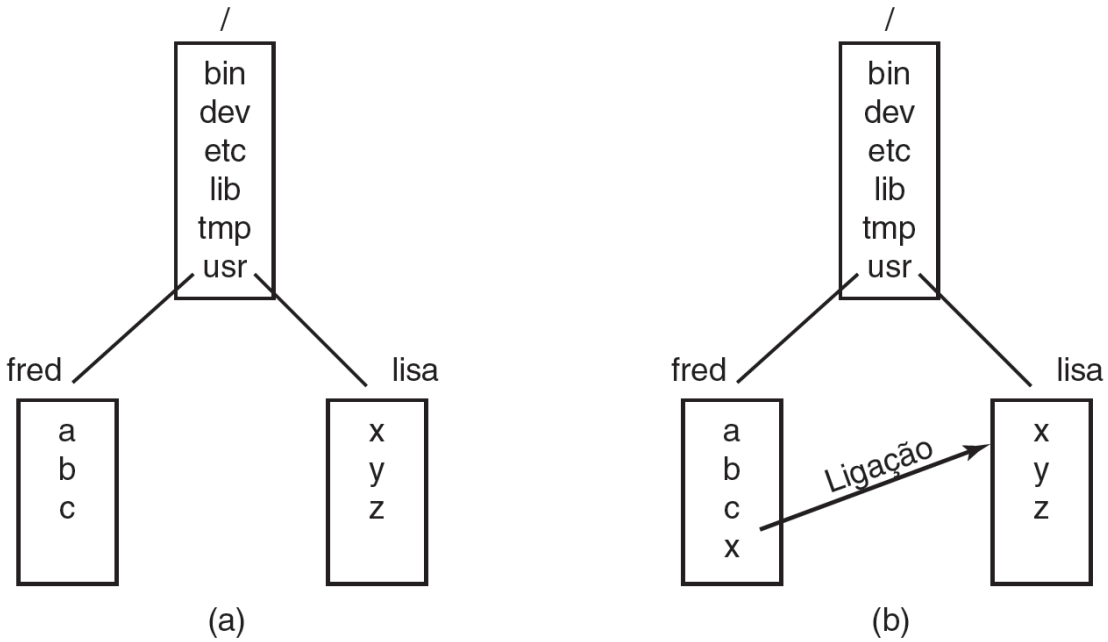


Figura 10.16 (a) Antes da ligação. (b) Após a ligação.

Diretórios: operações

Uma variação da ligação de arquivos é a **ligação simbólica** .

Neste caso um nome aponta para um arquivo que contêm o nome de um segundo.

- Open <nome> : segue o caminho e encontra o nome do segundo
  - Reinicializa o processo de localização
- Implementação menos eficiente