

# MEIOS DE ARMAZENAMENTO E MÉTODOS DE ACESSO

Prof. Dieisson Martinelli  
[dieisson.martinelli@udesc.br](mailto:dieisson.martinelli@udesc.br)

# Programação

- Meios de armazenamento
- Hierarquia de memória
- Discos rígidos
- Sistemas de arquivos
- Registros
- Métodos de acesso

## Meios de armazenamento

- Existem diversos **meios de armazenamento** em um sistema de computação, cada um com suas próprias características e particularidades, mas todos com um mesmo objetivo: **armazenar dados**, de forma temporária ou permanente

# Meios de armazenamento

- **Armazenamento Interno (volátil)**
  - Registradores
  - Cache L1 (cache interna do processador)
  - Cache L2 (cache externa na placa-mãe)
  - Memória RAM (memória principal)

# Meios de armazenamento

- **Armazenamento Externo (não volátil)**
  - Discos rígidos e SSDs
  - Unidades externas removíveis:
  - Pendrives (memória flash USB)
  - CD-ROM e DVD-ROM (memórias ópticas)
  - Fitas magnéticas

# Meios de armazenamento

- **Registradores:**
  - Pequenas unidades de armazenamento dentro da CPU.
  - Guardam temporariamente dados utilizados diretamente pelo processador.
- **Características:**
  - **Velocidade:** Altíssima (na mesma frequência do clock da CPU).
  - **Capacidade:** Muito limitada (geralmente dezenas ou centenas de bytes).
- **Função:**
  - Armazenar resultados intermediários de cálculos.
  - Armazenar temporariamente endereços e instruções que serão processadas.
- Exemplo: Ao realizar uma soma, os números são carregados em registradores até que a operação seja concluída.

# Meios de armazenamento

- Cache L1 (nível 1)
  - Memória muito rápida localizada dentro do processador.
  - Tem como objetivo acelerar o acesso aos dados e instruções usados frequentemente.
- Dividida em duas partes principais:
  - **Cache de Dados (L1-I)**: Armazena instruções que serão executadas a seguir.
  - **Cache de Instruções (L1-D)**: Guarda dados frequentemente acessados pelo processador.
- **Características:**
  - **Dados**: Guarda os dados frequentemente acessados.
  - **Velocidade**: Muito alta, quase tão rápida quanto os registradores.
  - **Capacidade**: Limitada (normalmente entre 16 KB e 128 KB por núcleo).
- **Função:**
  - Reduzir o tempo de espera do processador, aumentando o desempenho do sistema.
  - Oferecer acesso rápido a dados e instruções recorrentes.
- **Exemplo**: Se o processador perceber que um dado será frequentemente utilizado (por exemplo, resultado de uma operação recente), ele será armazenado na Cache L1 para rápido acesso.

# Meios de armazenamento

- Cache L2 (nível 2)
  - Memória intermediária que pode estar integrada ao processador ou na placa-mãe (em sistemas mais antigos).
  - Serve como "ponte" entre a rápida Cache L1 e a Memória RAM.
- Características:
  - **Velocidade:** Alta, mas inferior à L1.
  - **Capacidade:** Geralmente entre 256 KB e 12 MB.
- **Função:** Armazenar dados que não cabem na L1, atuando como intermediária entre a L1 e a RAM.
- **Exemplo:** Quando um dado não está disponível na Cache L1, o processador verifica primeiro a Cache L2, antes de buscar na memória RAM, ganhando eficiência e tempo.



# Meios de armazenamento

- **RAM** (Random Access Memory)
  - É a memória principal do sistema computacional.
  - Armazena temporariamente os dados e programas em execução.
- **Características:**
  - **Velocidade:** Mais lenta que as caches, porém ainda consideravelmente rápida.
  - **Capacidade:** Gigabytes (GB), dependendo do sistema.
- **Função:** Armazena dados e instruções usados por programas em execução.
- **Exemplo:** Ao abrir um jogo, arquivos essenciais são carregados do HD ou SSD diretamente na memória RAM, para permitir que o jogo funcione de forma rápida e fluida.

# Classificação dos meios de armazenamento

- Armazenamento **primário**:
  - Memória cache, registradores e memória principal
    - Dados acessados frequentemente (permite acesso rápido e manipulação de dados – funciona como uma mesa de trabalho)
- Armazenamento **secundário**:
  - Discos magnéticos e *flash*
    - Dados acessados com certa frequência (acesso mais lento que a memória principal)
- Armazenamento **terciário**:
  - Fitas magnéticas, CDs/DVDs, outras mídias removíveis
    - Dados que não são acessados com frequência



