Sistemas de Arquivos

Pedroso

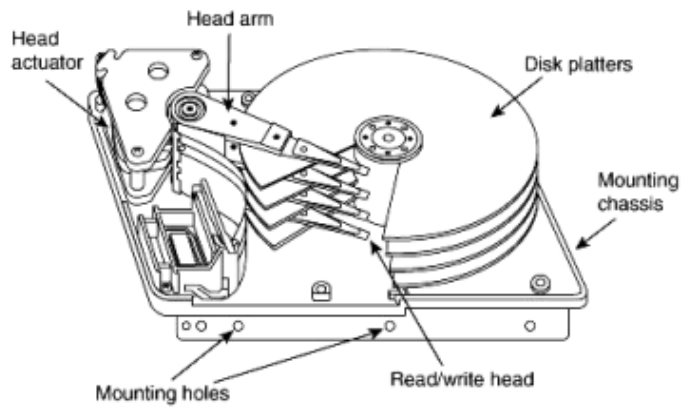
15 de outubro de 2025

Definição de Sistemas de Arquivos

- Um Sistema de Arquivos (Filesystem) é uma estrutura lógica utilizada por um Sistema Operacional (SO) para organizar, armazenar e gerenciar arquivos em um dispositivo de armazenamento.
- Ele atua como uma ponte entre o SO e o hardware, abstraindo a complexidade de como os dados são realmente gravados nos setores do disco.
- Funções principais:
  - ✔ Gerenciamento de espaço livre e ocupado.
  - ✔ Organização de arquivos em diretórios e subdiretórios.
  - ✔ Controle de acesso e permissões.
  - ✔ Manutenção de metadados (nome, tamanho, data de criação, etc.).

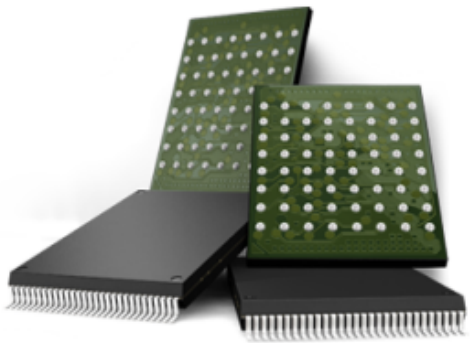
2. Hardware de Armazenamento

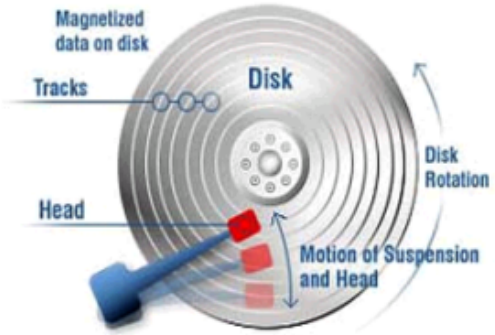
- HDD (Hard Disk Drive):
  - ✔ Tecnologia mecânica com discos magnéticos rotativos.
  - ✔ Acesso a dados é mais lento (milissegundos) devido à latência e ao tempo de busca.
  - ✔ Vantagem: baixo custo por gigabyte.



- SSD (Solid State Drive):
  - ✔ Tecnologia de memória flash (não-volátil).
  - ✔ Acesso a dados é eletrônico, muito mais rápido (microssegundos).
  - ✔ Vantagem: alta velocidade, durabilidade e silêncio.

Hardware de Armazenamento





Comparison Chart	SSD	HDD
Mechanism	Solid NAND Flash	Magnetic rotating platters
Speed (SATA II)	80-250MB/sec	65-85MB/sec
Average Seek Time	0	< 10 ms
Noise	None	Noisy
Power consumption	2W>, Low power consumption	10W, Generates more heat
Weight	Lightweight	Heavier
Endurance	MTBF > 2,000,000 Hours	MTBF < 700,000 Hours
Temperature	-40 ~ 85	0 ~ 60
Reliability	Anti-shock	Non-shock resistant
Shock & Vibration	Excellent	Poor

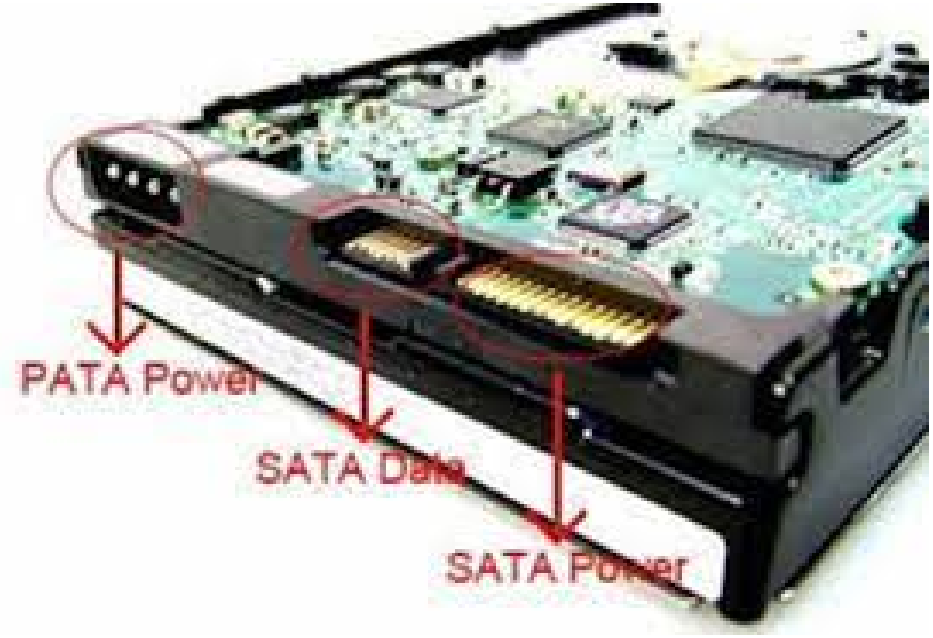
Fonte: Benefits of SSD vs. HDD, https://www.amplicon.es/actions/viewDoc.cfm?doc=SSD-vs-HDD-white-paper.pdf

Hardware de Armazenamento (cont.)

