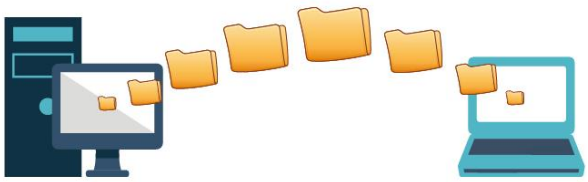
UNIDADE 3 – GERENCIAMENTO DE ARQUIVOS, ENTRADA E SAÍDA
MÓDULO 1 – ARQUIVOS E DIRETÓRIOS

01

1 - ATRIBUTOS DE ARQUIVOS

Uma das principais características dos sistemas operacionais é prover armazenamento persistente para os dados gerados pelos aplicativos de sistema ou do usuário, ou seja, disponibilizar um meio para que as informações sejam armazenadas de maneira permanente, independente do fornecimento de energia elétrica para o equipamento. Para prover esta funcionalidade, os Sistemas Operacionais modernos se baseiam na utilização de duas abstrações, os arquivos, que fisicamente são representados por uma sequência linear de bytes, e os diretórios, que são estruturas que permitem a organização dos arquivos.

De forma pragmática, **arquivos e diretórios são as principais estruturas utilizadas pelo Sistema Operacional para armazenar os dados em disco rígido.** Além dos dados pertencentes a cada arquivo, cabe ainda ao sistema operacional gerenciar o armazenamento de todos os metadados, ou atributos, associados a estas estruturas de dados, os métodos de acesso as informações dos arquivos e as operações básicas executadas sobre os mesmos, como a criação ou a exclusão.



Muitas destas características relacionadas a arquivos e diretórios, como os **atributos** relacionados ao nome, forma de implementação de métodos de acesso, as formas de definição dos tipos de arquivos e uma série de outras características relacionadas a arquivos e diretórios são diretamente vinculados a cada implementação de Sistema operacional. Estas e outras questões relacionadas a duas das principais estruturas de dados do Sistema Operacional, os arquivos e os diretórios, serão abordadas no decorrer deste módulo.

02

Quando um arquivo é criado, algumas informações que lhe são vinculadas são automaticamente geradas e gravadas em disco, são os chamados **“atributos”** do arquivo.

São diversos os atributos que podem ser associados aos arquivos: as informações sobre data e hora da criação, os dados sobre quem foi o usuário que criou o arquivo e sobre quem tem permissão de acesso são alguns exemplos. Uma lista contendo os atributos de arquivo utilizados nos principais sistemas operacionais do mercado é apresentada abaixo:

- Nome do arquivo;
- Número de série ou Identificador do arquivo;

- Dispositivo;
- Atributos de leitura e escrita;
- Tamanho do arquivo;
- Usuário dono do arquivo;
- Grupo dono do arquivo;
- Último acesso;
- Última modificação;
- Momento de Criação.

Nome do arquivo
O nome do arquivo é representado por uma sequência de caracteres que é exibida para o usuário e que segue as restrições de nomenclatura de cada sistema operacional.

Número de série ou Identificador do arquivo
O número de série ou identificador do arquivo diz respeito ao nome único utilizado para a identificação do arquivo pelo Sistema Operacional.

Dispositivo
O dispositivo identifica o hardware que contém o arquivo, já que um computador pode possuir múltiplos discos instalados.

Atributos de leitura e escrita
Os atributos de leitura e escrita fazem referência as permissões de execução das operações de leitura e/ou alteração dos arquivos.

Tamanho do arquivo
O tamanho do arquivo indica o espaço que o arquivo ocupa no dispositivo de armazenamento.

Usuário dono do arquivo
Todo arquivo tem um proprietário, que necessariamente é um dos usuários cadastrados no sistema operacional.

Grupo dono do arquivo
Além de um usuário, todo arquivo também tem um grupo que mantém permissões de proprietário sobre o arquivo.

Último acesso
Registra a data e hora do último acesso sofrido pelo arquivo.

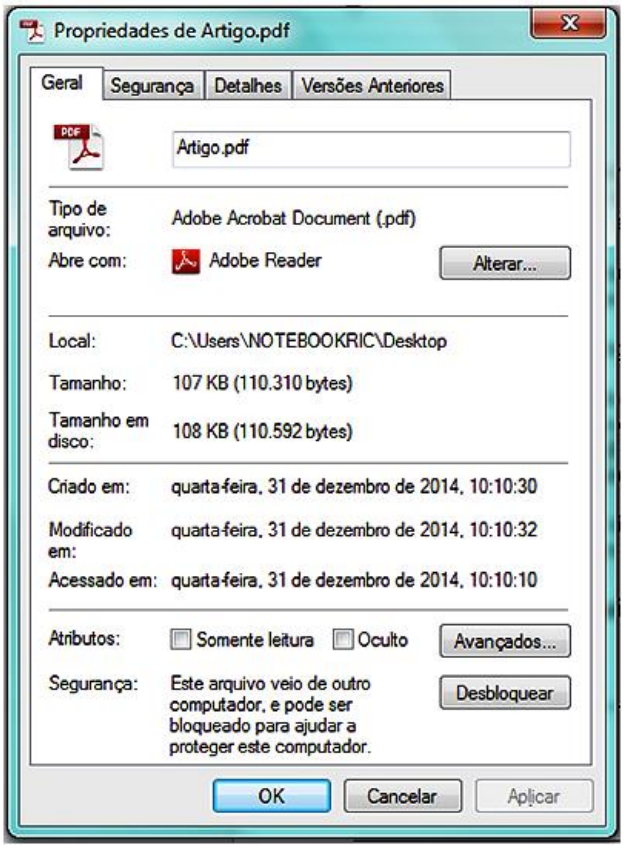
Última modificação
Registra a data e hora da última modificação realizada no arquivo.

Momento de Criação
Registra a data e hora da criação do arquivo.

03

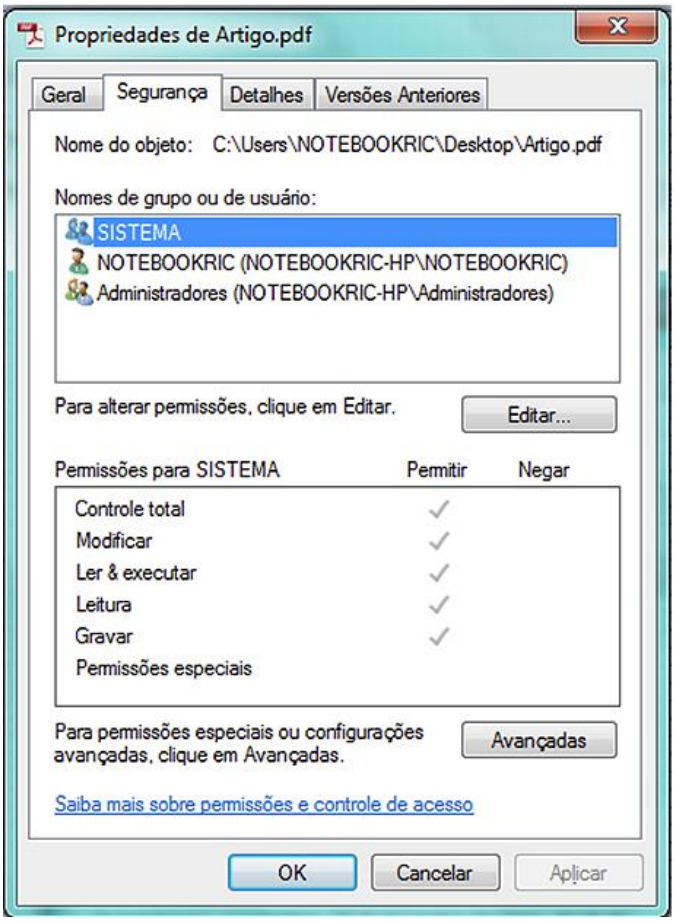
Apesar de estas serem características conhecidas, é importante ressaltar que não existe uma lista de atributos padrão, de modo que o conjunto de atributos associados a um determinado arquivo pode variar a depender da implementação do Sistema Operacional.

No MS Windows, por exemplo, é possível visualizar os atributos do arquivo clicando com o botão direito do mouse sobre o ícone que o representa e selecionando a opção “propriedades”. A figura abaixo representa a janela inicial de propriedades do arquivo do Windows, nela é possível observar informações como o nome do arquivo, o tipo, o local de armazenamento, o tamanho, a data e hora da criação, do último acesso e da última modificação.



04

A aba “segurança” completa os atributos do arquivo com as informações de controle de acesso, com a identificação dos usuários e grupos do sistema que possuem algum tipo de permissão sobre o arquivo, seja de leitura, modificação ou execução.



05

Já nas distribuições Linux, uma das formas mais conhecidas de se listar os atributos do arquivo é através da utilização do comando “*ls*”, fornecido pelo Sistema Operacional.

A execução do “*ls*” em associação com o parâmetro –“*l*”, faz com que sejam listados todos os arquivos e diretórios existentes dentro do diretório atual, exibindo os nomes dos arquivos, permissões de acesso, o tamanho de cada arquivo e a data e hora da última modificação, conforme pode ser visualizado na figura abaixo.

```
[guest@freelinuxconsole.info:2]$ ls -l
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  apropos
-rwxr-xr-x  1  root      wheel     bin/n.a.  2005/07/26 23:00:00  invaders
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  logname
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  news
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  uname
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  vi
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  view
-rwxr-xr-x  1  root      wheel     bin/n.a.  2003/11/05 12:00:00  which
[guest@freelinuxconsole.info:2]$
```

06

2 - Extensões de Arquivos e MIME Types

A extensão de um arquivo é um sufixo agregado ao nome do arquivo e que tem por função indicar o formato ao qual o arquivo está vinculado. Historicamente não é claro o momento exato do surgimento das extensões, entretanto, tem-se que a popularização do seu uso se deu com a publicação do Sistema Operacional Microsoft DOS.

Por padrão, as extensões de arquivo são separadas do nome base através da utilização de um ponto (“.”). Desta forma, para um dado arquivo de exemplo “RELATORIO1.DOC”, o termo “RELATORIO1” representaria o nome base e o termo “DOC” indicaria a extensão do arquivo.

Como comentado, cada extensão padrão associa um arquivo a um determinado formato, o que acaba por criar uma lista de extensões para classificação dos formatos dos arquivos. Documentos produzidos pela última versão do aplicativo Microsoft Word, por exemplo, são nomeados com o sufixo “docx”, já os arquivos criados a partir do aplicativo Corel Draw têm como sufixo “cdr”. Uma lista com algumas das extensões de arquivos mais conhecidas pode ser visualizada na tabela abaixo.

Extensão	Descrição
doc ou docx	Arquivos produzidos pelo aplicativo Microsoft Word
txt	Arquivos de texto puro produzido pelos mais variados aplicativos
exe	Arquivos executáveis do MS Windows
wav, mp3 e wma	Arquivos de áudio produzidos pelos mais variados aplicativos
odt	Padrão para documentos de texto do OpenDocument
sh	Arquivos de script utilizados nas distribuições Linux
bat	Arquivos de script executados no MS Windows
bmp, jpg, png, tif, gif	Padrões de arquivos de imagem produzidos nos mais variados aplicativos
Pdf	Padrão aberto para troca de documentos eletrônicos, inicialmente desenvolvido pela Desenvolvido pela Adobe Systems mas hoje mantido pela International Standards

	Organization (ISO)
avi, mpg, wmv	Arquivos de vídeo produzidos pelos mais variados aplicativos
mov	Formato multimídia utilizado pelo aplicativo Quicktime
ppt ou pptx	Arquivos de apresentação do Microsoft Powerpoint
xlx ou xlxs	Arquivos de planilha eletrônica do MS Excel
ods	Padrão para Arquivos de planilha eletrônica do OpenDocument
php	Arquivos de código fonte da linguagem php
zip	Formato de arquivos compactados no padrão ZIP

07

Além de permitir a identificação do formato do arquivo pelo sistema operacional, a utilização das extensões também facilita o manuseio dos arquivos por parte dos usuários, sobretudo daqueles arquivos cujas extensões são de conhecimento geral, como os formatos padrão de imagens, a exemplo do JPG e BMP, ou os formatos padrão de música, como o MP3.

Se em alguns sistemas operacionais, como o MS Windows, a utilização da extensão é obrigatória para que o SO possa identificar corretamente o aplicativo que irá tratar um determinado arquivo, em outros, como o Linux e o Mac OS, a utilização do sufixo é apenas uma convenção, não sendo uma premissa obrigatória do Sistema Operacional. Uma das principais alternativas adotadas pelas distribuições LINUX e pelo MAC OS para substituir a necessidade da extensão dos arquivos é a utilização dos tipos **MIME** (*Extensões Multi função para Correio de Internet*).

O MIME é uma informação resumida sobre o formato do arquivo e que se encontra embutida no início do próprio código do arquivo.

Originalmente, o MIME foi pensado como uma extensão do protocolo de e-mail, que tinha como principal funcionalidade permitir a troca pelos usuários dos mais variados tipos de arquivos. Apesar de a modelagem inicial visar o protocolo SMTP do correio eletrônico, o uso do tipo MIME (ou MIME Types) foi expandido e passou a ser utilizado nos protocolos de internet e também por alguns Sistemas Operacionais, como alternativa ao uso das extensões de arquivos.