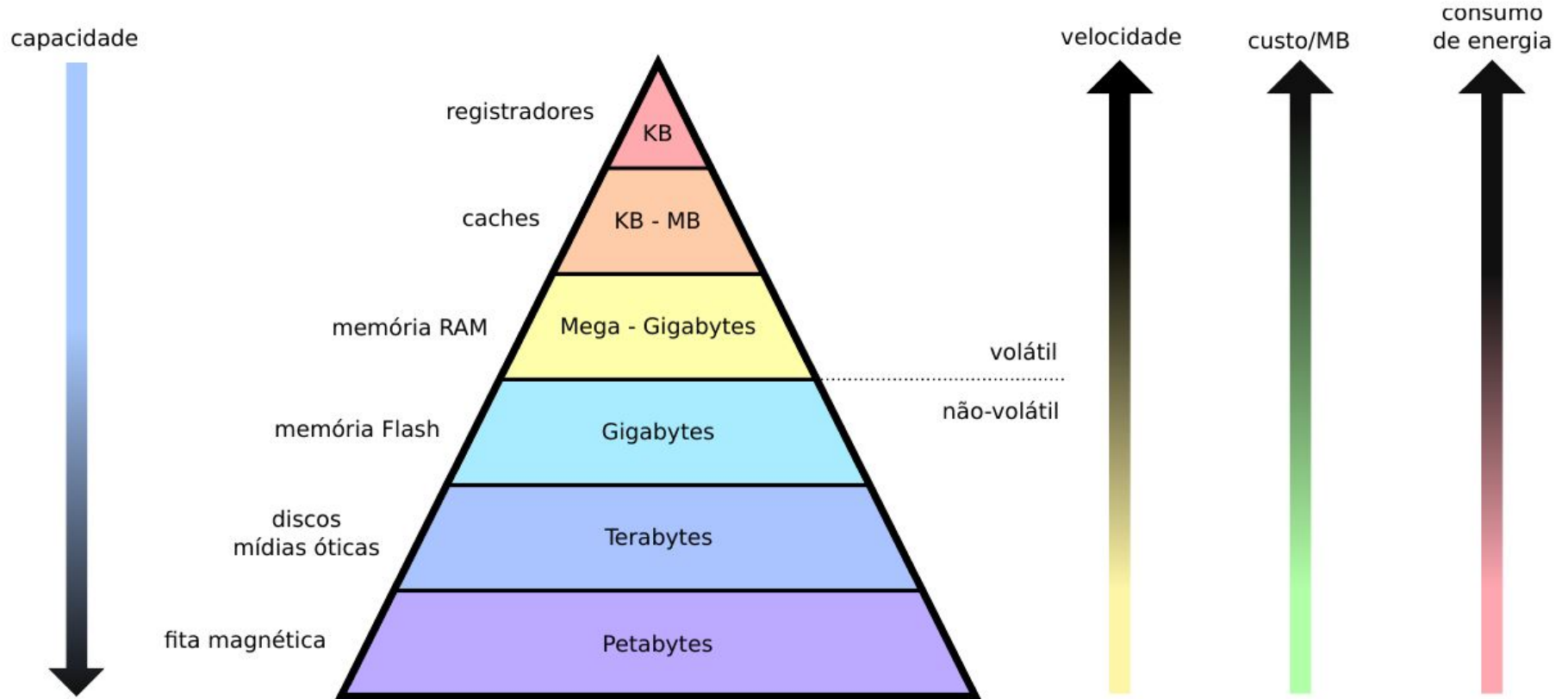
# Meios de armazenamento

- Vantagens e desvantagens
  - Dispositivos secundários e terciários em relação aos primários são:
    - Dispositivos **mais lentos**
    - Fornecem **maior capacidade** de armazenamento
    - São do tipo **não voláteis** (armazenamento permanente)
    - Oferecem **menor custo**

# Meios de armazenamento

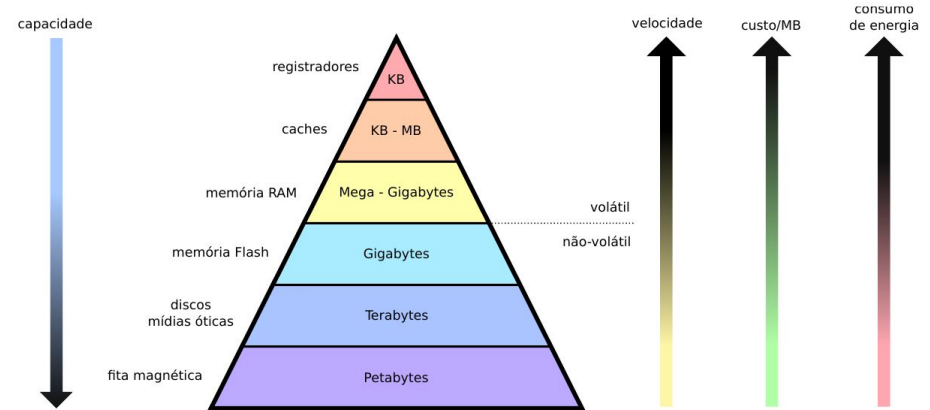
- Como os componentes de hardware utilizados p/ armazenamento de dados são construídos usando diversas tecnologias, eles possuem características distintas, como:
  - A **capacidade** de armazenamento
  - A **velocidade** de operação
  - O **consumo** de energia
  - O **custo** por byte armazenado
  - A **volatilidade**
- Essas características permitem definir uma hierarquia de memória, geralmente representada na forma de uma pirâmide

