

Sistemas operacionais

Aula 6 - Sistema de arquivos

INTRODUÇÃO



Um sistema operacional tem por finalidade permitir que os usuários do computador executem aplicações, como editores de texto, jogos, reprodutores de áudio e vídeo etc. Essas aplicações processam informações como textos, músicas e filmes, armazenados sob a forma de arquivos em um disco rígido ou outro meio.

Esta aula apresenta a noção de arquivo, suas principais características e formas de acesso, a organização de arquivos em diretórios e as técnicas usadas para criar e gerenciar arquivos nos dispositivos de armazenamento.

OBJETIVOS



Descrever os atributos de um arquivo.

Distinguir as organizações lógicas e físicas de arquivos.

Identificar os métodos de acesso aos arquivos.

GERÊNCIA DE ARQUIVOS

Iniciaremos nossa aula com um simples questionamento: Você sabe o que é um arquivo?

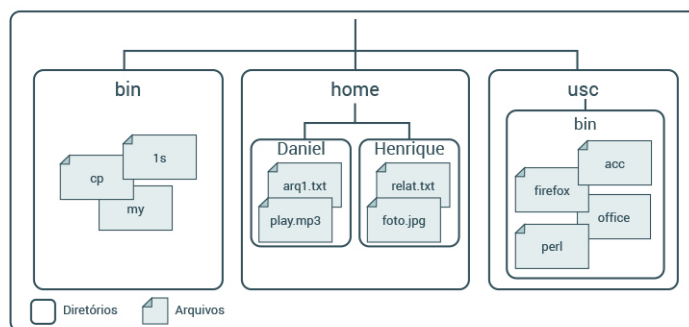


Fonte da Imagem: M.Stasy / Shutterstock

Um arquivo pode ser entendido como um conjunto de dados armazenados em um dispositivo físico não volátil, com um nome ou outra referência que permita sua localização posterior.

Do ponto de vista do usuário e das aplicações, o arquivo é a unidade básica de armazenamento de informação em um dispositivo não volátil, pois para eles não há forma mais simples de armazenamento persistente de dados.

Arquivos são extremamente versáteis em conteúdo e capacidade: podem conter desde um texto ASCII com alguns bytes até sequências de vídeo com dezenas de gigabytes, ou mais. Como um dispositivo de armazenamento pode conter milhões de arquivos, estes são organizados em estruturas hierárquicas denominadas *diretórios*, conforme ilustrado a seguir.



Arquivos organizados em diretórios dentro de um dispositivo

O Sistema de Arquivos pode ser entendido então como a organização física e lógica dos dados armazenados de forma persistente em um dispositivo físico não volátil.

Existem vários sistemas de arquivos nos Ss.Os. dentre os quais podemos citar FAT e NTFS do Windows.

ATRIBUTOS

Como vimos, um arquivo é uma unidade de armazenamento de informações que podem ser dados, código executável etc.

Cada arquivo é descrito por atributos, cujo conjunto varia de acordo com o sistema e arquivos utilizado. Os atributos mais usuais são:

Nome

String de caracteres que identifica o arquivo para o usuário, como "foto1.jpg", "relatório.pdf", "hello.c" etc.

Tipo

Indicação do formato dos dados contidos no arquivo, como áudio, vídeo, imagem, texto etc. Muitos sistemas operacionais usam parte do nome do arquivo para identificar o tipo de seu conteúdo, na forma de uma extensão: ".doc", ".jpg", ".mp3", etc.

Tamanho

Indicação do tamanho do conteúdo do arquivo, em bytes ou registros.

Datas

Para fins de gerência, é importante manter as datas mais importantes relacionadas ao arquivo, como suas datas de criação, de último acesso e de última modificação do conteúdo.

Proprietário

Em sistemas multiusuários, cada arquivo tem um proprietário, que deve estar corretamente identificado.

Permissões de acesso

Indicam que usuários têm acesso àquele arquivo e que formas de acesso são permitidas (leitura, escrita, remoção etc.).

Localização

Indicação do dispositivo físico onde o arquivo se encontra e da posição do arquivo dentro do mesmo.

Outros atributos

Vários outros atributos podem ser associados a um arquivo, por exemplo, para indicar se é um arquivo de sistema, se está visível aos usuários, se tem conteúdo binário ou textual etc. Cada sistema de arquivos normalmente define seus próprios atributos específicos, além dos atributos usuais.