Existe uma correlação entre os MIME Types e as extensões de arquivos, muito em função da manutenção da compatibilidade entre os sistemas Windows, que exercem o domínio entre os SO instalados em computadores domésticos, e os demais Sistemas Operacionais. A tabela abaixo apresenta alguns exemplos de sufixos com os MIME Types correlatos.

Sufixos	MIME Types
Doc	application/msword
Exe	application/octet-stream
Jpg	image/jpeg
Bmp	image/bmp
Mov	video/quicktime
Mp3	audio/mpeg3

08

3 - Operações básicas e métodos de acesso aos arquivos

Cada Sistema Operacional tem um conjunto próprio de operações que são permitidas em seus arquivos, com o objetivo de prover uma série de funcionalidades básicas, como a leitura e a escrita de novos dados. Entretanto, analisando diferentes implementações de SO, percebe-se que determinadas operações são encontradas com mais frequência. Uma lista com este grupo de operações e uma breve descrição é exibida abaixo.

• Criação

É a operação que inicia o ciclo de vida de um arquivo, promovendo a sua inclusão no dispositivo de armazenamento.

• Abertura

Operação necessária antes que se possa ler ou alterar efetivamente um dado arquivo, já que permite ao SO carregar os atributos do arquivo em memória principal.

• Leitura

Esta operação permite o acesso para leitura aos dados pertencentes a um arquivo.

• Escrita

Permite a inclusão de novos dados a um arquivo ou a alteração de informação já existente.

• Busca

Posiciona o ponteiro para uma determinada localização do arquivo de modo a permitir o acesso específico a uma localidade, seja para execução de uma operação de leitura ou de escrita.

• Encerramento

Quando não há mais operações a serem executadas sob o arquivo, é recomendável que se execute a operação de encerramento, ou fechamento, para que sejam liberados os recursos em memória principal alocados para viabilizar a execução de operações sobre o arquivo.

• Alteração do nome

Como o próprio nome já sugere, a operação é acionada quando o usuário necessita alterar o nome do arquivo.

• Remoção

Esta operação apaga todo o conteúdo do arquivo, promovendo a sua exclusão do dispositivo de armazenamento.

09

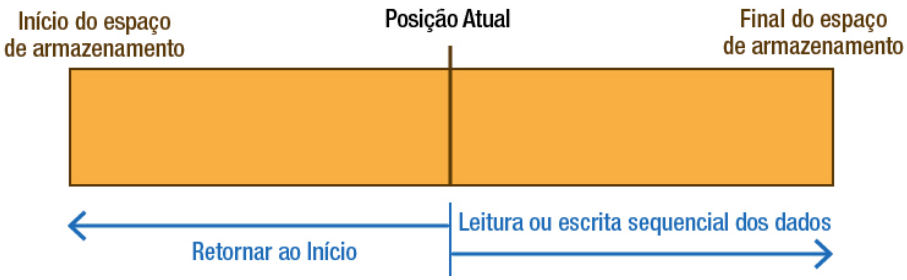
Métodos de Acesso aos Arquivos

Os métodos de acesso definem a forma como são realizadas as operações sobre os arquivos e a forma como as aplicações acessam os dados.

São três os principais métodos de acesso implementados nos sistemas de arquivo atuais:

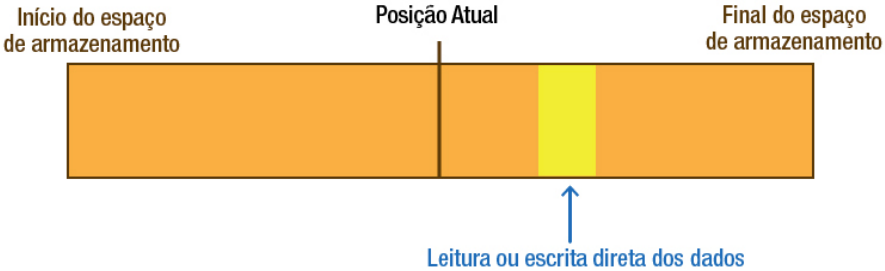
- o sequencial,
- o direto e
- o indexado.

No **acesso sequencial**, a operação de escrita de novos registros sempre ocorre ao final do arquivo e a operação de leitura sempre ocorre sequencialmente, vinculados à ordem em que os dados foram armazenados. Um desenho esquemático do método de operação é exibido na figura abaixo, o método sequencial é geralmente utilizado na gerência de acesso aos dispositivos de fita magnética.



10

Já no **acesso direto**, os registros podem ser acessados independentemente da ordem em que foram armazenados, desta forma o ponteiro é direcionado para a localização exata do arquivo antes que seja executada a operação desejada. Nesta modalidade, que é muito utilizada em discos, os arquivos são vistos como uma sequência ordenada de blocos de registros, conforme exibido na figura abaixo, que representa o método em análise.



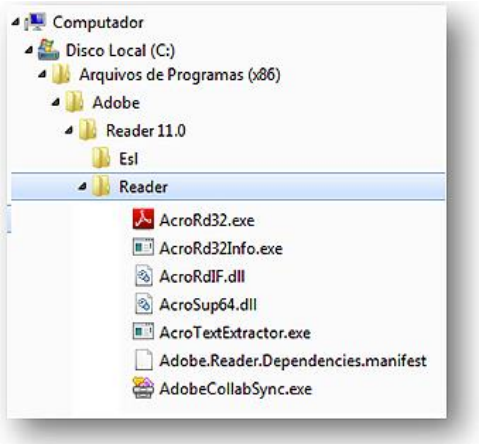
O método de **acesso indexado**, por sua vez, é construído sobre o conceito do acesso direto, só que tem a performance incrementada através da utilização de um índice para localização dos blocos de registros associados a um arquivo.

11

4 - DIRETÓRIOS

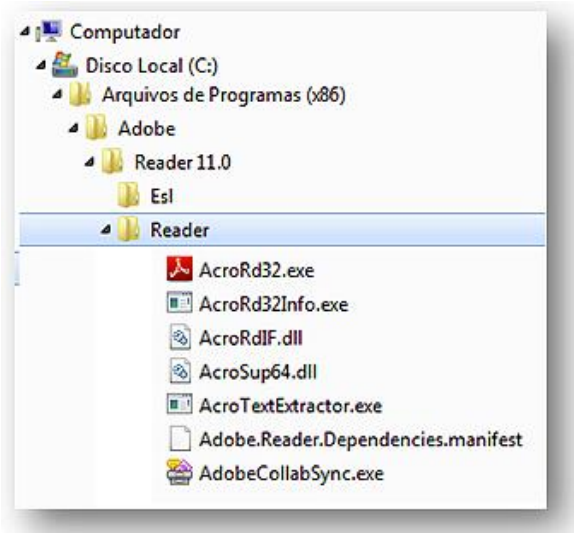
Os diretórios são estruturas que foram pensadas com dois objetivos principais, primeiro o de prover uma estrutura de organização dos arquivos por parte dos usuários, e segundo o de prover uma interface de nomenclatura que permitisse ao sistema operacional separar a organização lógica dos arquivos da sua localização física no disco.

Os diretórios tradicionalmente se organizam de forma hierárquica, formando uma espécie de árvore de diretórios. O exemplo exibido na figura abaixo traz uma representação da árvore de diretórios típica do Sistema Operacional Windows. Observe que o diretório “Disco Local (C:)” é o diretório raiz, ou seja, o primeiro nó da árvore, estando os demais diretórios organizados hierarquicamente abaixo deste, característica que está representada na figura através da utilização da indentação à direita.



12

Outra constatação importante acerca do exemplo dado é a existência recursiva de diretórios dentro de diretórios. Observe que o diretório “Arquivos de Programas (x86)” engloba o “Adobe”, este último engloba o “Reader 11.0”, e assim sucessivamente até que seja alcançado o último nível de diretórios. É justamente a organização desta árvore de diretórios que define o que é chamado de **endereço absoluto** do arquivo, ou seja, o caminho para que o usuário possa localizá-lo no sistema de arquivos do equipamento.



Seguindo a lógica proposta e tomando como exemplo o arquivo "AcroRd32.exe" e a estrutura de diretórios exposta na figura, o endereço absoluto para acesso a este arquivo seria:

C:\Arquivos de Programa (x86)\Adobe\Reader 11.0\Esl\Reader\AcroRd32.exe

É importante ressaltar que a utilização da barra invertida ("\") para a separação dos diretórios é uma característica específica do SO Windows, outros sistemas operacionais podem, portanto, utilizar padrão diferente, como, por exemplo, as distribuições Linux, que utilizam como separador a barra normal ("/").

13

Dentre outras motivações, os endereços absolutos se tornam essenciais em função da possibilidade de se ter mais de um arquivo com o mesmo nome armazenado numa mesma estrutura física de disco, mas não dentro de um mesmo diretório. A utilização dos endereços absolutos faz com que se mantenha a singularidade nos nomes dos arquivos a partir da definição não apenas do seu nome, da sua estrutura completa, incluindo a árvore de diretórios a que pertence.

Cotidianamente, as implementações de sistemas operacionais costumam definir uma série de diretórios padrões para facilitar a experiência do usuário na operação do SO. Um dos exemplos mais conhecidos é o esquema de **diretórios**, utilizado na maioria das distribuições Linux, o qual é exibido a seguir.

Assim como o Linux, o MS Windows também possui alguns diretórios com objetivo definido, são eles:

- Arquivos de Programa;
- Windows;
- Usuários.

Arquivos de Programa
Diretório que agrupa as pastas dos aplicativos instalados pelo usuário.

Windows
Arquivos referentes ao sistema operacional, como arquivos de sistema, fontes, drivers etc.

Usuários
Agrupa os arquivos pessoais relacionados a cada um dos usuários cadastrados no sistema operacional, incluindo documentos, imagens e downloads realizados.

14

RESUMO

Arquivos e diretórios são as principais estruturas utilizadas pelo Sistema Operacional para armazenar os dados em disco rígido, além dos dados pertencentes a cada arquivo, cabe, ainda, ao sistema operacional gerenciar o armazenamento de todos os atributos associados a estas estruturas de dados, os métodos de acesso às informações dos arquivos e as operações básicas executadas sobre os mesmos, como a criação ou a exclusão.

Os atributos dos arquivos estão associados a informações que caracterizam os arquivos e que são armazenadas pelo sistema operacional, a exemplo do nome, dos atributos de leitura e escrita, os dados sobre quem foi o usuário que criou o arquivo ou sobre quem tem permissão de acesso. Apesar das diferentes implementações de sistemas operacionais utilizarem o conceito de atributos de arquivos, é importante ressaltar que não existe uma lista de atributos padrão, de modo que o conjunto de atributos associados a um determinado arquivo pode variar a depender do SO.

Além dos atributos padrão, tradicionalmente os arquivos possuem um sufixo agregado ao seu nome que tem por função indicar o formato ao qual o arquivo está vinculado. Além de permitir a identificação do formato do arquivo pelo sistema operacional, a utilização das extensões também facilita o manuseio dos arquivos por parte dos usuários, sobretudo daqueles arquivos cujas extensões são de conhecimento geral.

Entretanto, se em alguns sistemas operacionais, como o MS Windows, a utilização da extensão é obrigatória, em outros, a utilização do sufixo é apenas uma convenção, e uma das principais alternativas adotadas pelas distribuições LINUX e pelo MAC OS para substituir a necessidade da extensão dos arquivos é a utilização dos tipos “MIME”, que é uma informação resumida sobre o formato do arquivo e que se encontra embutida no início do próprio código do arquivo.

Cada Sistema Operacional tem um conjunto próprio de operações que são permitidas em seus arquivos, com o objetivo de prover uma série de funcionalidades básicas, como a leitura e a escrita de novos dados, entretanto, determinadas operações são encontradas com mais frequência, como as operações de criação, abertura, leitura, escrita, busca, encerramento, alteração do nome e remoção do arquivo. Além das operações básicas, os métodos de acesso também são vinculados às implementações de SO. Os métodos de acesso definem a forma como são realizadas as operações sobre os arquivos e a forma como as aplicações acessam os dados, sendo três os principais métodos: o sequencial, o direto e o indexado.

Além dos arquivos, os Sistemas Operacionais precisam lidar também com os diretórios, que são estruturas que foram pensadas com os objetivos de permitir a organização dos arquivos por parte dos usuários e de prover uma interface de nomenclatura que permitisse ao sistema operacional separar a organização lógica dos arquivos da sua localização física no disco.

1 - Sistemas de arquivos e estruturação dos Discos Rígidos

No módulo anterior, foram apresentadas uma série de funcionalidades providas ao usuário pelo Sistema Operacional, relacionadas sobretudo a arquivos e diretórios. Entretanto, enquanto para os usuários o importante é a forma como os arquivos são nomeados e quais operações podem ser executadas sobre estes arquivos, para o sistema operacional a questão mais importante é como armazenar fisicamente os arquivos e diretórios e como gerenciar o espaço em disco de forma eficiente, eficaz e segura. Este módulo dará ênfase justamente nesta questão, promovendo uma discussão sobre a estruturação, funcionamento e forma de implementação dos sistemas de arquivos encontrados nos principais sistemas operacionais do mercado.