# Hierarquia de memória



# Hierarquia de memória

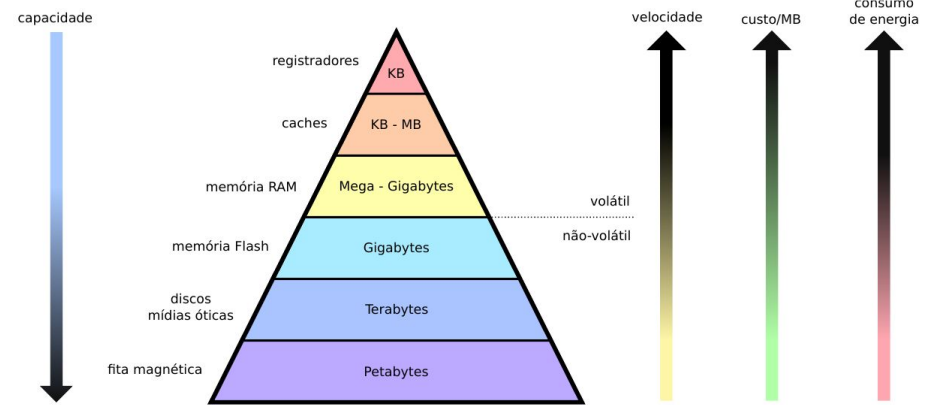
- Nessa pirâmide, **memórias mais rápidas** (como os registradores da CPU e os caches) são menores (têm menor capacidade de armazenamento), mais caras e consomem mais energia que memórias mais lentas, como a memória principal (RAM) e os discos



- As memórias mais rápidas **são voláteis**, ou seja, perdem seu conteúdo ao ficarem sem energia, quando o computador é desligado
- Memórias que preservam seu conteúdo mesmo quando não tiverem energia, como as unidades Flash e os discos rígidos, são denominadas **não-voláteis**

# Hierarquia de memória

- Outra característica importante das memórias é a rapidez de seu funcionamento, que pode ser traduzida em duas grandezas: o **tempo de acesso** (ou latência) e a **taxa de transferência**



- O **tempo de acesso** representa o tempo necessário para o sistema iniciar a transferência de dados após receber uma solicitação de leitura ou escrita.
- A **taxa de transferência** mede quantos bytes por segundo podem ser efetivamente lidos ou escritos após o início da transferência. Ela está diretamente relacionada à largura de banda do dispositivo e à velocidade do barramento de comunicação.

# Hierarquia de memória

- Valores de **tempo de acesso** e **taxa de transferência** típicos de alguns meios de armazenamento usuais

Meio	Tempo de acesso	Taxa de transferência
Cache L2	1 ns	1 GB/s (1 ns por byte)
Memória RAM	60 ns	1 GB/s (1 ns por byte)
Memória <i>flash</i> (NAND)	2 ms	10 MB/s (100 ns por byte)
Disco rígido SATA	5 ms (tempo para o ajuste da cabeça de leitura e a rotação do disco até o setor desejado)	100 MB/s (10 ns por byte)
DVD-ROM	de 100 ms a vários minutos (caso a gaveta do leitor esteja aberta ou o disco não esteja no leitor)	10 MB/s (100 ns por byte)



# Discos rígidos

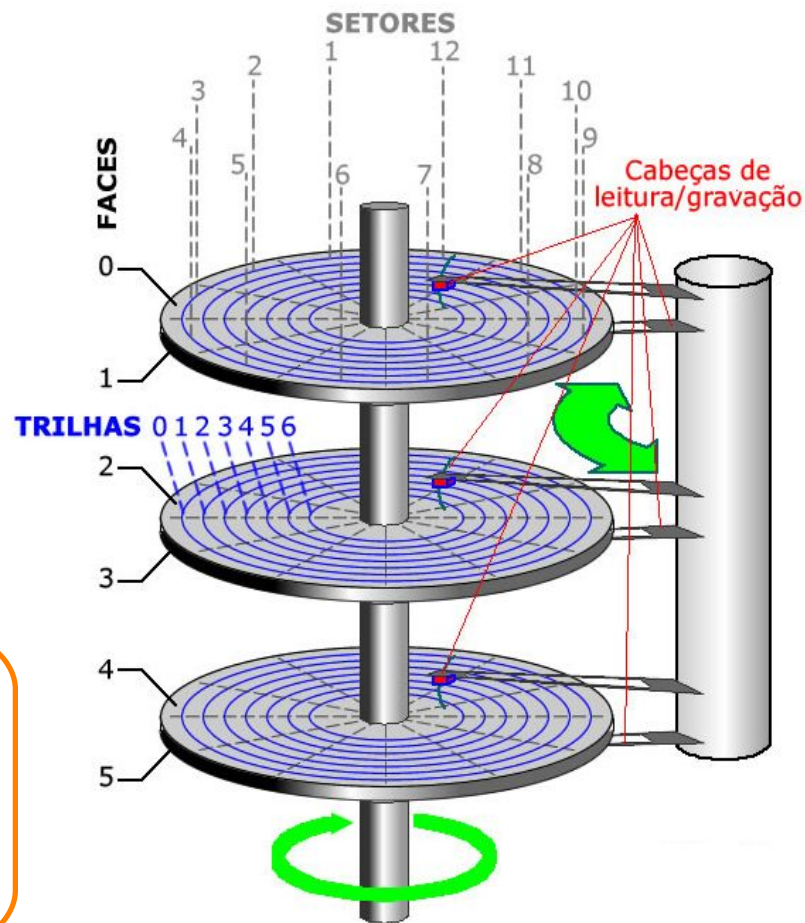
- Discos rígidos estão presentes na grande maioria dos computadores pessoais e servidores
  - Um disco rígido permite o **armazenamento persistente** (não-volátil) de grandes volumes de dados com baixo custo e tempos de acesso razoáveis
  - Além disso, a **leitura e escrita** de dados em um disco rígido é mais simples e flexível que em outros meios, como fitas magnéticas ou discos óticos (CDs, DVDs)
- Por essas razões, eles são intensivamente utilizados em computadores para o **armazenamento de arquivos** e frequentemente usados como **área de armazenamento** de páginas em sistemas de paginação em disco (*swapping* e *paging*)