- A tecnologia Linear Tape-Open (LTO) ainda é um dos padrões mais utilizados para backup e arquivamento de dados em larga escala. Embora pareça ser uma tecnologia antiga, ela continua evoluindo e oferece a melhor relação custo-benefício para guardar terabytes ou petabytes de dados por longos períodos. Vantagens:
  - ✔ Custo: O menor custo por gigabyte do mercado.
  - ✔ Capacidade: Uma única fita LTO pode armazenar dezenas de terabytes de dados não compactados.
  - ✔ Durabilidade: As fitas são muito resistentes e podem durar até 30 anos ou mais.
- Desvantagens:
  - ✔ Acesso sequencial: O acesso aos dados é lento, pois a fita precisa ser rebobinada até a posição correta.
  - ✔ Hardware: Exige drives de fita e bibliotecas de fita, que são caros.

A Evolução do SATA

- O padrão SATA evoluiu em diversas versões para acompanhar o aumento da velocidade dos dispositivos de armazenamento, especialmente os SSDs.
- Cada nova revisão dobrou a taxa de transferência teórica da versão anterior.
- A retrocompatibilidade é uma característica-chave, permitindo que dispositivos e controladores mais recentes funcionem com versões mais antigas.



SATA 1.0 (SATA-150)

- Lançado em 2003.
- Taxa de transferência: 1.5 Gbit/s.
- Taxa real de dados (teórica): 150 MB/s.
- Foi a primeira geração a substituir o PATA.
- Suficiente para a maioria dos HDDs da época, mas rapidamente se tornou um gargalo para os primeiros SSDs.

SATA 2.0 (SATA-300)

- Lançado em 2004.
- Taxa de transferência: 3.0 Gbit/s.
- Taxa real de dados (teórica): 300 MB/s.
- Melhoria essencial para aproveitar o potencial dos SSDs que começavam a surgir no mercado.
- Introduziu a tecnologia NCQ (Native Command Queuing), que otimiza a ordem dos comandos para aumentar o desempenho, especialmente em cargas de trabalho com múltiplos acessos.

SATA 3.0 (SATA-600)

- Lançado em 2009.
- Taxa de transferência: 6.0 Gbit/s.
- Taxa real de dados (teórica): 600 MB/s.
- Padrão dominante em desktops e notebooks por muitos anos.
- É o padrão mais comum para SSDs SATA e HDDs modernos.
- Continua sendo o padrão de fato para discos rígidos, que raramente alcançam a velocidade máxima do SATA 3.0.

SATA Express (SATAe)

- Lançado em 2013.
- Uma tentativa de fundir o padrão SATA com a interface PCIe (PCI Express).
- Taxa de transferência (teórica): 1.0 GB/s a 2.0 GB/s, utilizando duas pistas PCIe 2.0 ou 3.0.
- Permitia conectar SSDs de alta velocidade diretamente ao barramento PCIe.
- Foi superado pelo padrão NVMe (Non-Volatile Memory Express) , que se tornou a solução preferida para SSDs de alto desempenho em servidores de alto desempenho.

Tabela Comparativa

Versão	Lançamento	Taxa de Transferência (Teórica)
SATA 1.0	2003	1.5 Gbit/s (150 MB/s)
SATA 2.0	2004	3.0 Gbit/s (300 MB/s)
SATA 3.0	2009	6.0 Gbit/s (600 MB/s)
SATAe	2013	10.0 Gbit/s (1.0 GB/s)
NVMe	2011	112.0 Gbit/s (14.0 GB/s)

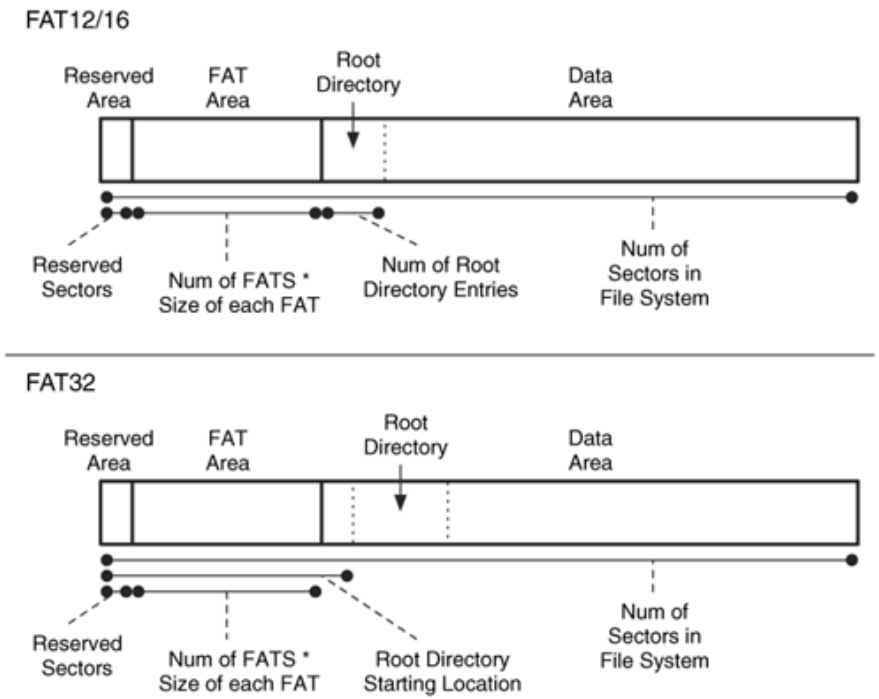
- As taxas de transferência dependem também do padrão do barramento PCI em uso. Exemplo:
  - ✔ PCIe 3.0: SSDs NVMe com essa interface geralmente atingem velocidades de leitura sequencial de até 3.500 MB/s
  - ✔ PCIe 4.0: SSDs NVMe atingem velocidades de leitura sequencial de até 7.000 MB/s.
  - ✔ PCIe 5.0: SSDs NVMe podem ultrapassar 14.000 MB/s.

