Capítulo 10: Interface do sistema de arquivos

Capítulo 10: Interface do sistema de arquivos

- Conceito de arquivo
- Métodos de acesso
- Estrutura de diretório
- Montando o sistema de arquivos
- Compartilhamento de arquivos
- Proteção

Objetivos

- Explicar a função dos sistemas de arquivos
- Descrever as interfaces dos sistemas de arquivo
- Discutir as escolhas do projeto do sistema de arquivos, incluindo métodos de acesso, compartilhamento de arquivos, bloqueio de arquivos e estruturas de diretório
- Explorar a proteção do sistema de arquivos

Conceito de arquivo

- Espaço de endereços lógicos contíguos
- Tipos:
  - Dados
    - numéricos
    - caractere
    - binários
  - Programa

Estrutura de arquivo

- Nenhuma – Seqüência de words, bytes
- Estrutura de registro simples
  - Linhas
  - Tamanho fixo
  - Tamanho variável
- Estruturas complexas
  - Documento formatado
  - Arquivo de carga relocável
- Pode simular dois últimos com o primeiro método, inserindo caracteres de controle apropriados
- Quem decide:
  - Sistema operacional
  - Programa

Atributos do arquivo

- **Nome** – somente informações mantidas em formato legível
- **Identificador** –tag exclusiva (número) identifica arquivo dentro do sistema de arquivos
- **Tipo** – necessário para sistemas que admitem diferentes tipos
- **Local** – ponteiro para local do arquivo no dispositivo
- **Tamanho** – tamanho de arquivo atual
- **Proteção** – controla quem pode realizar leitura, gravação, execução
- **Hora, data e identificação do usuário** – dados para proteção, segurança e monitoração de uso
- Informação sobre arquivos são mantidas na estrutura de diretório, que é mantida no disco

Operações do arquivo

- Arquivo é um **tipo de dado abstrato**
- **Criar**
- **Gravar**
- **Ler**
- **Reposicionar dentro do arquivo**
- **Excluir**
- **Truncar**
- *Open*(F_i) – procura a estrutura de diretório no disco para entrada F_i, e move conteúdo da entrada para memória
- *Close* (F_i) – move o conteúdo da entrada F_i na memória para a estrutura de diretório no disco

Abrir arquivos

- Vários dados são necessários para gerenciar arquivos abertos:
  - Ponteiro de arquivo: ponteiro para último local de read/write, por processo que tem o arquivo aberto
  - Contagem de arquivos abertos: contador do número de vezes que um arquivo está aberto – para permitir a remoção de dados da tabela de arquivos abertos quando últimos processos a fecham
  - Local do arquivo no disco: cache de informações de acesso a dados
  - Direitos de acesso: informação de modo de acesso por processo

Bloqueio de abertura de arquivo

- Fornecido por alguns sistemas operacionais e sistemas de arquivo
- Serve para mediar o acesso a um arquivo
- Obrigatório ou aconselhável:
  - **Obrigatório** – o acesso é negado dependendo dos bloqueios mantidos e requisitados
  - **Aconselhável** – os processos podem descobrir o status dos bloqueios e decidir o que fazer

Exemplo de bloqueio de arquivo –Java

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
      FileLock sharedLock = null;
      FileLock exclusiveLock = null;
      try {
      RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
            // get the channel for the file
            FileChannel ch = raf.getChannel();
            // this locks the first half of the file - exclusive
            exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
            /** Now modify the data . . . */
            // release the lock
            exclusiveLock.release();
```

Exemplo de bloqueio de arquivo – API Java (cont.)

