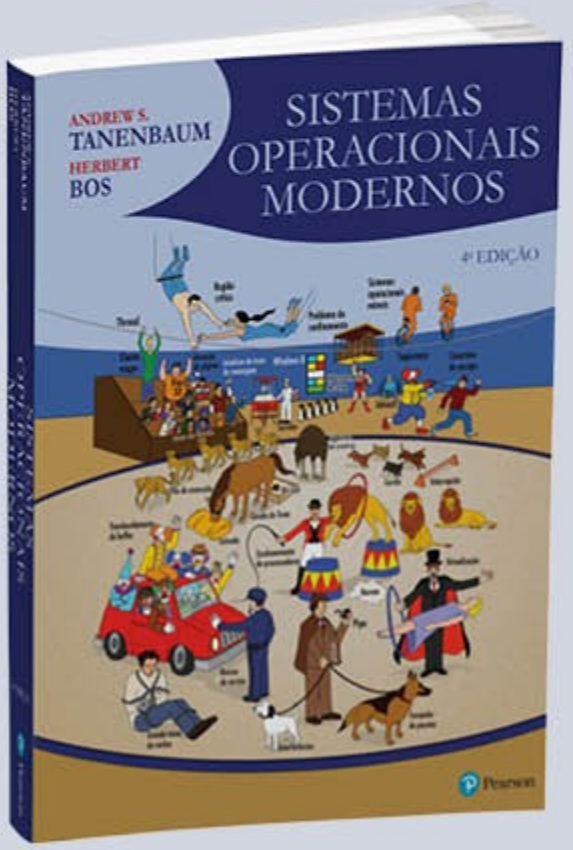
Capítulo 4: Sistemas de Arquivos



Sistemas de Arquivos

- As aplicações de computadores precisam **armazenar** e **recuperar** informações.
- A capacidade de armazenamento está **restrita** ao tamanho do **espaço do endereçamento virtual**.
- Outro problema em manter informações dentro do espaço de endereçamento de um processo é que, uma vez que o processo é **concluído**, as informações serão **perdidas**.

Sistemas de Arquivos

- Muitas vezes, é necessário que múltiplos processos acessem uma informação ao mesmo tempo. Se temos um diretório telefônico on-line armazenado dentro do espaço de um processo, só aquele processo pode acessá-lo. É preciso tornar a informação independente de qualquer processo.

- Três requisitos para o armazenamento de informações por um longo prazo:
  1. Deve ser possível armazenar uma quantidade muito grande de informações.
  2. As informações devem sobreviver ao término do processo que as está utilizando.
  3. Múltiplos processos têm de ser capazes de acessá-las ao mesmo tempo.

Sistemas de Arquivos

- Discos magnéticos foram usados por anos para esse armazenamento de longo prazo. Recentemente, unidades de estado sólido tornaram-se cada vez mais populares, já que não têm partes móveis que possam quebrar.
- Basta pensar em um disco como uma sequência linear de blocos de tamanho fixo e que dão suporte a duas operações:
    1. *Leia o bloco k.*
    2. *Escreva no bloco k.*

Sistemas de Arquivos

- Existem mais operações, mas com essas duas é possível solucionar o problema do armazenamento de longo prazo.
- Entretanto, essas operações são muito inconvenientes, mais ainda em sistemas grandes usados por muitas aplicações e múltiplos usuários. Questões que rapidamente surgem podem ser:
    1. Como você encontra informações?
    2. Como impedir que um usuário leia os dados de outro?
    3. Como saber quais blocos estão livres?

Sistemas de Arquivos

- As abstrações de processos (e threads), espaços de endereçamento e arquivos são os conceitos mais importantes relacionados com os sistemas operacionais.
- **Arquivos** são unidades lógicas de informação criadas por processos. Um disco normalmente conterá milhares ou mesmo milhões deles, cada um independente dos outros.
- **Processos** podem ler arquivos existentes e criar novos. Informações armazenadas em arquivos devem ser persistentes: não devem ser afetadas pela criação e término de um processo. Um arquivo deve desaparecer apenas quando o seu proprietário o remove explicitamente.

Sistemas de Arquivos

- Arquivos são gerenciados pelo sistema operacional. Como um todo, aquela parte do sistema operacional lidando com arquivos é conhecida como sistema de arquivos.

- Um arquivo fornece uma maneira para armazenar informações sobre o disco e lê-las depois. Isso deve ser feito de tal modo que isole o usuário dos detalhes de como e onde as informações estão armazenadas.

Nomeação de Arquivos

- Quando um processo cria um arquivo, ele lhe dá um nome. Quando o processo é concluído, o arquivo continua a existir e pode ser acessado por outros processos usando o seu nome.
- As regras para a nomeação de arquivos variam de sistema para sistema, mas todos os sistemas operacionais atuais permitem cadeias de uma a oito letras como nomes de arquivos legais. Dígitos e caracteres especiais também são frequentemente permitidos.