O **sistema de arquivos** é o elemento que fornece ao programador uma série de interfaces que permitem que os aplicativos que estão sendo codificados tenham acesso às funções de armazenamento e recuperação no disco rígido, independentemente do fabricante e do modelo do hardware que está sendo utilizado pelo computador.

Os discos rígidos, por sua vez, são os dispositivos de memória responsáveis por armazenar a maior parte dos arquivos e diretórios em um computador e, por conseguinte, por hospedar a implementação do sistema de arquivos do SO. A grande maioria dos discos pode ser segmentada em uma ou mais partições, de forma a permitir a configuração de múltiplos sistemas de arquivos em um mesmo disco. Para poder gerenciar e controlar a existência concorrente destas partições, o disco se vale do **MBR**, ou *Master Boot Record*. Apesar de poder conter múltiplas partições, é importante ressaltar que cada disco possui apenas uma MBR.

02

O ***Master Boot Record*** está presente no primeiro setor do disco, sendo o responsável pela inicialização do sistema.

Na prática, o MBR é o local que é lido e carregado pela BIOS quando o equipamento é ligado. É durante o processo de particionamento do disco que é criado o MBR, estrutura composta por três elementos:

- uma pequena quantidade de código de inicialização, chamado de ***"master boot code"***;
- a **tabela de partições** do disco;
- e a **assinatura** do disco.

O ***master boot code*** tem como principal função promover o início do sistema. Para isso, executa uma varredura na tabela de partições visando localizar o setor de início da partição ativa e, em seguida, carrega uma cópia do setor de inicialização, originalmente presente no disco, na memória principal, transferindo o controle para o código executável recentemente carregado em memória.

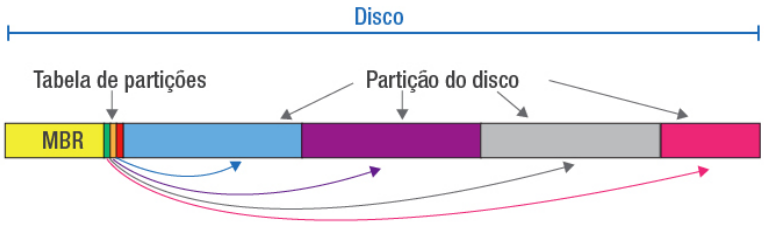
A necessidade de se ter uma **tabela de partições** se dá em função da possibilidade da existência de múltiplas partições em um único disco e da necessidade de se identificar a localização física de cada uma destas partições.

Além de conter o código de inicialização, as partições são, na prática, uma estrutura de organização do disco onde, após a configuração do sistema de arquivos a ser utilizado, serão armazenados os dados de usuário e sistema.

O último componente da MBR é a **assinatura do disco**, que nada mais é do que um identificador único que individualiza cada um dos discos para o sistema operacional.

03

Observe a figura a seguir, que representa o desenho esquemático da configuração padrão de um disco rígido, com a posição da MBR, da tabela de partições e da relação desta com as partições do disco.



Como já explanado, em cada partição pode ser implementado um diferente sistema de arquivos, sendo que os mais conhecidos são aqueles que integram os principais sistemas operacionais do mercado, a exemplo do *File Allocation Table (FAT)* e do *New Technology File System (NTFS)*, que são utilizados pelos sistemas Windows, e das diferentes versões do *Extended File System (EXT)*, que compõe a maior parte das distribuições Linux.

Cada um destes sistemas de arquivos será estudado com mais detalhes no decorrer da disciplina.

04

2 - File Allocation Table (FAT)

O sistema de arquivos **FAT - *File Allocation Table*** foi desenvolvido no final dos anos 1970, tendo se tornado conhecido no início dos anos 80, após a sua inclusão no sistema operacional MS DOS. A sua primeira versão foi o FAT12, sistema de arquivos voltado para discos flexíveis, ou disquetes. Atualmente, existem mais três versões, o FAT16 o FAT32 e o extFAT, ou FAT64, sendo que **a diferença básica entre elas é o tamanho, em bits**, das entradas na estrutura da tabela de alocação de arquivos em disco. A arquitetura *File Allocation Table* foi desenvolvida originalmente para computadores aderentes à arquitetura IBM PC.

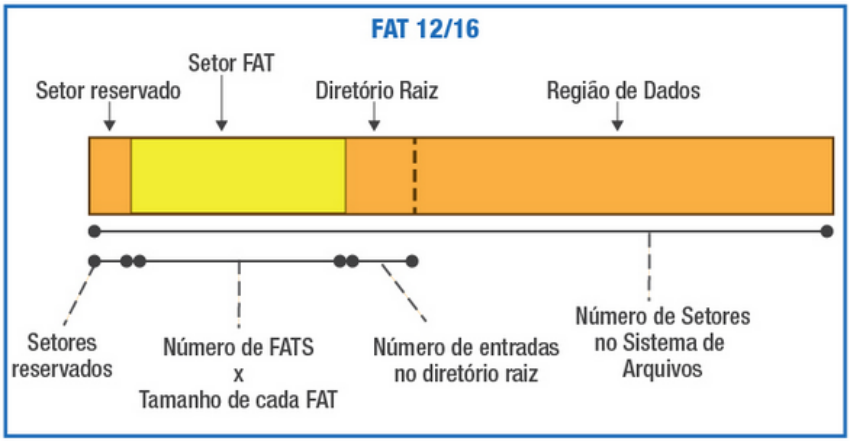
O **sistema de arquivos FAT** é formado por quatro diferentes quadrantes, que são organizados no volume do disco na seguinte sequência:

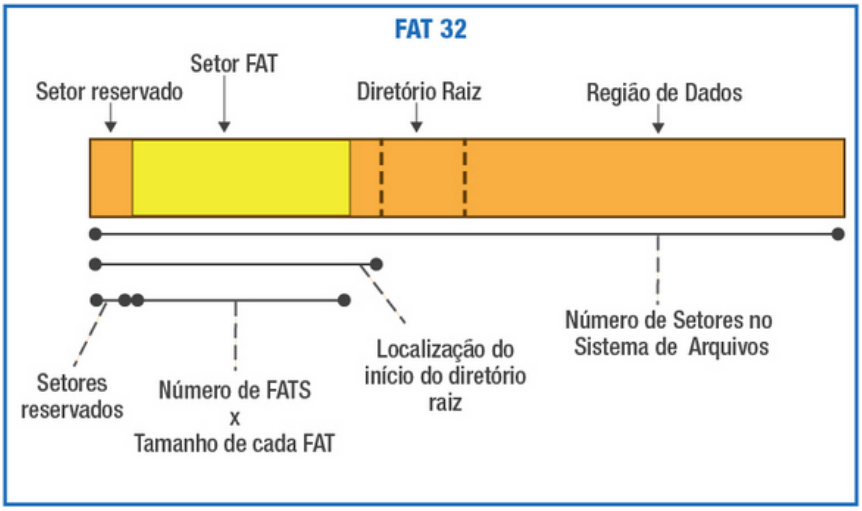
- Setor reservado ou de *"boot"*,
- Setor FAT,
- Região do Diretório Raiz e

- Região de dados dos arquivos e diretórios.

É importante ressaltar que a Região do Diretório Raiz não existe na versão FAT32 e extFAT.

Os modelos esquemáticos das diferentes versões do sistema de arquivos *File Allocation Table* são exibidos abaixo.





Setor de boot

Originalmente, o **setor de *boot*** está localizado na primeira seção do volume de disco e é o responsável por armazenar as informações sobre todo o restante da organização do sistema de arquivos, como, por exemplo, o tamanho dos setores do disco, a quantidade de setores por cluster, a versão do sistema de arquivos (FAT12, FAT16 etc.), o rótulo do volume e a identificação do volume.

Setor FAT

Outra estrutura importante na composição do volume de disco é o **setor FAT**, elemento responsável por armazenar a localização de todos os clusters do disco, se tornando o único meio para se localizar arquivos e diretórios armazenados. As tabelas FAT, por sua vez, são similares a listas ordenadas de dados, sendo que cada entrada lógica da tabela é traduzida para o endereço físico de um determinado conjunto de dados armazenado em disco.

Diretório raiz
O **diretório raiz** é o diretório primário de qualquer disco formato com o padrão FAT e, diferentemente dos demais diretórios onde são armazenados os dados, tem que obrigatoriamente estar presente no sistema de arquivos. Além disso, nas versões FAT12 e FAT16, o diretório raiz está posicionado em um local fixo no disco, logo após a última tabela de alocação de arquivos, e tem um tamanho padrão pré-definido. Diferentemente das versões mais antigas, o diretório raiz na versão FAT32 pode ter um tamanho variável, de forma similar a qualquer outro diretório existente no disco.

Região de dados de arquivos e diretórios
É o último setor encontrado nos volumes formatados com o padrão FAT. A **região de gravação de arquivos e diretórios**, como o próprio nome já diz, é responsável por armazenar os dados de todos os demais arquivos e diretórios existentes no disco.

05

Existem dois padrões de nomenclatura utilizados nos sistemas de arquivo FAT, os **nomes curtos** e os **nomes longos**. Os nomes curtos de arquivos e diretórios são limitados a oito caracteres, não contando os três caracteres associados a extensão dos arquivos e o caractere “.”, que é utilizado como separador. Além disso, o tamanho total do endereço absoluto dos arquivos é limitado a oitenta caracteres.

Além dos tradicionais caracteres alfanuméricos tradicionais (letras de A a Z e números de 0 a 9), no FAT os nomes curtos admitem a utilização dos seguintes símbolos especiais na nomenclatura dos arquivos:

- $ % ' - ^ ` ! ( ) { } # & _ @ ~
- Caracteres com código ASCII maiores do que 127.

A utilização dos nomes curtos esteve associada ao lançamento do Sistema Operacional MS DOS, que rodava sobre FAT16. Entretanto, com o desenvolvimento do SO Windows surgiu a necessidade de utilização de nomes de tamanho maior do que os oito caracteres utilizados no DOS.

O novo padrão utilizado, definido como nomes longos, permitia a utilização de 255 (duzentos e cinquenta e cinco) caracteres para definição do nome do arquivo e de 260 (duzentos e sessenta) caracteres para definição do endereço absoluto.

Apesar de manter a compatibilidade com os caracteres utilizados para criação dos nomes curtos, seis novos caracteres foram adicionados ao padrão de nomes longos: + [ ] , ; =.

06

Uma característica importante de ambos os padrões de nomenclatura é o fato de que todas as operações de busca realizadas tanto em nomes curtos como em longos são **"CASE INSENSITIVE"**, ou seja, não diferenciam maiúsculas de minúsculas.

Nesta linha, a busca por um arquivo através das entradas "Documento" ou "DOCUMENTO" trariam o mesmo resultado.

Outra observação relevante é que, mesmo com a introdução dos nomes longos nas versões mais recentes do sistema FAT, para toda operação de criação de um novo arquivo com nome longo também lhe é gerado e armazenado no disco um nome curto.

A necessidade da criação deste "atalho" de no máximo oito caracteres se deu para que se pudesse prover a compatibilidade entre as diferentes versões dos sistemas.

Apesar de se apresentar como um Sistema de arquivos leve e rápido, os padrões FAT12, FAT16 e FAT32 apresentam dois grandes problemas:

- o primeiro relacionado ao fato de que **não permitem o gerenciamento de permissões de acesso a arquivos ou pastas**,
- o segundo está relacionado ao **tamanho máximo do arquivo individual, que é limitado a 4 Gb na versão FAT32**.

Por conta disto, e visando aproveitar o melhor desempenho do padrão FAT em relação a sistemas mais complexos e pesados como o NTFS, a Microsoft lançou em 2006 o exFAT, ou FAT64, que é um sistema de arquivos voltado a dispositivos com menor capacidade de processamento e memória e que era fornecido com versão embarcada do Windows CE.

07

3 - New Technology File System (NTFS)

No início da década de 1990, a Microsoft resolveu investir no desenvolvimento de Sistemas Operacionais voltados ao ambiente dos *data centers*corporativos, visto que as versões do MS DOS e do Windows 3.x não tinham condições de competir em igualdade com as distribuições UNIX neste mercado.

Uma das principais fraquezas da versão Windows 3.x se concentrava justamente no seu sistema de arquivos FAT que, apesar do desempenho, **não provia com segurança características fundamentais** para a operação em ambiente corporativo como, por exemplo:

- Controle de acesso e segurança

As versões do FAT não implementavam a funcionalidade de controle de acesso a arquivos e diretórios, o que praticamente inviabilizava o desenvolvimento de aplicações corporativas que possuíssem requisitos de segurança no acesso aos dados.

• Confiabilidade

Uma das principais características esperadas em sistemas corporativos é que seja hábil a se recuperar de problemas evitando a ocorrência de perda de dados.