```
// this locks the second half of the file - shared
  sharedLock = ch.lock(raf.length()/2+1, raf.length(),
        SHARED);
        /** Now read the data . . . */
        // release the lock
        exclusiveLock.release();
  } catch (java.io.IOException ioe) {
        System.err.println(ioe);
  }finally {
        if (exclusiveLock != null)
        exclusiveLock.release();
        if (sharedLock != null)
        sharedLock.release();
  }
     }
    }
```

Tipos de arquivo – Nome, extensão

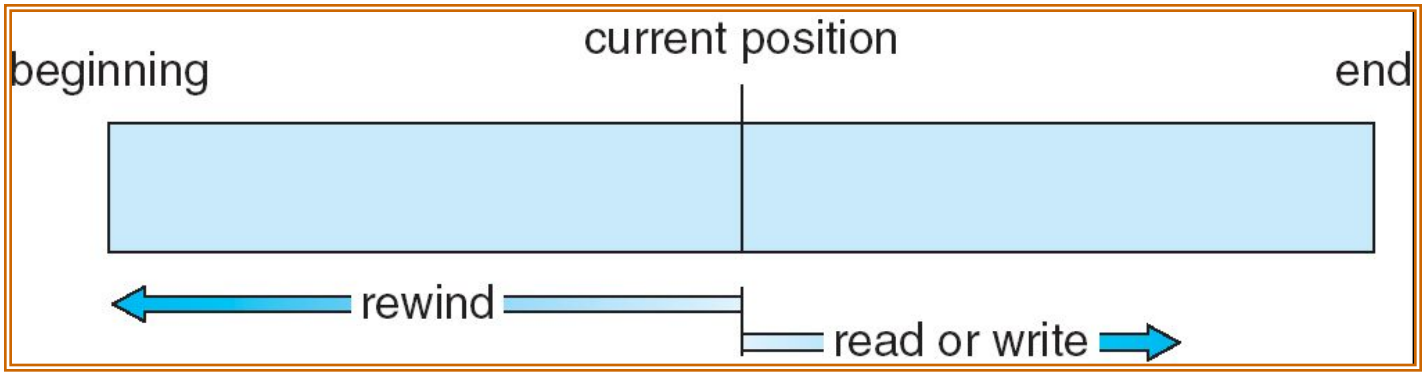
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Métodos de acesso

- **Acesso seqüencial**
  - lê próximo
  - grava próximo
  - reinicia
  - nenhuma leitura após última gravação
    - (regrava)
- **Acesso direto**
  - lê *n*
  - grava *n*
  - posiciona para *n*
    - lê próximo
    - grava próximo
  - regrava *n*

n = número de bloco relativo

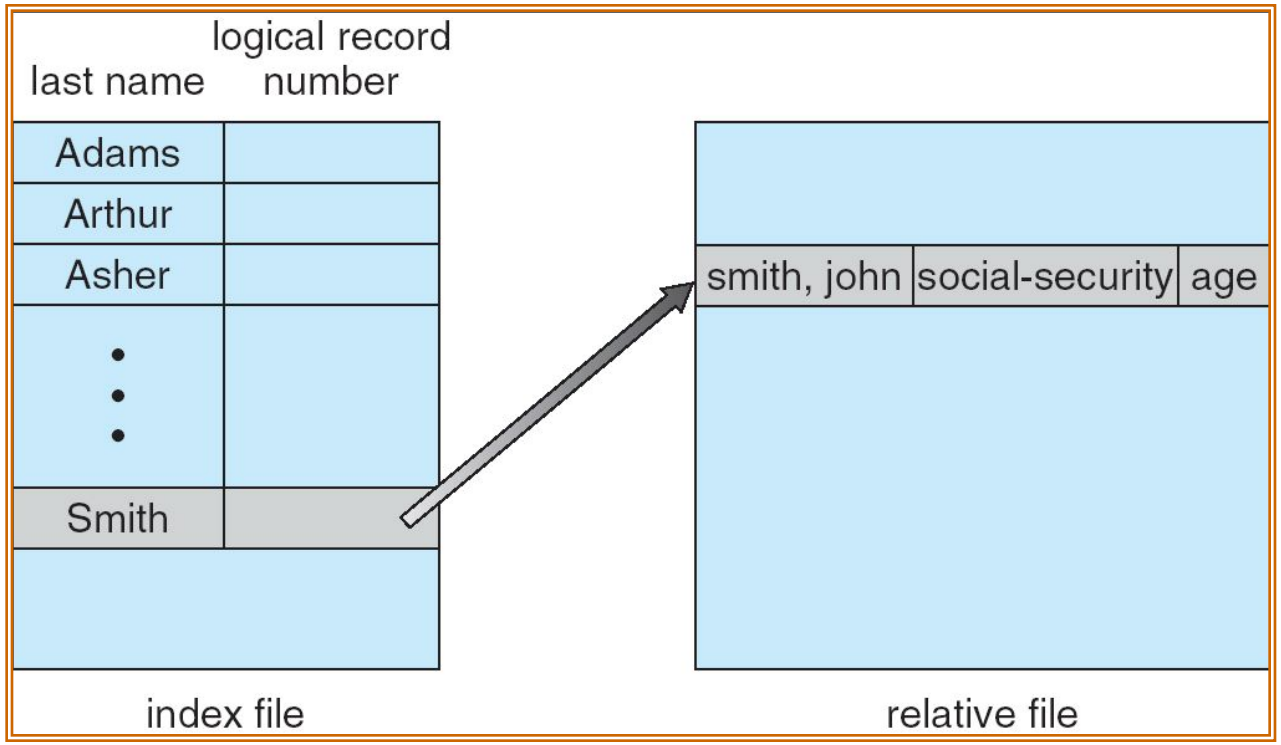
Arquivo de acesso seqüencial



Simulação de acesso seqüencial em um arquivo de acesso direto

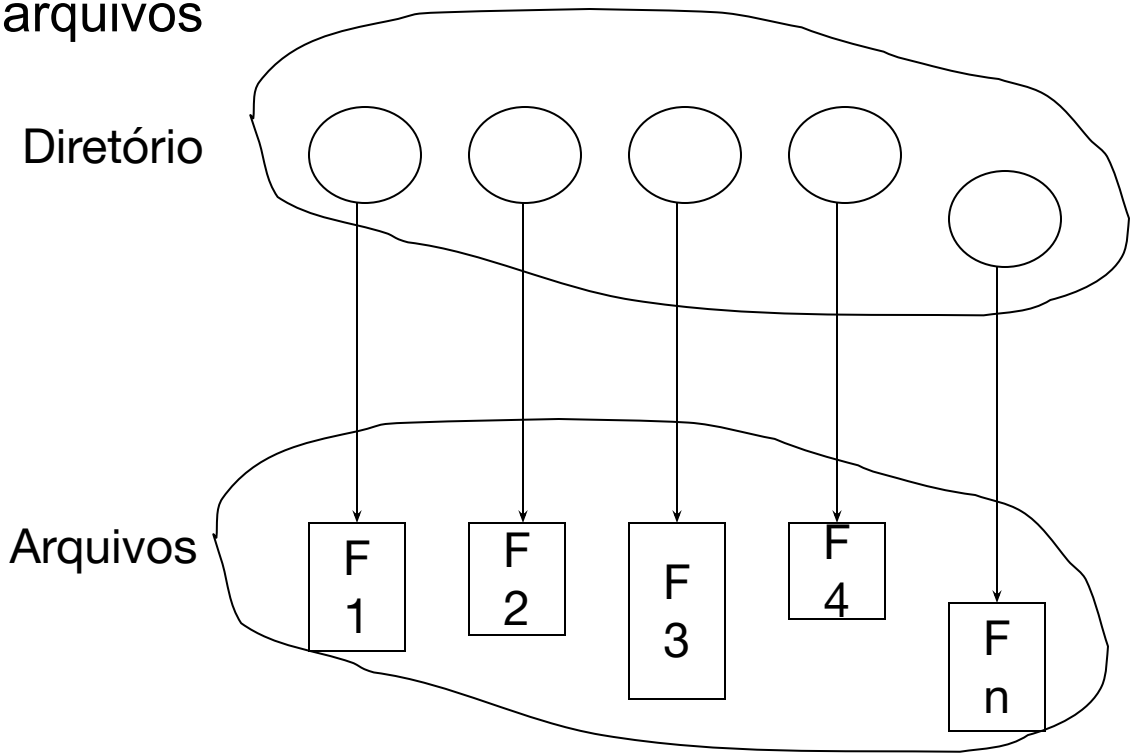
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Exemplo de arquivo de índice e relativo