Nomeação de Arquivos

- Alguns sistemas de arquivos distinguem entre letras maiúsculas e minúsculas. O UNIX faz essa distinção; o MS-DOS, não.
- Um sistema UNIX pode ter todos os arquivos a seguir como três arquivos distintos: **maria**, **Maria** e **MARIA**. No MS-DOS, todos esses nomes referem-se ao **mesmo** arquivo.
- Muitos sistemas operacionais aceitam nomes de arquivos de duas partes, com as partes separadas por um ponto, como em *prog.c*. A parte que vem depois do ponto é a extensão do arquivo.

Nomeação de Arquivos

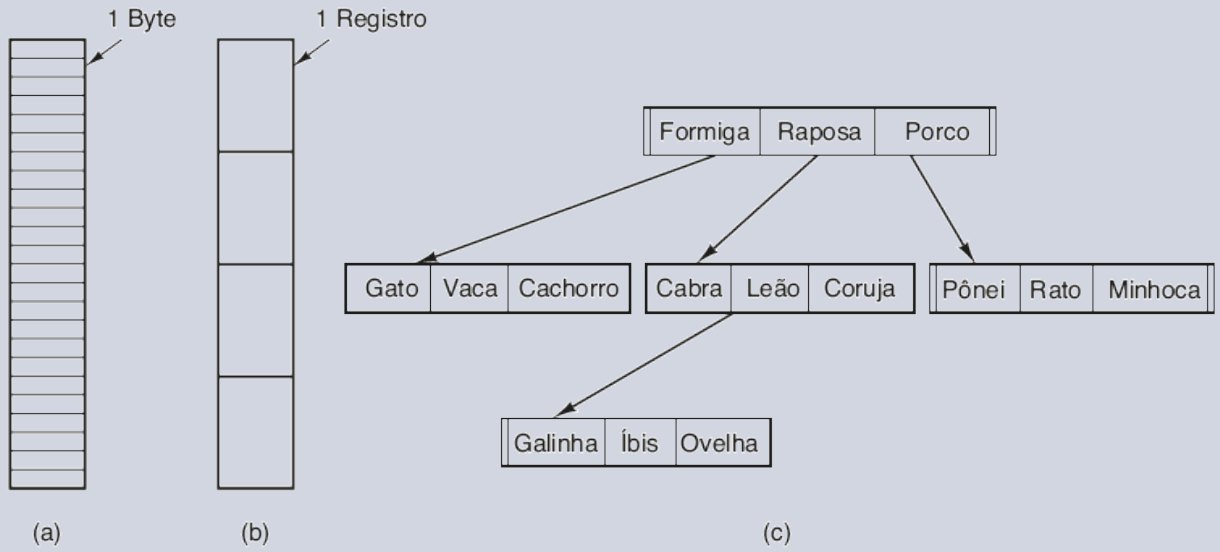
- Algumas das extensões de arquivos mais comuns e seus significados:

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

Estrutura de Arquivos

- Arquivos podem ser estruturados de várias maneiras. Três possibilidades comuns estão descritas na figura:

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



Estrutura de Arquivos

- O arquivo em **(a)** é uma sequência desestruturada de bytes. O sistema operacional não sabe ou não se importa com o que há no arquivo; o que ele vê são bytes. Qualquer significado deve ser imposto por programas em nível de usuário. Ter o sistema operacional tratando arquivos como nada mais que sequências de bytes oferece a máxima flexibilidade.

- O primeiro passo na estruturação está ilustrado em (b). Nesse modelo, um arquivo é uma sequência de registros de tamanho fixo, cada um com alguma estrutura interna.

Estrutura de Arquivos

- O fundamental para que um arquivo seja uma sequência de registros é a ideia de que a operação de leitura retorna um registro e a operação de escrita sobrepõe ou anexa um registro.

- O terceiro tipo de estrutura de arquivo é mostrado em **(c)**. Nessa organização, um arquivo consiste em uma árvore de registros, não necessariamente todos do mesmo tamanho, cada um contendo um campo chave em uma posição fixa no registro. A árvore é ordenada no campo chave, a fim de permitir uma busca rápida por uma chave específica.

Tipos de Arquivos

- Muitos sistemas operacionais aceitam vários tipos de arquivos. O UNIX (incluindo OS X) e o Windows, por exemplo, apresentam arquivos regulares e diretórios. O UNIX também tem arquivos especiais de caracteres e blocos.

- Os arquivos regulares são aqueles que contêm informações do usuário. Todos os arquivos da figura anterior são arquivos regulares.
- Os diretórios são arquivos do sistema para manter a estrutura do sistema de arquivos.
- Os arquivos especiais de caracteres são relacionados com entrada/saída e usados para modelar dispositivos de E/S seriais como terminais, impressoras e redes.
- Os arquivos especiais de blocos são usados para modelar discos.