# Estrutura de um disco rígido convencional

- Os discos rígidos magnéticos (HDs) são divididos em:
  - **Trilhas**
  - **Cilindros**
  - **Setores** (subdivisão de uma trilha)
- O conjunto destas informações é denominado **geometria do disco**

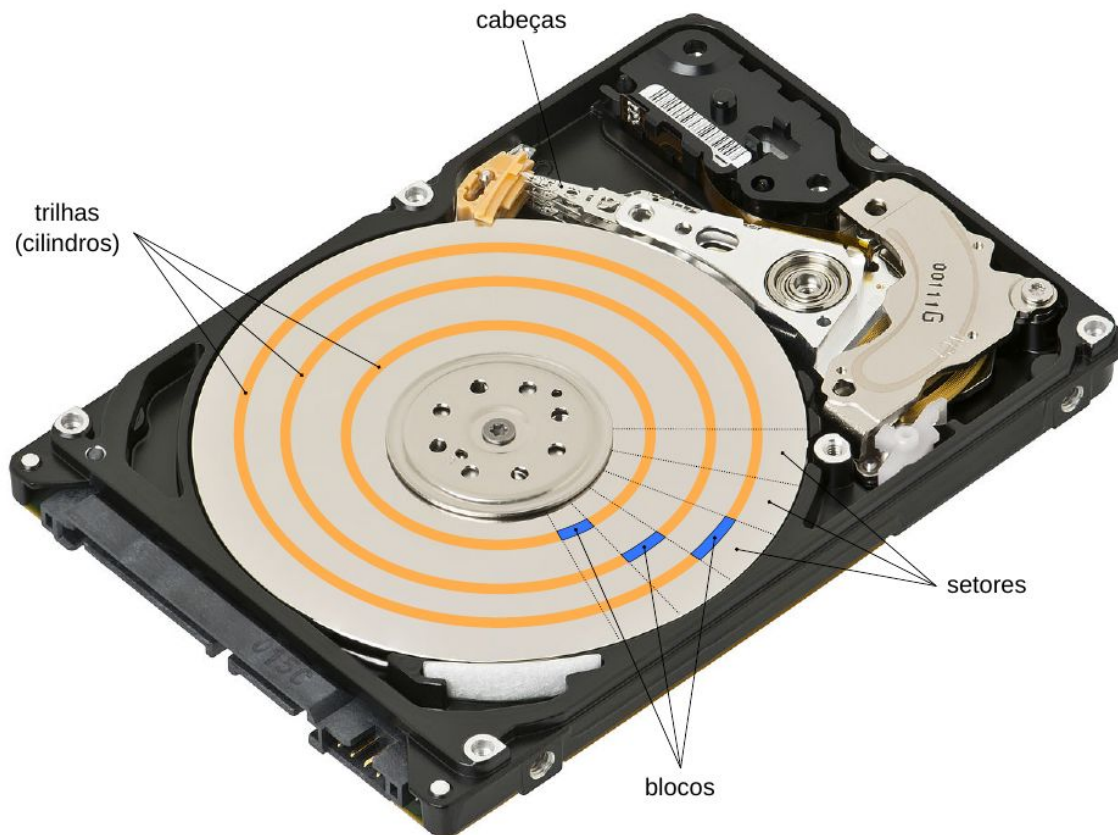
O HD possui uma **cabeça de leitura** para **cada face de disco**, mas **todas as cabeças** estão **presas** na mesma **peça de metal**, o **braço de leitura**; por isso **não possuem movimento independente**: para onde uma vai, todas vão.

O termo “**cilindro**”, corresponde à **todas as trilhas de mesmo número** (quando há mais de um disco metálico no disco rígido)  
Ex.: o cilindro 0 é formado por todas as trilhas número 0 (zero), em todas as faces de disco.



# Estrutura de um disco rígido convencional

- A intersecção de uma **trilha** e um **setor** em uma face define um **bloco físico**
  - Alguns autores denominam essa intersecção como “**setor de trilha**”
- Um disco típico contém várias faces e milhares de trilhas e de setores por face, resultando em milhões de blocos disponíveis
  - Cada bloco pode ser individualmente acessado (lido ou escrito) através de seu **endereço**



# Funcionamento de um disco rígido convencional

- **CHS (Cylinder-Head-Sector)**

- Método tradicional de endereçamento dos blocos.
- Para localizar um bloco, era preciso indicar:
- Cilindro (trilha)
- Cabeça (face)
- Setor
- Cada combinação apontava para uma posição física específica no disco.