Partições de Disco

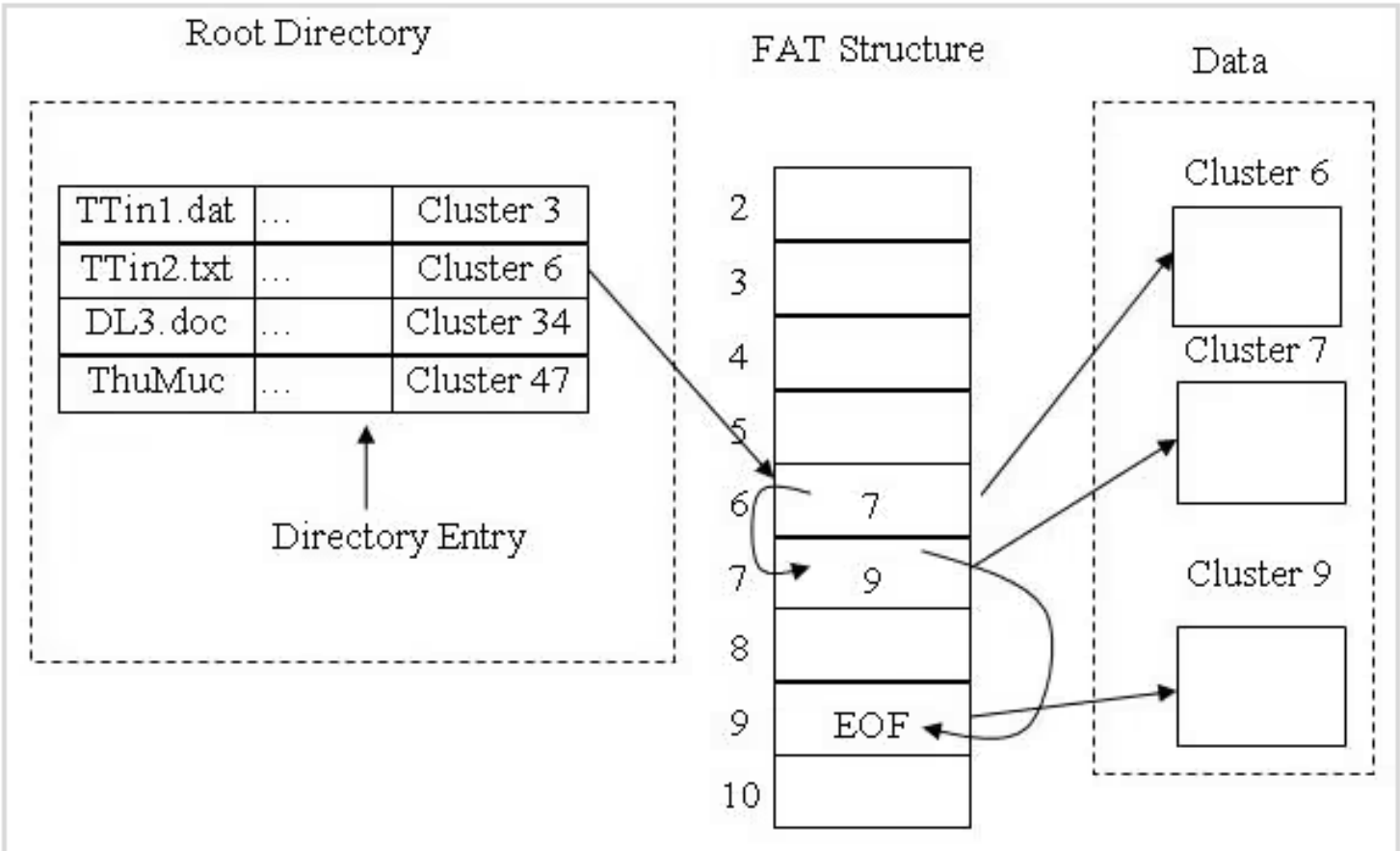
- Definição: Uma divisão lógica de um disco rígido físico.
- História: Originalmente, discos inteiros eram formatados como uma única unidade. Com o aumento da capacidade, as partições se tornaram necessárias para:
  - ✔ Organizar dados.
  - ✔ Instalar múltiplos sistemas operacionais (multi-boot).
  - ✔ Separar dados do sistema de dados do usuário.
- Implementação: O esquema de particionamento (MBR ou GPT) define a estrutura e o tamanho de cada partição no início do disco.
- Uso: Cada partição pode ser formatada com um sistema de arquivos diferente, como NTFS, ext4 ou HFS+.

FAT - File Allocation Table

- Sistema de arquivos antigo, simples e robusto.
- Áreas do Disco:
  - ✔ Setor de Boot: Informações sobre o SO e a tabela de partições.
  - ✔ Área FAT: A tabela que mapeia clusters de dados para arquivos.
  - ✔ Área de Dados: Onde os arquivos e diretórios são armazenados.