Tipos de Arquivos

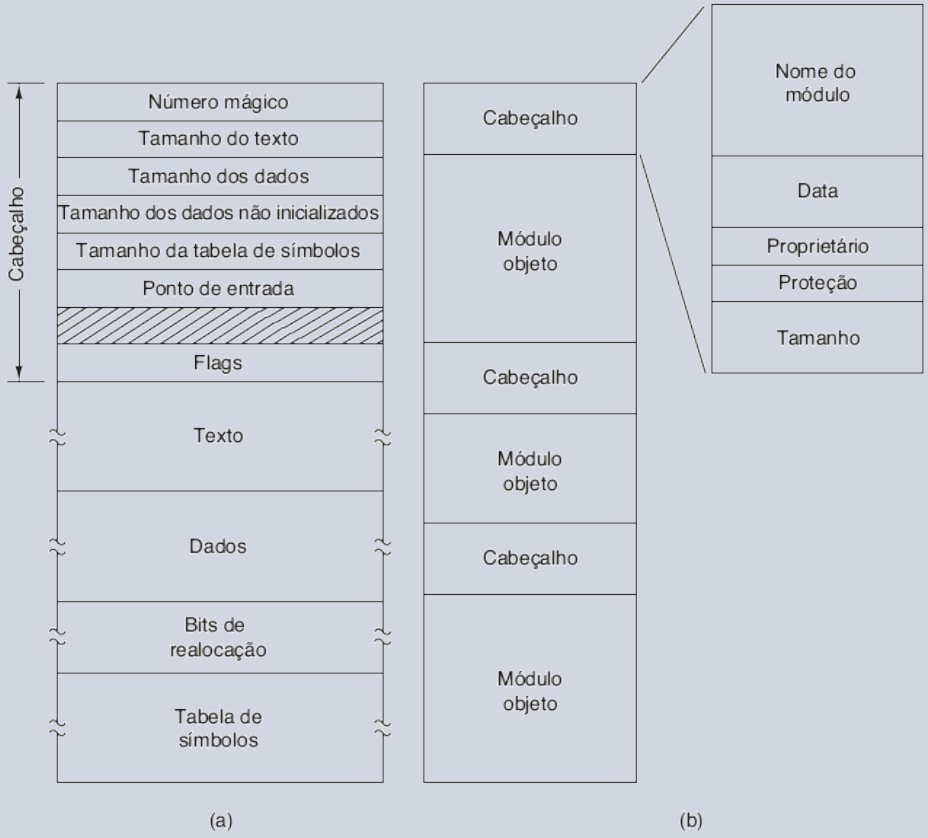
- Arquivos regulares geralmente são arquivos ASCII ou arquivos binários.
- Arquivos ASCII consistem de linhas de texto. Em alguns sistemas, cada linha termina com um caractere de retorno de carro (*carriage return*). Em outros, o caractere de próxima linha (*line feed*) é usado.
- Quando arquivos são binários, isso significa apenas que eles não são arquivos ASCII. Listá-los em uma impressora resultaria em algo completamente incompreensível. Em geral, eles têm alguma estrutura interna conhecida pelos programas que os usam.

Tipos de Arquivos

- Em **(a)** da próxima figura, há um arquivo binário executável simples, tirado de uma versão inicial do UNIX. Embora o arquivo seja apenas uma sequência de bytes, o sistema operacional o executará somente se ele tiver o formato apropriado.
- Ele tem cinco seções: cabeçalho, texto, dados, bits de realocação e tabela de símbolos. O cabeçalho começa com o chamado número mágico, identificando o arquivo como executável.

Tipos de Arquivos

(a) Um arquivo executável. (b) Um repositório (archive).



Acesso aos Arquivos

- Os primeiros sistemas operacionais forneciam apenas um tipo de acesso aos arquivos: sequencial. Nesses sistemas, um processo lia todos os bytes ou registros em um arquivo em ordem, começando do princípio, mas não podia pular nenhum ou lê-los fora de ordem.
- Arquivos sequenciais podiam ser trazidos de volta para o ponto de partida, então podiam ser lidos quando necessário. Arquivos sequenciais eram convenientes quando o meio de armazenamento era uma fita magnética, em vez de um disco.

Acesso aos Arquivos

- Quando os discos passaram a ser usados para armazenar arquivos, tornou-se possível ler os bytes ou registros de um arquivo fora de ordem, ou acessar os registros pela chave em vez de pela posição. Arquivos ou registros que podem ser lidos em qualquer ordem são chamados de arquivos de acesso aleatório.

Atributos de Arquivos

- Todo arquivo possui um nome e sua data. Além disso, os sistemas operacionais associam outras informações com os arquivos, como data e horário em que foi modificado pela última vez, e tamanho do arquivo. Esses itens extras são atributos do arquivo ou metadados.