- **LBA (Logical Block Addressing)**

- Sistema moderno que endereça blocos de forma linear (0, 1, 2, 3, ...).
- Simplifica a gestão pelo sistema operacional, eliminando a necessidade de cálculos complexos.

- **Integração entre LBA e a Estrutura Física**

- Apesar do disco manter sua estrutura física (faces, trilhas e setores), o firmware do disco converte automaticamente os endereços LBA para o formato CHS, sem que o usuário ou o sistema operacional percebam essa transformação.

# Operações do disco para a memória principal

- **Os dados precisam estar na memória principal para processamento**
  - Antes de serem manipulados, os dados devem ser transferidos do disco para a memória principal.
- **Leitura e escrita ocorrem em blocos físicos ou clusters**
  - O disco armazena dados em blocos físicos individuais ou em clusters (conjunto de blocos contíguos).
  - Mesmo que apenas um pequeno dado seja necessário, o bloco inteiro é transferido para a memória.
- **Operações de Entrada e Saída (I/O)**
  - Qualquer leitura ou escrita no disco é considerada uma operação de I/O, sendo gerenciada pelo sistema operacional.
- **Tempo de Acesso (Latência)**
  - **Tempo de Procura:** Tempo necessário para a cabeça de leitura/gravação alcançar o bloco desejado no disco.
  - **Tempo de Rotação:** Tempo de espera até que o bloco gire e fique alinhado com a cabeça de leitura.
  - **Tempo de Transferência:** Tempo necessário para ler ou gravar os dados no bloco enquanto ele passa sob a cabeça do disco.

## Outras características dos discos

- Para cada bloco a ser **lido/escrito**, a cabeça de leitura/gravação deve se **posicionar na trilha** desejada e aguardar o disco girar até **encontrar o setor** desejado
- Por serem **dispositivos eletromecânicos**, os discos rígidos são extremamente lentos, se comparados à velocidade da memória ou do processador
- Os valores médios típicos de atrasos por tempo de acesso para **discos de uso doméstico** são entre 10ms e 15ms
  - Os atrasos por tempo de acesso podem ter um forte impacto no desempenho de um sistema (por exemplo, tornar um servidor web lento para responder aos acessos de clientes)

## Outras características dos discos

- Os discos rígidos típicos são compostos por um ou mais discos metálicos que giram juntos em **alta velocidade** (usualmente entre 4.200 e 15.000 RPM)
- Até 2010, os discos rígidos usavam **blocos físicos** de 512 bytes, mas o padrão da indústria migrou nos últimos anos para blocos de 4.096 bytes
- Um disco rígido contém várias faces e milhares de trilhas e de setores por face, resultando em milhões de blocos de dados disponíveis. Cada bloco pode ser individualmente acessado (lido ou escrito) através de seu **endereço**
- O sistema operacional fornece uma abstração de hardware para os discos rígidos, que é conhecida por **sistema de arquivos**

# Sistema de arquivos

- Um **sistema de arquivos** é responsável por organizar os dados armazenados em disco de forma lógica, facilitando o acesso e a gestão das informações.
- Organização padrão:
  - **Arquivos**
    - Unidade básica de armazenamento de dados acessível ao usuário.
    - Contém informações organizadas de forma estruturada.
  - **Diretórios**
    - Facilitam a navegação e a estruturação dos dados no disco.
    - Permitem classificar e agrupar arquivos de maneira organizada.
- Fornecer uma **visão lógica** (abstrata) do disco rígido, proporcionando o uso eficiente e a organização dos dados
- Escalonamento (capacidade de organizar dados em níveis – subpastas)



# Funções do Sistema de Arquivos

- **Visão Lógica do Disco**

- Abstrai a complexidade física do armazenamento, permitindo ao usuário acessar arquivos de maneira intuitiva.

- **Eficiência no Uso dos Dados**

- Gerencia o armazenamento para otimizar espaço e melhorar o desempenho.

- **Escalonamento (Hierarquia de Diretórios)**

- Permite a organização de arquivos em múltiplos níveis (subpastas).
- Facilita a categorização e localização de informações de forma estruturada.