• Tamanho dos nomes dos arquivos e das partições

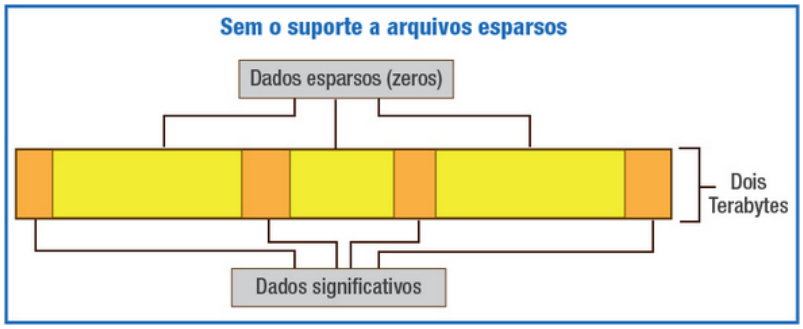
No início da década de 90, a versão em uso do FAT era a de 16 bits que, como já apresentado, tinha os nomes de arquivos limitados a oito caracteres e o tamanho das partições limitado a 4GB.

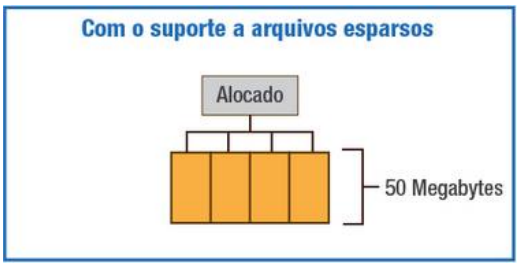
08

Visando endereçar estes problemas relacionados ao modelo FAT, a Microsoft realizou o lançamento do **NTFS 1.1**, sistema de arquivos desenvolvido para acompanhar o Windows NT 4, sistema operacional lançado para uso em servidores corporativos. No ano 2000 foi lançada uma inovadora versão do New Technology File System, o NTFS5, que corrigiu uma série de problemas encontrados na versão anterior e adicionou diversas novas funcionalidades. São algumas das principais inovações advindas com o NTFS5:

- **Criptografia**;
- **Quota de Disco**;
- "**Journals**".

O NTFS5 trouxe, ainda, o suporte a arquivos esparsos, funcionalidade que permite aos programas criar arquivos muito grandes e consumir o espaço em disco apenas quando necessário. Este recurso se dá em função de que muitos arquivos têm a característica de conter áreas que estão momentaneamente sem dados, e que são representadas por longas sequências de zeros, conforme exibido na figura a seguir.





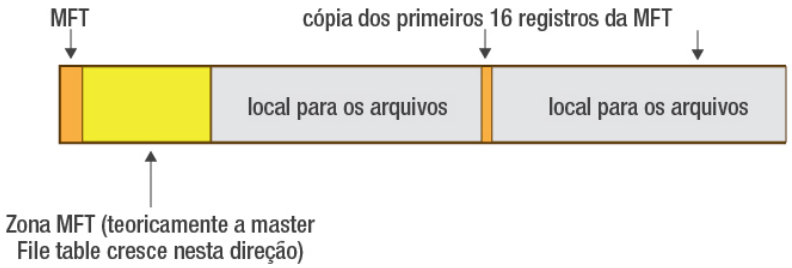
Criptografia
O sistema de arquivos incluiu a funcionalidade de armazenar automaticamente os arquivos em disco criptografados.

Quota de Disco
Esta funcionalidade permitiu que os administradores do sistema pudessem atribuir limites para armazenamento em disco para usuários ou grupos do sistema operacional.

"Journals"
As partições NTFS5 passaram a ser configuradas para manter o registro das alterações realizadas em arquivos e diretórios, criando um registro das ações que tinham sido executadas de modo a tornar o sistema mais resistente a falhas.

09

Assim como ocorre com o sistema de arquivos FAT, o NTFS divide a área útil de armazenamento em blocos. Estruturalmente, o NTFS é simbolicamente dividido em duas partes, uma área reservada para a Tabela Principal de Arquivos (*Master File Table – MFT*) e uma outra utilizada para o armazenamento dos arquivos e diretórios propriamente dito, conforme exibido na figura abaixo.



Um quadro comparativo das principais funcionalidades dos sistemas FAT e NTFS, em suas diversas versões, é exibido abaixo.

	FAT12	FAT16	FAT32	exFAT	NTFS	NTFS5

Compressão	Não	Não	Não	Não	Sim	Sim
Criptografia	Não	Não	Não	Não	Não	Sim
Quota de disco	Não	Não	Não	Não	Não	Sim
Arquivos esparsos	Não	Não	Não	Não	Não	Sim
Tolerância a falhas	Não	Não	Não	Sim (com o TFAT)	Sim	Sim
Performance em volumes pequenos	Alta	Alta	Alta	Alta	Baixa	Baixa

10

4 - EXTENDED FILE SYSTEM (EXT)

O ***extended file system*** se tornou conhecido por ser o sistema de arquivos padrão das mais conhecidas distribuições Linux. A primeira versão do EXT foi desenvolvida em 1992, inspirada no sistema de arquivos utilizado nos sistemas UNIX, o UFS (UNIX File System).

Apesar de apresentar uma série de melhorias em relação ao MINIX, sistema de arquivos que até então era utilizado no Linux, o EXT em sua primeira versão ainda apresentava uma série de **problemas** como, por exemplo:

- a frequente ocorrência de fragmentação do sistema de arquivos e
- limitações relacionadas ao tamanho dos discos.

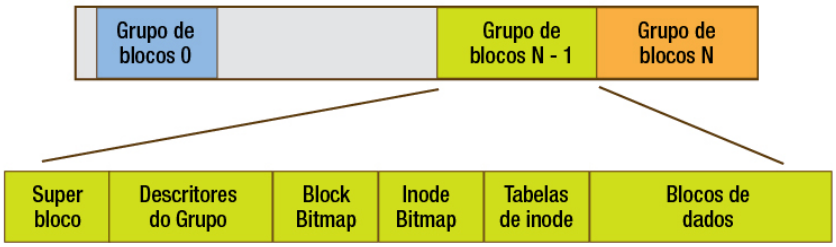
Por conta destes problemas estruturais, o EXT teve uma vida curta, sendo substituído pelo EXT2 já em janeiro de 1993.

Dentre os sistemas de arquivos utilizados nas distribuições Linux, certamente o ***Second Extended File System*** **(EXT2)** foi o que obteve maior sucesso. Assim como a grande maioria dos sistemas de arquivos, o EXT2 se baseia na premissa de que os arquivos e diretórios são armazenados em blocos de dados presentes no disco, todos de mesmo tamanho, apesar da definição do tamanho padrão ser configurável.

É importante ressaltar, entretanto, que nem todos os blocos presentes na partição de disco são utilizados para o armazenamento de dados, já que o correto funcionamento do EXT2 exige que sejam armazenados também os metadados dos arquivos e diretórios.

11

De uma forma geral, o EXT2 divide a área disponível na partição do disco em diversos grupos de blocos que são utilizados para armazenar o conteúdo dos arquivos e os seus metadados, conforme exibido abaixo.



Como pôde ser visualizado, o padrão EXT2 divide cada um dos grupos de blocos em:

• **Super Bloco**

Contém a descrição dos dados básicos do sistema de arquivo, como a configuração do tamanho do bloco ou a quantidade de blocos por grupo. Apesar de usualmente apenas o superbloco do grupo de blocos 0 ser lido quando o sistema é montado, os N grupos de bloco do disco mantêm uma cópia das informações.

• **Descritores do Grupo**

Armazenam os dados que descrevem as informações dos grupos de blocos como, por exemplo, os endereços do*"block bitmap"* e *"inode bitmap"*. Assim como acontece com o superbloco, as informações dos descritores de grupo são duplicadas em todos os grupos de blocos.

• ***Block Bitmap***

Gerencia o *status* da alocação dos blocos do grupo de blocos.

• ***Inode Bitmap***

Gerencia o *status* da alocação dos inodes do grupo de blocos.

• ***Tabelas de Inode***

Agrupa os *inodes* que descrevem os arquivos armazenados no grupo de blocos.

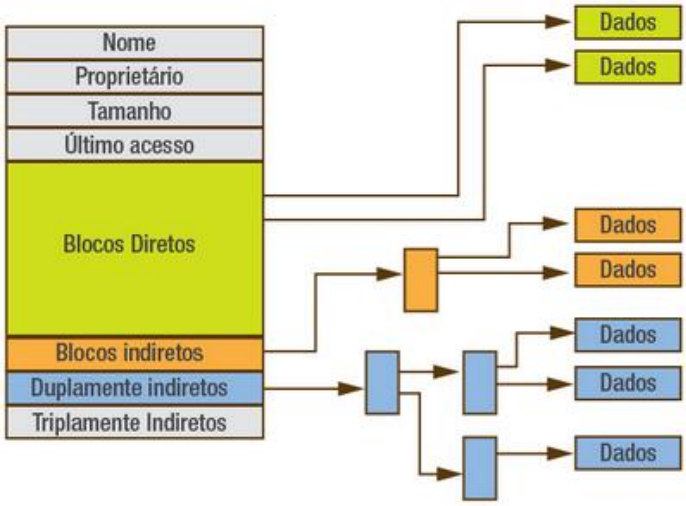
12

No modelo definido pelo Extended File System, os arquivos são descritos através dos ***inodes***, cujos registros são mantidos agrupados pelo sistema em tabelas.

Inodes são estruturas de dados que armazenam tanto os dados que compõem os atributos do arquivo (como as permissões de acesso, a identificação do proprietário, o tamanho do objeto, a data de modificação, ou a data do último acesso), quanto a informação sobre os endereços dos blocos de dados ocupados pelo arquivo descrito.

A figura abaixo apresenta detalhadamente a estrutura de um *inode*.

A figura exibe, ainda, a possibilidade de referência direta ou indireta aos blocos de armazenamento da partição.

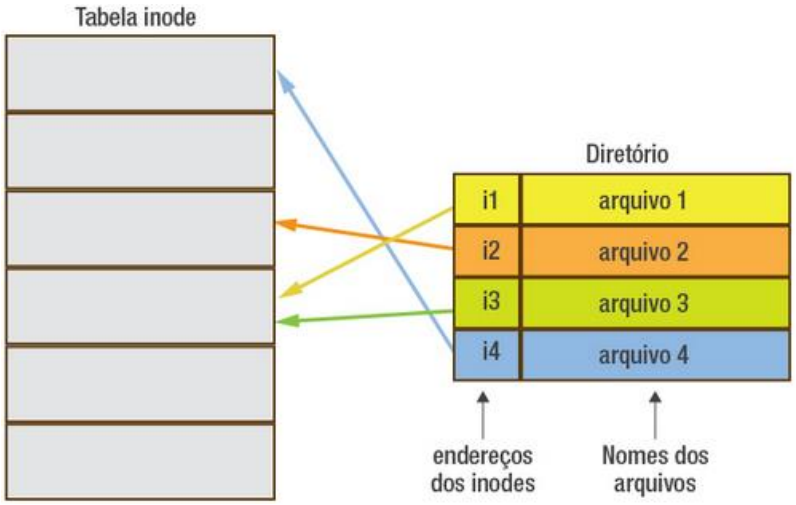


Os doze primeiros ponteiros presentes na estrutura dos *inodes* apontam diretamente para a localização de blocos físicos onde os dados do arquivo estarão armazenados, já os três ponteiros restantes direcionam para níveis de apontamentos indiretos. Assim, ao invés de apontar diretamente para um bloco de dados, os ponteiros indiretos apontam para blocos que armazenam ponteiros para novos blocos de endereços (blocos duplamente ou triplamente indiretos).

Os apontamentos indiretos são uma alternativa para arquivos cujo tamanho ultrapassa doze vezes o tamanho máximo alocado para os blocos no sistema de arquivos.

13

Assim como os arquivos, os diretórios também são representados no modelo EXT2 como *inodes* só que, ao invés de apontarem para blocos que armazenam dados, têm como conteúdo uma lista de entradas que descrevem os nomes e os endereços dos *inodes* dos arquivos pertencentes ao diretório. A figura abaixo traz o esquema de funcionamentos dos diretórios no modelo EXT2.



Em função do sucesso do EXT2, a terceira versão do *extended file system* teve um lapso temporal de quase oito anos, sendo lançada apenas em 2001. A principal novidade do EXT3 em relação à versão anterior foi a inclusão de uma área dedicada no disco onde era feito o registro de todas as mudanças

executadas no sistema de arquivos, o que reduzia a possibilidade que o sistema fosse corrompido em caso de uma queda inesperada. Este conceito foi definido como *"Journaling"*.

14

A versão mais recente do *extended file system*, o EXT4, foi lançada em 2008. Em comparação com o EXT3, foi percebida a inclusão de uma série de **funcionalidades** com o objetivo de melhorar a performance e a segurança do sistema de arquivos, tais como:

- a possibilidade de alocação tardia,
- um sistema mais rápido de arquivos de verificação,
- a alocação multibloco e
- a utilização da soma de verificação, ou checksum, na tarefa de journaling.

Além disso, o tamanho máximo individual de cada arquivo foi extendido para 16 TB, e o tamanho total do sistema de arquivos foi alçado a 1 EB (ou 1024 PB). A tabela abaixo apresenta a evolução de algumas características relacionadas aos sistemas de arquivos MINIX e as diversas versões do *extended file system*.