Atributos de Arquivos

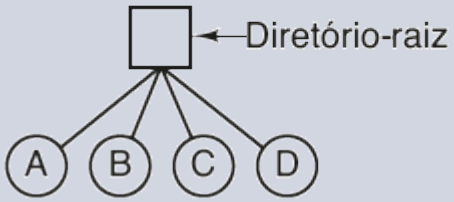
Alguns possíveis atributos de arquivos.

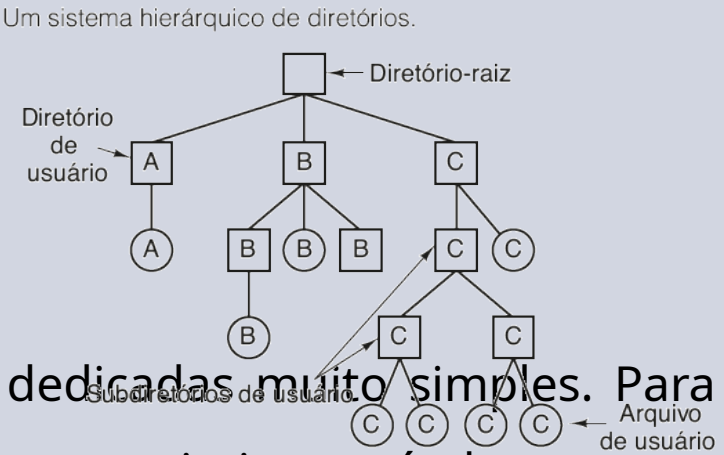
Atributo	Significado
Proteção	Quem tem acesso ao arquivo e de que modo
Senha	Necessidade de senha para acesso ao arquivo
Criador	ID do criador do arquivo
Proprietário	Proprietário atual
Flag de somente leitura	0 para leitura/escrita; 1 para somente leitura
Flag de oculto	0 para normal; 1 para não exibir o arquivo
Flag de sistema	0 para arquivos normais; 1 para arquivos de 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 da chave	Número de bytes na 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

Diretórios

- Para controlar os arquivos, sistemas de arquivos normalmente têm diretórios ou pastas, que são em si arquivos. A forma mais simples de um sistema de diretório é ter um diretório contendo todos os arquivos, um diretório-raiz.

Um sistema de diretório em nível único contendo quatro arquivos.





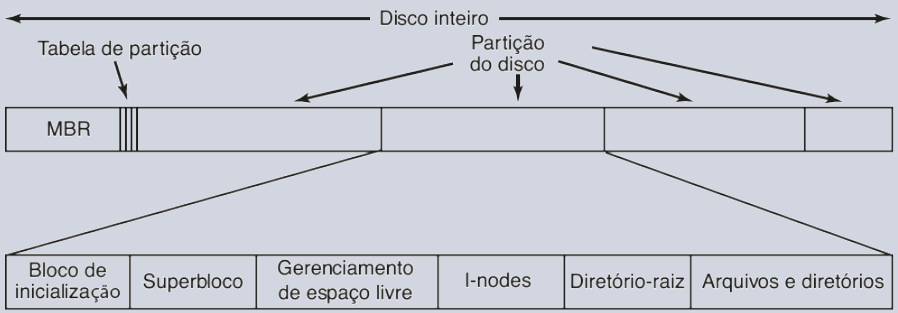
Diretórios

- O nível único é adequado para aplicações dedicadas muito simples. Para usuários modernos, com milhares de arquivos, seria impossível encontrar qualquer coisa se os arquivos estivessem em um único diretório. É necessária uma maneira para agrupar arquivos relacionados em um mesmo local.
- Com hierarquia, o usuário pode ter tantos diretórios quantos forem necessários. Além disso, se múltiplos usuários compartilham um servidor de arquivos comum, cada usuário pode ter um diretório-raiz privado para sua própria hierarquia.
- Essa abordagem é mostrada na figura. Aqui, cada diretório A, B e C contido no diretório-raiz pertence a um usuário diferente, e dois deles criaram subdiretórios para projetos nos quais estão trabalhando.

Implementação do sistema de arquivos

- Implementadores estão interessados em como os arquivos e os diretórios estão armazenados, como o espaço de disco é gerenciado e como fazer tudo funcionar de maneira eficiente e confiável.
- Sistemas de arquivos são armazenados em discos. A maioria dos discos pode ser dividida em uma ou mais partições, com sistemas de arquivos independentes em cada partição. O Setor 0 do disco é chamado de MBR (*Master Boot Record* — registro mestre de inicialização) e é usado para inicializar o computador.

Um esquema possível para um sistema de arquivos.



Implementação do sistema de arquivos

- Muitas vezes o sistema de arquivos vai conter alguns dos itens mostrados na figura. O primeiro é o superbloco. Ele contém todos os parâmetros-chave a respeito do sistema de arquivos e é lido para a memória quando o computador é inicializado ou o sistema de arquivos é tocado pela primeira vez.
- Em seguida podem vir informações a respeito de blocos disponíveis no sistema de arquivos, na forma de um mapa de bits ou de uma lista de ponteiros, por exemplo. O restante do disco contém todos os outros diretórios e arquivos.