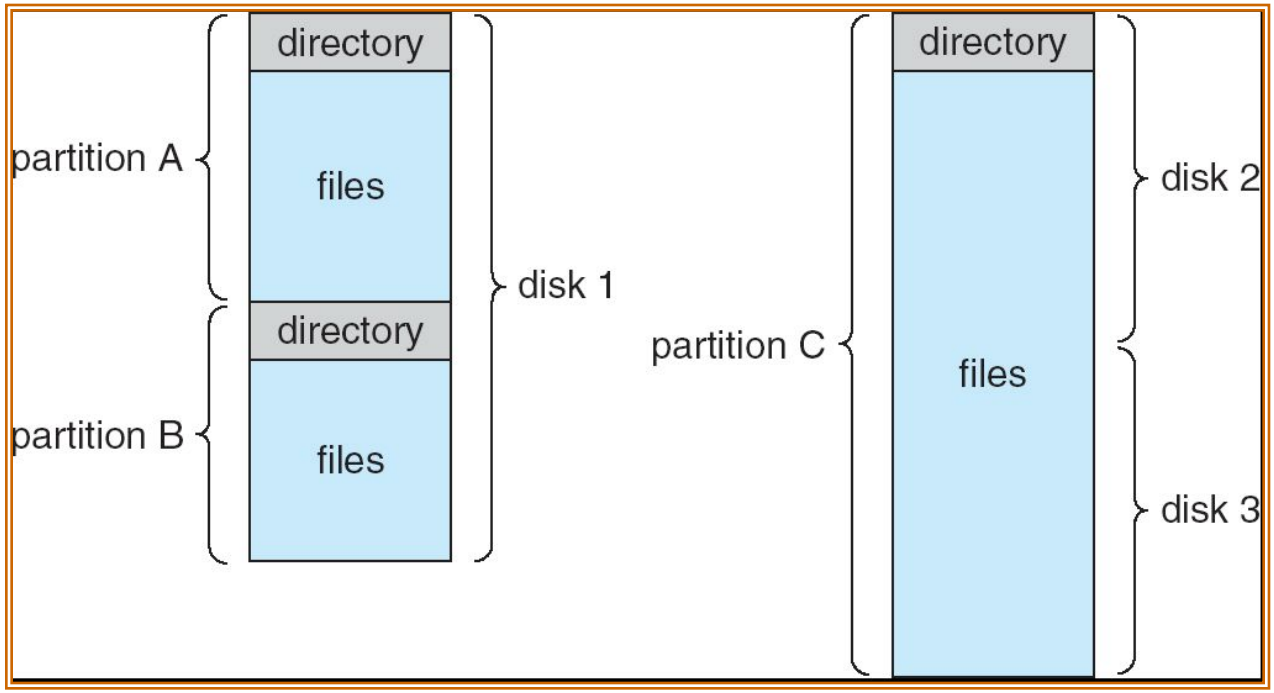
Estrutura de diretório

- Uma coleção de nós contendo informações sobre todos os arquivos



A estrutura de diretório e os arquivos residem no disco
Backups dessas duas estruturas são mantidas em fitas

Organização típica do sistema de arquivos



Operações realizadas no diretório

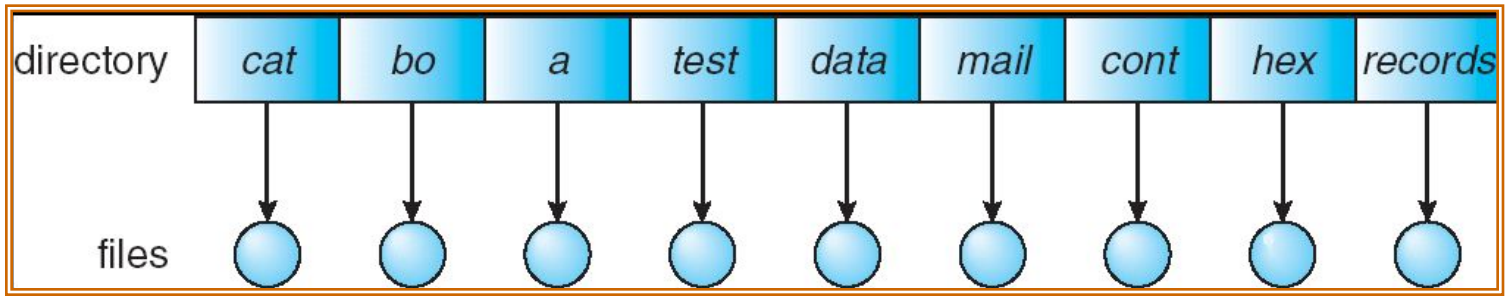
- Procurar um arquivo
- Criar um arquivo
- Excluir um arquivo
- Listar um diretório
- Renomear um arquivo
- Atravessar o sistema de arquivos

Organizar o diretório (logicamente) para obter

- Eficiência – localizando um arquivo rapidamente
- Nomeação – conveniente para usuários
  - Dois usuários podem ter o mesmo nome para diferentes arquivos
  - O mesmo arquivo pode ter vários nomes diferentes
- Agrupamento – agrupamento lógico de arquivos por propriedades, (por exemplo, todos os programas Java, todos os jogos, …)

Diretório de único nível

- Um único diretório para todos os usuários

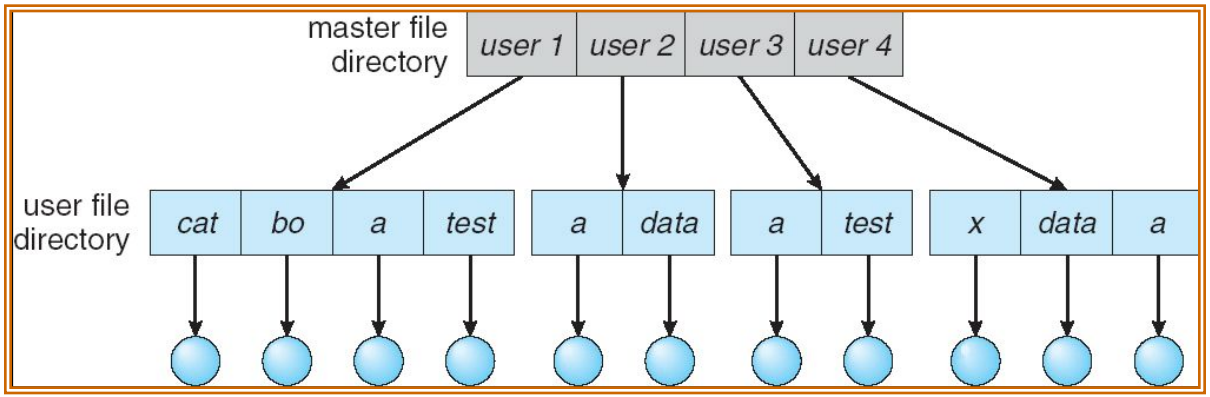


Problema de nomeação

Problema de agrupamento

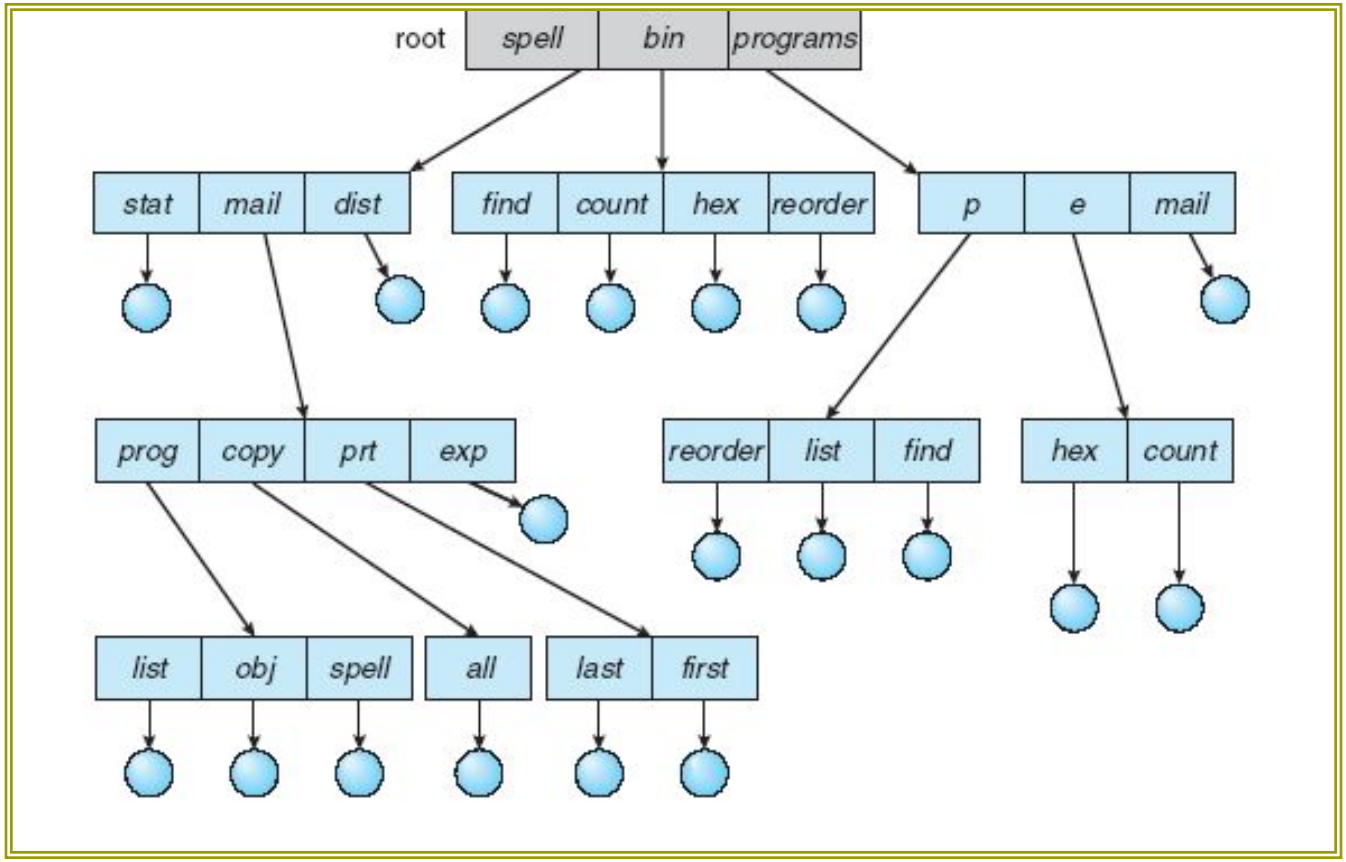
Diretório de dois níveis

- Diretório separado para cada usuário