	MINIX	EXT	EXT2	EXT3	EXT4
Tamanho máximo do arquivo em disco	64 MB	2 GB	2 TB	2 TB	16 TB
Tamanho Máximo do disco	64 MB	2 GB	32 TB	32 TB	1 EB
Journaling	Não	Não	Não	Sim	Sim
Alocação Multibloco	Não	Não	Não	Não	Sim
Alocação tardia	Não	Não	Não	Não	Sim
Journal checksum	Não	Não	Não	Não	Sim

15

5 - Virtual File System (VFS)

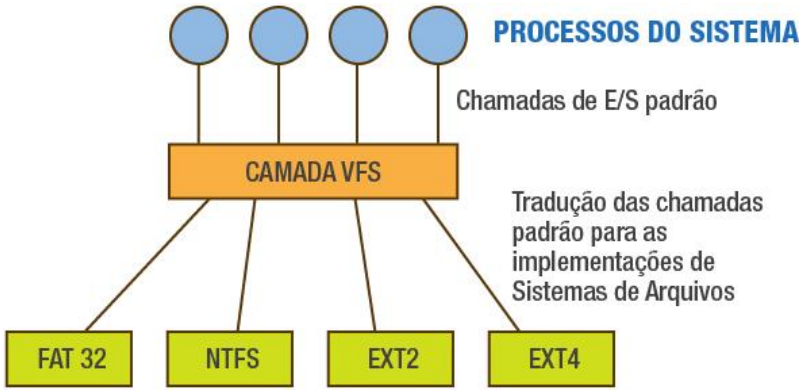
Cada sistema operacional é configurado para trabalhar com um determinado sistema de arquivos. Como vimos, as primeiras versões do MS Windows, por exemplo, utilizavam o sistema File Allocation Table, conquanto as versões mais recentes já utilizam o New Technology File System. As distribuições Linux, por sua vez, costumam utilizar versões do Extended File System.

Conforme já comentado, um dado disco pode possuir múltiplas partições, sendo que cada uma delas pode conter uma implementação diferente do sistema de arquivos. A grande questão que se põe é como prover acesso aos diferentes sistemas de arquivos em sistemas operacionais que implementam nativamente apenas um tipo específico de sistema de arquivos.

Foi neste contexto, de promover a interoperabilidade entre diferentes sistemas de arquivos, que o Virtual File System foi proposto.

A ideia do **Virtual File System** é a de se criar uma camada acima dos sistemas de arquivos de forma a abstrair as peculiaridades de cada implementação através do fornecimento de comandos padrão que serão utilizados pelos processos em execução no sistema operacional.

O modelo esquemático do VFS é exibido na figura a seguir.



16

RESUMO

O sistema de arquivos é o elemento que fornece ao programador uma série de interfaces, as quais permitem que os aplicativos que estão sendo codificados tenham acesso às funções de armazenamento e recuperação no disco rígido, independentemente do fabricante e do modelo do hardware que está sendo utilizado pelo computador. Os sistemas de arquivos são tradicionalmente implementados nos discos rígidos, que são os dispositivos de memória responsáveis por armazenar a maior parte dos arquivos e diretórios em um computador. A grande maioria dos discos pode ser segmentada em uma ou mais partições, de forma a permitir a configuração de múltiplos sistemas de arquivos em um mesmo disco.

Os sistemas de arquivos mais conhecidos são aqueles que integram os principais sistemas operacionais do mercado, a exemplo do *File Allocation Table (FAT)* e do *New Technology File System (NTFS)*, que são utilizados pelos sistemas Windows, e das diferentes versões do*Extended File System (EXT)*, que compõe a maior parte das distribuições Linux.

O sistema de arquivos *File Allocation Table* foi desenvolvido no final dos anos 1970, sendo que sua primeira versão foi o FAT12, sistema de arquivos voltado para discos flexíveis. Atualmente, existem mais três versões deste sistema, o FAT16 o FAT32 e o extFAT (ou FAT64). A arquitetura *File Allocation Table* foi desenvolvida originalmente para computadores aderentes a arquitetura IBM PC. Já o NTFS foi lançado no início da década de 1990, como o início da estratégia da Microsoft em investir no desenvolvimento de Sistemas Operacionais voltados ao ambiente dos data centers corporativos. No ano 2000 foi lançada uma inovadora versão do sistema, o NTFS5, que corrigiu uma série de problemas

encontrados na versão anterior e adicionou diversas novas funcionalidades, como criptografia, quotas de disco e a utilização de “journals”. O NTFS5 trouxe, ainda, o suporte a arquivos esparsos.

O *extended file system, por sua vez*, se tornou conhecido por ser o sistema de arquivos padrão das mais conhecidas distribuições Linux. Dentre as suas diversas versões, certamente o *Second Extended File System* (EXT2) foi o que obteve maior sucesso, o que fez com que a terceira versão do *extended file system* tivesse um lapso temporal de quase oito anos, sendo lançada apenas em 2001. A versão mais recente do extended file system, o EXT4, chegou ao mercado no ano de 2008.

Com a existência de diversos sistemas de arquivos e com a possibilidade de que cada partição do disco possa conter uma implementação diferente do sistema de arquivos, surgiu a necessidade de que o SO pudesse prover acesso aos diferentes sistemas de arquivos em sistemas operacionais que implementam nativamente apenas um tipo específico de sistema de arquivos. Neste contexto é que foi proposto o Virtual File System, modelo que propõe a criação de uma camada acima dos sistemas de arquivos, que tem a atribuição de abstrair as peculiaridades de cada implementação dos sistemas de arquivos através do fornecimento de comandos padrão que são utilizados pelos processos em execução no sistema operacional.

UNIDADE 3 – GERENCIAMENTO DE ARQUIVOS, ENTRADA E SAÍDA
MÓDULO 3 – PRINCÍPIOS DE ENTRADA E SAÍDA (I/O)
01

1 - GERENCIAMENTO DE ENTRADA E SAÍDA

O gerenciamento de entrada e saída é uma das principais atividades realizadas pelo sistema operacional, principalmente pela dificuldade que envolve a interação dos aplicativos com uma quantidade incomensurável de dispositivos e periféricos, dos mais diversos modelos e fabricados pelos mais variados fabricantes.

Nesta linha, o SO atua provendo uma interface que abstrai a complexidade de operação dos dispositivos em baixo nível, dando a impressão aos aplicativos de usuário que o próprio sistema operacional é quem executa as operações, **independentemente do dispositivo** que está sendo acessado.

Uma das ações que deve ser implementada pelo SO para prover a abstração do elemento de *hardware* é a adoção de um padrão de**nomenclatura uniforme** para as classes de dispositivos.

Como não há como uma forma de alterar ou reescrever o código do SO ou dos aplicativos do usuário sempre que um novo dispositivo for adicionado ao computador, os nomes que referenciam cada um destes dispositivos devem ser conhecidos no momento da codificação dos *softwares*.

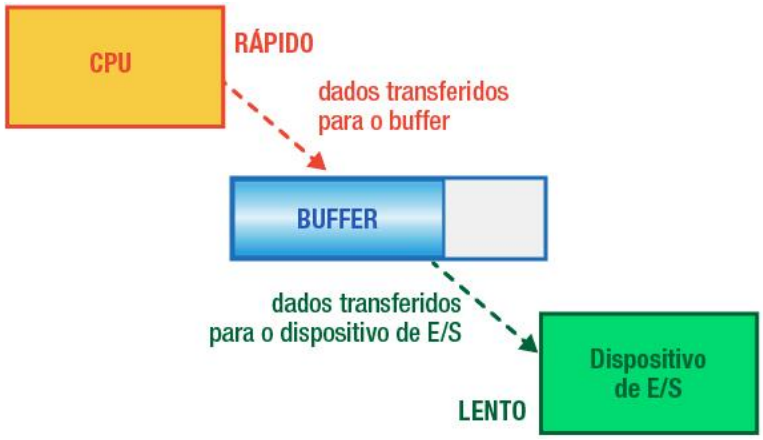
Outro conceito também relacionado a operacionalização da gerência de Entrada e Saída é o de **armazenamento intermediário**, ou *buffering*. O fato de que os dispositivos de E/S normalmente não têm grande capacidade de armazenamento de dados, associado ao de que muitas vezes durante o processo de E/S ainda não se sabe a destinação final dos dados, fez que surgisse a necessidade de criação de um espaço na hierarquia de memória do computador para a manutenção dos dados extras

oriundos dos dispositivos de E/S, provendo a abstração de que estes dispositivos têm mais espaço de armazenamento do que realmente possuem.

02

A utilização do armazenamento intermediário auxiliou, ainda, na solução de outro grande problema, a diferença de velocidade entre o tempo do processamento realizado pela CPU e o tempo consumido com as operações de E/S. Como alguns dispositivos têm sérias restrições de tempo real, a utilização do *buffering* promoveu uma melhora no desempenho global do sistema à medida que reduziu a ocorrência de diversos problemas como, por exemplo, os erros originados quando o ritmo de leitura é maior do que o de gravação.

A figura abaixo demonstra uma operação de transferência de dados para um determinado dispositivo de E/S como, por exemplo, uma impressora. Em condições normais, uma situação onde a velocidade da CPU em enviar os dados é maior do que a capacidade da impressora em consumir os dados faria com que a CPU ficasse ocupada e subutilizada até que toda a impressão fosse finalizada. A utilização do *buffering*permite a liberação da CPU para execução de outras tarefas assim que esta acaba de carregar os dados na área de armazenamento temporário, sem a necessidade de que a impressão esteja finalizada.



03

Uma última tarefa que deve ser provida pelo subsistema de Entrada e Saída e que compõe a gerência de E/S do Sistema Operacional é o controle e **gerenciamento de erros** na execução das operações de E/S.

Neste sentido, o SO deve tentar sempre tratar as ocorrências nas camadas mais próximas ao *hardware*. Apenas nos casos onde não for possível a utilização desta abordagem, o problema deve chegar até o nível da aplicação do usuário.

Como pôde ser visto, o escopo da atividade de Gerenciamento de E/S é vasto e intrincado. Além disso, os desafios não são triviais, o que faz com que o papel do Sistema Operacional seja fundamental para promover a abstração da operação dos dispositivos e a consequente redução da complexidade inerente ao processo de construção dos aplicativos de usuário.

04

2 - Dispositivos de Entrada e Saída (E/S)

Os dispositivos de entrada e saída costumam ter duas diferentes classificações, uma relacionada ao processo de entrada e saída de dados e outra relacionada aos atores envolvidos no processo.

Em relação ao **processo de E/S**, os dispositivos podem ser classificados como:

- **de entrada**,
- **de saída**, ou
- **híbrido**.

Já em relação aos **atores envolvidos**, os dispositivos são categorizados como:

- os que **interagem com o usuário**,
- os que **interagem com as máquinas** ou
- os **de comunicação**.

Os dispositivos que interagem com o usuário, como o próprio nome já sugere, estão relacionados com a promoção da interação do operador do computador com o *hardware* computacional. Esta categoria tem como principal função **prover um meio de comunicação entre o homem e a máquin**a, seja no sentido homem-máquina, como, por exemplo, o teclado e o mouse, ou no sentido máquina-homem, como as impressoras e monitores de vídeo.

Os dispositivos voltados à máquina, por sua vez, têm como principal objetivo prover a comunicação entre os diferentes itens de *hardware*. Um dos principais exemplos desta categoria são os discos rígidos, estrutura de *hardware* que provê informações que serão transferidas para a memória principal e processadas na CPU, elementos também de *hardware*.

A última categoria da classificação do em relação aos atores envolvidos é composta pelos dispositivos de comunicação, que têm como meta prover a comunicação dos componentes de *hardware* com dispositivos remotos. Nesta linha, têm-se como principais exemplos as placas de fax/modem e as placas de rede.

05

Já o sistema de classificação em relação ao processo de E/S, conforme já comentado, enquadra os dispositivos de E/S como de entrada, de saída, ou híbrido.

Os **dispositivos de entrada** são aqueles que se caracterizam por prover a comunicação no sentido usuário – computador, sendo os principais exemplos o teclado e o mouse.

Os **dispositivos de saída**, por sua vez, se caracterizam por prover a comunicação no sentido contrário, ou seja, computador – usuário. Os principais exemplos desta categoria são o monitor e a impressora.

Já os dispositivos **híbridos** têm como característica atuar tanto como um receptor quanto um emissor de dados, sendo o disco rígido um dos principais exemplos. Uma lista de dispositivos de E/S com as suas respectivas classificações pode ser visualizada na tabela abaixo.

Dispositivo	Classificação quanto ao processo de E/S	Classificação quanto ao ator envolvido
Impressora	saída	usuário
Mouse	entrada	usuário
Teclado	entrada	usuário
Disco Rígido	híbrido	máquina
Placa de Rede Ethernet	híbrido	comunicação
Placa de Fax/Modem	híbrido	comunicação
Scanner	entrada	usuário
Pen Drive	híbrido	máquina
Placa de Rede Sem Fio	híbrido	comunicação
Monitor	saída	usuário

06

3 - Funções para execução da entrada e saída

A função de E/S sofreu uma série de evoluções ao longo dos anos. Inicialmente o processador controlava diretamente o periférico, depois surgiu a utilização de interrupções para gerência do processo, para, por fim, ser proposta a técnica que ficou conhecida como acesso direto à memória.