Implementação do sistema de arquivos

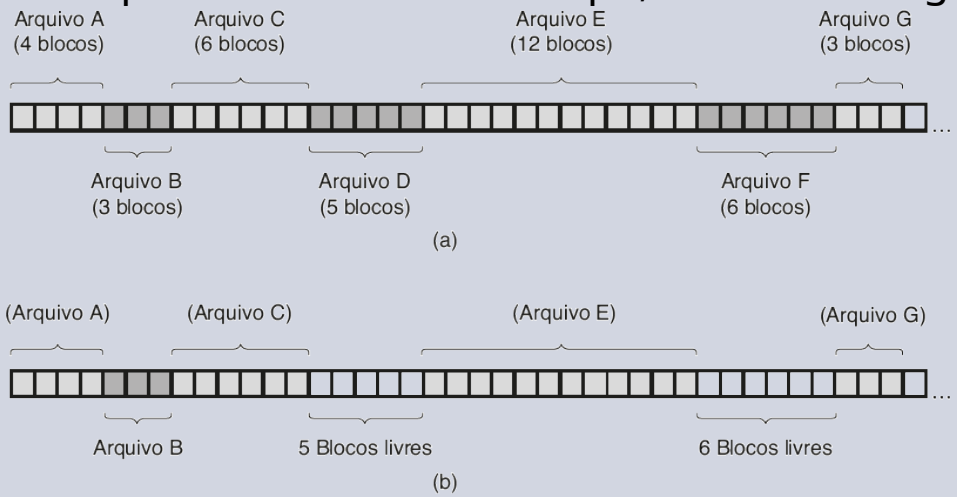
- Vários métodos são usados, em diferentes sistemas operacionais, para controlar quais blocos de disco vão com quais arquivos na implementação do armazenamento de arquivos.

- Alocação contígua: O esquema de alocação mais simples é armazenar cada arquivo como uma execução contígua de blocos de disco. A alocação de espaço de disco contíguo é simples de implementar porque basta se lembrar de dois números para monitorar onde estão os blocos de um arquivo: o endereço em disco do primeiro bloco e o número de blocos no arquivo. Dado o número do primeiro bloco, o número de qualquer outro bloco pode ser encontrado por uma simples adição.

Implementação do sistema de arquivos

- Alocação contígua: O desempenho da leitura é excelente, pois o arquivo inteiro pode ser lido do disco em uma única operação. Uma busca é necessária (para o primeiro bloco). Depois, não são mais necessárias buscas ou atrasos rotacionais, então os dados são lidos com a capacidade total do disco. Infelizmente, a alocação contígua tem um ponto fraco: com o tempo, o disco se fragmenta.

(a) Alocação contígua de espaço de disco para sete arquivos. (b) O estado do disco após os arquivos D e F terem sido removidos.



Um exemplo de alocação em armazenamento contíguo na figura, em **(a)**.

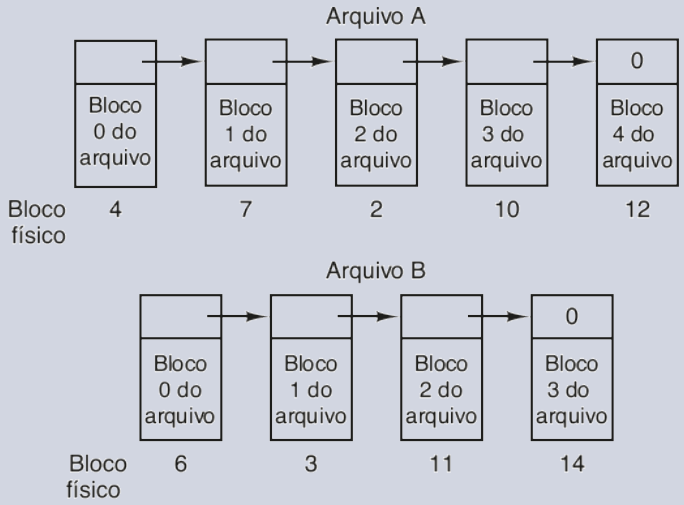
Implementação do sistema de arquivos

- Alocação contígua: Os primeiros 40 blocos de disco são mostrados, começando com o bloco 0 à esquerda. De início, o disco estava vazio. Então um arquivo A, de quatro blocos de comprimento, foi escrito a partir do início (bloco 0). Após isso, um arquivo de seis blocos, B, foi escrito começando logo depois do fim do arquivo A.
- Alocação contígua: Examine **(b)** na figura. Dois arquivos, D e F, foram removidos. Quando um arquivo é removido, seus blocos são naturalmente liberados, deixando uma lacuna de blocos livres no disco. O disco não é compactado imediatamente para eliminá-la, já que isso envolveria copiar todos os blocos seguindo essa lacuna. Como resultado, o disco consiste em arquivos e lacunas, como ilustrado na figura.