A Tabela, a seguir, mostra algumas extensões de arquivo e seu tipo:

Extensão	Tipo de arquivo	Função
Exe, com, bin	Executável	Programa em linguagem de máquina pronto para executar.
Obj, o	Objeto	Programa compilado em linguagem de máquina não linkado.
C, cpp, pas, asm	Código fonte	Código fonte do programa em diversas linguagens.
Bat, sh	Batch	Comandos para o interpretador de comandos do S.O.
Txt, doc	Texto	Dados de processadores de texto.
Lib, dll, a	Biblioteca	Bibliotecas de rotinas para linkagem estática ou dinâmica.
Ps, dvi, gif, bmp	Print ou view	Binários ou ASCII em formato de impressão ou visualização.
Arc, zip, arj, tar	Comprimidos	Conjunto de arquivos comprimidos para armazenamento.

OPERAÇÕES

As aplicações e o sistema operacional utilizam um conjunto de operações para manipular arquivos.

As operações básicas, envolvendo arquivos, são:



Criar

A criação de um novo arquivo implica em alocar espaço para ele no dispositivo de armazenamento e definir seus atributos (nome, localização, propriedade, permissões de acesso etc.).



Abrir

Antes que uma aplicação possa ler ou mover dados em um arquivo, ela deve solicitar ao sistema operacional a "libertação" desse arquivo. O sistema só então verifica se o arquivo existe, verifica se as permissões associadas ao arquivo permitem aquele acesso, localiza seu conteúdo no dispositivo de armazenamento e cria uma referência para ele na memória da aplicação.



Leitura

Permite transferir dados presentes no arquivo para uma área de memória da aplicação.



Escrever

Permite transferir dados na memória da aplicação para o arquivo no dispositivo físico; os novos dados podem ser adicionados no final do arquivo ou sobrescrever dados já existentes.



Mudar atributos

Para modificar outras características do arquivo, como nome, localização, propriedade, permissões etc.



Fechar

Após concluir o uso do arquivo, a aplicação deve informar ao sistema operacional que o mesmo não é mais necessário, a fim de liberar as estruturas de gestão do arquivo na memória do núcleo.



Remover

Para eliminar o arquivo do dispositivo, descartando seus dados e liberando o espaço ocupado por ele.

Quanto à forma de organização dos registros de um arquivo, existem várias possibilidades. Veremos algumas formas a seguir:

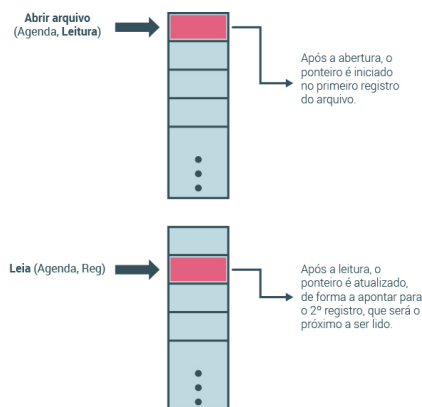
ORGANIZAÇÃO SEQUENCIAL

Neste tipo de arquivo, os registros são acessados, quer para leitura quer para escrita, de forma sequencial, isto é, a escrita de um registro só é feita após o último registro escrito e a leitura de um registro só é possível após todos os registros anteriores terem sido lidos.

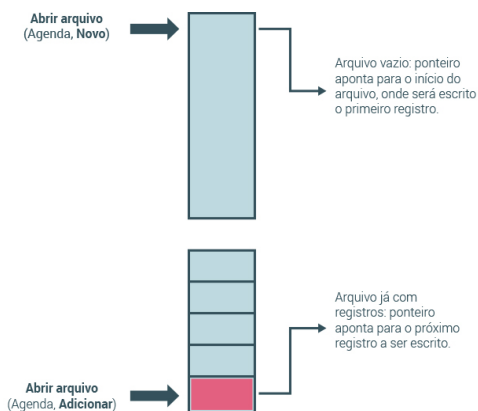
Existe um ponteiro "invisível" que assinala para o registro corrente e que é incrementado de forma automática após cada leitura ou escrita. Dizemos que o ponteiro é invisível porque não é necessário lidar com ele no algoritmo. O sistema de arquivos da linguagem que se utilizar (e que oferecer organização sequencial de arquivos) lida com o ponteiro automaticamente a cada operação de abertura, leitura ou escrita do arquivo.