- Nome do caminho
- Pode ter o mesmo nome de arquivo para usuário diferente
- Pesquisa eficiente
- Sem capacidade de agrupamento

Diretórios estruturados em árvore



Diretórios estruturados em árvore (cont.)

- Pesquisa eficiente
- Capacidade de agrupamento
- Diretório atual (diretório de trabalho)
  - cd /spell/mail/prog
  - lista de tipo

Diretórios estruturados em árvore (cont.)

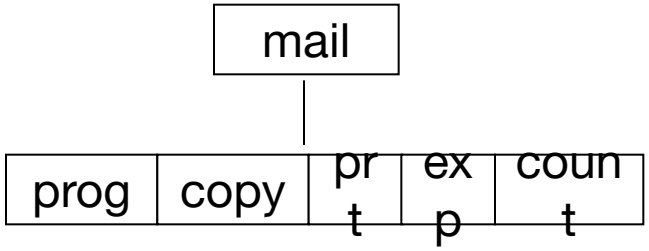
- Nome de caminho **absoluto** ou **relativo**
- A criação de um arquivo novo é feita no diretório atual
- Exclusão de um arquivo

  rm <nome de arquivo>
- A criação de um novo subdiretório é feita no diretório atual

  mkdir <nome diretório>

  Exemplo: se no diretório atual /mail

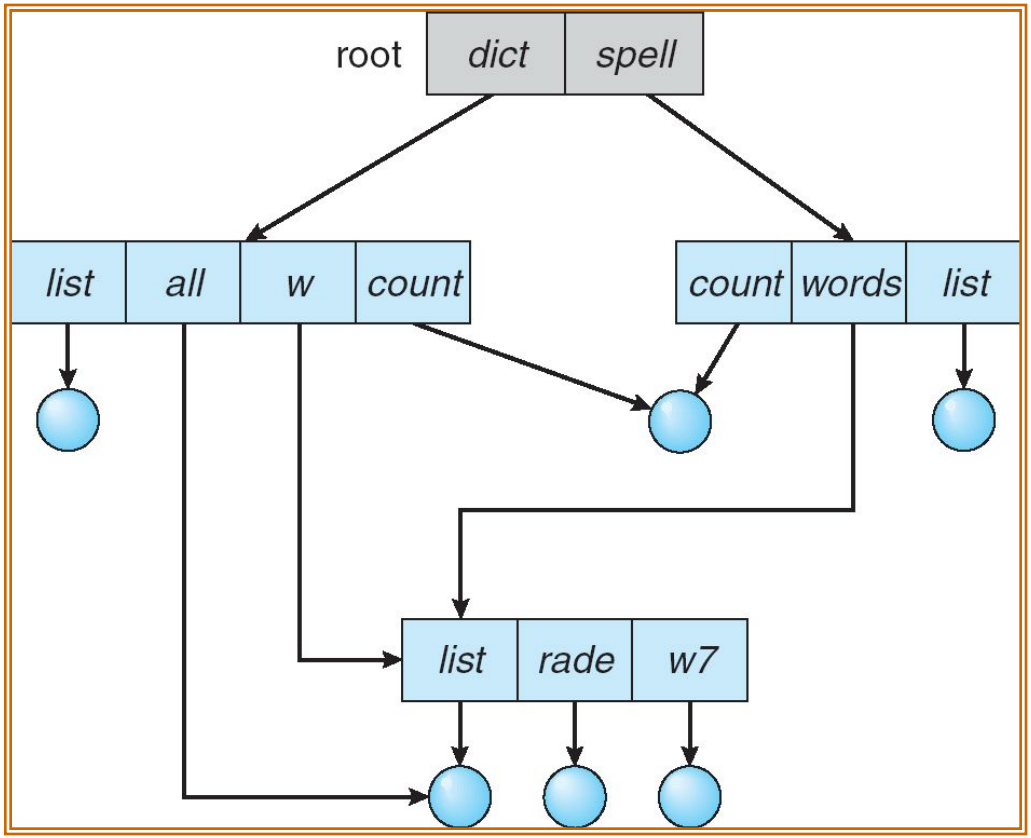
  mkdir count



Excluir “mail” ⇒ excluir a sub-ávore inteira iniciada com “mail”

Diretórios de grafo acíclico

- Têm subdiretórios e arquivos compartilhados

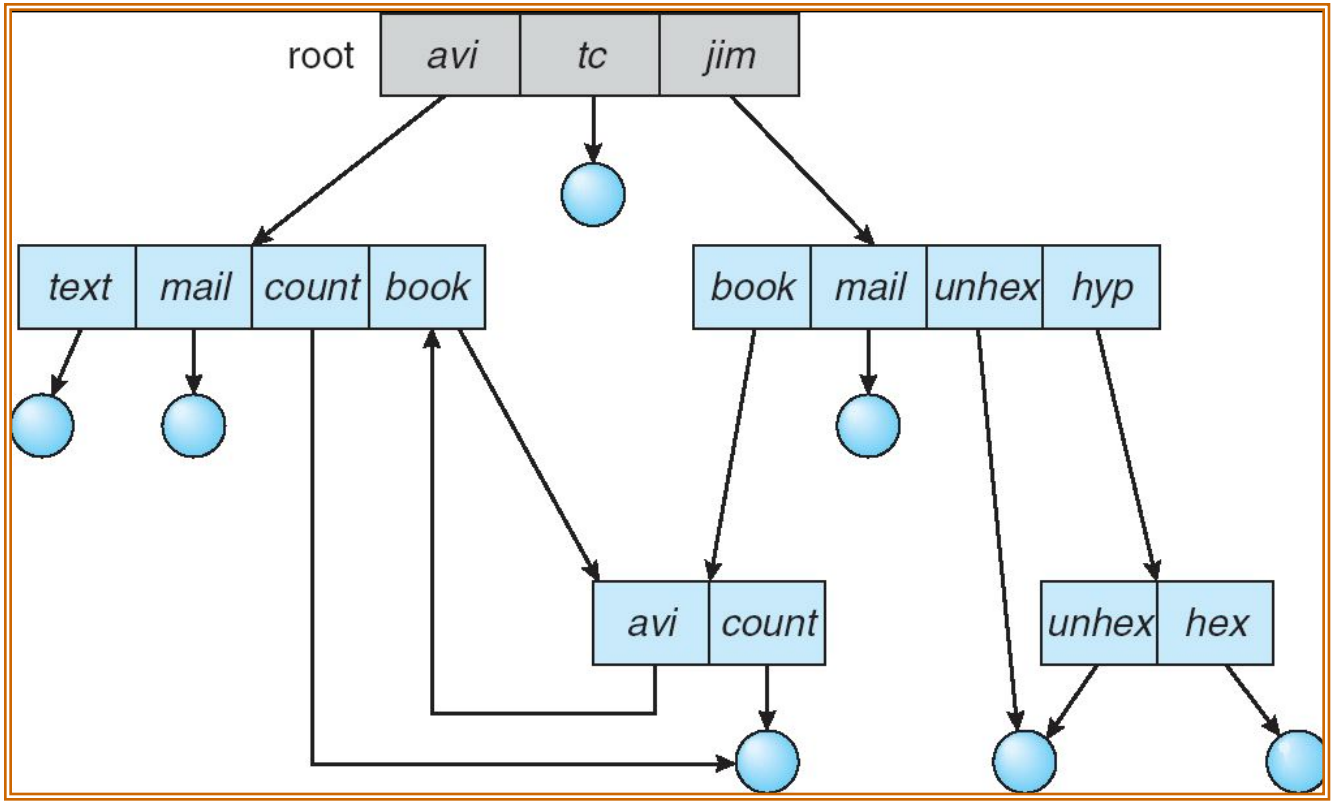


Diretórios de grafo acíclico (cont.)

- Dois nomes diferentes (aliasing)
- Se *dict* exclui *list* ⇒ ponteiro pendente

  Soluções:
  - Backpointers, e podemos excluir todos ponteiros Registros de tamanho variável