Implementação do sistema de arquivos

- Alocação por lista encadeada: O segundo método para armazenar arquivos é manter cada um como uma lista encadeada de blocos de disco, como mostrado na figura.

Armazenando um arquivo como uma lista encadeada de blocos de disco.



A primeira palavra de cada bloco é usada como um ponteiro para a próxima. O resto do bloco é reservado para dados. Diferentemente da alocação contígua, todos os blocos do disco podem ser usados nesse método. Nenhum espaço é perdido para a fragmentação de disco.

Implementação do sistema de arquivos

- Alocação por lista encadeada usando uma tabela na memória: Ambas as desvantagens da alocação por lista encadeada - acesso aleatório de extrema lentidão e quantidade de dados que um bloco pode armazenar não mais uma potência de dois - podem ser eliminadas colocando-se as palavras do ponteiro de cada bloco de disco em uma tabela na memória. A figura seguinte mostra como são as tabelas para o exemplo da figura anterior.

Alocação por lista encadeada usando uma tabela de alocação de arquivos na memória principal.

Implementação do sistema de arquivos

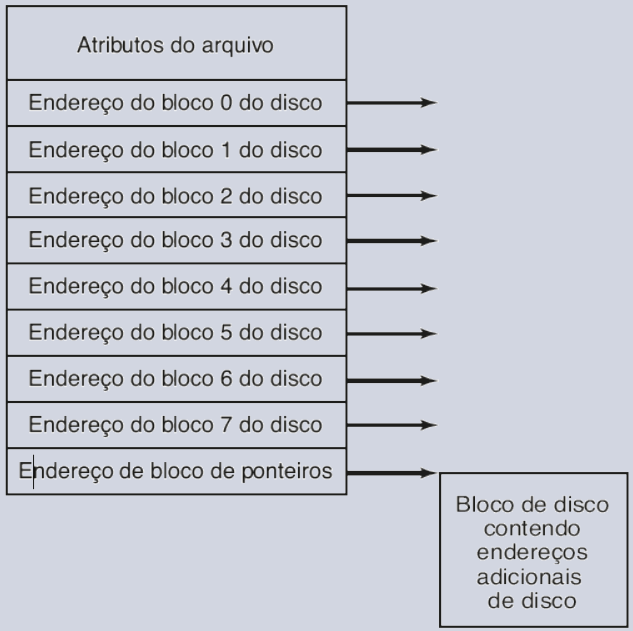
- Alocação por lista encadeada usando uma tabela na memória: O arquivo A usa os blocos de disco 4, 7, 2, 10 e 12, nessa ordem, e o arquivo B usa os blocos de disco 6, 3, 11 e 14, nessa ordem. Usando a tabela da figura, podemos começar com o bloco 4 e seguir a cadeia até o fim. Essa tabela na memória principal é chamada de FAT (File Allocation Table — tabela de alocação de arquivos). A principal desvantagem desse método é que a tabela inteira precisa estar na memória o todo o tempo para fazê-la funcionar.

- I-nodes: último método para monitorar quais blocos pertencem a quais arquivos, associa cada arquivo a uma estrutura de dados chamada de i-node (index-node — nó-índice), que lista os atributos e os endereços de disco dos blocos do disco. Um exemplo simples é a figura. Dado o i-node, é possível encontrar todos os blocos do arquivo.

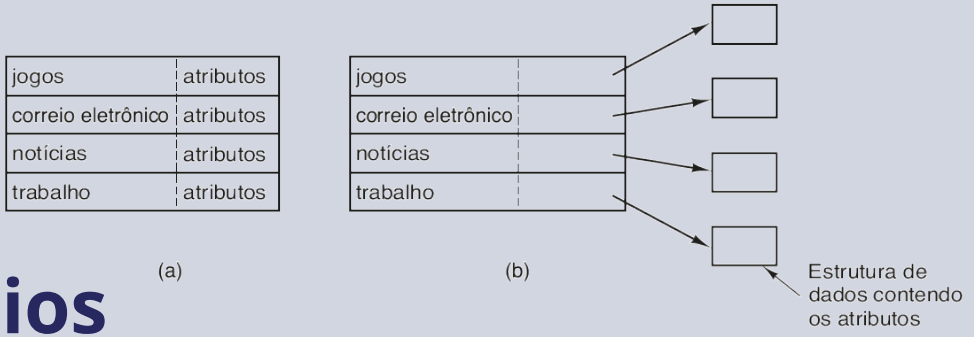
Implementação do sistema de arquivos

- I-nodes: A grande vantagem desse esquema sobre os arquivos encadeados usando uma tabela na memória é que o i-node precisa estar na memória apenas quando o arquivo correspondente estiver aberto. Um problema com i-nodes é que cada um tem espaço para um número fixo de endereços de disco, e quando um arquivo cresce além de seu limite, a solução é reservar o último endereço de disco não para um bloco de dados, mas para o endereço de um bloco contendo mais endereços de blocos de disco, como mostrado na figura.