Quando abrimos um arquivo, devemos informar se desejamos ler ou escrever e, a partir de então, só podemos realizar esse tipo de operação (leitura ou escrita) até o fechamento do arquivo.

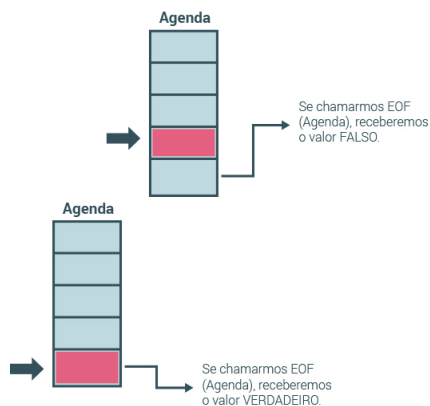
O esquema mostrado, a seguir, ilustra o funcionamento das operações de leitura em um arquivo sequencial chamado Agenda:



Na escrita, o ponteiro pode ser posto no início do arquivo (se for um arquivo novo) ou no final do arquivo (se quisermos acrescentar registros aos já existentes). A diferenciação é feita no comando de abertura. Veja:



Para lidar com arquivos sequenciais, existe uma função lógica (Verdadeiro/Falso) indicando se o ponteiro chegou ao fim do arquivo (na leitura). Chamamos essa função de **EOF (glossário)**. Por exemplo, imagine as seguintes situações mostradas a seguir:



Essa função é usada como critério de parada em repetições para testar se já atingimos o fim do arquivo sequencial.

Fazer um algoritmo para ler um nome e procurar as demais informações desse nome em um arquivo sequencial de informações pessoais, por exemplo. Veja, a seguir:

```

1 Programa Localiza
2 Declaração de tipos
3   RegAgenda = Registro
4   Nome : string
5   Endereco : string
6   Telefone : string
7   fim_registro
8   ArqAgenda = Arquivo sequencial de RegAgenda
9 Declaração de variáveis
10  Agenda : ArqAgenda
11  NomeProcurado : string
12  RegistroLido : RegAgenda
13 Inicio
14  Leia (NomeProcurado)
15  Abrir arquivo (Agenda, Leitura)
16  Repita
17    Leia (Agenda, RegistroLido)
18    até (RegistroLido.Nome = NomeProcurado) OU (EOF (Agenda))
19    Se (RegistroLido = NomeProcurado)
20    então
21      Escreva (RegistroLido.Nome, RegistroLido.Endereco, RegistroLido.Telefone)
22    senão
23      Escreva ( 'Não existe o nome procurado')
24  Fechar (Agenda)
25 Fim
26

```

ORGANIZAÇÃO DIRETA

Neste tipo de arquivo, podemos acessar um determinado registro no meio do arquivo pelo seu número (posição do registro dentro do arquivo). O arquivo, uma vez aberto, aceita tanto operações de leitura quanto de escrita.

Por exemplo, vamos supor que temos um arquivo de alunos com informações sobre nome, endereço, número do aluno e telefone. Suponha que o número corresponda à posição do registro do respectivo aluno dentro do arquivo.

Precisaremos de um algoritmo para ler as informações do aluno, dado seu número, ficaria bem mais simples que com arquivos sequenciais. Vejamos:

```

1 Programa ProcuraAluno
2 Declaração de tipos
3   RegAluno = Registro
4   Nome : string
5   Endereço : string
6   Número : inteiro
7   Telefone : string
8   fim_registro
9   ArqAluno = Arquivo direto de RegAluno
10 Declaração de variáveis
11   Aluno : ArqAluno
12   NumeroProcurado : inteiro
13   RegistroLido : RegAluno
14 Início
15   Leia (NumeroProcurado)
16   Abrir arquivo (Aluno)
17   Leia (Aluno [NumeroProcurado], RegistroLido)
18   Escreva (RegistroLido.Nome, RegistroLido.Endereço, RegistroLido.Telefone)
19   Fechar (Aluno)
20 Fim
21

```


ORGANIZAÇÃO INDEXADA

Neste tipo de arquivo, existe um local do registro chamado *campo chave* que permite o acesso a um registro determinado. Agora não temos mais um índice necessariamente numérico, como nos arquivos diretos.

