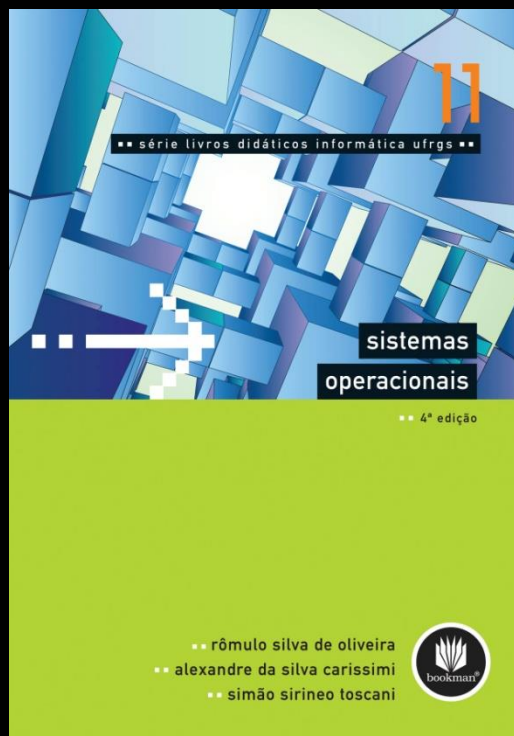


■ ■ série de livros didáticos informática ufrgs



bookman[®]
EMPRESA DO GRUPO ARTMED
www.bookman.com.br

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS



Sistemas Operacionais Cap 8 - Sistemas de Arquivos

Rômulo Silva de Oliveira
Alexandre da Silva Carissimi
Simão Sirineo Toscani

Introdução

- O sistema de arquivos é a parte mais visível do sistema operacional
- Cria um recurso lógico a partir de recursos físicos através de uma interface coerente e simples, fácil de usar
- Mecanismo para armazenamento e acesso a dados e a programas
- Duas partes básicas:
 - Arquivos
 - Armazenamento de dados e de programas
 - Diretórios
 - Organização e informações sobre arquivos

Objetivos do sistema de arquivos

- Fornecer mecanismos para usuários manipular arquivos e diretórios
- Garantir a validade e coerência de dados
 - Minimizar ou eliminar o risco de perda/alteração de dados
- Otimizar o acesso
- Fornecer suporte a outros sistemas de arquivos
- Suporte a vários usuários (multiprogramação)
 - Uso compartilhado (proteção e acesso concorrente)

Requisitos mínimos: ponto de vista do usuário

- Cada usuário deve ser capaz de:
 - Criar, apagar, ler e alterar arquivos
 - Controlar as permissões de acesso a seus arquivos
 - Nomear arquivos de forma simbólica
 - Estruturar os arquivos de forma a adequá-los a suas necessidades específicas
 - Criação de diretórios e subdiretórios
 - Realizar *back-ups* e recuperar arquivos em caso de problemas

Requisitos mínimos: ponto de vista do sistema

- O sistema operacional deve ser capaz:
 - Descrever a localização de todos os arquivos e de seus atributos
 - Via diretório
 - Gerenciar espaço físico do disco
 - Alocar blocos livres a arquivos em criação/expansão
 - Liberar blocos de arquivos removidos
 - Mecanismos para localizar eficientemente blocos (setores) que compõem arquivos

Conceitos básicos

- Arquivos
 - Recipientes que contêm dados
- Diretórios
 - Conjuntos de referências a arquivos
- Partição
 - Abstração que permite a partir do disco físico criar discos lógicos

Conceito de arquivo

- Informação pode ser armazenada em diferentes tipos de mídia
 - O sistema operacional deve oferecer uma visão uniforme da informação independente do dispositivo físico de armazenamento
 - Visão lógica é o arquivo
- Arquivos são mapeados para dispositivos físicos
- Arquivos possuem:
 - Nome
 - Atributos
 - Estrutura interna
 - Tipo
 - Método de acesso
 - Operações

Nomes de arquivos

- O sistema de arquivos define um espaço de nomes
 - Conjunto de regras e convenções para identificar simbolicamente um arquivo
- Variam de sistema para sistema
 - Distinção entre letras maiúsculas e minúsculas
 - Obrigatoriedade ou não de uma extensão
 - As vezes extensões são apenas convenções
 - Tamanho máximo de nome e da extensão (se houver)

Atributos de um arquivo

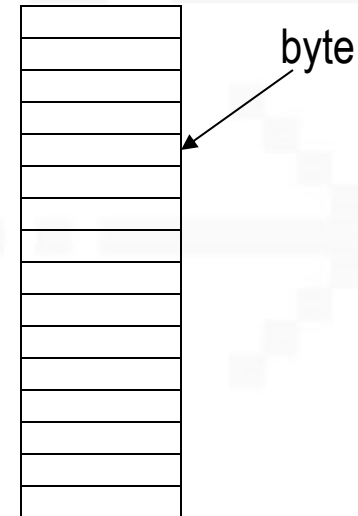
- Informações sobre arquivos
 - Nome: informação simbólica empregada para referenciar o arquivo
 - Tipo: binário, texto, executável, caracter, bloco
 - Localização: posição do arquivo em um determinado dispositivo E/S
 - Tamanho: número de *bytes* que compõem o arquivo
 - Proteção: controla acesso de leitura, escrita e execução ao arquivo
 - Hora e data de criação, identificação do usuário: informações destinadas à proteção, segurança e monitoração
- Varia de sistema operacional a sistema operacional
- Atributos são mantidos em uma estrutura a parte
 - Diretório

Estruturas de arquivos

- Sequência de bytes
- Sequência de registros
- Árvore

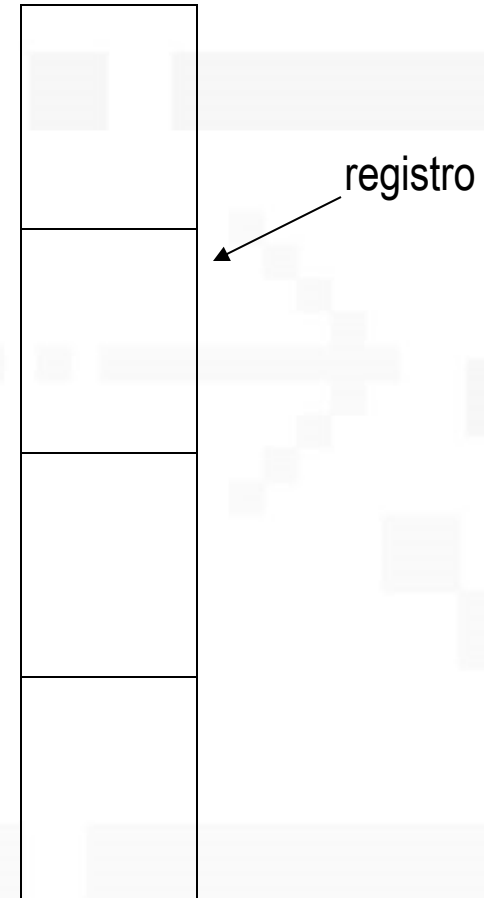
Seqüência de bytes

- Sistema operacional não “interpreta” o conteúdo do arquivo
 - Enxerga apenas *bytes*
- Interpretação é a nível do programa de usuário
- Vantagem:
 - Flexibilidade
 - Exemplo: Arquivos de imagem (JPEG, PNG), arquivos executáveis, arquivos de áudio (MP3).



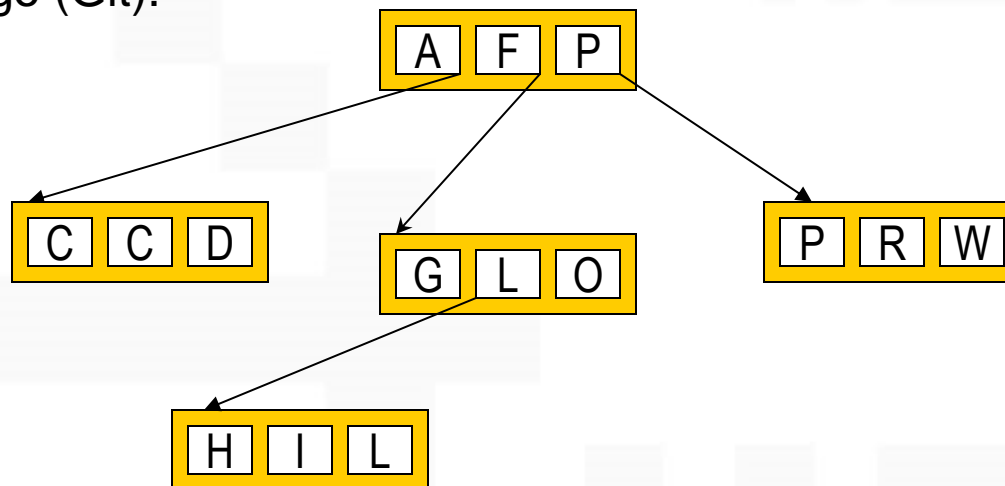
Seqüência de registros

- Arquivo é “interpretado” como uma seqüência de registros, isto é
 - Tamanho fixo
 - Estrutura interna
- Operações leem/escrevem registros
- Exemplo: Bancos de dados relacionais, arquivos CSV, planilhas eletrônicas.