# Características dos sistemas de arquivos

- O sistema de arquivos também possibilita o gerenciamento do conteúdo dos discos do sistema. Esse gerenciamento inclui tanto o **controle de acesso** (segurança), quanto o controle da localização e manuseio dos conteúdos de cada arquivo
- Vantagens do ponto de vista do **sistema operacional**:
  - Possibilidade de gerenciamento do espaço físico, com uma estrutura que permite gerenciar blocos livres / ocupados, endereçamentos, metadados
- Vantagens do ponto de vista do **usuário**:
  - Possibilidade de nomear um arquivo e manipular: criar, apagar, ler, mover e escrever, controlar seus direitos de acesso
  - Efetuar backups, obter estatísticas, compartilhamento, procura e classificação

# Sistema de arquivos: interface do usuário

- No SO o acesso a arquivos é realizado através de **chamadas de sistema** (*system call*). Há um conjunto de funções privilegiadas acessadas apenas em **modo Kernel** e um programa de usuário (**modo usuário**) pode usar estas funções apenas indiretamente através de chamadas de sistema
- O sistema operacional efetua o vínculo entre **nome simbólico** e lugar de armazenamento no disco (o espaço físico: blocos ou *clusters*)
- Exemplos de chamadas de sistema para manipulação de arquivos:
  - Métodos *read( )*, *open( )*, *write( )*; com um simples *click* ou via *shell*
- Para o usuário o arquivo é caracterizado:
  - Por um **nome** (visualizado com ou sem extensão)
  - Por **atributos** (somente leitura, leitura e escrita, arquivo de sistema, etc)
  - Por uma **organização lógica** (a estrutura de diretórios, uma extensão)

# Arquivos e registros

- **Arquivos:**

- Possuem um espaço de endereçamento lógico contíguo
  - Por exemplo: FAT16 utilizava *clusters* de 32KB, entretanto apresentava alguns problemas, como desperdício de espaço (ex. arquivo de 100KB)
- Arquivos são formados por um conjunto de registros relacionados
  - Por exemplo: conteúdo do arquivo com dados de clientes

- **Registros:**

- São unidades lógicas manipuladas por um programa, constituídas por uma sequência de itens chamados de **campos** ou **atributos**
  - Por exemplo: <código, nome, data\_nascimento>

# Campo

- ▶ **Campo** é o espaço de um ou mais caracteres onde são armazenados os dados de mesma natureza; é a menor unidade lógica de informação em um **arquivo**

- Possuem: nome, tipo, tamanho

- Exemplo:

Nome do Campo	Tipo	Tamanho
Código	Numérico	4
Nome	Texto	40
Data_Nascimento	Texto	10

- Campo corresponde a cada uma das informações que se deseja modelar a respeito da entidade ou objeto considerado

# Formato dos registros

- Registros com campos de **tamanho fixo**:
  - O comprimento do campo é fixo (ex.: número de caracteres p/ um nome), cada campo tem o mesmo comprimento em todos os registros
  - Cada campo ocupa no arquivo de registro um **tamanho fixo**, pré-determinado. O fato do tamanho ser conhecido facilita a recuperação de cada campo em um arquivo de registros
  - O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (**desperdício**). É inapropriado quando para grandes quantidades de dados c/ tamanho variável
  - Razoável apenas se o comprimento dos campos é realmente fixo, ou apresenta pouca variação
- Registros com campos de **tamanho variável**:
  - O comprimento dos campos podem ter tamanhos variáveis em diferentes registros

# Formato dos registros

- Artifícios para identificação de campos:
  - Campos com **indicador de comprimento**.
    - Nesse método, o tamanho de cada campo é armazenado antes do próprio dado.
    - Isso permite que o sistema saiba exatamente quantos bytes pertencem ao campo antes de ler seu conteúdo.
    - **Exemplo:** Os dois primeiros bytes de um campo podem indicar o seu tamanho. Se um campo armazena um nome e os primeiros dois bytes contêm "08", o sistema sabe que os próximos 8 caracteres correspondem ao nome.
    - **Desvantagem:** É necessário padronizar a forma como esse tamanho será armazenado para evitar confusão entre os dados e a informação de comprimento.
    - **Vantagem:** Flexível, pois permite campos de tamanho variável.

# Formato dos registros

- Artifícios para identificação de campos:
  - Campos Separados por Delimitadores
    - Utiliza caracteres especiais que indicam onde um campo termina e outro começa.
    - Esses caracteres não podem fazer parte dos dados, para evitar confusão.
    - **Exemplo:** Se tivermos um registro de cliente com nome e idade, ele pode ser armazenado assim: "João Silva/32#"
    - **Desvantagem:** É necessário escolher delimitadores únicos que nunca apareçam nos dados.
    - **Vantagem:** Simples e eficiente para leitura sequencial de dados.



# Formato dos registros

- Artifícios para identificação de campos:
  - Uso de Tags para Identificação de Registros e Campos
    - Utiliza o formato “chave = valor” para identificar os campos, muitas vezes combinado com símbolos como colchetes { }..
    - Muito utilizado em arquivos JSON, XML e metadados de sistemas modernos.
    - **Exemplo:** { nome="João Silva", idade=32, cidade="São Paulo" }
    - **Desvantagem:** Ocupa mais espaço, pois armazena os nomes dos campos junto com os valores.
    - **Vantagem:** Fácil de entender e interpretar, permite flexibilidade na ordem dos campos.