FAT - File Allocation Table



FAT - Variações e Desvantagens

- FAT-12: Utilizada em disquetes. Limitações severas de tamanho de arquivo e partição.
- FAT-16: Suporta partições maiores. Clusters maiores, levando a desperdício de espaço (internal fragmentation) para arquivos pequenos.
- FAT-32: Melhoria significativa, suportando partições de até 2TB e arquivos de até 4GB.
- exFAT: Remove a limitação de tamanho de arquivo de 4 GB da FAT-32, com tamanho máximo de partição de 128 petabytes (PB). É a versão em maior uso atualmente.
- Desvantagens da FAT:
  - ✔ Falta de segurança (permissões de arquivo limitadas).
  - ✔ Ausência de journaling (maior risco de perda de dados em falhas).
  - ✔ Alta fragmentação de arquivos.
  - ✔ Limitação de 4GB para arquivos.

1. O que é um Sistema de Arquivos baseado em Inodes?

- É um modelo de organização de dados amplamente utilizado em sistemas operacionais Unix-like, como o Linux, macOS e FreeBSD.
- O nome vem de Nós-Índice ou "nó-I", que se refere à estrutura de metadados central: o inode.
- Diferente de sistemas como a FAT, que armazenam informações de arquivos em uma tabela linear, os sistemas baseados em inodes separam as informações de metadados dos dados do arquivo.

Sistemas de Arquivos baseados em Nós Índice

- Um inode é uma estrutura de dados que armazena todas as informações sobre um arquivo ou diretório, exceto o seu nome e seu conteúdo.
- Cada arquivo no sistema tem um inode único, identificado por um número (o número do inode).
- O nome do arquivo, que vemos no diretório, é apenas um "ponteiro"para o seu inode correspondente. Isso permite que um mesmo arquivo tenha vários nomes (hard links).

O que o Inode Armazena?

- **Tipo de arquivo:** Se é um arquivo regular, diretório, link simbólico, etc.
- **Permissões de acesso:** Quem pode ler, escrever ou executar o arquivo (rwx).
- **Proprietário e Grupo:** O ID do usuário e do grupo que são donos do arquivo.
- **Tamanho do arquivo:** O tamanho em bytes.
- **Carimbos de tempo:** A data de criação, de modificação e do último acesso.
- **Contagem de links:** Quantos nomes de arquivo (links) apontam para este inode.
- **Ponteiros para os blocos de dados:** O mais importante! Uma lista de endereços que aponta para os blocos de dados no disco onde o conteúdo do arquivo está realmente armazenado.

Áreas do Disco em Sistemas de Inodes

- A estrutura de um disco formatado com um sistema de arquivos como o Ext4 (um dos mais populares no Linux) é dividida em grupos de blocos. Cada grupo contém:
- Superbloco: Metadados globais sobre o sistema de arquivos (tamanho, quantidade de inodes e blocos).
- Grupo de Inodes: A tabela que armazena os inodes.
- Bitmap de Inodes: Um mapa que indica quais inodes estão livres ou ocupados.
- Bitmap de Blocos: Um mapa que indica quais blocos de dados estão livres ou ocupados.
- Blocos de Dados: A área onde os dados (o conteúdo) dos arquivos são armazenados.

Como a Leitura de um Arquivo Funciona?

1. O usuário solicita um arquivo, por exemplo, /home/user/documento.txt.
2. O sistema operacional procura o diretório /home e, em seguida, o diretório /user. Cada diretório é uma lista de nomes de arquivos e seus respectivos números de inode.
3. Ao encontrar documento.txt, o sistema obtém o número do inode correspondente.
4. Com o número do inode, o sistema vai até a Tabela de Inodes para ler o inode do arquivo.
5. Dentro do inode, o sistema encontra os endereços dos blocos de dados no disco.
6. O sistema, então, lê os dados desses blocos e entrega o conteúdo do arquivo para o usuário.

Vantagens do Modelo de Inodes

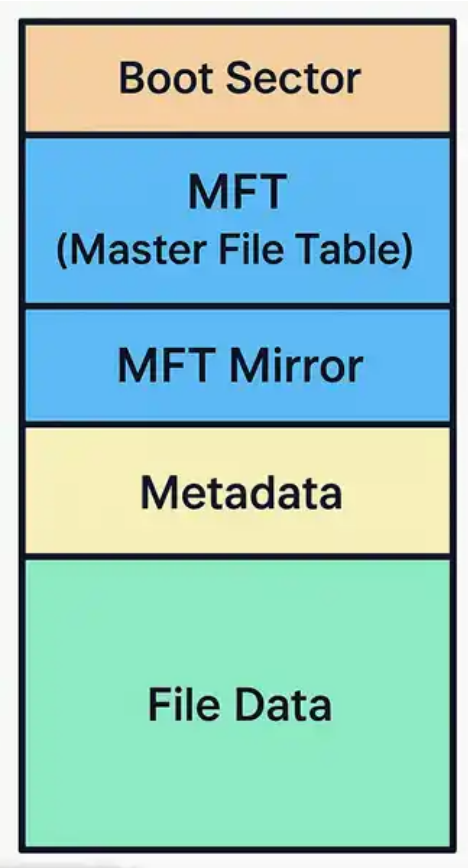
- **Flexibilidade:** Permite que um único arquivo tenha vários nomes e links (hard links), compartilhando o mesmo inode.
- **Eficiência:** A separação de metadados e dados facilita o gerenciamento de permissões e informações do arquivo sem precisar mover os dados.
- **Robustez:** A estrutura de metadados é bem definida e mais resistente a alguns tipos de corrupção. O uso de journaling, como no Ext4, torna o sistema ainda mais confiável.

Desvantagens do Modelo de Inodes

- Limitação de Inodes: A quantidade de inodes em uma partição é definida na formatação. Se todos os inodes forem usados, não é possível criar novos arquivos, mesmo que haja espaço livre em disco.
- Tamanho de Arquivos Pequenos: Para arquivos muito pequenos, o overhead de um inode completo pode ser maior que o tamanho dos dados, levando a um pequeno desperdício de espaço.