Árvore

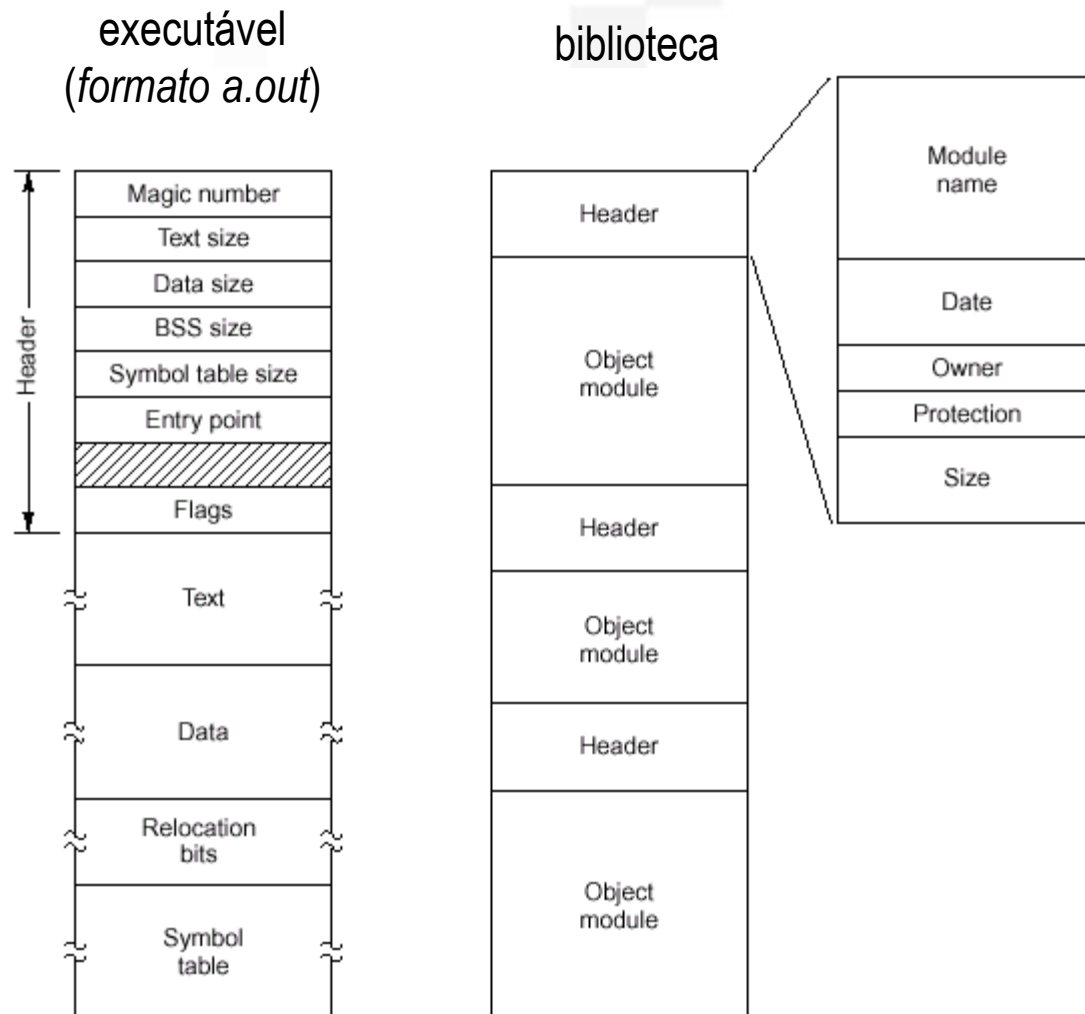
- Conjunto de registros não necessariamente de mesmo tamanho
 - Possuem um campo de acesso (chave)
- Comum em *mainframes*
 - Método ISAM (*Indexed Sequential Access Method*)
 - Exemplo: Sistemas de arquivos, sistemas de versionamento de código (Git).



Tipos de arquivo

- Sistema operacional suporta vários tipos de arquivos
- Tipos comuns:
 - Regular
 - Arquivos de dados em ASCII e binário
 - Diretório
 - Arquivos que mantêm a estrutura (organização) do sistema de arquivos
 - Arquivos especiais de caracter/bloco
 - Vinculados a dispositivos de entrada e saída

Exemplos de arquivos UNIX



Métodos de acesso

- Forma pela qual o conteúdo de um arquivo é acessado
- Métodos elementares de acesso:
 - Acesso sequencial
 - Acesso relativo

Acesso sequencial

- Acesso a um arquivo é feito através de primitivas (chamadas de sistema) do tipo *read* e *write*
- Cada chamada de sistema *read* retorna ao processo os dados seguintes àqueles que foram lidos na chamada anterior
- Método não adequado a todas aplicações
 - e.g.: acesso e atualização a cadastros de funcionários, *backup*, *log*

Acesso relativo

- Provê uma chamada de sistema específica para indicar o ponto em que um arquivo deve ser lido/escrito
- Implementado através da abstração de “posição corrente no arquivo”

Outros tipos de acesso

- Os métodos sequenciais e relativos não resolvem todos os tipos de acesso
 - e.g.: localizar um registro a partir do conteúdo
- Necessidades de métodos de acesso mais sofisticados, tais como sequencial indexado, indexado, direto, *hash*, etc.
 - Normalmente implementados por programas específicos
 - Baseados nos métodos de acesso sequencial e relativo

Operações básicas sobre arquivos

- Arquivo é um tipo abstrato de dados sobre o qual se pode efetuar uma série de operações
 - Criação (*create*)
 - Escrita (*write*) e leitura (*read*)
 - Reposicionamento em um ponto qualquer do arquivo (*file seek*)
 - Remoção (*delete*)
 - Abertura (*open*) e encerramento (*close*)
 - Adicionalmente: truncagem (*truncate*); renomeação (*rename*); *appending*, etc.
- Geralmente correspondem a chamadas de sistema
 - Operações mais complexas podem ser criadas utilizando-se das operações básicas

Controle de acesso

- Importante controlar o acesso aos arquivos devido a questões de segurança e de confidencialidade
- Objetivo é evitar acessos indevidos a arquivos
- Baseado na identificação dos usuários
 - Sistema de autenticação padrão (*login name* + senha)
 - Usuários possuem direitos de acessos
- Solução típica:
 - Lista de acesso e grupo

Listas de acesso

- Consiste em associar a cada arquivo e/ou diretório uma lista de acesso que determina que tipos de acessos são permitidos para cada usuário
- Maior inconveniente é o tamanho da lista
- Uma solução consiste em:
 - Criar classes de usuários
 - e.g.: proprietário, grupo, universo
 - Tipos de acessos
 - e.g: *read, write, modify, execute*

Exemplo: UNIX

- Cada objeto oferece 3 *bits* (rwx) para três domínios diferentes: proprietário, grupo e outros
- Problema de flexibilidade
 - Quando um usuário pertence a vários grupos ele é identificado por um grupo primário e o arquivo (/etc/groups) mantém todos os grupos a que ele pertence

```
rwx r-- r--      1 mary staff      214056   May 30 22:19 windbind.pdf
```

Outra abordagem: senhas

- Associar uma senha a cada arquivo
 - A grande desvantagem é o número de senhas
- Declaração de compartilhamento de arquivo e/ou subdiretório
 - Esquema utilizado pelo *Macintosh* e pelo *Windows*

Implementação de arquivos

- Arquivos são implementados através da criação, para cada arquivo no sistema, de uma estrutura de dados
- Descritor de arquivo é um registro que mantém informações sobre o arquivo
- Informações típicas (atributos):
 - Nome do arquivo
 - Tamanho em *bytes*
 - Data e hora da criação, do último acesso, da última modificação
 - Identificação do usuário que criou o arquivo
 - Listas de controle de acesso
 - Local do disco físico onde o conteúdo do arquivo foi colocado
 - Etc.