  - Backpointers usando uma organização de cadeia de margaridas
  - Solução entrada-mantém-contador
- Novo tipo de entrada de diretório
  - **Link** – outro nome (ponteiro) para arquivo existente
  - **Resolve o link** – siga ponteiro para localizar o arquivo

Diretório grafo geral



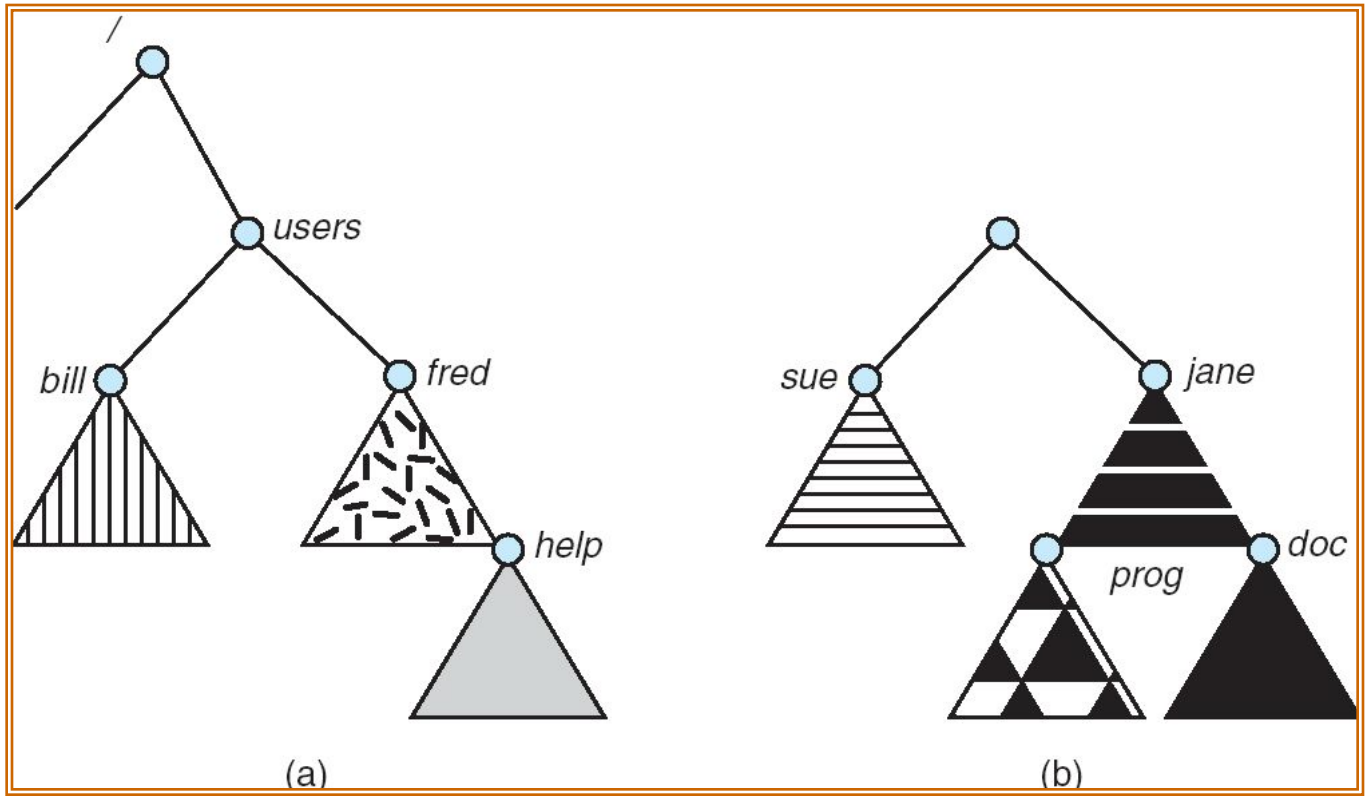
Diretório grafo geral (cont.)

- Como garantimos nenhum ciclo?
  - Permitir apenas links para arquivo, e não subdiretórios
  - Toda vez que um novo link é acrescentado, use um algoritmo de detecção de ciclo para determinar se tudo está OK

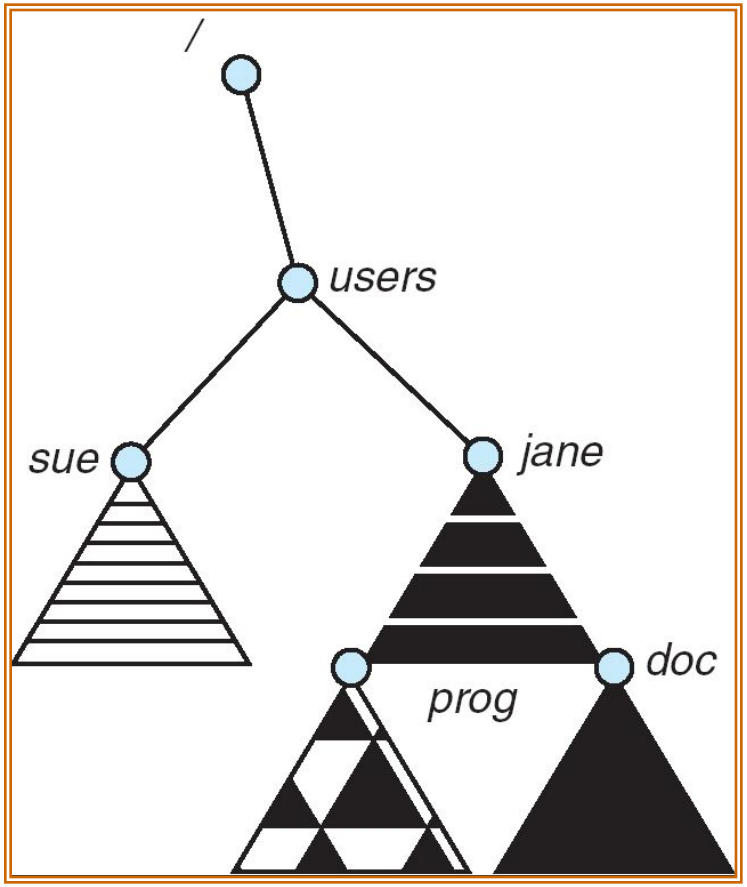
Montagem do sistema de arquivos

- Um sistema de arquivos precisa ser **montado** antes de poder ser acessado
- Um sistema de arquivos desmontado (p.e., Figura 11.11(b)) é montado em um **ponto de montagem**

Partição (a) existente (b) desmontada



Ponto de montagem



Compartilhamento de arquivos

- Compartilhamento de arquivos em sistemas multiusuário é desejável
- Compartilhamento pode ser feito por um esquema de **proteção**
- Em sistemas distribuídos, arquivos podem ser compartilhados por uma rede