NTFS

- Sistema de arquivos padrão do Windows, introduzido com o Windows NT.
- Áreas do Disco:



- ✔ Master File Table (MFT): Coração do NTFS. Metadados de todos os arquivos e diretórios.
- ✔ Área de Dados: Clusters para os dados dos arquivos.
- ✔ O sistema de arquivos NTFS é composto por arquivos de metadados internos, que são essenciais para o seu funcionamento. São invisíveis para o usuário, ex:
  - ✓ $MFT: O próprio MFT.
  - ✓ $LogFile: Mantém um log de transações, permitindo o journaling.
  - ✓ $Boot: Informações de boot.
  - ✓ $Bitmap: Um mapa de bits que rastreia os clusters (blocos de dados) que estão em uso e os que estão livres no volume.
  - ✓ $Volume: Contém informações sobre o volume, como o nome, a versão do NTFS e o estado do disco.

NTFS - Capacidades, Vantagens e Desvantagens

- Capacidades:
  - ✔ Suporte a partições e arquivos muito grandes (até 16 EB).
  - ✔ Compressão de arquivos nativa.
  - ✔ Criptografia (EFS).
- Vantagens:
  - ✔ Journaling: Maior integridade e recuperação rápida após falhas.
  - ✔ Permissões de arquivo granulares (ACLs).
  - ✔ Links simbólicos e hard links.
- Desvantagens:
  - ✔ Não é nativamente suportado por sistemas operacionais como macOS e Linux (necessita de drivers).
  - ✔ Complexidade maior comparada a sistemas mais simples.

EXT3 e EXT4

- Padrões para sistemas baseados em Linux.
- EXT3: Evolução do ext2 com a adição de journaling.
- EXT4: Melhoria do ext3 com foco em desempenho e capacidade.

EXT4 - Características

- Áreas do Disco:
  - ✔ Group Descriptors: Metadados sobre cada grupo de blocos.
  - ✔ Inode Table: Tabela de inodes (estruturas que armazenam metadados dos arquivos).
  - ✔ Block Bitmap / Inode Bitmap: Mapeiam blocos e inodes livres/ocupados.
  - ✔ Data Blocks: Os dados dos arquivos.
- Arquivos Principais:
  - ✔ Inodes (estruturas de metadados).
  - ✔ super_block: Metadados globais sobre o filesystem.

EXT4 - Capacidades, Vantagens e Desvantagens

- Capacidades:
  - ✔ Suporte a partições de até 1 EB e arquivos de 16 TB.
  - ✔ Alocação de extents (chunks contíguos de blocos), reduzindo a fragmentação.
- Vantagens:
  - ✔ Journaling rápido e eficiente.
  - ✔ Backwards compatibility com ext2/ext3.
  - ✔ Robusto e amplamente utilizado em ambientes de servidor.
- Desvantagens:
  - ✔ Sem desfragmentação online nativa (embora menos necessária).
  - ✔ Sem compressão ou criptografia nativa no nível do filesystem.

O que é Journaling?

- O Journaling (ou registro em diário) é uma técnica de sistemas de arquivos que garante a integridade dos dados após uma falha do sistema.
- Ele funciona como um "livro de registro"de todas as transações que o sistema de arquivos pretende realizar.
- É a principal diferença entre sistemas de arquivos modernos (como NTFS, ext4) e os mais antigos (FAT).

Como o Journaling Funciona?

1. **Registro da Transação:** Antes de realizar qualquer alteração, o sistema de arquivos escreve no journal (um arquivo especial) as etapas da transação que será feita (por exemplo, "criar arquivo X", "gravar dados no bloco Y").
2. **Execução da Transação:** O sistema, então, executa a operação real, modificando os metadados e os dados no disco.
3. **Commit da Transação:** Quando a operação é concluída com sucesso, o sistema marca a transação no journal como "concluída"(um "commit").

Journaling em Cenários de Falha

- Cenário de Sucesso: A transação é concluída e marcada no journal. Não há problema.
- Cenário de Falha: Uma falha de energia ocorre antes do "commit"final.
- Ao reiniciar o sistema, ele verifica o journal e encontra a transação incompleta. Ele pode, então, "desfazer"a operação (rollback) ou completá-la, garantindo que o sistema de arquivos não fique em um estado inconsistente e corrompido.
- **Vantagem Principal:** Evita que o sistema tenha que varrer todo o disco (como no CHKDSK do Windows antigo) para recuperar arquivos após uma falha, tornando a inicialização muito mais rápida.

Fragmentação

- A Fragmentação ocorre quando um único arquivo é armazenado em vários pedaços não contíguos (separados) no disco.
- Isso acontece porque o sistema de arquivos tenta usar o primeiro espaço livre que encontra para armazenar os dados, em vez de um único bloco contínuo.
- A fragmentação é um problema dos discos rígidos (HDD), mas não dos discos de estado sólido (SSD)!!!

Como a Fragmentação Acontece?

1. O arquivo é criado e armazenado em um bloco contíguo de espaço livre.
2. O usuário cria e deleta vários outros arquivos pequenos, criando "buracos"de espaço livre no disco.
3. O arquivo original é expandido (ex: um vídeo é editado, uma planilha cresce).
4. O sistema de arquivos não encontra espaço contíguo suficiente para a nova parte do arquivo e a armazena em um dos "buracos"livres.
5. Com o tempo, um único arquivo pode ter vários pedaços espalhados pelo disco.

Escolha do Tamanho do Cluster (Bloco Lógico)

- O tamanho do cluster (ou bloco de alocação) é definido na formatação do disco.
- Ele representa a menor unidade de espaço que pode ser alocada para um arquivo.