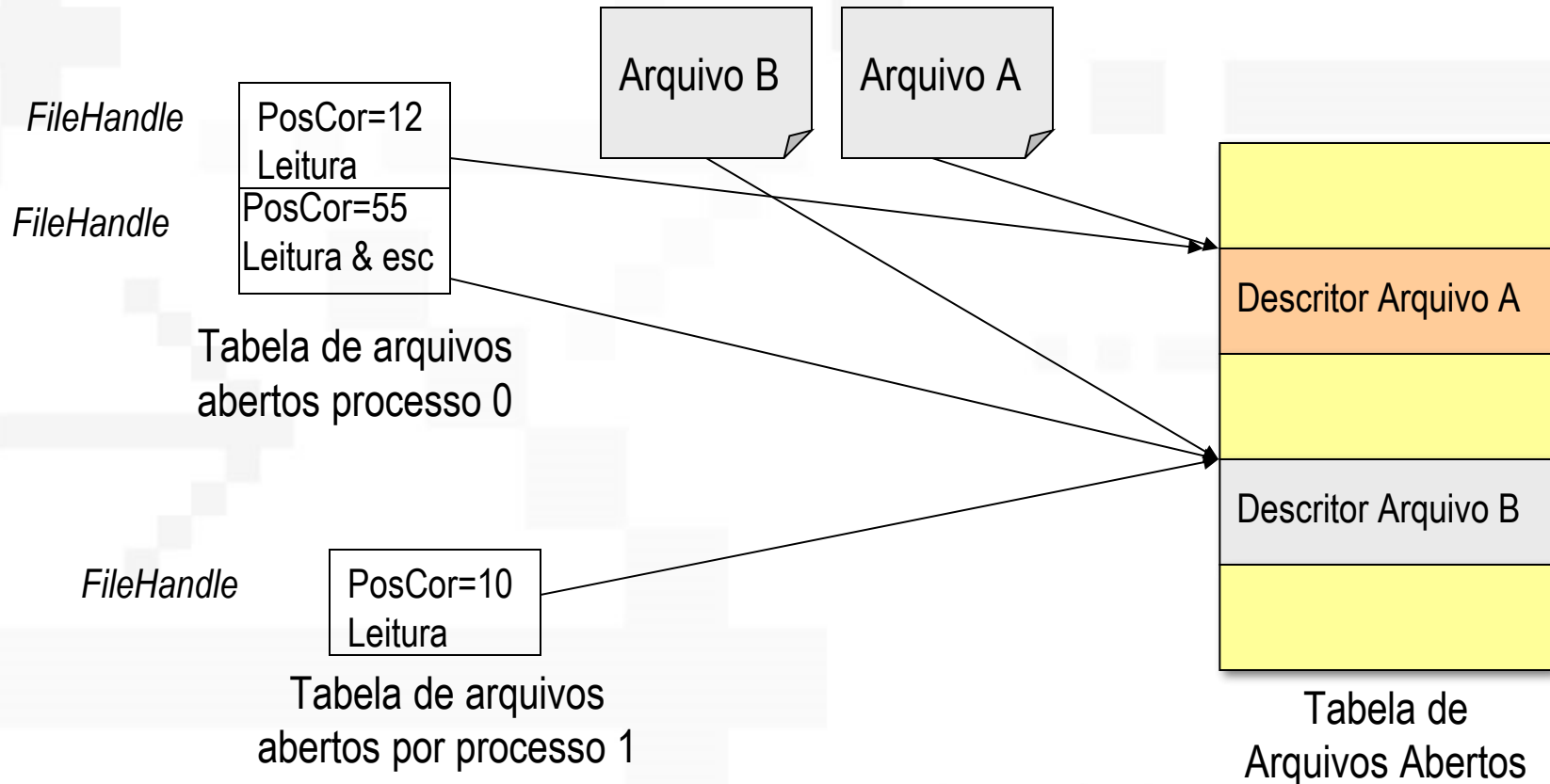
Tabelas de descritores de arquivos

- Descritores de arquivos são armazenados no próprio disco
 - Na realidade no mesmo disco lógico (partição)
- Problema de desempenho
 - Acesso ao disco para ler o descritor de arquivos é lento
 - Solução é manter descritor em memória enquanto o arquivo estiver em uso
 - Indicação se arquivo está em uso normalmente é feito pelo próprio usuário (aplicação) através de chamadas do tipo *open* e *close*
- Sistema de arquivos mantém os descritores de arquivos em memória em uma estrutura de dados do sistema operacional
 - Tabela de descritores de arquivos abertos (TDAA)

Tabelas de arquivos abertos por processo

- Informações relacionadas com arquivos são de dois tipos:
 - Não variam conforme o processo que está acessando o arquivo
 - e.g.: tamanho do arquivo
 - Dependem do processo que está acessando o arquivo
 - e.g.: posição corrente
- Informações dependentes do processo são armazenadas em uma tabela à parte mantida pelo processo (TAAP)
 - e.g.: posição corrente no arquivo, tipo de acesso e apontador para a entrada correspondente na TDAA
- Entrada na TAAP serve para referenciar o arquivo
 - *File handle*

Emprego conjunto das tabelas TAAP e TDAA



Leituras complementares

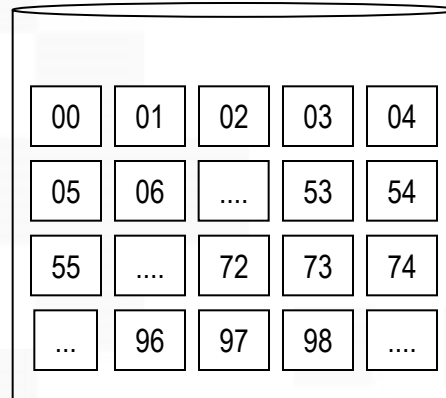
- R. Oliveira, A. Carissimi, S. Toscani; Sistemas Operacionais. Editora Bookman, 2010.
 - Capítulo 8, seções 8.1, 8.2, e 8.3
- A. Silberchatz, P. Galvin; Operating System Concepts. Addison-Wesley, (4th edition).
 - Capítulo 10, seções 10.1, 10.2, 10.4, e 10.5

Gerenciamento do dispositivo de armazenamento

- Desafio: arquivos devem ser armazenados no disco!!
 - Unidade de manipulação dos dados no dispositivo físico (bloco)
- Pontos a serem tratados:
 - Relação número de setores do disco que compõem um bloco
 - Não necessita ser 1:1
 - Alocação de blocos no disco
 - Recuperação de blocos liberados
 - Localização de dados no disco
- Existe uma relação entre a política de alocação com a política de gerência de espaço livre

Alocação do espaço em disco

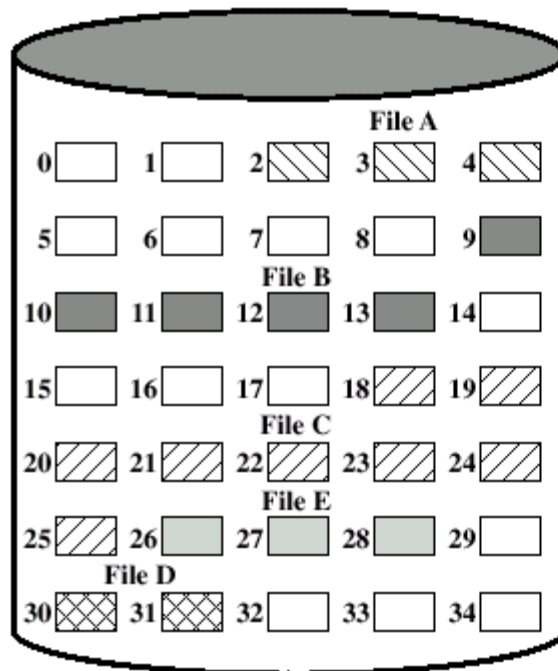
- Como alocar espaço em disco de forma que os arquivos sejam armazenados de forma eficiente e que permita acesso rápido
 - Alocar blocos livres suficientes para armazenar o arquivo
 - Blocos lógicos do disco são numerados sequencialmente
- Duas formas básicas:
 - Contígua (alocação contígua)
 - Não contígua (alocação encadeada e alocação indexada)



Alocação contígua

- Arquivo é uma sequência de blocos lógicos contíguos alocados no momento da criação
- Endereços no disco são lineares
 - bloco lógico i e $i+1$ são armazenados fisicamente em sequência
 - Reduz a necessidade de *seek* já que blocos estão na mesma trilha
 - No pior caso necessita apenas a troca de cilindro
- Arquivo é descrito através de uma entrada na forma:
 - Bloco físico inicial
 - Tamanho do arquivo em blocos

Esquema alocação contígua



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

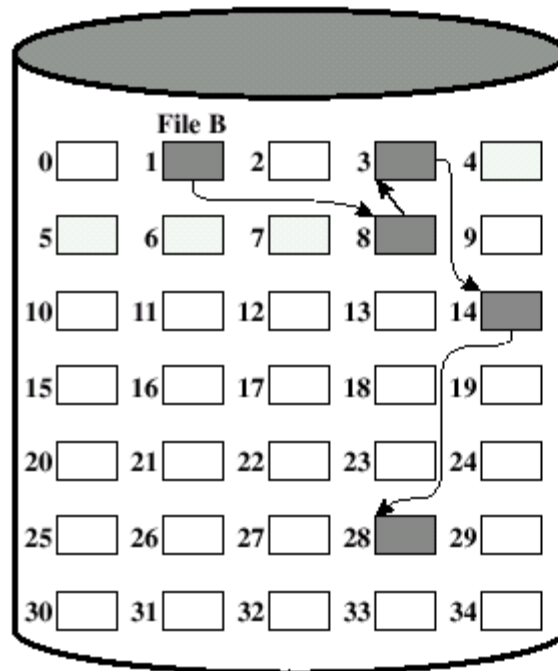
Problemas com alocação contígua

- Problema 1: encontrar espaço para um novo arquivo
 - Técnicas de gerência de memória
 - e.g.; *first-fit*, *best-fit*, *worst-fit*
 - Gera fragmentação externa
 - Necessidade de compactação
- Problema 2: determinar o espaço necessário a um arquivo
 - Arquivos tendem a crescer, e se não há espaço contíguo disponível?
 - Aborta execução do programa com erro
 - Recopia o programa para uma zona maior
 - Pré-alocar um espaço máximo para o arquivo
 - Fragmentação interna

Alocação encadeada

- Soluciona os problemas da alocação contígua
 - Relação a dimensionamento do tamanho e crescimento de arquivos
- Alocação é baseada em uma unidade de tamanho fixo (bloco lógico)
 - Análogo à paginação
- Arquivo é uma lista encadeada de blocos
 - Cada bloco contém um ponteiro para o próximo bloco
- Arquivo é descrito em uma entrada na forma:
 - Bloco inicial do arquivo
 - Bloco final do arquivo ou tamanho do arquivo em blocos

Esquema de alocação encadeada



File Allocation Table

File Name	Start Block	Length
...
File B	1	5
...