O **controle direto da CPU** é estabelecido através da técnica de entrada e saída programada, também conhecida como pooling. Neste modelo, a Unidade Central de Processamento garante que o programa não avança até que a operação de E/S esteja totalmente completada.

O que acontece na prática é que o programa efetua uma chamada de sistema solicitando o dispositivo e, assim que obtém a posse do mesmo, faz uma nova chamada para que a operação de E/S seja executada. Apenas neste momento o processo de Entrada e Saída passa a ser gerenciado pelo SO.

A principal característica desta técnica é o fato de que o Sistema Operacional demanda que a CPU verifique, durante a execução da operação de E/S e de forma cíclica e contínua, se o dispositivo ainda está disponível. Este comportamento, chamado de **espera ocupada** (busy waiting), é uma das principais desvantagens do modelo programado, já que gera desperdício do tempo de processamento da CPU.

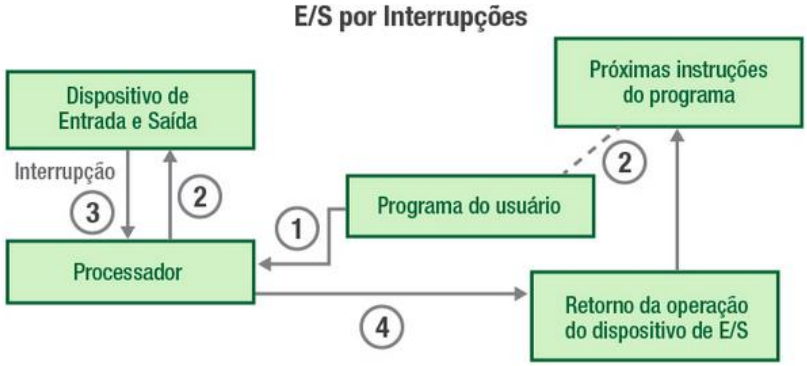
De forma geral, a E/S programada se caracteriza pela facilidade de implementação e baixo custo. Como desvantagem, pode-se citar a ineficiência desta abordagem, sobretudo pelo desperdício de processamento ocasionado pela verificação cíclica da condição do dispositivo.

Um esquema de funcionamento da técnica de e/s programada é exibido na figura.

07

Diferentemente do modelo programado, a utilização de **interrupções** não impede necessariamente que o programa continue a sua execução, exceto nos casos onde a entrada ou saída é pré-requisito para a execução da próxima instrução. Além disso, com o uso do modelo de interrupções a CPU não necessita verificar constantemente e de forma cíclica o estado da operação de E/S.

Neste modelo o fluxo é invertido, já que é o dispositivo de E/S que passa a ter a obrigação de informar a CPU, através da execução de uma interrupção, quando do término da execução da sua tarefa, o que faz com que a CPU possa utilizar o tempo de processamento que seria perdido na execução de outras operações. O desenho esquemático do modelo de E/S por interrupções é exibido na figura abaixo.

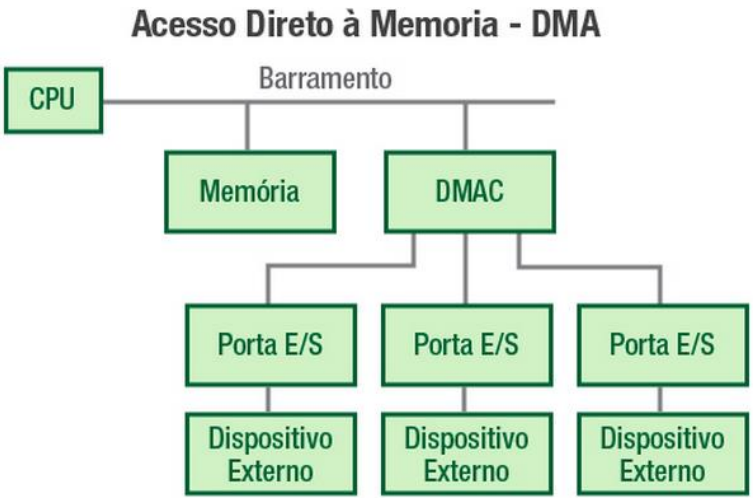


Apesar de evitar o processo de espera ativa, a CPU continua responsável por efetuar as transferências de dados entre os diversos atores do processo no modelo de entrada e saída com o uso de interrupções. O **Acesso Direto à Memória** (DMA) surgiu justamente como uma alternativa para solucionar este problema, já que neste modelo, depois de iniciada a transação, o controle da transferência dos dados passa automaticamente da CPU para o Dispositivo Controlador (DMAC).

08

No modelo DMA, durante a operação de E/S o DMAC assume o controle do barramento e passa a ter acesso direto à memória, sem qualquer suporte ou intervenção por parte da CPU. Outro ponto positivo é o fato de que apenas uma interrupção é gerada, ao final do processo ou na ocorrência de algum tipo de erro, independentemente do tipo de operação de E/S. No modelo por interrupções, uma nova interrupção é gerada a cada evento de E/S.

Imagine, por exemplo, que um determinado processo solicita a impressão de um documento com 50 linhas, sendo que cada linha tem 40 caracteres. Com o uso do modelo de interrupções, uma nova interrupção seria gerada para cada novo caractere, o que demandaria um total de 2000 interrupções para que o processo de impressão fosse completado. Já no modelo DMA, apenas uma interrupção seria gerada, após a impressão do último caractere. O desenho esquemático do funcionamento do modelo DMA é abaixo.



09

4 - Camadas de *Software* de Entrada e Saída

Softwares de Entrada e Saída são comumente organizados em função de quatro camadas, com atribuições bem definidas, mas cujas interfaces podem variar a depender de cada Sistema Operacional.

A Figura a seguir traz o desenho esquemático das camadas de E/S.

As interações dos ***softwares* de E/S em nível de usuário** com o subsistema de entrada e saída normalmente ocorrem através da realização de chamadas ao Sistema Operacional, mais especificamente chamadas de **biblioteca**, que fazem referência à chamadas de sistema.

Sistema Operacional
Essa organização é conforme o Livro Modern Operating Systems, de Tanenbaum.

10

Para se ter uma ideia desta abstração, imagine que durante a construção de um aplicativo em nível de usuário, codificado em linguagem C, o usuário necessita gravar um arquivo em disco. Para tanto, durante a escrita do aplicativo, o programador adiciona uma chamada de alto nível as funções *fopen*, *fprint* e *close*, oriundas das bibliotecas da linguagem de programação, conforme apresentado no código abaixo.

```
FILE *arquivo=fopen("relatorio.txt", "w"); //abre o arquivo relatorio.txt no disco
fprintf(relatorio.txt, "Compras 2015"); // adiciona o texto Compras 2015
fclose(relatorio.txt); // fecha o arquivo relatorio.txt
```

Observe que, durante a escrita do código, o programador não precisa ter conhecimento de qual o fabricante ou sequer o modelo do disco rígido que estará instalado no equipamento que executará o código, precisa apenas do conhecimento acerca de funções das bibliotecas da linguagem de programação que está utilizando.

Apesar de aparentemente parecer que esta abstração é provida pela própria linguagem, na realidade as funções da biblioteca chamam procedimentos de baixo nível do Sistema Operacional, que são pertencentes a camada de *software* de SO independente de dispositivo.

A camada de **Software de SO independente de dispositivo** é o local onde são centralizadas as principais ações de gerência do subsistema de entrada e saída.

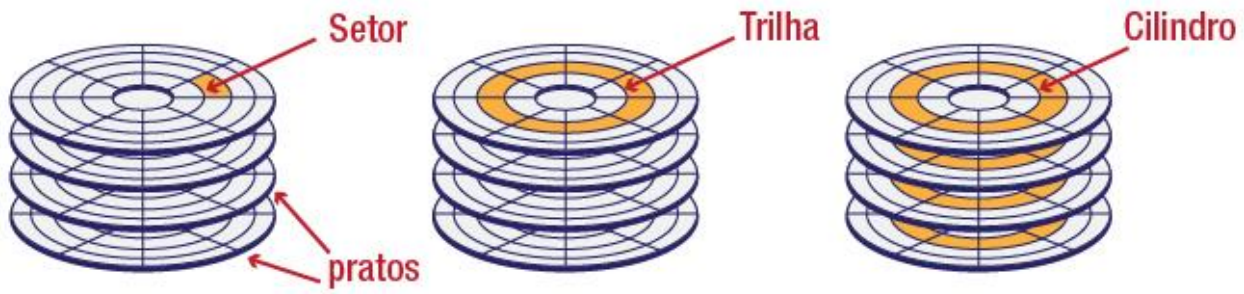
11

Dentre as funções desta camada intermediária (camada de **Software de SO independente de dispositivo**), pode-se citar:

- Mapear os nomes simbólicos dos dispositivos utilizados nas camadas de usuário para os drivers específicos localizados na camada de drivers (tomando como exemplo o SO Linux, o nome simbólico */dev/hda*, que representa uma chamada a um dispositivo de disco, deve ser mapeado para o driver específico fornecido pelo fabricante do dispositivo).
- Gerenciar e prover o armazenamento intermediário de dados, intermediando o fluxo de informações direcionado aos dispositivos de E/S.
- Coletar os erros e reportar ao usuário de maneira consistente, quando necessário.
- Gerenciar a alocação e liberação de dispositivos de Entrada e Saída.

- Prover um tamanho de bloco independente de dispositivo, de modo a permitir o fluxo de dados entre as camadas.

Esta camada provê, ainda, uma interface uniforme para a chamada dos drivers de dispositivo. Isto é fundamental para o funcionamento do Sistema operacional, já que não há como se alterar o código do SO para que se adapte a cada novo driver de dispositivo instalado. Seguindo esta linha, todo fabricante de dispositivo deve ter conhecimento prévio das chamadas do sistema operacional para que consiga construir um driver que seja aderente às interfaces padrão do SO (Figura abaixo). Este requisito corrobora a necessidade de que o fabricante tenha que codificar diferentes drivers para um mesmo dispositivo, a depender do Sistema Operacional.



12

A camada de *software* de SO independente de dispositivo, por sua vez, tem ligação direta com a **camada de drivers de dispositivo**.

Objetivamente, a camada de drivers corresponde ao componente que implementa a abstração das rotinas de baixo nível dos dispositivos de E/S.

Na prática, é como se cada dispositivo anexado ao computador precisasse de uma parcela específica de código para que pudesse ser utilizado, como não há como reescrever o código do SO a cada novo elemento de E/S instalado, o código é fornecido através de componentes de*software* que são agregados ao SO – os **drivers de dispositivo**.

Além de abstrair a complexidade do *hardware*, os drivers de dispositivo têm uma série de outras funções. As principais **tarefas** associadas a estes componentes de *software* são:

- Executar o gerenciamento de entrada e saída.
- Incrementar a velocidade de E/S através da otimização da operação.
- Realizar o controle e gerenciamento dos erros de *hardware*.
- Permitir o acesso concorrente ao *hardware* por diferentes processos.
- Prover o gerenciamento transparente do dispositivo, evitando a necessidade de conhecimento das operações de baixo nível.

13

Em alguns sistemas operacionais, os drivers são comumente classificados em duas diferentes categorias a depender do tipo de dispositivo.

• drivers associados aos dispositivos de caractere

São aqueles cuja comunicação é realizada por meio do envio e recebimento de caracteres únicos, como acontece, por exemplo, com o teclado e o mouse.

• drivers associados aos dispositivos de bloco

Têm como exemplo os discos rígidos e utilizam em sua comunicação blocos inteiros de dados que agregam um conjunto de caracteres por vez.

A inclusão de novos drivers de dispositivo, a depender do Sistema Operacional, pode requerer a recompilação do código do SO, já que muitas vezes este componente de *software* é executado em modo kernel do processador. Além disso, em alguns sistemas operacionais, como o Windows e o Linux, existe uma configuração conhecida como *"plug-and-play"*, onde os drivers de dispositivo ficam em estado de hibernação e são executados apenas sob demanda, quando o dispositivo de E/S específico é detectado pelo SO.

A última camada de *software* do subsistema de E/S é a de **gerenciamento de interrupções**, que, como o próprio nome já diz, tem como principal tarefa o controle das interrupções do SO. Este é o mais baixo nível dentre as Camadas de *Software* de Entrada e Saída existentes, ou seja, é a camada mais próxima dos dispositivos de E/S.

O gerenciamento de interrupções é fundamental para incrementar o desempenho do SO, pois a maior parte das operações de E/S se utiliza de interrupções em sua gerência. Isto se dá porque a diferença de velocidade entre os elementos de E/S e o processador torna premente a utilização de interrupções como forma de evitar a indesejável espera da CPU pela finalização das operações de Entrada e Saída.

14

RESUMO

Uma das principais atribuições dos sistemas operacionais modernos é promover o gerenciamento do processo de entrada e saída dos computadores. Esta tarefa tem como objetivo permitir que os aplicativos de usuários possam utilizar os dispositivos de *hardware* e periféricos sem a necessidade de conhecer detalhes da operação destes elementos em baixo nível.