Para podermos utilizar qualquer campo como índice (campo chave), o sistema de gerenciamento de arquivos da linguagem (que é transparente para o usuário) utiliza outro arquivo, chamado *arquivo de índice*, que é **ordenado** pelo campo chave. Esse arquivo de índice, possui um campo com a chave e outro com o número do registro correspondente àquela chave no arquivo principal.

Por exemplo, suponha que temos um arquivo com nome, endereço e telefone, e o campo chave é *nome*. Observe como seria a organização para os dados:

Arquivo Principal			Arquivo de Índice	
João	393-1111	R. Açu, 2	2	Ana
Ana	934-2222	R. Ita, 234	3	Carlos
Carlos	854-0000	R. Leda, 8/102	1	João
José	645-1111	R. Itua, 34	6	Jonas
Vítor	222-3333	Pç. Onze, 99	4	José
Jonas	935-1234	Av. Passos, 4/211	5	Vítor


Ordenado

Note que o arquivo de índice é ordenado. Assim, quando precisa localizar um registro no arquivo principal, o sistema de arquivos percorre primeiro o de índice, procurando pelo nome chave. Após encontrar a chave, pega o índice (que representa a posição) do registro completo referente àquela chave no arquivo principal.

Imagine que queremos mudar o endereço de um determinado nome de uma agenda. Veja como podemos fazer com arquivo indexado:


```

1 Programa MudaEndereco
2 Declaração de tipos
3   RegAgenda = Registro
4     Nome : string
5     Endereco : string
6     Telefone : string
7   fim_registro
8   ArqAgenda = Arquivo indexado de RegAgenda [ Nome]
9 Declaração de variáveis
10  Agenda : ArqAgenda
11  NomeProcurado : string
12  NovoEndereco : string
13  RegistroLido : RegAgenda
14 Início
15   Leia (NomeProcurado)
16   Leia (NovoEndereco)
17   Abrir arquivo (Agenda)
18   Leia (Agenda[NomeProcurado], RegistroLido)
19   RegistroLido.Endereco <- NovoEndereco
20   Escreva (Agenda[NomeProcurado], RegistroLido)
21   Fechar (Agenda)
22 Fim
23

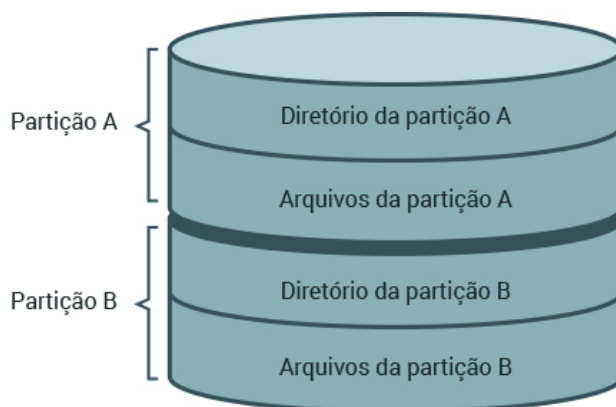
```

ORGANIZAÇÃO LÓGICA

O sistema de arquivos é quebrado em *partições* conhecidas como *volumes*.

Existe pelo menos uma partição e é nessa estrutura de mais baixo nível que os arquivos efetivamente residem. Um mesmo dispositivo físico (uma unidade de disco magnético, por exemplo) pode conter mais de uma partição, cada uma delas tratada como um dispositivo virtual.