Prós e contras da alocação encadeada

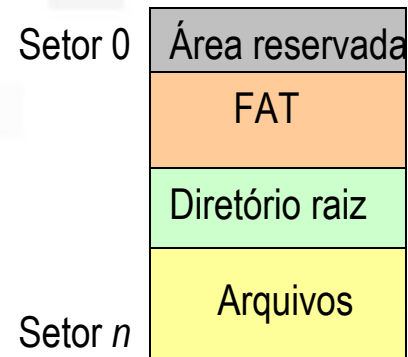
- Elimina a fragmentação externa
- Arquivos podem crescer indefinidamente
 - Não há necessidade de compactar o disco
- O acesso a um bloco i implica em percorrer a lista encadeada
 - Afeta o desempenho
 - Adequado para acesso sequencial a arquivos
- Confiabilidade
 - Erro provoca a leitura/escrita em bloco pertencente a outro arquivo

Exemplo: *File Allocation Table* (FAT)

- Variação de alocação encadeada
- FAT é uma tabela de encadeamento de blocos lógicos
 - Uma entrada na FAT para cada *bloco lógico* do disco (sistema de arquivos)
 - Composta por um ponteiro (endereço do bloco lógico)
 - Arquivo é descrito por uma sequência de entradas na FAT, cada entrada apontando para a próxima entrada

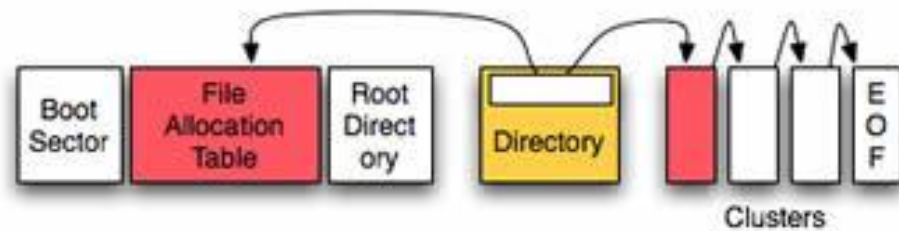
Sistema de arquivos FAT (MS-DOS)

- Organização lógica do disco:



- Diretório raiz possui tamanho fixo em função da capacidade do disco
 - Cada entrada possui 32 *bytes*
- Tamanho da *File Allocation Table* (FAT) é proporcional a capacidade do disco
- Alocação é baseada em *clusters* (bloco lógico)
 - 2ⁿ setores (depende da capacidade do disco)

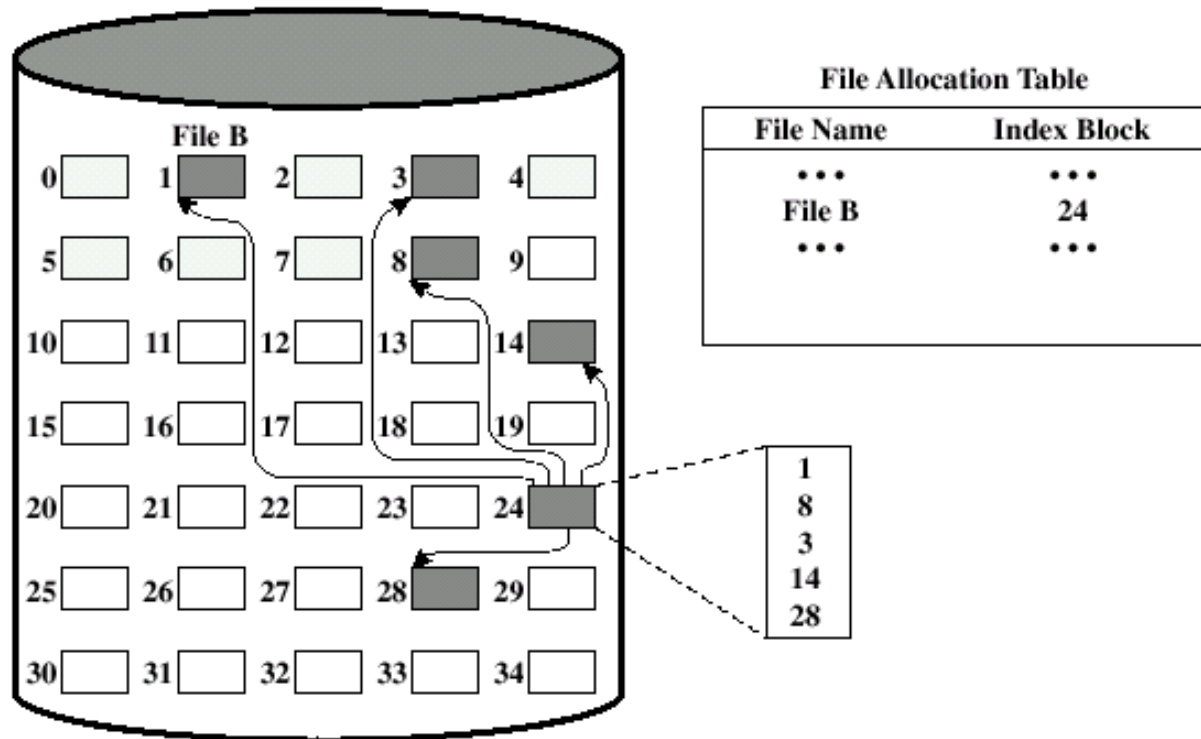
FAT FILE SYSTEM



Alocação indexada

- Busca resolver o problema de “ponteiros esparramados” pelo disco que a alocação encadeada provoca
- Mantém, por arquivo, um índice de blocos que o compõe
- O índice é mantido em um bloco
- Diretório possui um ponteiro para o bloco onde está o índice associado a um determinado arquivo

Esquema de alocação indexada

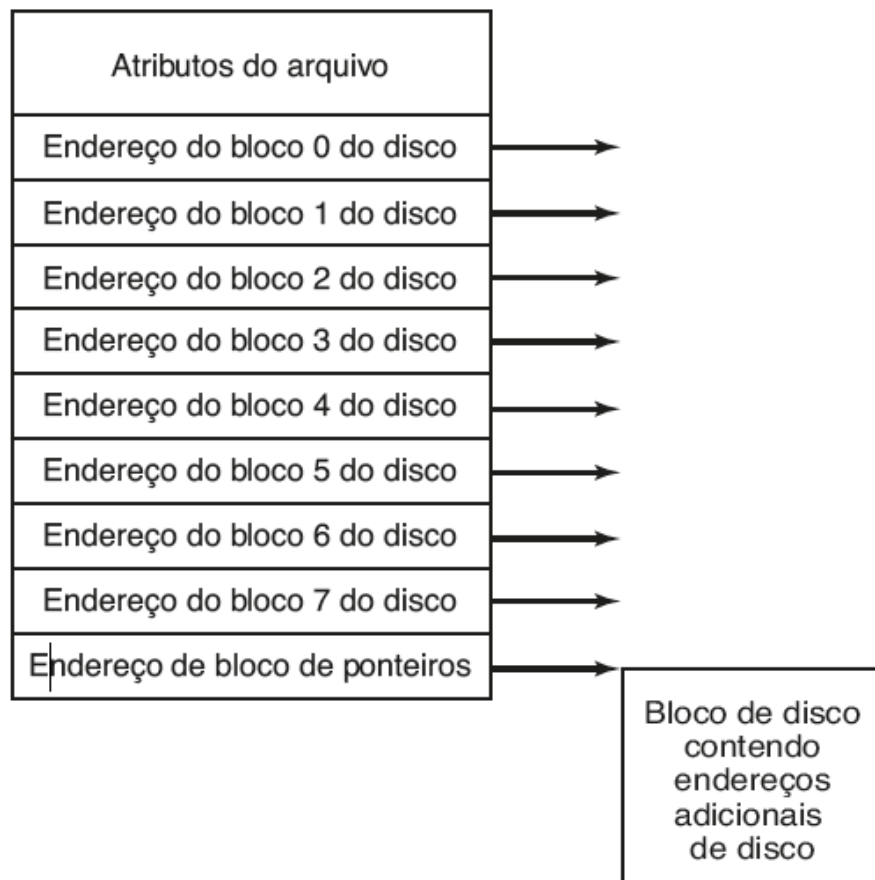


Prós e contras da alocação indexada

- Permite o acesso randômico a blocos independentes de sua posição relativa no arquivo
- Tamanho máximo do arquivo é limitado pela quantidade de entradas suportadas pelo bloco
 - Muito pequeno (limita tamanho do arquivo)
 - Muito grande (desperdiça espaço em disco)
- Solução é utilizar dois tamanhos de blocos, um para índice e outro para dados
 - e.g.: i-nodes e bloco de dados em sistemas UNIX

Exemplo: estrutura de *i-nodes* (UNIX)

Um exemplo de i-node.



Problema com os métodos de alocação não contígua

- Necessidade de acessar áreas específicas do disco para ler as informações de encadeamento
 - Quantidade de *seeks* depende do tipo da estrutura (FAT ou descritores)
- Solução é manter em memória
 - Tradicionais problemas de área de memória ocupada e de confiabilidade

Resumo dos tipos de alocação

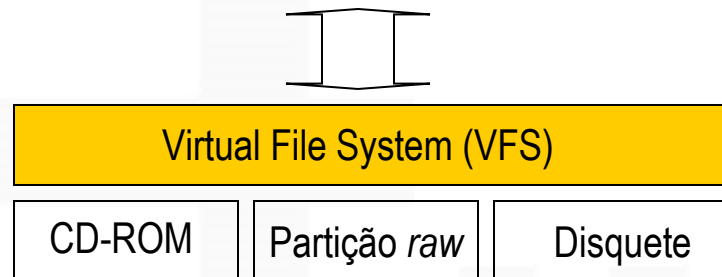
- Alocação contígua
 - Só armazena endereço do primeiro bloco
 - Acesso randômico é possível (bloco inicial + deslocamento)
 - Gera fragmentação externa no disco
- Alocação encadeada
 - Armazena endereço do primeiro bloco
 - Problema de desempenho (seek)
 - Não recomendado para acesso randômico
- Alocação indexada
 - Visa solucionar problemas dos tipos anteriores
 - Análise de desempenho (tamanho + tempo de acesso) é complexa
 - Depende da estrutura de índice e do tamanho de arquivo

Conclusão: qual o melhor método de alocação?