- Network File System (NFS) é um método comum de compartilhamento de arquivo distribuído

Compartilhamento de arquivos – múltiplos usuários

- **User IDs** identificam usuários, permitindo que proteções e permissões sejam feitas por usuário
- **IDs de grupo** permitem que usuários estejam em grupo, permitindo direitos de acesso em grupo

Compartilhamento de arquivos – Sistemas de arquivo remotos

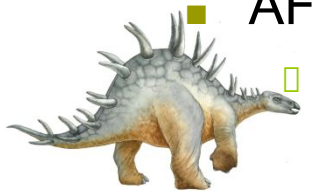
- Usa redes para permitir acesso do sistema de arquivos entre sistemas
  - Manualmente por programas como FTP
  - Automaticamente por **sistemas de arquivo distribuídos**
  - Semi-automaticamente pela **World Wide Web**
- Modelo **cliente-servidor** permite que clientes montem sistemas de arquivo remotos por servidores
  - Servidor pode atender múltiplos clientes
  - Identificação de cliente e usuário no cliente é insegura ou complicada
  - **NFS** é o protocolo padrão de compartilhamento de arquivos cliente-servidor no UNIX
  - **CIFS** é protocolo padrão do Windows
  - Chamadas padrão de arquivo do sistema operacional são traduzidos para chamadas remotas