Clusters Pequenos (ex: 4 KB)

- **Vantagem:** Menor desperdício de espaço (fragmentação interna) para arquivos pequenos.
- **Desvantagem:** Maior número de clusters para arquivos grandes, aumentando a chance de fragmentação. O sistema de arquivos precisa gerenciar mais entradas, o que pode impactar o desempenho em discos grandes.

Clusters Grandes (ex: 64 KB)

- **Vantagem:** Mais eficiente para arquivos grandes, pois o sistema de arquivos precisa gerenciar menos clusters. Reduz a fragmentação.
- **Desvantagem:** Maior desperdício de espaço para arquivos pequenos. Por exemplo, um arquivo de 1 KB ocupa um cluster inteiro de 64 KB, desperdiçando 63 KB.

Consequências da Fragmentação

- Desempenho Reduzido: Em HDDs, o braço de leitura precisa se mover constantemente para acessar os diferentes pedaços do arquivo. Isso aumenta o tempo de busca e diminui drasticamente a velocidade de leitura e escrita.
- Não Afeta SSDs: Em SSDs, não há partes móveis. O acesso a dados em diferentes locais da memória flash é quase instantâneo, tornando a fragmentação um problema insignificante para o desempenho.
- Desfragmentação: Processo de rearranjar os pedaços de um arquivo para que fiquem contíguos. É uma tarefa importante para otimizar o desempenho de HDDs.

Outros Sistemas de Arquivos

- APFS (Apple File System):
  - ✔ Criado para macOS, iOS.
  - ✔ Otimizado para SSDs, com clonagem de arquivos e snapshots.
- ZFS (Zettabyte File System):
  - ✔ Alta capacidade e robustez.
  - ✔ Inclui gerenciamento de volume, proteção contra corrupção de dados (checksums) e snapshots.
  - ✔ Aplicações: servidores, storage de alta performance.
- ReFS (Resilient File System):
  - ✔ Sistema de arquivos da Microsoft focado em resiliência e integridade de dados.
  - ✔ Utilizado principalmente em Windows Server para armazenamento de grandes volumes.
- ExFAT (Extended FAT):
  - ✔ Uma versão moderna da FAT.
  - ✔ Quebra a limitação de 4GB para arquivos.
  - ✔ Vantagem: alta compatibilidade entre Windows, macOS e Linux.

Estrutura de Diretórios do Unix (FHS)

- O Sistema de Arquivos Unix é organizado em uma estrutura de árvore, iniciando-se no diretório Raiz (/).
- Todos os arquivos e diretórios são acessíveis a partir da Raiz.
- **Caminhos**:
  - ✔ Absolutos: Começam na Raiz (`/home/usuario/arquivo`).
  - ✔ Relativos: Começam do diretório atual (`documentos/relatorio.pdf`).
- **Referências Importantes**:
  - ✔ **.** : Representa o diretório atual.
  - ✔ **..** : Representa o diretório pai (nível acima).
  - ✔ ~ : Representa o diretório *home* do usuário logado.

Comandos Essenciais de Navegação

- `pwd` (Print Working Directory)
  - ✔ Exibe o caminho absoluto do diretório atual.
  - ✔ Exemplo: `/home/aluno/Documentos`
- `cd` (Change Directory)
  - ✔ Usado para navegar entre diretórios.
  - ✔ `cd /etc` : Vai para o diretório `/etc`.
  - ✔ `cd ..` : Sobe um nível na hierarquia.
  - ✔ `cd` ou `cd ~` : Retorna ao diretório *home* do usuário.
- `ls` (List)
  - ✔ Lista o conteúdo de um diretório.
  - ✔ `ls -l` : Lista em formato longo (detalhado, incluindo permissões).
  - ✔ `ls -a` : Lista também os arquivos ocultos (que começam com `.`).

Montagem de Partições

- Montagem é o processo de tornar um sistema de arquivos acessível em um ponto específico da hierarquia de diretórios.
- O ponto da hierarquia onde o sistema de arquivos é anexado é chamado Ponto de Montagem.
- Arquivos de configuração de montagem automática: `/etc/fstab` (File System Table).
- `mount`
  - ✔ Usado para anexar um dispositivo ou sistema de arquivos a um diretório.
  - ✔ Exemplo: `mount /dev/sdb1 /mnt/usb`
  - ✔ `mount` (sem argumentos) lista todos os sistemas de arquivos atualmente montados.
- `umount`
  - ✔ Usado para **desmontar** (desanexar) o sistema de arquivos. Essencial antes de remover um dispositivo.
  - ✔ Exemplo: `umount /mnt/usb`
- `df` (Disk Free)
  - ✔ Exibe a quantidade de **espaço livre** em disco para os sistemas de arquivos montados.

Montagem de Partições Exemplo

```
df -hT
Filesystem  Type     Size  Used  Avail  Use%  Mounted on
tmpfs       tmpfs    783M  2.0M   781M    1%  /run
/dev/sda2   ext4     219G  129G    80G   62%  /
tmpfs       tmpfs    3.9G   53M   3.8G    2%  /dev/shm
tmpfs       tmpfs    5.0M   16K   5.0M    1%  /run/lock
tmpfs       tmpfs    783M  144K   783M    1%  /run/user/1000
/dev/sdc4   ext4      51G  2.1M    49G    1%  /media/writabl
/dev/sdc1   vfat     5.8G  204M   5.6G    4%  /media/DATA001
/dev/sdb1   vfat      40M  116K    39M    1%  /hdd/1
/dev/sdb2   fuseblk   14G  7.3G   5.9G   56%  /hdd/2
/dev/sdb3   fuseblk   94G   38G    56G   41%  /hdd/3
/dev/sdb5   ext4     459G  256G   180G   59%  /hdd/5
/dev/sdb6   ext4     321G  163G   142G   54%  /hdd/6
```

Obs. O tmpfs é um sistema de arquivos especial que armazena todos os seus dados diretamente na memória volátil (RAM).