- Depende do tipo de acesso que o sistema faz a seus arquivos
 - Sequencial *versus* randômico
- Fator adicional:
 - Evolução tecnológica (novos *hardwares*) e de desempenho forçam a coexistência de diferentes sistemas de arquivos
- Necessidade de “fazer conviver” diferentes sistemas de arquivos em um mesmo computador
 - Suporte a múltiplos sistemas de arquivos

Suporte a múltiplos sistemas de arquivos

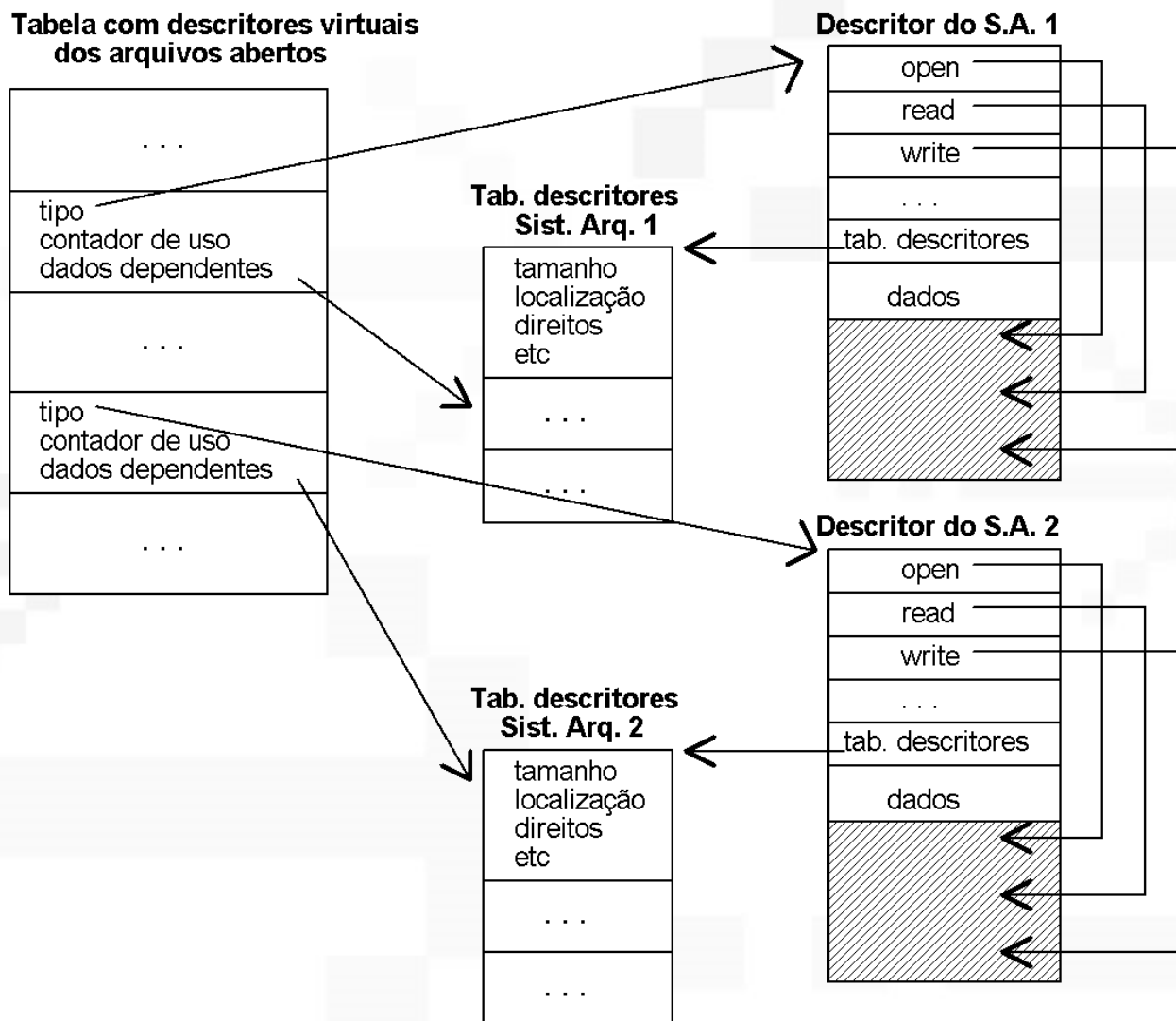
- Fazer com que o sistema operacional suporte diversos sistemas de arquivos diferentes simultaneamente
- Solução inspirada na gerência de periféricos
 - Parte independente do dispositivo
 - Serviços idênticos independente do tipo de sistema de arquivos
 - Parte dependente do dispositivo
 - Interface padrão



Implementação de múltiplos sistemas de arquivos

- Cada partição possui um único sistema de arquivos
- Tabela com descritores virtuais de arquivos abertos
 - Parte independente do sistema de arquivos
 - Uma entrada ocupada para cada arquivo aberto (descritor virtual)
- Descritor virtual
 - Informações comuns a todo sistema de arquivo (proteção, nro de acessos, ...)
 - Apontador para uma estrutura “Tipo do sistema de arquivos”
 - Apontador para o descritor do sistema de arquivos real
 - Lista de ponteiros para rotinas que implementam o código necessário à execução de uma dada chamada de sistema (*read*, *write*, *close*,...)
 - Informações sobre a gerência desse sistema de arquivos (blocos livres, ocupados, estrutura de diretórios, ...)

Múltiplos sistemas de arquivos: estrutura de dados



Organização da cache de disco

- Objetivo é manter na memória principal uma certa quantidade de blocos do disco
- Não adiciona nem elimina funcionalidades ao sistema de arquivos
 - Função é melhorar o desempenho do sistema de arquivos
- Não confundir com a cache do processador
- Normalmente a cache de disco é mantida em uma área da memória principal e é controlada pelo sistema operacional
 - Pode ser global ou exclusiva (uma por sistema de arquivo suportado)

Funcionamento da cache de disco

- Em uma requisição de E/S verifica se o bloco está na cache
 - Sim: realiza o acesso a partir dessa cópia em memória
 - Não: realiza o acesso a partir do disco e carrega o bloco para a cache
- A modificação de valores é feito em blocos na cache
 - Problema de quando atualizar o disco após um bloco ter sido alterado
- Problema da perda de informações e da consistência do sistema de arquivos em caso de pane do sistema (falta de energia)

Políticas de atualização da cache

- Posterga ao máximo
- Atualiza a cada intervalo de tempo
- Atualiza imediatamente no disco
- Atualiza imediatamente apenas informações sensíveis a consistência do sistema do arquivo

Política de substituição

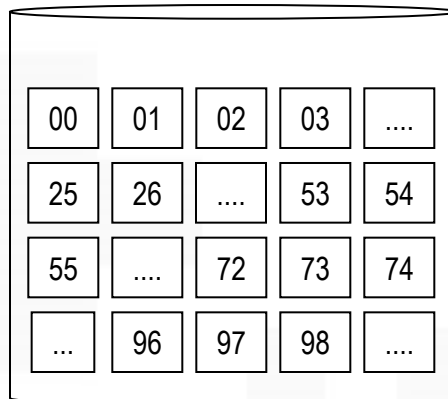
- A cache de disco é um recurso limitado
- O que fazer quando um novo bloco deve ser inserido na cache e não há espaço livre ?
 - Problema similar à gerência de memória virtual (substituição de páginas)
- Tipicamente a política *Least-Recently-Used* (LRU) é empregada

Implementação da política LRU

- Facilmente implementada através de uma lista duplamente encadeada
 - Quando o bloco é acessado ele é removido de sua posição na lista e colocado no início da lista
 - Todo bloco novo (acessado pela primeira vez) também é inserido no início da lista
 - O bloco menos recentemente acessado é o último da lista
- Existe o problema de localizar rapidamente um bloco na lista
 - Emprego de função *hash*

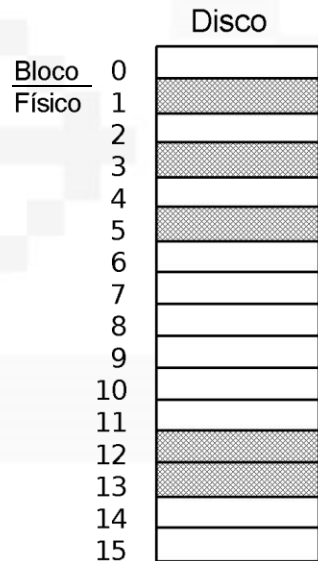
Gerenciamento do espaço livre

- Necessário manter a informação de blocos livres e ocupados
- Métodos básicos:
 - Mapa de *bits* (*bitmap*)
 - Lista de blocos livres
- Ambos métodos consideram que os blocos no disco são numerados sequencialmente



Mapa de *bits* (*bit map*)

- Forma simples de gerenciar o espaço em disco
- Cada bloco do disco possui um *bit* indicando se o bloco está livre ou ocupado



Mapa de Bits.

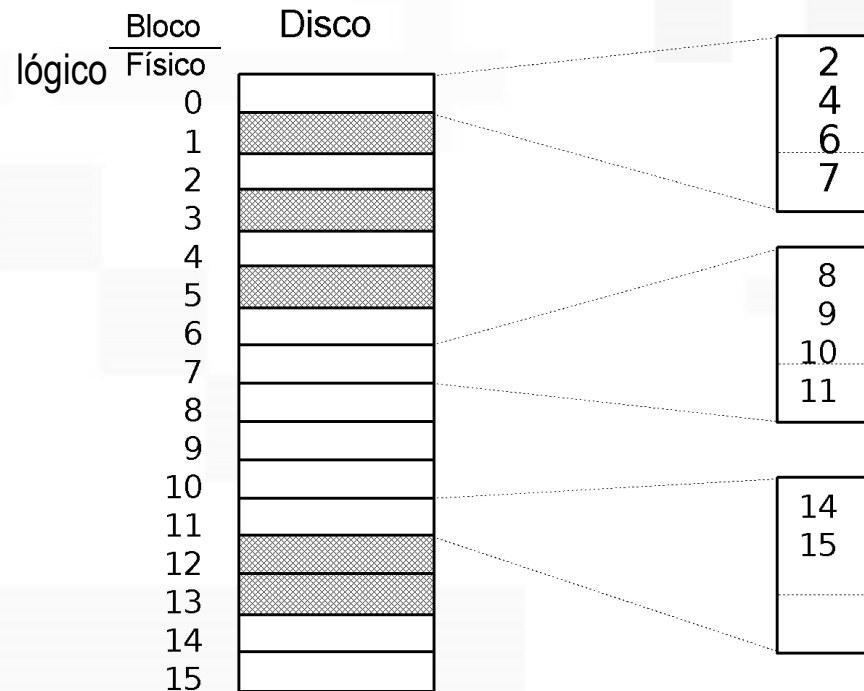
0	0	1	0	1	0	1	0
0	0	1	1	0	0	0	0

$$tamanho_bit_map = \frac{Capacidade_disco(bytes)}{8 \times tamanho_bloco(bytes)}$$