O sistema de gerenciamento de Entrada e Saída tem quatro tarefas basilares, promover as operações de E/S **independentemente do dispositivo** que está sendo acessado, prover adoção de um padrão de **nomenclatura uniforme** para as classes de dispositivos, fornecer um meio de **armazenamento intermediário** para o processo de E/S e efetuar o controle e **gerenciamento de erros** na execução das operações de E/S.

Um dos principais elementos do subsistema de E/S são os dispositivos de entrada e saída, elementos que são comumente categorizados por duas diferentes metodologias, uma relacionada ao **processo de E/S**, onde os dispositivos podem ser classificados como **de entrada, de saída**, ou **híbrido**, e outra em relação aos **atores envolvidos**, onde os dispositivos são categorizados como os que **interagem com o usuário**, os que **interagem com as máquinas** ou os dispositivos **de comunicação.**

Associados aos dispositivos de E/S, o Sistema Operacional tem que prover uma série de funções que promovem a sua operação. A função de E/S sofreu uma série de evoluções ao longo dos anos, sendo que atualmente têm-se três diferentes técnicas: a **Entrada e Saída programada**, em que há o controle direto da CPU sobre o processo de E/S, o modelo de **interrupções**, no qual o fluxo é invertido e o dispositivo de E/S é que passa a ter a obrigação de informar a CPU o término da execução da sua tarefa, e o **acesso direto a memória**, no qual o DMAC assume o controle do barramento e passa a ter acesso direto à memória sem qualquer suporte ou intervenção por parte da CPU.

Segundo *Tanenbaum*, o processo de Entrada e Saída pode ser segmentado em quatro camadas, com atribuições bem definidas, mas cujas interfaces podem variar a depender de cada Sistema Operacional, são elas: **softwares de E/S em nível de usuário; Software de SO independente de dispositivo; Drivers de dispositivo;** e a de **gerenciamento de interrupções**.

UNIDADE 3 – GERENCIAMENTO DE ARQUIVOS, ENTRADA E SAÍDA

MÓDULO 4 – DISCOS

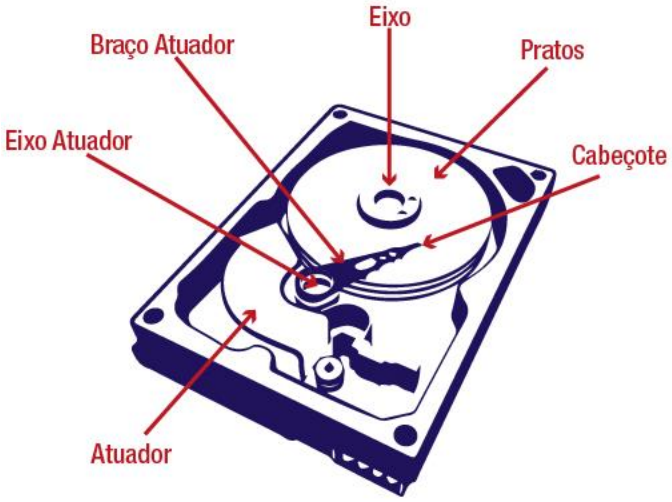
01

1 - ESTRUTURA DE UM DISCO RÍGIDO

Os discos rígidos estão entre os principais elementos de *hardware* do computador, assumindo um papel de extrema relevância tanto no processo de gerenciamento de memória quanto no de gerenciamento de entrada e saída.

Internamente, como o próprio nome já sugere, os discos rígidos se organizam com base em uma série de outros discos, chamados de “pratos”, que se posicionam um acima do outro através de um eixo central, ou motor. Cada um dos pratos possui duas superfícies, a superior e a inferior, que são utilizadas para prover o armazenamento físico dos arquivos e diretórios. A quantidade de pratos existentes em cada disco varia a depender da tecnologia utilizada e da capacidade de armazenamento disponibilizado por cada *hardware* especificamente.

Para executar o processo de gravação e leitura dos dados, o disco rígido possui um cabeçote de leitura para cada prato. Estes elementos são fixados no atuador através de braços, o que permite a mobilidade necessária para que seja possível ler todo o conteúdo da superfície do prato enquanto este é rotacionado pelo eixo.

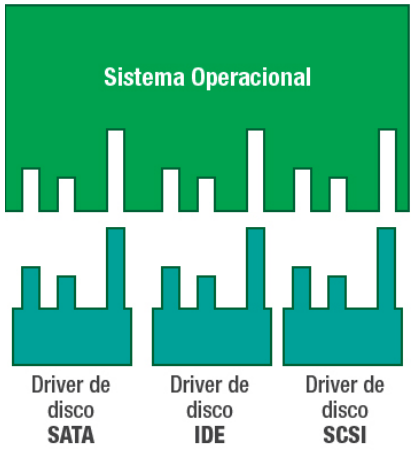


02

As superfícies dos pratos dos discos são organizadas em trilhas que, por sua vez, são divididas em setores.

Um conjunto de trilhas de mesma posição nos diferentes pratos é chamada de **cilindro.**

Estes elementos são os responsáveis por permitir a identificação dos locais para onde deve se movimentar os cabeçotes de leitura do disco rígido.



03

2 - POLÍTICAS DE AGENDAMENTO DE DISCO

A performance dos discos rígidos é normalmente mensurada em função de três fatores:

• Tempo de busca

O tempo gasto na movimentação do braço do disco até que o cabeçote de leitura atinja o setor onde está a informação.

• Latência de rotação

Tempo de espera pela rotação do eixo até que o setor específico seja alcançado.

• Tempo de transferência

Tempo de transferência dos dados armazenados do cilindro até o componente eletrônico do disco rígido.

Os principais atrasos nas operações de disco se referem aos tempos gastos na busca e na latência de rotação. De forma a tentar reduzir o custo destas operações, muitos discos rígidos implementam políticas de agendamento. Estes algoritmos de agendamento do disco são componentes do sistema operacional que podem ser alocados em um módulo separado, de modo a viabilizar a sua substituição a qualquer tempo, ou como parte do núcleo do SO.

Os métodos de agendamento mais conhecidos são:

- *First Come First Served (FCFS),*
- *Shortest Seek Time First (SSTF),*
- *Elevador (SCAN),*
- *Look*,
- *Circular SCAN (C-SCAN).*

A performance de cada um dos algoritmos vai depender da quantidade e do tipo das requisições de acesso ao disco, em sistemas que tem um carregamento pesado no disco, por exemplo, os algoritmos SCAN e C-SCAN apresentam um melhor desempenho, mas em outras situações outros algoritmos podem ser os mais indicados.

04

a) *First Come First Served* (FCFS)

O FCFS é o modelo mais simples de agendamento, tendo como base a premissa de que as operações são executadas na ordem em que as solicitações chegam ao disco, assim como ocorre com o algoritmo FIFO utilizado no agendamento de processos pelo Sistema Operacional e já apresentado em módulo anterior.

Apesar de não apresentar o melhor desempenho entre os algoritmos, existentes, o FCFS evita problemas clássicos de busca em disco, como, por exemplo, o da inanição (starvation), já que todas as requisições efetuadas acabam sendo respondidas, mesmo que não da forma mais eficaz.

Para exemplificar o funcionamento do algoritmo FCFS, imagine uma fila de requisições para leitura ou escrita em blocos pertencentes aos cilindros 76, 15, 84, 22, 71 e 149. Tomando como base a premissa de

que o cabeçote de leitura estava no cilindro de número 30 antes do início da operação, tem-se que foram realizados os seguintes movimentos:

Movimento	Deslocamento
Do 30 para o 76	46
Do 76 para o 15	61
Do 15 para o 84	69
Do 84 para o 22	62
Do 22 para o 71	49
Do 71 para o 149	78
Total de deslocamentos	**365**

De acordo com o cálculo registrado da tabela, percebe-se que 365 cilindros foram percorridos até que o cabeçote do disco conseguisse cumprir as seis requisições de entrada.

05

b) *Shortest Seek Time First* (SSTF)

Tem como base o mesmo fundamento do algoritmo de agendamento de processos *Shortest Job First*, ou seja, executa primeiro a requisição que requer o menor movimento do braço do disco independentemente da direção, contando-se a partir da posição do cabeçote de leitura antes do início da operação.

Para exemplificar o funcionamento do SSTF, imagine a mesma fila de requisições utilizada no algoritmo anterior, e que doravante será a base para comparação dos demais algoritmos que serão apresentados - 76, 15, 84, 22, 71 e 149. Tomando como base que o cabeçote de leitura estava no cilindro de número 30, tem-se que foram realizados os seguintes movimentos para concluir a operação de E/S:

Movimento	Deslocamento
Do 30 para o 22	8
Do 22 para o 15	7
Do 15 para o 71	56
Do 71 para o 76	5
Do 76 para o 84	8
Do 84 para o 149	65
Total de deslocamentos	**149**

O que se observa neste algoritmo é que a quantidade de deslocamentos necessários para executar a requisição, para o exemplo dado, é inferior a metade do valor encontrado no algoritmo FCFS. Apesar

disto, o SSTF apresenta duas desvantagens que não são encontradas no modelo anterior, a primeira é o fato de que a mudança constante de direção do movimento de leitura pode ocasionar uma perda adicional de tempo, e a segunda é que se uma determinada área de disco continuar a receber requisições com muita frequência, o cabeçote pode permanecer indefinidamente nesta área, causando o problema conhecido como *starvation*.

06

c) Elevador (SCAN)

O funcionamento deste algoritmo é similar ao de um elevador convencional, elemento que motivou a sua nomenclatura. Neste método, o cabeçote de leitura percorre o disco na direção da requisição mais próxima indo até o último cilindro nesta direção para, só então, inverter o sentido de leitura, indo novamente até o último cilindro só que na direção inversa.

Utilizando a mesma lista de requisições adotada nos algoritmos anteriores - 76, 15, 84, 22, 71 e 149, e tomando como premissa que o último cilindro do disco é o de número 200, o algoritmo executaria os seguintes movimentos para finalizar com sucesso as requisições:

Movimento	Deslocamento
Do 30 para o 22	8
Do 22 para o 15	7
Do 15 para o 0	15
Do 0 para o 71	71
Do 71 para o 76	5
Do 76 para o 84	8
Do 84 para o 149	65
Do 149 para o 200	51
Total de deslocamentos	**230**

No exemplo dado, o algoritmo do elevador apresentou desempenho inferior ao algoritmo *Shortest Seek Time First* para mesma entrada. Entretanto, diferentemente do SSTF, o algoritmo do elevador tem como vantagem o fato de que a sua lógica evita a ocorrência do problema da inanição.

07

d) Look

O algoritmo Look é muito similar ao algoritmo do elevador, a principal diferença se dá pelo fato de que finaliza o seu movimento em uma determinada direção a partir do momento que não existem mais requisições nesta direção, diferentemente do outro algoritmo que só finaliza o movimento ao atingir o último ou o primeiro cilindro do disco.

Utilizando a mesma lista de requisições adotada nos algoritmos anteriores - 76, 15, 84, 22, 71 e 149, e tomando como premissa que o último cilindro do disco é o de número 200, o algoritmo executaria os seguintes movimentos para finalizar com sucesso as requisições:

Movimento	Deslocamento
Do 30 para o 22	8
Do 22 para o 15	7
Do 15 para o 71	56
Do 71 para o 76	5
Do 76 para o 84	8
Do 84 para o 149	65
Total de deslocamentos	**149**

No cálculo de movimentos apresentado para a entrada padrão, o algoritmo Look apresentou desempenho superior ao do elevador e similar ao SSTF, só que com o diferencial de que a sua lógica evita a ocorrência do problema da inanição.

08

e) Circular SCAN (C-SCAN)

O C-SCAN é uma adaptação do algoritmo do elevador, desta forma, assim como o modelo correlato, o cabeçote inicia o seu deslocamento indo no sentido da requisição cujo cilindro está mais próximo.

A principal diferença é que, ao chegar ao primeiro ou ao último bloco (a depender da direção tomada), o cabeçote não inverte o sentido do movimento, e sim continua na mesma direção pulando para o cilindro que representa a outra ponta da trilha, perfazendo uma espécie de percurso circular.

Movimento	Deslocamento
Do 30 para o 22	8
Do 22 para o 15	7
Do 15 para o 0	15
Do 200 para o 149	51
Do 149 para o 84	65
Do 84 para o 76	8
Do 76 para o 71	5
Total de deslocamentos	**159**

Apesar de não ser o algoritmo ótimo, o C-SCAN apresenta um dos melhores desempenhos dentre os algoritmos de agendamento de disco, contando, ainda, com o diferencial de também evitar o problema da inanição.