/run: Contém dados de tempo de execução (runtime) do sistema, como IDs de processos e informações de sockets. /dev/shm: É usado para a memória compartilhada. /run/lock e /run/user/1000: Usados para arquivos de bloqueio e dados específicos da sessão do usuário.

Permissões em Sistemas Unix

- O controle de acesso é dividido em três entidades e três tipos de permissões:

Entidade	Permissão 1	Permissão 2	Permissão 3
Dono (u) (User)	`r` (Read)	`w` (Write)	`x` (Execute)
Grupo (g) (Group)	`r` (Read)	`w` (Write)	`x` (Execute)
Outros (o) (Others)	`r` (Read)	`w` (Write)	`x` (Execute)

- **Permissões Detalhadas**:
  - ✔ `r` (Leitura): Visualizar conteúdo (arquivo) ou listar o diretório.
  - ✔ `w` (Escrita): Modificar/apagar conteúdo (arquivo) ou criar/apagar arquivos no diretório.
  - ✔ `x` (Execução): Rodar um script/programa (arquivo) ou entrar/acessar um diretório.
- O primeiro caractere da saída do comando *ls -l* indica o tipo (- para arquivo, `d` para diretório, `l` para link).

Notação Octal das Permissões

- As permissões são frequentemente representadas por um valor *octal* de 3 dígitos (um para cada entidade).
- Cada permissão tem um valor numérico: **r=4**, **w=2**, **x=1** (pode interpretar como binário também).
- A soma dos valores indica a permissão total para o arquivo.

Representação	Soma	Significado
`rwx`	$4+2+1=\mathbf{7}$	Leitura, Escrita e Execução
`rw-`	$4+2+0=\mathbf{6}$	Leitura e Escrita
`r-x`	$4+0+1=\mathbf{5}$	Leitura e Execução
`r--`	$4+0+0=\mathbf{4}$	Somente Leitura

- Exemplo: `-rwxr-x--x` (Valor Octal: **751**)
  - ✔ `rwx` (Dono): 7
  - ✔ `r-x` (Grupo): 5
  - ✔ `--x` (Outros): 1
- Comando para alterar: `chmod 751 arquivo.sh`

Links Dinâmicos (Soft Links ou Links Simbólicos)

- Um Link Simbólico (*Symlink*) é um ponteiro ou *atalho* para outro arquivo ou diretório.
- É um arquivo especial que contém o *caminho* do arquivo ou diretório de destino.
- Permite o acesso a arquivos em **diferentes sistemas de arquivos**.
- Se o arquivo original for movido ou apagado, o link é quebrado (fica *órfão*).

```
$ ln -s /caminho/do/original /caminho/do/link
$ ln -s /usr/bin/python3 /usr/local/bin/python
```

- **Link vs. Hard Link (Estático):** Hard Links apontam para o mesmo *inode* (dados reais do arquivo) e só funcionam no mesmo sistema de arquivos.

Hard Link (Físico) vs. Soft Link (Simbólico)

Hard Link (Link Físico)

- **Comando:** `ln arquivo_origem link_novo`
- **Exemplo:** `ln dados.txt dados_estatico.txt`
- Não cria um novo arquivo, apenas um novo nome.
- Ambos os nomes apontam para o **mesmo número de INODE**.
- Se o arquivo original for deletado, o link **continua funcionando**, pois os dados só são apagados quando todos os links (nomes) são removidos.

Soft Link (Link Simbólico)

- **Comando:** `ln -s arquivo_origem link_novo`
- **Exemplo:** `ln -s dados.txt dados_dinamico.txt`
- Cria um **novo e pequeno arquivo** com um INODE diferente.
- O novo arquivo armazena apenas o **caminho** do arquivo original.
- Se o arquivo original for deletado, o Soft Link fica **quebrado** e não funciona mais.

Campos da Saída do `ls -ls`

```
$ ls -ls
4 -rwxrw-r-- 1 pedroso staff  714 Aug 14 14:27 es.asm
```

- **4** (Primeiro Campo):
  - ✔ Tamanho do arquivo em blocos (geralmente 1KB ou 512B).
  - ✔ Na saída do `ls -ls` (long format com tamanho em blocos), este é o **tamanho em blocos**.
- **-rwxrw-r–** (Permissões):
  - ✔ $-$: Tipo de arquivo (Arquivo regular neste caso).
  - ✔ **rwx**: Permissões para o **dono** (`pedroso`).
  - ✔ **rw**$-$: Permissões para o **grupo** (`staff`).
  - ✔ **r** $-$ $-$: Permissões para **outros** (`others`).

Campos da Saída do `ls -ls` (Final)

```
$ ls -ls
4 -rwxrw-r-- 1 pedroso staff  714 Aug 14 14:27 es.asm
```

- **1** (Contagem de Links):
  - ✔ Número de links físicos (*hard links*) para o arquivo.
- **714** (Tamanho em Bytes):
  - ✔ Tamanho do arquivo em bytes.
- **Aug 14 14:27** (Data e Hora):
  - ✔ Data e hora da última modificação do arquivo (*timestamp*).
- **es.asm** (Nome do Arquivo):
  - ✔ O nome do arquivo ou diretório.

Permissões Windows (NTFS): ACLs

- **Modelo Principal**: Lista de Controle de Acesso (ACL).
- **Entradas de Controle de Acesso (ACEs)**: Definem as permissões para um usuário ou grupo específico.
- **Identificadores (SIDs)**: Os usuários e grupos são identificados por SIDs (Security Identifiers).
- **Permissões Padrão**:
  - ✔ Leitura (*Read*).
  - ✔ Gravação (*Write*).
  - ✔ Execução (*Execute*).
  - ✔ Modificar (*Modify*).
  - ✔ Controle Total (*Full Control*).
- **Permissões Especiais**: Permitem granularidade extrema (Ex: *Excluir Subpastas e Arquivos*).