Lista de blocos livres

- Os blocos livres são organizados em uma lista
- Lista é mantida no próprio disco
 - Problema é o tamanho da lista
 - Paliativo: a medida que o espaço em disco é ocupado a lista diminui de tamanho liberando espaço do disco
- Solução alternativa é manter uma lista de áreas livres ao invés de uma lista de blocos livres
 - Endereço do bloco inicial da área livre e o seu tamanho

Gerência de espaço livre através de blocos livres



Leituras complementares

- R. Oliveira, A. Carissimi, S. Toscani; Sistemas Operacionais. Editora Bookman, 2010.
 - Capítulo 8, seções 8.5 e 8.6
- A. Silberchatz, P. Galvin; Operating System Concepts. Addison-Wesley, 4th edition.
 - Capítulo 11 seção 11.3

Diretório

- Problema:
 - Quantidade (grande) de arquivos implica na necessidade de organizá-los
- Sistema de arquivos oferece duas formas de organização
 - Partição
 - Diretório
- Partição divide um disco em discos lógicos (virtuais), mas não resolve a organização de arquivos dentro desse disco lógico
 - No mínimo uma em um sistema
 - Onde “residem” os arquivos e os diretórios

O conceito de diretório

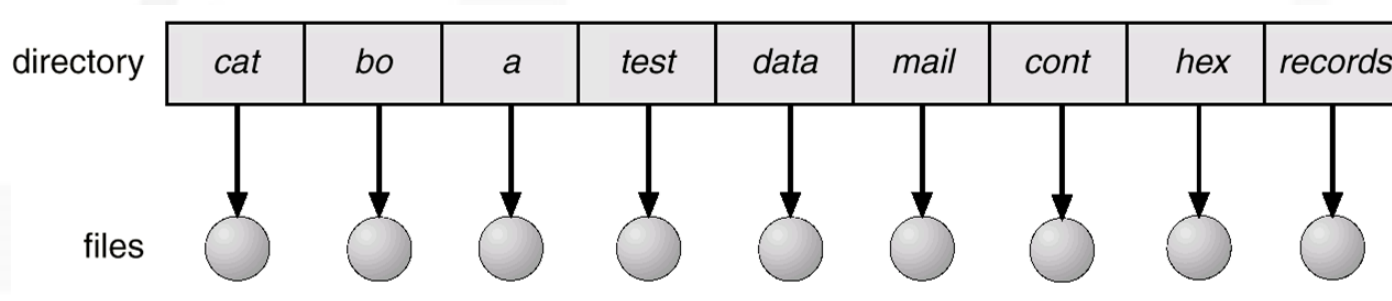
- Estrutura de dados que contém informações sobre arquivos
 - Atributos básicos: nome, tipo, ...
 - Localização: dispositivo físico, end. Início, tamanho,...
 - Controle de acesso: proprietário, informações de acesso, ações permitidas,...
 - Utilização: data criação/modificação, nro. de processos que o usam, *locking*,...
- Diretório é um arquivo pertencente ao sistema operacional
 - Acesso é feito via serviços do sistema operacional
- Tipos de operações em um diretório
 - Pesquisar
 - Criar e remover arquivos
 - Listar diretório
 - Atualizar diretório

Organização de diretório

- Cada entrada do diretório é um arquivo
- Existem duas formas básicas para se organizar um diretório
 - Linear
 - Em árvore

Diretório linear

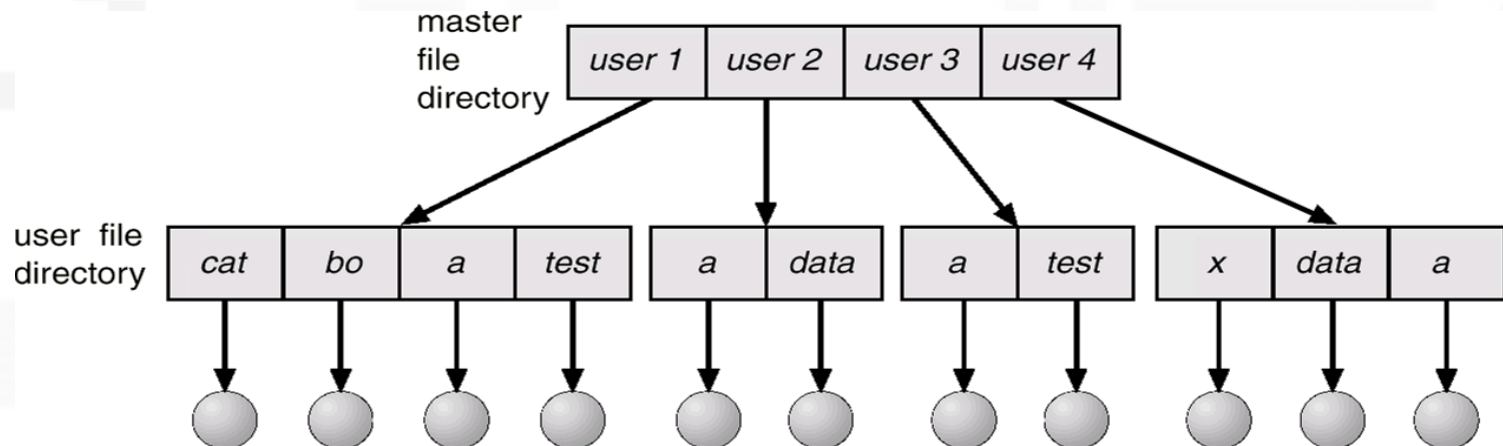
- Mais simples
- O diretório corresponde a uma lista de todos os arquivos do disco
- Desvantagem:
 - Problema de nomeação e agrupamento
 - 2 ou mais usuários não podem ter arquivos com o mesmo nome (colisão)



Silberchatz, Galvin, Gagne. Applied Operating System Concepts (1st Ed.) John Wiley & Sons, 2000.

Diretório linear a dois níveis

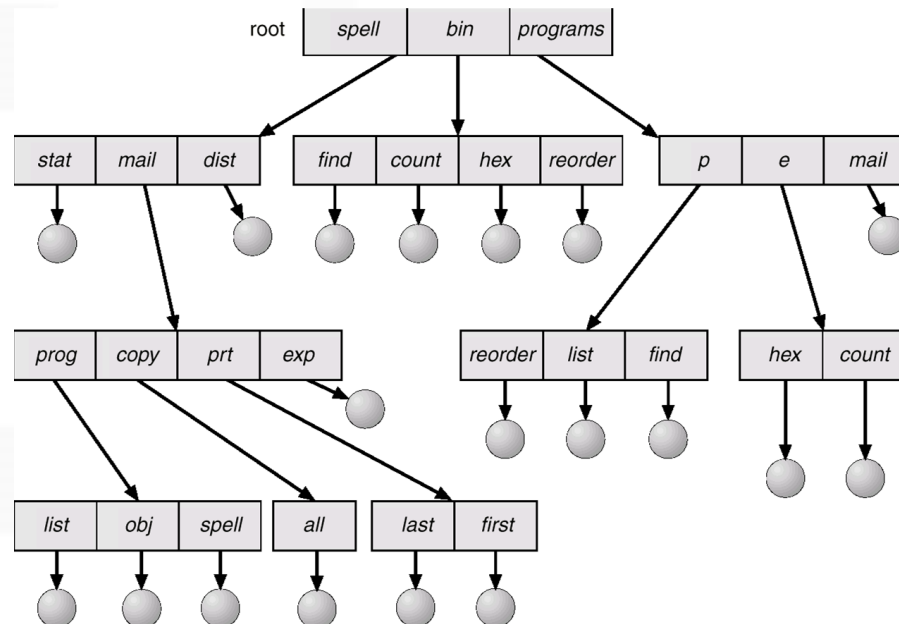
- Cada usuário possui o seu próprio diretório
 - Informação mantida na raiz (*master directory*)
 - Cada entrada corresponde a um subdiretório (usuário)
- Resolve parcialmente o problema de “colisão” de nomes e mas não resolve o problema de organização dos arquivos



Silberchatz, Galvin, Gagne. Applied Operating System Concepts (1st Ed.) John Wiley & Sons, 2000.

Diretório em árvore

- Generalização do diretório linear a dois níveis
 - Permite aos usuários criar subdiretórios e organizar seus arquivos
- Possui um diretório raiz (*master*)



Silberchatz, Galvin, Gagne. Applied Operating System Concepts (1st Ed.) John Wiley & Sons, 2000.