# Chave

- Uma chave é um valor armazenado em um ou mais campos que permite localizar um ou mais registros em um arquivo.
  - Ex: nome (João), porém não define um registro único; ao contrário de código (10321), que representa um único registro do arquivo

Código	*Nome	Data_Nascimento
10123	João	01-03-1990
10321	José	30-03-1985
10333	Maria	11-11-1991
10432	João	31-12-1980

- Dependendo do tipo de campo e da organização dos registros, pode-se definir diferentes tipos de chaves

# Tipos de chaves

- **Chave primária:**
  - Possibilita a identificação única dos registros
  - Apresenta obrigatoriamente um valor diferente para cada registro
    - Exemplo: código, CPF
- **Chave secundária:**
  - Permite a recuperação de registros, mas não necessariamente é única
    - Exemplo: nome, data\_nascimento, salário
- **Chave de acesso (índice):**
  - Chave utilizada para acessar o registro desejado (apontar para um local específico do arquivo de registros)
- **Chave de ordenação:**
  - Chave utilizada para estabelecer a ordem dos registros

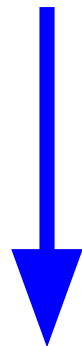
# Tipos de chaves

- Cada tipo de chave tem um papel fundamental na organização, busca e recuperação de dados dentro de um sistema de arquivos.
  - A chave primária garante unicidade.
  - A chave secundária permite buscas mais flexíveis.
  - A chave de acesso (índice) acelera a localização dos registros.
  - A chave de ordenação define como os registros são organizados no arquivo.
- Com essas chaves, conseguimos gerenciar grandes volumes de dados de forma eficiente, tornando o acesso e a manipulação das informações muito mais rápidos e organizados

## Métodos de acesso

- **Acesso sequencial:**

- O acesso sequencial refere-se ao armazenamento e recuperação de registros de acordo com uma ordem (ex.: "um depois do outro")
- Isso significa que, para acessar um determinado registro, pode ser necessário percorrer os registros anteriores até encontrá-lo
- Determinados registros geralmente são armazenados numa ordem ascendente ou descendente, por um código de registro



Código	Veículo	Ano
00001	Gol	2009
00002	Palio	2007
00003	Fox	2011
00004	Celta	2008
...	...	...

# Métodos de acesso

- **Acesso sequencial:**
  - No acesso sequencial a leitura é feita comparando-se o argumento de pesquisa com cada registro lido, lido de forma sequencial
  - Na inclusão deve ser gerado um novo arquivo a partir do atual, para intercalar o novo registro (qualquer modificação no arquivo gera um novo arquivo)
  - Na exclusão também deve ser gerado um novo arquivo a partir do atual, eliminando o registro desejado

# Métodos de acesso

- **Acesso sequencial:**

- Outra forma de exclusão para arquivos em disco é, indicar, em um campo adicional, um estado de registro excluído (um campo para “marcar” a exclusão, sem excluir de verdade)

Código	Veículo	Ano	Status
00001	Gol	2009	1
00002	Palio	2007	0
00003	Fox	2011	0
00004	Celta	2008	1
...	...	...	...

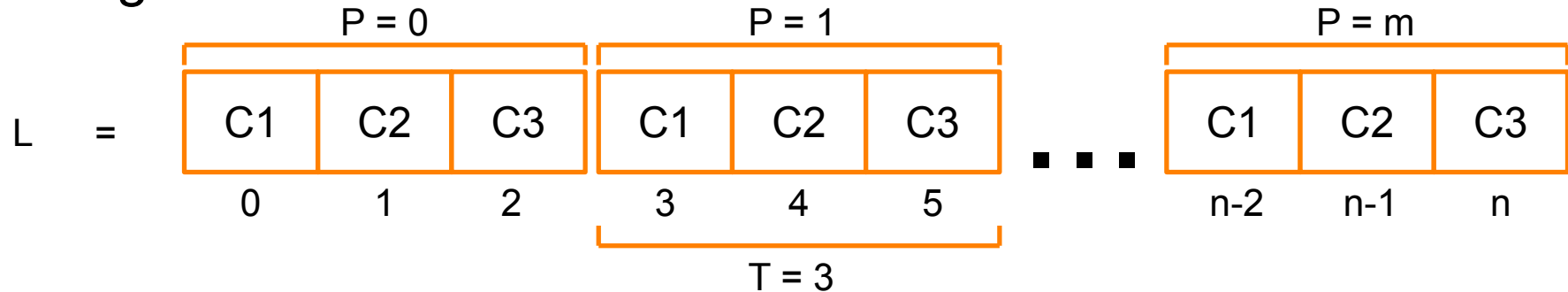
(Campo “Status”: 1 = registro excluído, 0 = registro ativo)

# Métodos de acesso

- **Acesso sequencial:**

- Um exemplo de acesso a registros de tamanho fixo é dado pela fórmula:  $L = (T * P)$ 
  - Onde, L = Localização inicial, T = Tamanho do registro, P = Posição