09

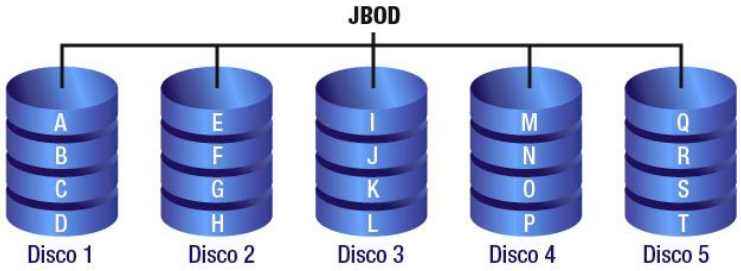
3 - ESTRUTURAS PARA ORGANIZAÇÃO DOS DISCOS

Quando se fala em estruturas de organização de discos rígidos, dois são os principais modelos que vêm à mente, o **RAID** e o **JBOD**.

a) JBOD - ***J****ust a* ***B****unch* ***O****f* ***D****isks*

Em tradução literal significa “apenas um monte de discos” e atua como um agregador de discos, fazendo que o sistema operacional consiga lidar com o armazenamento em múltiplos volumes.

A arquitetura JBOD, entretanto, não permite qualquer ganho de desempenho ou sequer implementa qualquer método que viabilize características esperadas em um arranjo de discos como, por exemplo, a tolerância a falhas. Conforme pode ser visualizado na figura abaixo, neste modelo os discos são tratados de forma independente e a informação é armazenada sequencialmente nos discos.



Desta forma, no padrão JBOD, caso um dos discos seja danificado, toda a informação armazenada é perdida. Além disso, não há qualquer preocupação em segmentar a informação e armazena-la em múltiplos discos como forma de melhorar a performance do sistema através do incremento das velocidades de leitura e escrita.

10

b) RAID - ***R****edundant* ***A****rray of* ***I****ndependent* ***D****isks*

O modelo RAID surgiu como uma forma de prover alguma organização e controle ao modelo JBOD. Diferentemente de ser apenas um arranjo de um monte de discos, o aparecimento da arquitetura RAID veio acompanhada de uma série de funcionalidades que buscavam reduzir a diferença de performance entre a velocidade de processamento da CPU e a velocidade de acesso aos discos.

Uma das ideias básicas proposta pelo RAID foi a de priorizar a utilização de conjuntos de discos menores, mas que trabalhavam de forma associada, ao invés da instalação de discos de grande volume

e alto custo.

A segmentação da informação em múltiplos volumes não apenas poderia aumentar a velocidade de leitura e escrita em função da maior quantidade de cabeçotes, como abriria a possibilidade de se realizar a leitura simultânea em múltiplos discos.

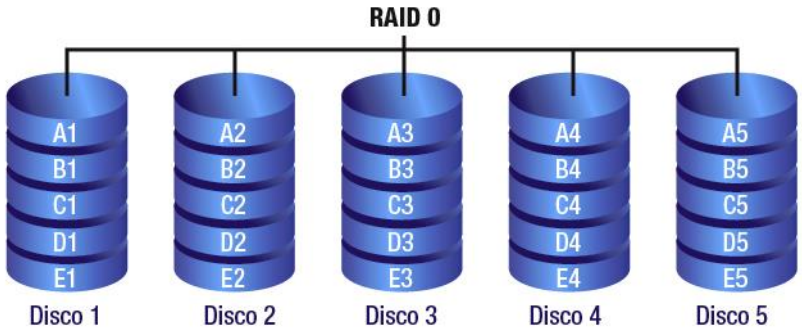
Foram propostos diversos modelos de organização RAID, alguns com foco em desempenho e outros mais voltados ao controle de erros. As arquiteturas RAID, que são normalmente identificadas através da inserção de um número após a sigla (por exemplo, RAID 0 ou RAID 1), serão estudadas detalhadamente mais adiante.

11

4- ARQUITETURAS RAID

O modelo RAID 0 se caracteriza por não implementar qualquer redundância de dados, dando ênfase apenas à melhoria do desempenho das operações executadas em disco.

Como pode ser observado na figura abaixo, este modelo segmenta a informação e efetua a gravação sequencial dos dados nos múltiplos discos.



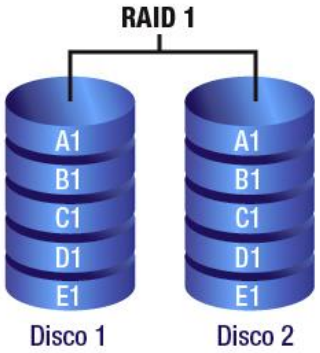
Este arranjo permite que múltiplos cabeçotes de leitura e escrita pertencentes a diferentes discos sejam utilizados para operacionalizar as operações, reduzindo o tempo que seria necessário para realizar a leitura ou a escrita dos dados em relação a operação realizada em um único disco. A grande desvantagem desta abordagem é o fato de que a perda de um dos discos pode implicar na perda de toda a informação, já que não há qualquer redundância da informação armazenada.

12

Diferentemente do RAID 0, o RAID 1 implementa mecanismos de redundância que evitam a perda da informação em caso de perda de um dos discos.

A estrutura do RAID 1 é organizada através da criação de cópias dos dados em múltiplos discos, ou seja, os discos são configurados sempre em pares, de modo que um dos membros sempre é uma cópia do outro.

Uma das principais desvantagens da utilização do RAID 1 é o fato de que o modelo faz com que metade do espaço útil de armazenamento seja perdido. Já como ponto positivo, tem-se que, a depender da natureza da operação, a performance de leitura dos blocos de dados pode ser até no máximo a soma do desempenho da execução da operação isoladamente em cada um discos.



As arquiteturas de RAID 2 e 3 são similares ao modelo do RAID 5, só que ao invés de utilizarem a segmentação de disco utilizando blocos, implementam a segmentação no nível de bits, no RAID 2, e no nível de bytes, no RAID 3. Além disso, o RAID 2 utiliza ECC (*Error Correcting Code*) ao invés de paridade, para controle de erros, e a configuração do RAID 3 exige um disco de paridade dedicado, ao invés de utilizar discos compartilhados como no RAID 5.

13

Assim como as arquiteturas de RAID 2 e 3, o RAID 4 também se assemelha ao 5, já que utiliza a segmentação da informação em blocos. Entretanto, assim como RAID 3, esta arquitetura exige um disco de paridade dedicado para funcionar corretamente.

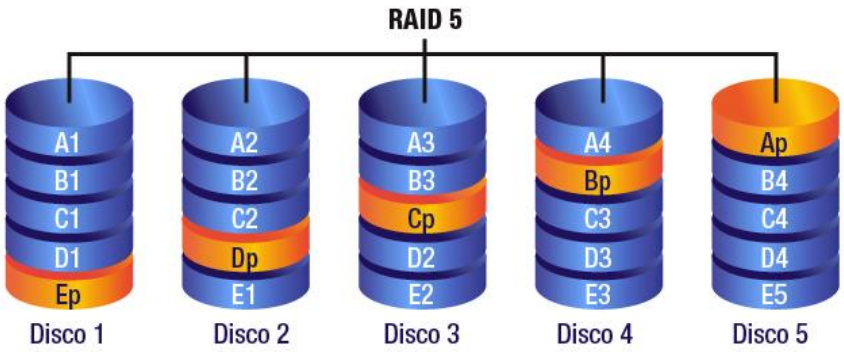
De concreto, tem-se que, atualmente, as arquiteturas RAID 2, 3 e 4 são raramente encontradas em aplicações comerciais. Veja a seguir alguns motivos.

O RAID 2 caiu rapidamente em desuso em função da sua complexidade de implementação e pelo fato de que o algoritmo de redução de erros que utiliza já se encontra implementado na maioria dos discos atuais.	Já o RAID 3 é pouco utilizado sobretudo pelo desempenho ruim em aplicações que requerem pequenas leituras e escritas, problema ocasionado pelo fato de que a arquitetura não consegue gerenciar requisições simultâneas.	O RAID 4, por sua vez, tem como principal problema o fato de que o disco de paridade tem que participar de todas as operações de escrita, o que acaba por criar um gargalo e prejudicar o desempenho do sistema como um todo.

Enquanto alguns modelos caíram no desuso, o RAID 5, ao contrário, tornou-se a configuração de discos mais utilizada em servidores corporativos.

O modelo se baseia na segmentação da informação em blocos que são gravados em todos os discos. O

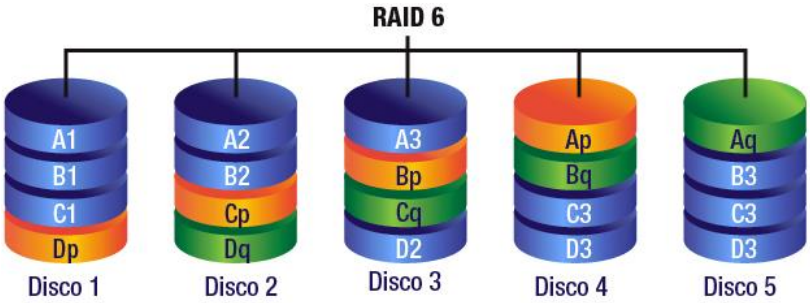
controle de falhas é realizado através da utilização de um bloco de paridade, sem a necessidade de utilização de um disco dedicado. Saiba+



Saiba+
Em caso de falha de apenas um disco, a arquitetura RAID 5 permite que a informação faltante seja reconstruída através da utilização dos dados armazenados nos blocos de paridade distribuídos pelos outros discos. No caso de mais de um disco ser perdido, não há como a informação ser recuperada.

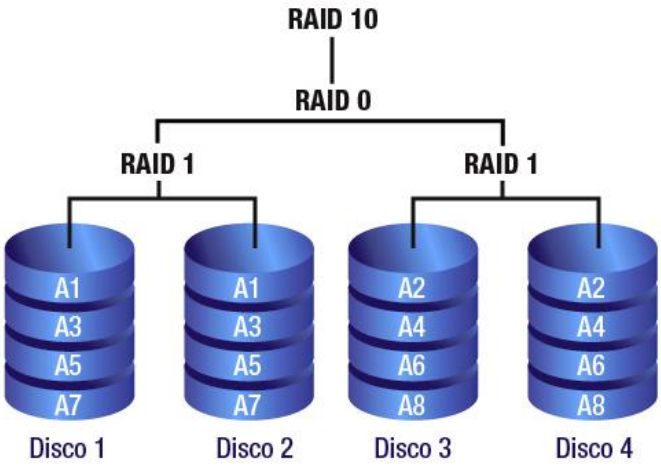
14

O RAID 6 tem estrutura similar ao RAID 5, com a diferença que implementa uma estrutura complementar de paridade que permite, em alguns casos, que a informação possa ser recuperada mesmo no caso de falha de mais de um disco.



15

Além dos modelos RAID tradicionais, alguns modelos híbridos foram propostos para gerenciamento dos arranjos de disco. Um dos mais conhecidos é o RAID 10, que também é chamado de RAID 1+0. Como o próprio nome já diz, o modelo é uma junção do RAID 1 com o RAID 0, o que permite que se associe os benefícios da performance de um modelo com a tolerância a falhas do outro. O maior problema do RAID 1+0 é, assim como no RAID 1, o desperdício de espaço útil de armazenamento.



A tabela abaixo apresenta um breve resumo das características de performance, redundância e eficiência de cada um dos níveis RAID apresentados neste módulo.

RAID	Qtd mínima de discos	Performance	Redundância	Eficiência
0	1	Alta	Baixa	Alta
1	2	Média	Alta	Alta
2	3	Média	Alta	Alta
3	3	Média	Alta	Alta
4	3	Baixa	Média	Alta
5	3	Média	Alta	Alta
6	4	Média	Alta	Alta
10	4	Muito Alta	Muito Alta	Alta

16

RESUMO

Os discos rígidos são um dos principais elementos de *hardware* do computador, assumindo um papel relevante tanto no subsistema de **gerenciamento de memória** quanto no de **entrada e saída**. Internamente, os discos se organizam em função de uma série de componentes, sendo os principais os **pratos**, o **eixo central** ou motor e **os cabeçotes de leitur**a. Por sua vez, as superfícies dos pratos dos discos são organizadas em **trilhas** que se dividem em **setore**s. Um conjunto de trilhas de mesma posição nos diferentes pratos é chamado de **cilindro**.

Os principais atrasos nas operações de disco se referem aos tempos gastos na busca e na latência de rotação. De forma a tentar reduzir o custo destas operações, muitos discos rígidos implementam

políticas de agendamento. Os métodos de agendamento de disco mais conhecidos são o***First Come First Served*** *(FCFS), o* ***Shortest Seek Time First*** *(SSTF), o do* ***Elevador*** *(SCAN), o Look, e o* ***Circular SCAN*** *(C-SCAN)*, sendo que a performance de cada um dos algoritmos vai depender da quantidade e do perfil das requisições de acesso ao disco solicitadas.

Já quando se fala em estruturas de organização de discos rígidos, os principais modelos que vem à mente são o RAID e o JBOD. O **JBOD** atua como um agregador de discos, fazendo com que o sistema operacional consiga lidar com o armazenamento em múltiplos volumes sem, no entanto, agregar ganho de desempenho ou implementar qualquer método de tolerância a falhas.

Já o modelo **RAID** surgiu como uma forma de se prover alguma organização e controle ao modelo JBOD. A arquitetura trouxe uma série de funcionalidades aos arranjos de discos, que auxiliaram na redução da diferença de performance entre e a velocidade de processamento da CPU e a velocidade de acesso aos discos e na promoção de técnicas de tolerância a falhas. O modelo RAID mais utilizado atualmente é o RAID 5.