Conceitos associados a um diretório em árvore

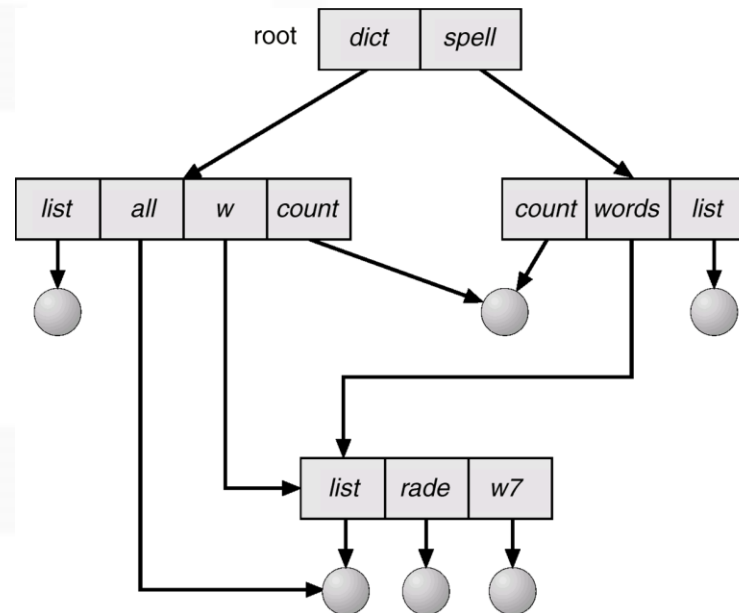
- Qualquer arquivo (ou subdiretório) pode ser identificado de forma não ambígua através de seu caminho (*pathname*)
 - Conceito de diretório corrente, caminho absoluto e caminho relativo
- Diretório corrente (diretório de trabalho):
 - Qualquer nó da árvore
- Caminho absoluto
 - Quando se referencia um arquivo a partir da raiz da árvore
 - e.g.: /spell/mail/prt/first
- Caminho relativo
 - Quando se referencia um arquivo a partir do diretório corrente
 - e.g.: prt/first

Prós e contras da estrutura em árvore

- Vantagem:
 - Procura eficiente por arquivos
 - Possibilidade de agrupamento de arquivos
- Desvantagem:
 - Compartilhamento de arquivos
- Questão é: copiar ou não arquivos a compartilhar?
 - Conceito de *search path*
 - Lista de diretórios (caminhos absolutos) a pesquisar um arquivo

Diretório estruturado em grafos acíclicos

- Generalização da estrutura em árvore
 - Provê compartilhamento através de caminhos alternativos para um arquivo



Silberchatz, Galvin, Gagne. Applied Operating System Concepts (1st Ed.) John Wiley & Sons, 2000.

Aliases

- Compartilhamento pode ser obtido através de *aliases*
- *Link* é uma forma comum de alias
 - Ponteiro para outro arquivo ou subdiretório
- *Link* é uma entrada na estrutura de diretório
 - *Soft link* (simbólico): fornece o caminho do arquivo
 - *Hard link*: fornece a localização (bloco) do arquivo no disco
- Remover um *link* implica em remover apenas a sua entrada na estrutura de diretório, não o arquivo que aponta

Problema da remoção de arquivos

- Solução 1:
 - Acesso a um *link* simbólico *dangling* é detectado no momento do acesso ao arquivo (não resolvido para nome válido)
- Solução 2:
 - Preservar o arquivo enquanto houver referências a ele
 - Contador de *links* ativos
 - Lista de *links*
- Solução 3:
 - Não permitir compartilhamento

Prós e contras de diretórios estruturados em grafos

- Vantagem:
 - Compartilhamento de arquivos
- Desvantagem:
 - Estrutura mais complexa de manter
 - Um arquivo pode possuir mais de um caminho de acesso
 - e.g.: Problemas para contabilização de acessos, *back-ups*, etc...
 - Remoção de um arquivo compartilhado
 - Problema de *dangling pointer*
 - Criação de laços através de aliases
 - Necessita algoritmo para verificar se não cria um laço (desempenho)

Exemplos de aliases: UNIX

%ln index hlink

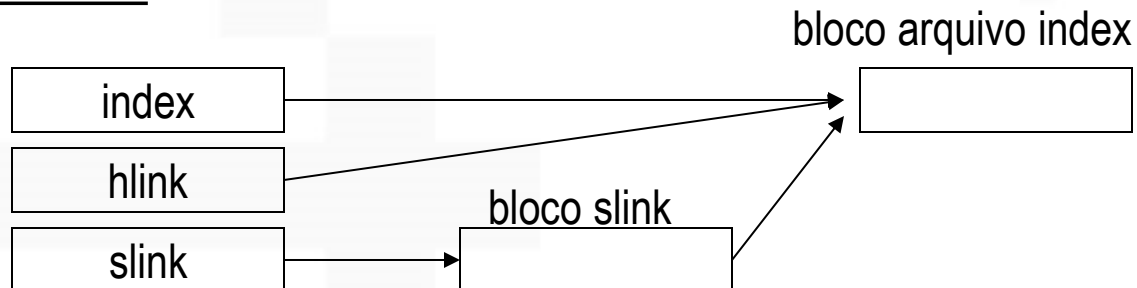
%ln -s index slink

%ls -l

-rw- - - - -	2	chavez	chem	5228	Mar 12 11:36	index
-rw- - - - -	2	chavez	chem	5228	Mar 12 11:36	hlink
lrwx rwx rwx	1	chavez	chem	5	Mar 12 11:36	slink→index

Contador de referências

Diretório



Exemplos de aliases: UNIX

%ln index hlink

%ln -s index slink

%ls -l

-rw- - - - -

-rw- - - - -

lrwx rwx rwx



2
2
1

Contador de referências

chavez

chem

5228

Mar 12 11:36 index

chavez

chem

5228

Mar 12 11:36 hlink

chavez

chem

5

Mar 12 11:36 slink→index

Diretório

index

hlink

slink

bloco slink

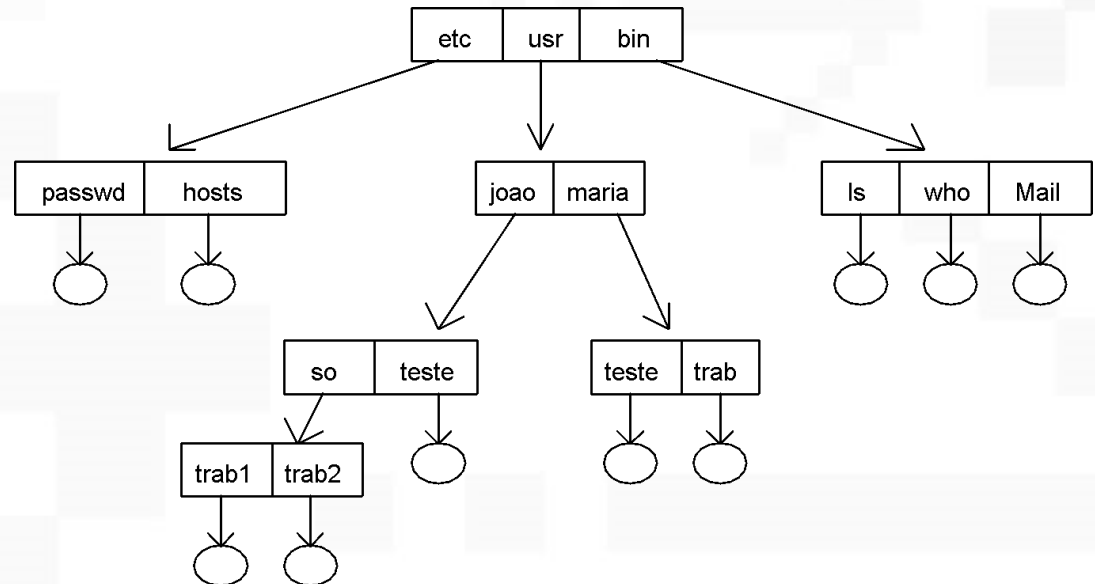
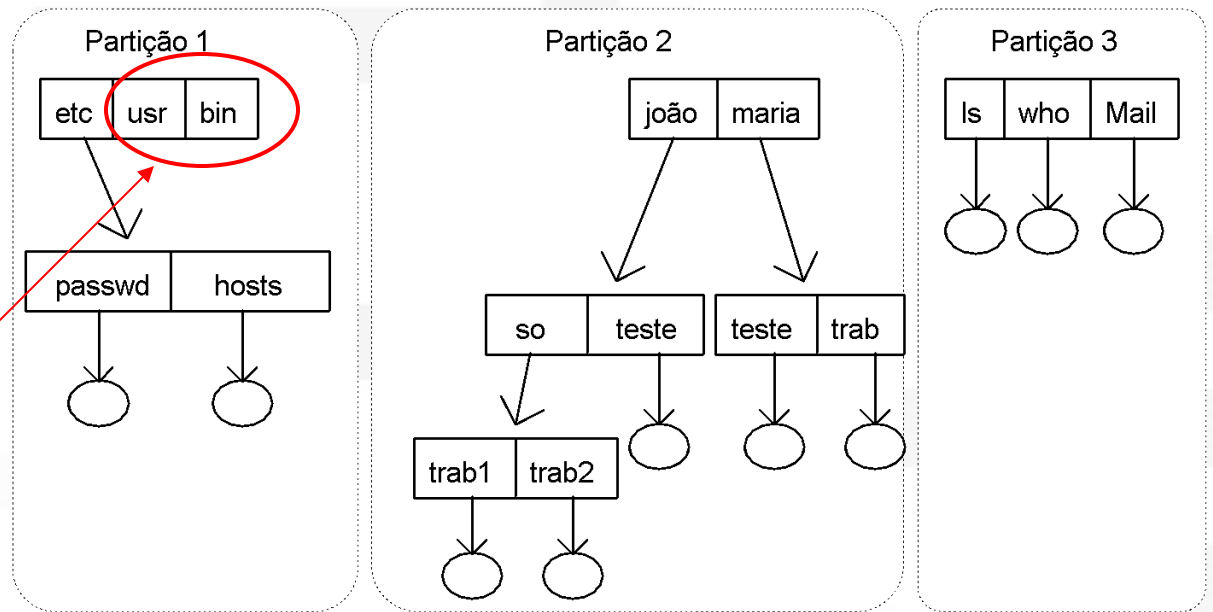
bloco arquivo index

Organização de diretórios do UNIX

- Baseado em partições
- Diretório raiz do sistema de arquivos corresponde a uma partição especial (*root*)
- Conceito de ponto de montagem
- Pontos importantes:
 - Cada partição possui seu próprio sistema de arquivos
 - Capacidade de integrar diferentes sistemas de arquivos em uma mesma hierarquia
 - Sistemas de arquivos podem ser diferentes
 - e.g.: ext2, FAT12, FAT32, NTFS, etc...