- Distributed Information Systems **(serviços de nome distribuídos)** como LDAP, DNS, NIS, Active Directory implementam acesso unificado às informações necessárias para computação remota

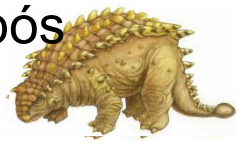
Compartilhamento de arquivos – modos de falha

- Sistemas de arquivo remotos acrescentam novos modos de falha, devido à falha na rede, falha no servidor
- Recuperação de falha pode invocar informação de estado sobre status de cada requisição remota
- Protocolos sem estado, como NFS, incluem toda a informação em cada requisição, permitindo a recuperação fácil, porém com menos segurança

Compartilhamento de arquivos – Semântica de consistência

- **Semântica de consistência** especifica como múltiplos usuários devem acessar um arquivo compartilhado simultaneamente
  - Semelhante aos algoritmos de sincronismo de processo do Cap. 7
    - Tende a ser menos complexo, devido à E/S de disco e latência de rede (para sistemas de arquivo remotos)
  - Andrew File System (AFS) implementava semântica complexa de compartilhamento de arquivo remoto
  - Sistema de arquivos do Unix (UFS) implementa:
    - Gravações em um arquivo aberto, visível imediatamente a outros usuários do mesmo arquivo aberto