Um exemplo de i-node.



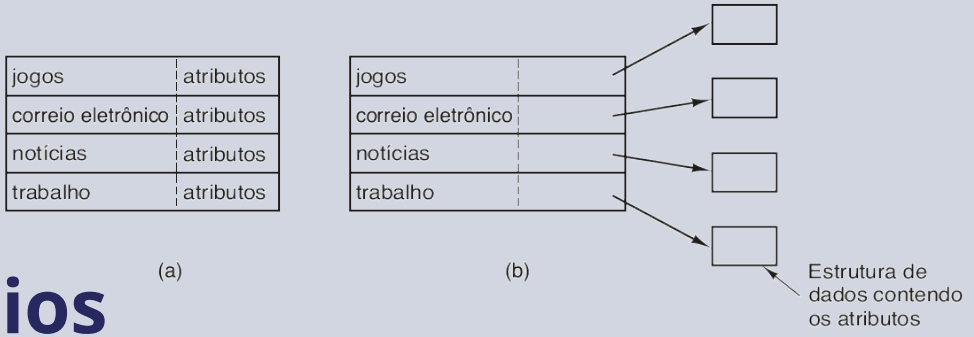
(a) Um diretório simples contendo entradas de tamanho fixo com endereços de disco e atributos na entrada do diretório.
(b) Um diretório no qual cada entrada refere-se a apenas um i-node.



Implementando diretórios

- Todo sistema de arquivos mantém vários atributos do arquivo, que devem ser armazenados em algum lugar. Uma possibilidade é fazê-lo diretamente na entrada do diretório. Alguns sistemas fazem precisamente isso. Essa opção é mostrada em **(a)** da figura. Nesse design simples, um diretório consiste em uma lista de entradas de tamanho fixo, um por arquivo, contendo um nome de arquivo (de tamanho fixo), uma estrutura dos atributos do arquivo e um ou mais endereços de disco (até algum máximo) dizendo onde estão os blocos de disco.

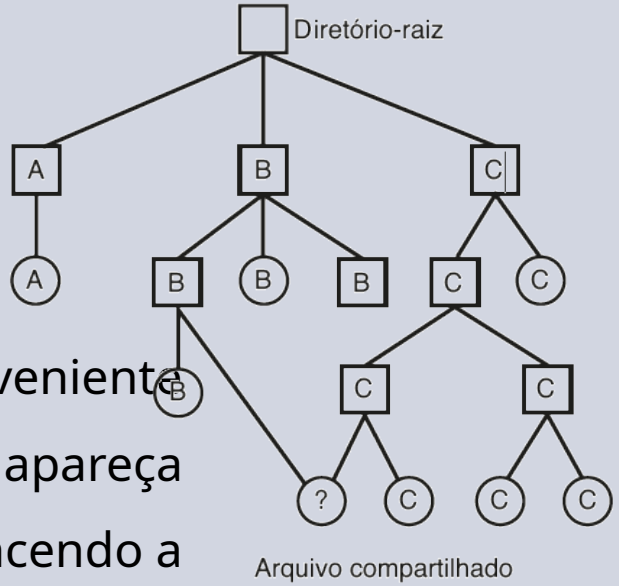
(a) Um diretório simples contendo entradas de tamanho fixo com endereços de disco e atributos na entrada do diretório.
(b) Um diretório no qual cada entrada refere-se a apenas um i-node.



Implementando diretórios

- Para sistemas que usam i-nodes, outra possibilidade para armazenar os atributos é nos próprios i-nodes, em vez de nas entradas do diretório. A entrada do diretório pode ser mais curta: apenas um nome de arquivo e um número de i-node. Essa abordagem está ilustrada em **(b)** da figura.

Arquivos compartilhados



Sistema de arquivos contendo um arquivo compartilhado.

- Usuários precisam compartilhar arquivos. É conveniente que um arquivo compartilhado apareça simultaneamente em diretórios diferentes pertencendo a usuários distintos.
- A figura mostra o sistema de arquivos de uma figura anterior novamente, apenas com um dos arquivos do usuário C agora presente também em um dos diretórios do usuário B. A conexão entre o diretório do usuário B e o arquivo compartilhado é chamada de ligação. O sistema de arquivos em si é agora um Gráfico Acíclico Orientado (Directed Acyclic Graph — DAG).

Gerenciamento e otimização de sistemas de arquivos

- O gerenciamento de espaço de disco é uma preocupação importante para os projetistas de sistemas de arquivos. Duas estratégias gerais são possíveis para armazenar um arquivo de n bytes: ou são alocados n bytes consecutivos de espaço, ou o arquivo é dividido em uma série de blocos (não necessariamente) contíguos. A mesma escolha está presente em sistemas de gerenciamento de memória entre a segmentação pura e a paginação.