Montagem de partições em um subdiretório

Pontos de montagem

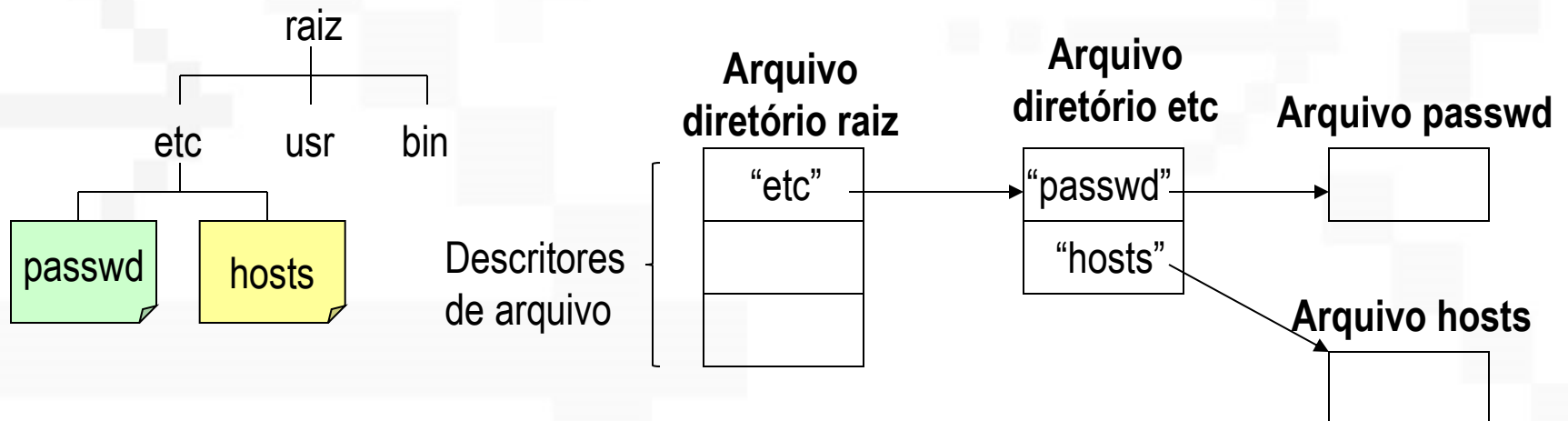


Implementação de diretórios

- Diretórios são arquivos especiais cujo conteúdo é manipulado pelo sistema operacional
- Sendo um arquivo:
 - Utiliza os mesmos mecanismos de alocação, liberação e localização de blocos do disco que arquivos “comuns”
 - Possuem um descritor de arquivo
- Duas formas básicas de implementação de diretórios:
 - Conjunto de arquivos de descritores de arquivos
 - Vetor de descritores

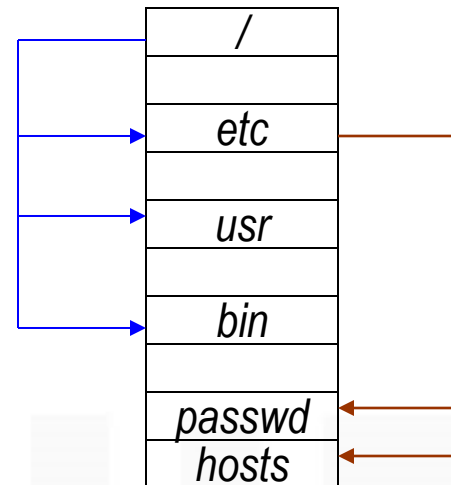
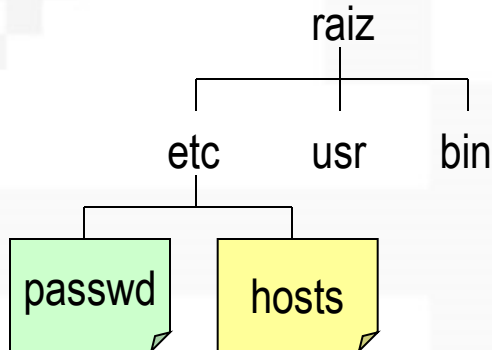
Conjunto de arquivos de descritores de arquivos

- Estrutura de diretório corresponde a um conjunto de arquivos do tipo diretório
 - Cada arquivo diretório possui descritores de arquivos



Vetor de descritores

- Uma parte do disco é reservada para o armazenamento de descritores de arquivos (diretórios”, regulares, etc...)
- Forma um diretório único (o vetor)
 - Arquivos e diretórios são identificados por sua posição nesse vetor
- Supõem-se que o primeiro descritor descreve o diretório raiz (“/”)



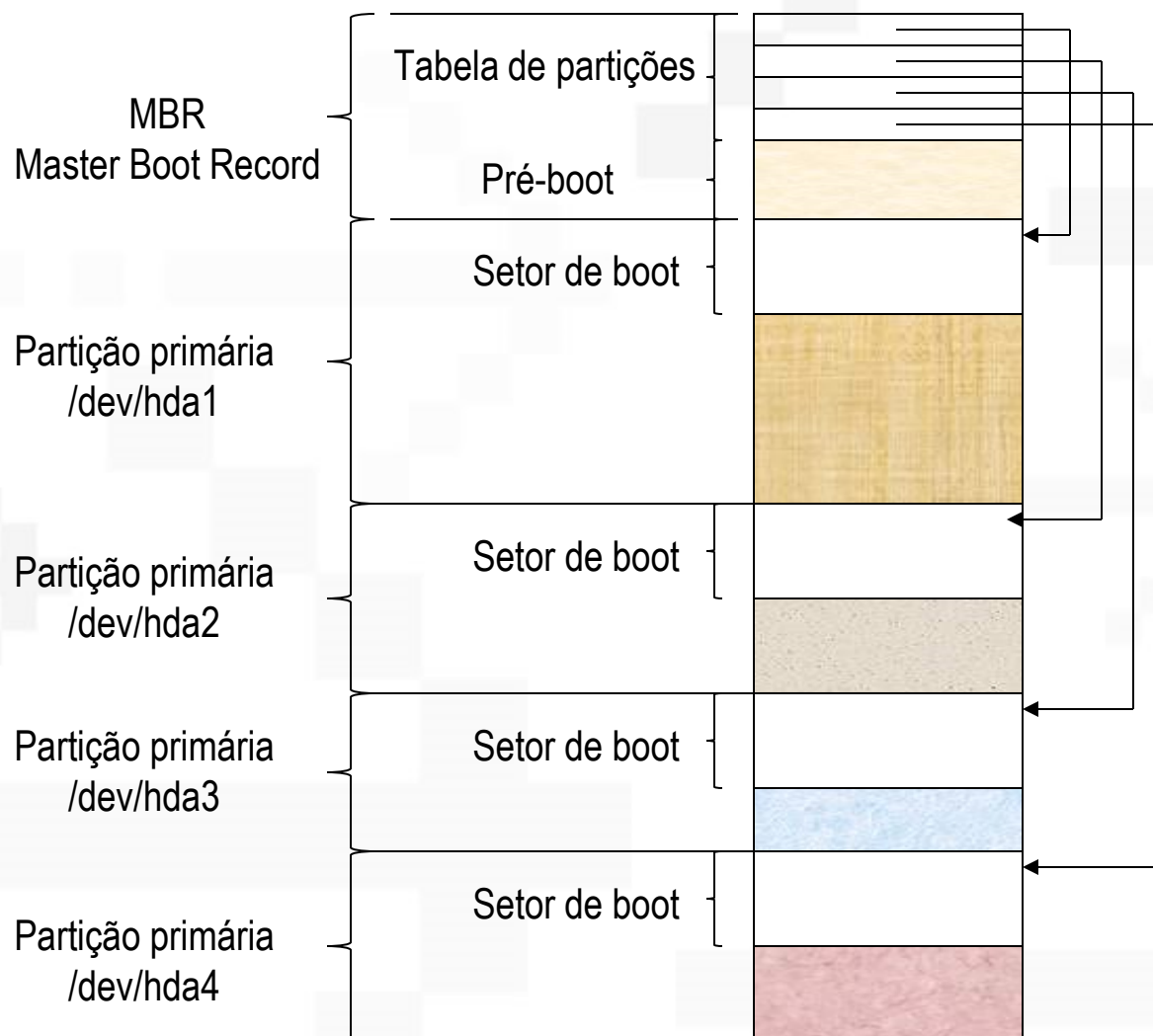
Implementação de diretórios como tabelas

- Um diretório nada mais é que uma tabela
- Três implementações mais utilizadas:
 - Lista não ordenada
 - Lista ordenada
 - Tabela de dispersão (tabela *hash*)
- Vantagens e desvantagens dessas implementações são as “tradicionais”:
 - Simplicidade *versus* desempenho

Organização interna de uma partição

- Uma partição é um disco lógico
- Cada partição é autocontida, isto é, todas as informações para acesso aos arquivos da partição estão contidas na própria partição
 - Diretórios e subdiretórios
 - Descritores de arquivos da partição
 - Blocos de dados
 - Lista de blocos livres da partição
- Formatação lógica corresponde à inicialização dessas estruturas de dados
- Normalmente um setor (bloco) especial do disco informa quais são as partições e quais parcelas do disco a partição ocupa

Partições primárias em um disco IDE



Leituras complementares

- R. Oliveira, A. Carissimi, S. Toscani; Sistemas Operacionais. Editora Bookman, 2010.
 - Capítulo 8, seções 8.7, 8.8 e 8.9
- A. Silberchatz, P. Galvin, G. Gagne; Applied Operating System Concepts. Addison-Wesley, 2000, (1st edition).
 - Capítulo 11
- W. Stallings; Operating Systems. (4th edition). Prentice Hall, 2001.
 - Capítulo 12