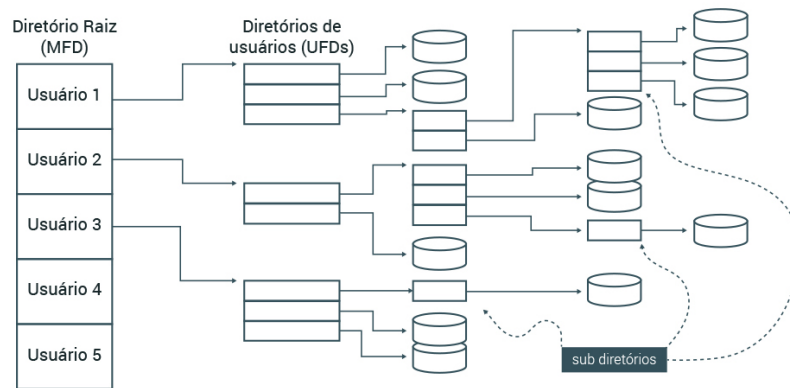
Informações sobre os arquivos são guardadas em entradas no *diretório de dispositivos* ou *tabela de volumes*. O diretório de dispositivos, também conhecido simplesmente como *diretório*, pode ser visto como uma estrutura de dados contendo informações sobre os arquivos tais como localização física, nome, atributos e demais informações que o sistema queira guardar sobre os arquivos. Veja um exemplo de disco com duas partições:



Nas organizações de diretórios mais antigas, todos os arquivos estavam contidos em um só diretório. Isso introduzia limitações no número de arquivos e, principalmente, na nomeação dos arquivos, já que o nome deve ser único dentro do diretório. Assim, os diretórios evoluíram para estruturas com diversos níveis.

Esses **diretórios (glossário)** são, normalmente, implementados como estruturas de árvores. Na estrutura principal ou *Master File Directory* (MFD), existem ponteiros para os diretórios de cada usuário ou *User File Directories* (UFDs) que, por sua vez, possuem apontadores para os demais subdiretórios, que podem ser criados por cada usuário e assim por diante, em uma estrutura que culmina pela referência dos arquivos físicos propriamente ditos.

Veja, a seguir, um exemplo de implementação de diretório de arquivos em árvore:



DIRETÓRIOS COMPARTILHADOS

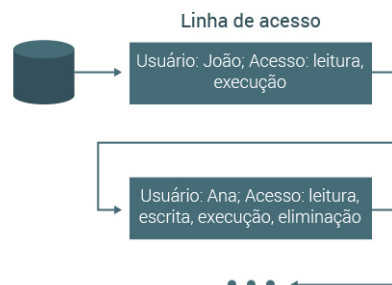
São estruturas que permitem que subdiretórios comuns sejam compartilhados. Um arquivo compartilhado existe no sistema em dois lugares, mas não como uma cópia, ou seja, só existe um arquivo físico. Assim, mudanças realizadas por um usuário são visualizadas por outro. Arquivos e subdiretórios compartilhados podem ser implementados de duas formas: **grupos de acesso** ou **listas de controle**.

Em ambos os mecanismos, é necessário estabelecer níveis de proteção diferentes para os diferentes usuários. Direitos de leitura, escrita, execução, criação e remoção de arquivos ou diretórios devem ser estabelecidos para cada usuário.

Grupos de acesso		Listas de controle
<p>No esquema de <i>listas de controle</i>, cada arquivo ou diretório a ser compartilhado possui uma lista associada, onde são relacionados os usuários e as operações que cada um desses usuários pode fazer.</p> <p>As desvantagens são o tamanho dessa estrutura que pode ser grande se o arquivo ou diretório for compartilhado por muitos usuários. Além disso, há o problema do acesso sequencial à lista de compartilhamento, que também é lento.</p>	X	<p>No esquema de <i>grupos de usuários</i>, os usuários que queiram compartilhar arquivos devem pertencer a um mesmo grupo. Quando da criação do arquivo é adicionada uma estrutura que descreve quais os níveis de proteção para: o dono (<i>owner</i>), o grupo (<i>group</i>) e todos (<i>all</i>).</p>

Para facilitar a compreensão, veja na ilustração como cada mecanismo funciona:

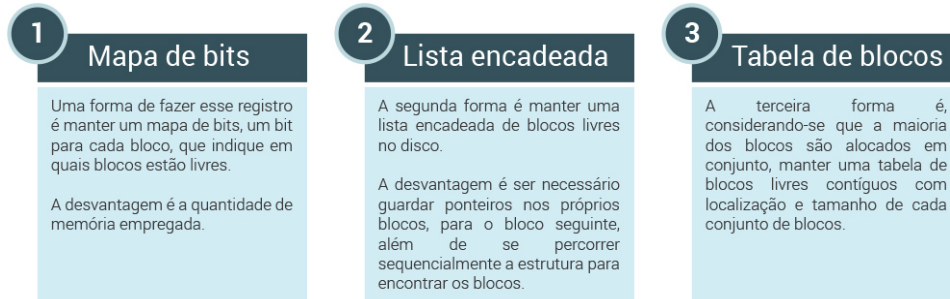
Grupo de usuários	
Nível	Acesso
Owner	Leitura, execução, escrita, eliminação
Grupo	Leitura, execução
All	Execução