Comparação NTFS vs Unix

Característica	Unix/Linux	Windows (NTFS)
Base	Classes Fixas (dono, grupo, outros)	ACLs (Entidades Ilimitadas)
Modos de Permissão	3 (r, w, x)	Dezenas (Muito Granular)
Prioridade de Acesso	Ordem Fixa (dono → grupo → ooutros)	ACEs (Ordem de Herança/Negação)
Herança	Limitada (via `umask`)	Padrão (configurável)
Negação Explícita	Não (apenas ausência)	Sim (ACE de Negação tem prioridade)

Enquanto o Unix prioriza a simplicidade e o desempenho, o NTFS prioriza a flexibilidade e o controle refinado (granularidade).

Função dos Principais Diretórios

- `/etc` (Editable Text Configuration)
  - Contém *arquivos de configuração* específicos da máquina (e.g., `passwd`, `group`, `fstab`).
- `/bin` e `/sbin` (Binaries e System Binaries)
  - `/bin`: Essenciais para o usuário (e.g., `ls`, `cat`, `cp`).
  - `/sbin`: Essenciais para o administrador (e.g., `mount`, `fdisk`).
- `/dev` (Devices)
  - Contém *arquivos especiais* que representam dispositivos de hardware (e.g., discos `/dev/sda`, terminais `/dev/tty`).

Função dos Principais Diretórios (Cont.)

- `/usr` (User System Resources)
  - Contém dados e programas **compartilháveis e somente leitura**. É o maior diretório.
  - `/usr/bin`: Binários não essenciais (aplicativos).
  - `/usr/lib`: Bibliotecas.
  - `/usr/share`: Documentação, arquivos de configuração não específicos.
- `/var` (Variable)
  - Contém dados que *mudam frequentemente* e cujo tamanho pode crescer (e.g., logs, spool de email, cache web).
  - `/var/log`: Arquivos de log do sistema.
- `/mnt` e `/media`
  - Pontos de *montagem temporária* para sistemas de arquivos (HDs externos, pendrives, CDs).

Configuração de Usuários: `/etc/passwd`

- Arquivo que armazena informações básicas de todos os usuários do sistema.
- Cada linha é uma entrada de usuário, com 7 campos separados por **dois-pontos** (`:`):

```
usuario:senha:UID:GID:GECOS:DiretorioHome:ShellLogin
```

- `usuario`: Nome de login.
- `senha`: Geralmente é um `x`, indicando que a senha criptografada está no arquivo `/etc/shadow` (por segurança).
- `UID` (User ID): Identificador numérico do usuário.
- `GID` (Group ID): ID do grupo primário do usuário.
- `GECOS`: Informações adicionais (nome completo, telefone, etc.).
- `DiretorioHome`: Caminho do diretório pessoal do usuário.
- `ShellLogin`: Shell que será executado no login (e.g., `/bin/bash`).

Configuração de Senhas: `/etc/shadow`

- Armazena informações de segurança de usuários, principalmente as **senhas criptografadas (hashing)**.
- Somente o usuário `root` tem permissão de leitura, garantindo a proteção contra ataques de dicionário ou força bruta.
- Cada linha representa um usuário e possui **9 campos** separados por dois pontos (`:`).

```
usuario:senhaHash:ultAlt:min:max:aviso:inat:exp:res
```

1. **Nome de Usuário**: Deve corresponder a uma entrada em `/etc/passwd`.
2. **Senha Criptografada (Hash)**:
   - ✔ Prefixo indica o algoritmo de hash (ex: $6 para SHA-512).
   - ✔ Se for `*` ou `!`, a conta está desabilitada/bloqueada.
3. **Última Alteração** (`ultAlt`):
   - ✔ Número de dias desde 1º de Janeiro de 1970 (Era Unix) em que a senha foi alterada pela última vez.

Configuração de Grupos: `/etc/group`

- Arquivo que armazena informações dos grupos existentes no sistema.
- Cada linha é uma entrada de grupo, com 4 campos separados por **dois-pontos** (`:`):

```
grupo:senha:GID:membros
```

- `grupo`: Nome do grupo.
- `senha`: Geralmente `x` (se houver senha do grupo, geralmente não é usada).
- `GID` (Group ID): Identificador numérico do grupo.
- `membros`: Lista de usuários que são membros adicionais deste grupo, separados por vírgulas.
  - ✔ Usuários cujo grupo primário (`GID` no `/etc/passwd`) é este, não precisam ser listados aqui.

Processo de Boot e Arquivos de Startup (`/etc/rc.d`)

- O processo de inicialização do Unix/Linux passa por várias etapas:
  1. **BIOS/UEFI** e carregamento do Bootloader (GRUB, LILO).
  2. O Bootloader carrega o Kernel do sistema operacional.
  3. O Kernel inicializa o hardware e executa o primeiro processo, o `init` (PID 1).
- O processo `init` (ou seus sucessores como `systemd` ou `sysvinit`) é responsável por inicializar todos os serviços do sistema.
- O `init` lê o arquivo `/etc/inittab` (em sistemas SysVinit) para determinar o *runlevel* (nível de execução).
- Os scripts de startup dos serviços estão tipicamente em:
  - ✔ `/etc/init.d`: Contém os scripts principais dos serviços.
  - ✔ `/etc/rc.d/` ou `/etc/rc`*N*`.d`: Diretórios contendo **links simbólicos** para os scripts em `/etc/init.d`.
- Os links simbólicos possuem nomes padronizados (e.g., `S20apache2` e `K80apache2`) para definir a ordem de execução (**S**tart e **K**ill) em um determinado *runlevel* (*N*).