    - Compartilha ponteiro de arquivo para permitir que múltiplos usuários leiam e gravem simultaneamente
  - AFS tem semântica de sessão
    - Gravações visíveis apenas a sessões começando após o arquivo ser fechado





Proteção

- Owner/creator do arquivo deve ser capaz de controlar:
  - o que pode ser feito
  - por quem
- Tipos de acesso
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Listas e grupos de acesso

- Modo de acesso: read, write, execute
- Três classes de usuários

  RWX

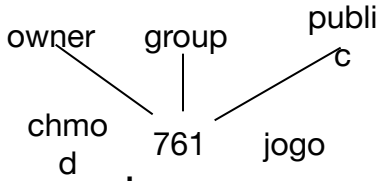
  a) **acesso owner** 7 ⇒ 1 1 1

  RWX

  b) **acesso group** 6 ⇒ 1 1 0

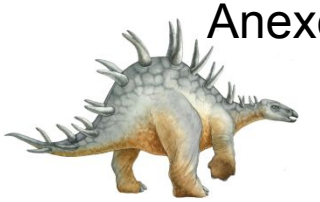
  RWX

  c) **acesso public** 1 ⇒ 0 0 1
- Peça ao gerente para criar um grupo (nome exclusivo), digamos G, e inclua alguns usuários ao grupo
- Para determinado arquivo (digamos, *jogo*) ou subdiretório, defina um acesso apropriado.

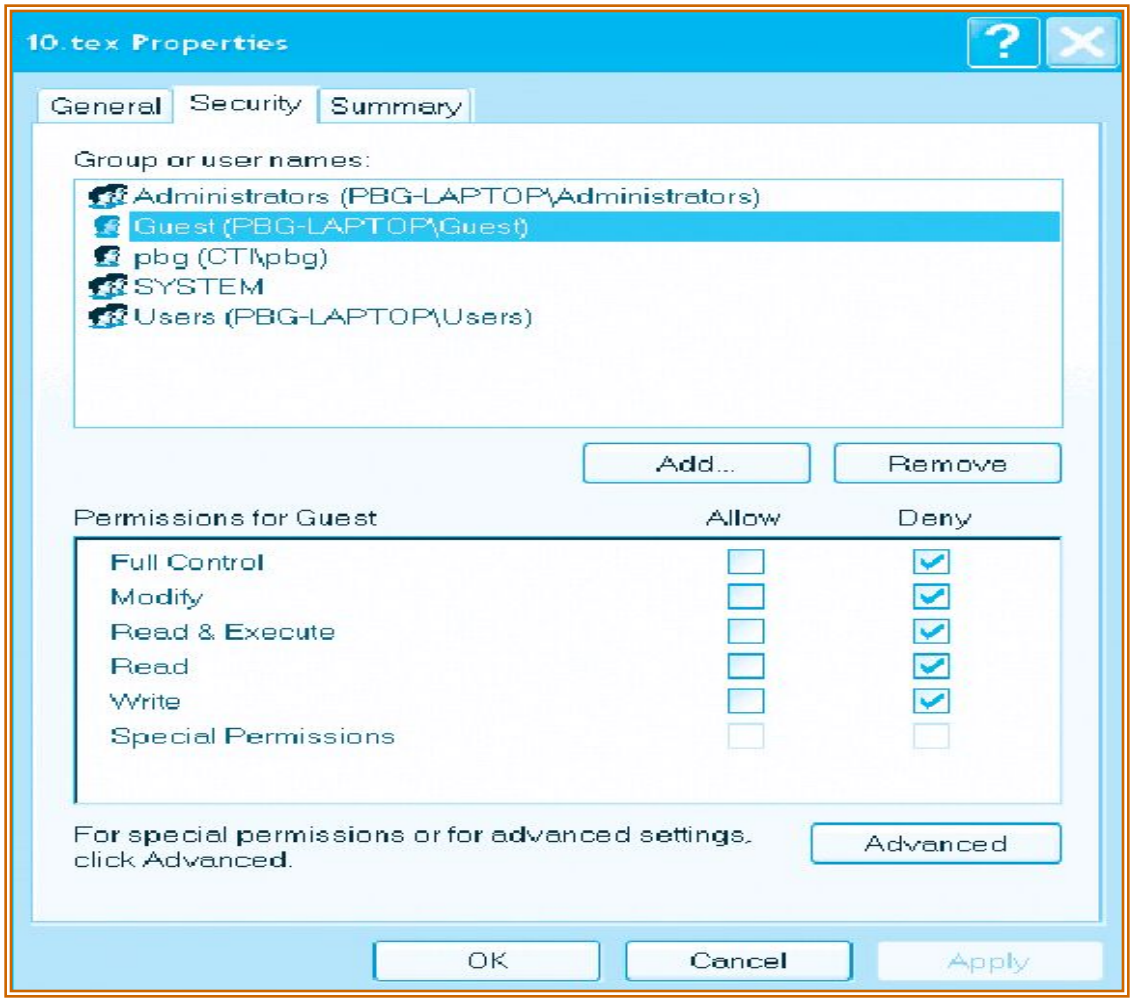


Anexe um grupo a um arquivo

chgrp G jogo



Gerenciamento de lista de controle de acesso no Windows XP



Exemplo de listagem de diretório no UNIX

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Final do Capítulo 10