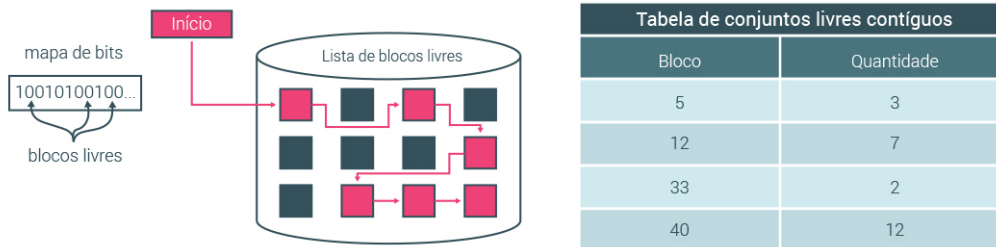
ORGANIZAÇÃO FÍSICA

Os sistemas precisam manter o controle da localização física dos arquivos nos dispositivos. Os dispositivos físicos são, normalmente, divididos em blocos e a estrutura de gerência de arquivos deve mapear em quais blocos físicos se encontra um determinado arquivo.

Para saber quais blocos estão livres, é preciso manter um registro de blocos livres. Esse registro pode ser feito de três formas. São elas:



Veremos, agora, essas três soluções:

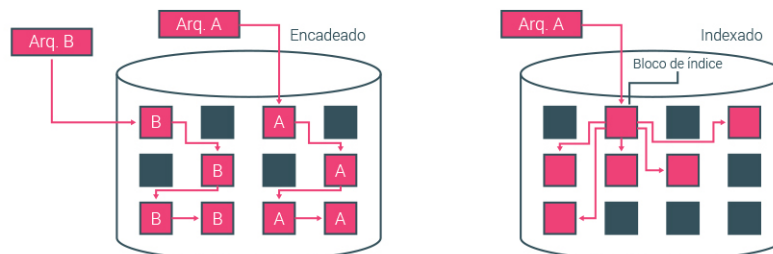


Para alocar espaço para um arquivo no disco, os primeiros sistemas procuravam por um espaço contíguo de blocos, de forma que só era necessário guardar o bloco inicial e o tamanho em blocos que o arquivo ocupava.

Naturalmente, isso é bastante ineficiente porque demanda escolher um bloco que tenha tamanho satisfatório para o arquivo (políticas de First-fit, Best-fit ou Worst-fit, como visto para partições de memória), além dos tradicionais problemas de fragmentação que ocorrem em longo prazo (depois de muitos arquivos terem sido criados e apagados). A solução foi criar esquemas de alocação não contígua.

Uma saída é a alocação encadeada, onde toda entrada de arquivo no diretório possui um apontador para um bloco inicial do disco que, por sua vez, aponta para o próximo bloco do arquivo e assim por diante. A desvantagem é, novamente, a obrigatoriedade do acesso sequencial.

Outra solução é o acesso indexado, em que um determinado bloco, chamado *bloco de índice*, contém os ponteiros para os demais blocos do arquivo no disco, veja a seguir:



ATIVIDADE

Para complementar seus estudos, faça uma pesquisa e veja as diferenças entre o sistema de Arquivos FAT e NTFS. Depois, digite no campo abaixo o que você descobriu.

Resposta Correta

Glossário

EOF

Abreviatura do termo em inglês *end of file*, que significa “fim de arquivo”.

DIRETÓRIOS

Na maioria dos sistemas, os diretórios são também tratados como arquivos, possuindo identificação e atributos, tais como proteção, nome do criador, data de criação, nível de acesso e compartilhamento etc.

Normalmente, nesses sistemas, é necessário incorporar ao nome do arquivo uma estrutura de nomes que indica qual é a localização do arquivo dentro da estrutura de diretórios. Esse adendo, que antecede o nome do arquivo propriamente dito, é conhecido como *caminho* ou *path* do arquivo.