- **Registros:**



- **Observação:** ler arquivo de T em T bytes



# Métodos de acesso

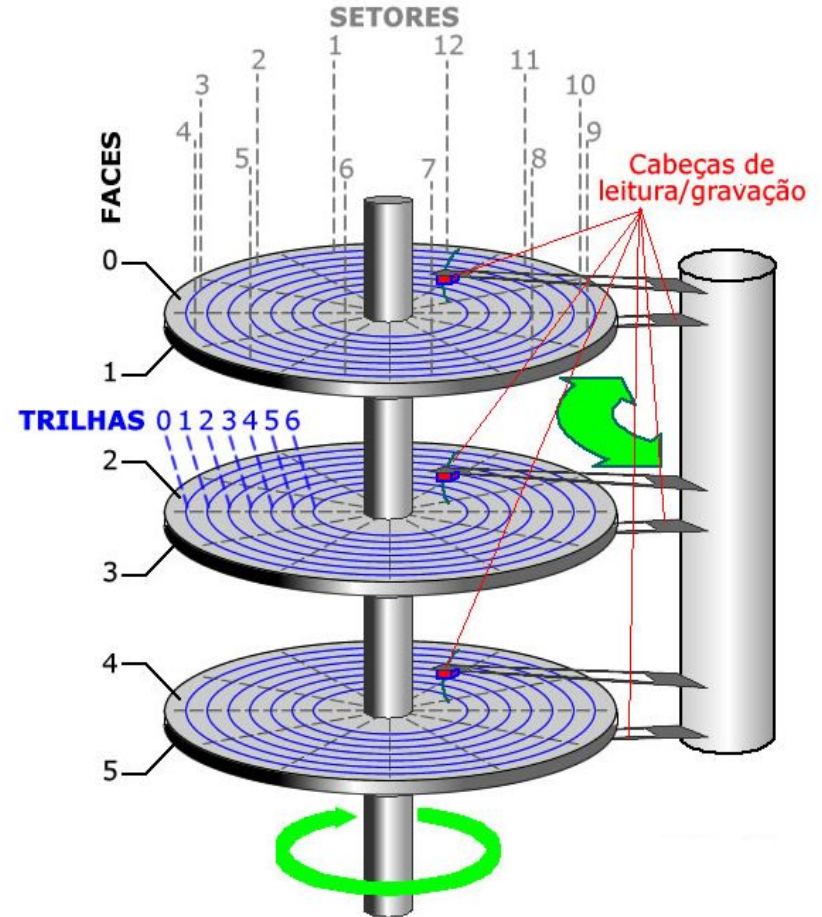
- O acesso sequencial é usado principalmente nos **meios de processamento por lotes** (onde, geralmente, ocorre uma modificação em todos os registros de um arquivo ou na grande maioria deles)
  - Por exemplo, a atualização de determinado percentual no campo “salário”
- Vantagens
  - Fácil de programar
  - Requer estruturas de arquivos simples
  - Pode ser aplicada a qualquer arquivo
- Desvantagem
  - Pode ser ineficiente em aplicações onde apenas uma pequena proporção dos registros são afetados por um dado lote de transações devido a necessidade de percorrer todo o arquivo para atualizar uns poucos registros (lentidão)

# Métodos de acesso

- Acesso **sequencial indexado**:
  - É formado por um **arquivo sequencial** (arquivo de registros) e por um **índice** (arquivo de índices)
- Como funciona?
  - Cada registro possui uma chave de acesso, como um código único.
  - O índice armazena essas chaves associadas ao endereço do registro dentro do arquivo sequencial.
  - Em vez de percorrer todos os registros um por um, o sistema consulta primeiro o índice e, com base nele, acessa diretamente a posição do registro desejado.

## Métodos de acesso

- Um exemplo de acesso **sequencial indexado** ao disco rígido
  - Cilindro é o conjunto de todas as trilhas de mesmo número, verticalmente alinhadas
  - A memória principal mantém um **índice de cilindros**
  - Cada cilindro aponta para um **índice de páginas**, que são os setores (c/ espaço típico de 512 bytes para dados, dependendo da formatação)
    - Obs.: antigo padrão CHS, atualmente é utilizado LBA (*logical block address*)



# Métodos de acesso

- Um exemplo de acesso **sequencial indexado** ao disco rígido:

X'	48	121	184	...
P	1	2	3	...

Índice de cilindros (memória principal)

Índice Chave X aponta p/ Índice de Página P  
Cada Chave X'' representa um Cilindro

X''	1	2	3	...
P	3	14	25	...

Índice de páginas do cilindro (setores)

3	5	7	11
---	---	---	----

14	17	20	21
----	----	----	----

25	29	32	36
----	----	----	----

setores do disco

## Exercícios

1

Dado um arquivo sequencial, com registros de tamanho fixo, contendo números de CPF (11 bytes) seguidos de datas de nascimento (8 bytes), escreva uma fórmula possível para acessar o 3º registro deste arquivo:

```
02147834523140219840818947232809041979016443766302110199609178
98344018051986167734221102312200100173255450080919680409018769
02511198100789089990090919991753135151210101987.....
```

```
.....
.....
.....
```

(arquivo parcial de registros)

# Execícios

2

Utilizando o mesmo arquivo (*string*) do exercício 1, faça um método que mostre na tela os registros da seguinte forma:

- 021.478.345-23 14/02/1984
- 081.894.723-28 09/04/1979
- 016.443.766-30 21/10/1996
- 091.789.834-40 18/05/1986
- 167.734.221-10 23/12/2001
- 001.732.554-50 08/09/1968
- 040.901.876-90 25/11/1981
- 007.890.899-90 09/09/1999
- 175.313.515-12 10/10/1987