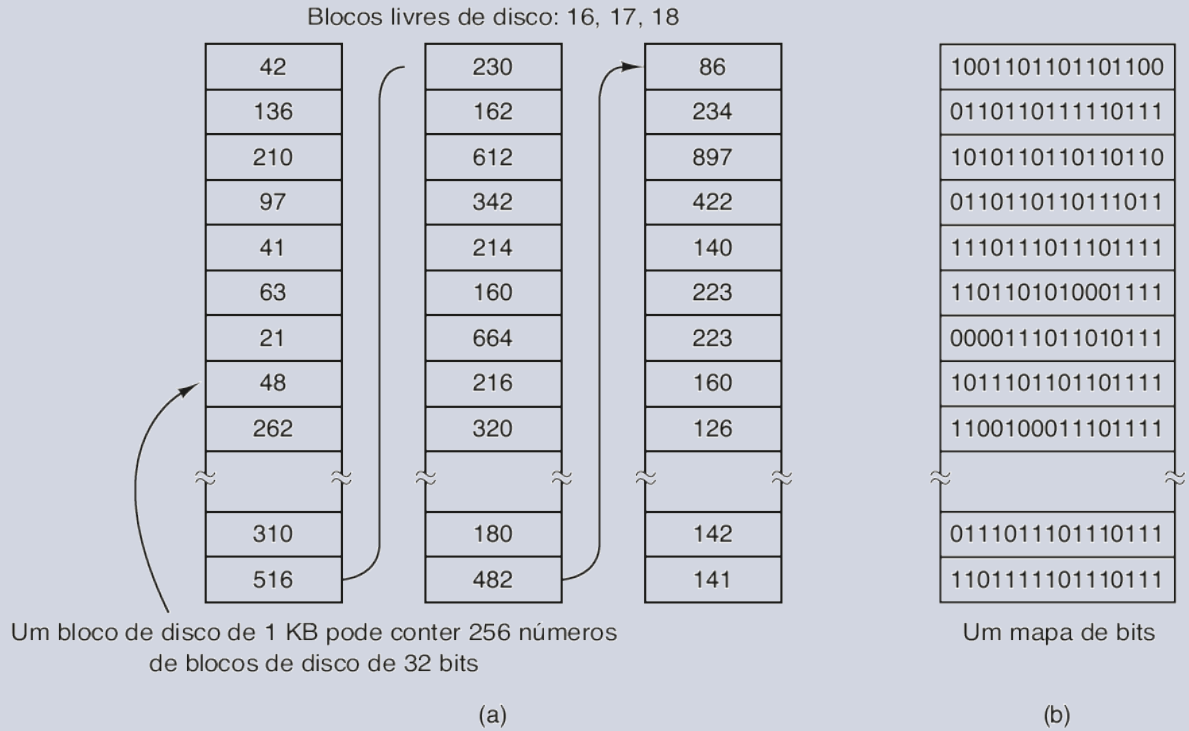
- Feita a opção de armazenar arquivos em blocos de tamanho fixo, devemos decidir qual tamanho o bloco terá. Dado o modo como os discos são organizados, o setor, a trilha e o cilindro são candidatos óbvios para a unidade de alocação. Em um sistema de paginação, o tamanho da página também é um argumento importante.

Gerenciamento e otimização de sistemas de arquivos

- Uma vez que um tamanho de bloco tenha sido escolhido, a próxima questão é como monitorar os blocos livres. Dois métodos são amplamente usados, como mostrado na figura. O primeiro consiste em usar uma lista encadeada de blocos de disco, com cada bloco contendo tantos números de blocos livres de disco quantos couberem nele. A outra técnica de gerenciamento de espaço livre é o mapa de bits. Um disco com n blocos exige um mapa de bits com n bits.

Gerenciamento e otimização de sistemas de arquivos

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



Desempenho do sistema de arquivo

- O acesso ao disco é muito mais lento do que o acesso à memória. Se apenas uma única palavra for necessária, o acesso à memória será da ordem de um milhão de vezes mais rápido que o acesso ao disco. Como consequência dessa diferença em tempo de acesso, muitos sistemas de arquivos foram projetados com várias otimizações para melhorar o desempenho.

1. **Cache de blocos (ou cache de buffer):** técnica mais comum, usada para reduzir os acessos ao disco.
2. **Leitura antecipada de blocos:** segunda técnica para melhorar o desempenho percebido do sistema de arquivos, que tenta transferir blocos para a cache antes que eles sejam necessários para aumentar a taxa de acertos.
3. **Redução do movimento do braço do disco:** reduz o montante de movimento do braço do disco colocando blocos que têm mais chance de serem acessados em sequência próximos uns dos outros, de preferência no mesmo cilindro.