Marisangila Alves, MSc
marisangila.alves@udesc.br
marisangila.com.br

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

UDESC
Universidade do Estado de Santa Catarina

2025/2

Sistemas Operacionais

Gerenciamento de Sistemas de Arquivos

Sumário

Sistemas de Arquivos

> *Um arquivo é um mecanismo de abstração. Ele fornece uma maneira para armazenar informações sobre o disco e lê-las depois. Isso deve ser feito de tal modo que isole o usuário dos detalhes de como e onde as informações estão armazenadas, e como os discos realmente funcionam*
>
> (TANENBAUM, 2010)

São organizados em estruturas hierarquicas (árvores) chamadas **diretório**

O que é Sistemas de arquivos?

- Estrutura de dados armazenada;
- Implementações: NTFS, FAT, Ext4, HFS;
- Arquivos possuem Conteúdo, Atributos e Operações

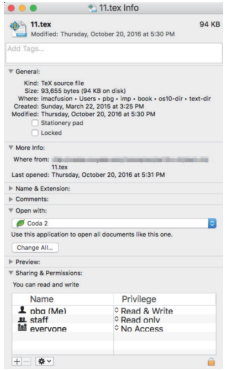


Figura 1: Detalhes de arquivo no MacOS (SILBERSCHATZ; GALVIN; GAGNE, 2001).

```
1        $stat arquivo.pdf
```

Convenções:

- no Windows: flag para ocultar arquivo.
- no Linux: nomear com ponto no início;
- Extensão: 3 últimos caracteres do nome do arquivo são reservados.

```
1        $ls -a
```

- **Criar:** alocar entrada para ele no sistema de arquivos.
- **Abrir:** preparar o SO para usar o arquivo:
  - Verificar se o arquivo existe;
  - Verificar as permissões de acesso;
  - Localizar seu conteúdo no dispositivo;
  - Criar descritores no núcleo e na aplicação.
- **Ler:** transferir dados do arquivo para a memória.
- **Escrever:** transferir dados da memória para o arquivo.
- **Fechar:** liberar as estruturas criadas ao abri-lo.
- **Remover:** eliminar o arquivo do sistema de arquivos.
- **Mudar atributos:** mudar nome, proprietário, etc.

- O arquivo é um sequência bytes;
- O SO não tem responsabilidade de ler todos os tipos de arquivos;
- Nativamente, sabe interpretar executáveis (no Windows .EXE);
- Para outros formatos de arquivos, as aplicações devem definir definir sua estrutura de escrita e leitura.
- O *kernel* não conhece formatos de aruqivo, eles são interpretados em espaço de usuário.

- Documentos: PDF, ODT, DOCS, XLSX;
- Imagens: JPG, PNG, GIF;
- Áudio: MP3, WAV;
- Desenhos: SVG;
- Vídeo: MKV, MP4;
- JSON, XML, HTML, CSV;
- TXT, código fonte.

- Arquivos de Texto:
- Os caracteres são codificados (ASCI, UTF8).

```
$hd arquivo.txt
```

- Arquivos de código fonte;

Arquivo do Tipo ELF

```
1        $file arquivo.txt
```

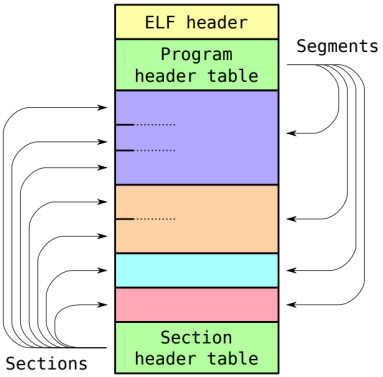


Figura 2: Estrutura ELF (Maziero, 2019).

Magic Numbers

O MacOS usa um atributo para identificar o tipo de arquivo!

- **Documento PDF:** bytes iniciais `%PDF`.
- **Imagem GIF:** bytes iniciais `GIF89a`.
- **Imagem JPEG:** bytes iniciais `0xFF 0xD8 0xFF`.
- **Classe Java:** bytes iniciais `0xCA 0xFE 0xBA 0xBE`.
- **Arquivo ZIP:** bytes iniciais `0x50 0x4B 0x03 0x04`.
- **Arquivo ELF:** bytes iniciais `0x7F 45 4C 46`.

```
1        $hd arquivo.pdf less
2        $hd arquivo.out less
```

- **application/java-archive:** arquivo de classes Java.
- **application/msword:** documento do Microsoft Word.
- **audio/mpeg:** áudio em formato MP3.
- **image/png:** imagem em formato PNG.
- **text/csv:** texto em formato CSV.
- **text/html:** texto em formato HTML.
- **text/plain:** texto puro.
- **text/rtf:** texto em formato RTF (*Rich Text Format*).
- **text/x-csrc:** código-fonte em linguagem C.

Observação

Esses tipos e subtipos MIME seguem a **RFC 2046 (Multipurpose Internet Mail Extensions – MIME)**, utilizada em anexos de e-mail e no protocolo HTTP.

```
1        $ls /dev
2        $cat /proc/crpuinfo
3        $cat /proc/meminfo
4        $cat /proc/<PID>/maps
5        $cat /proc/<PID>
6        $/dev/random
7        $ echo "teste" > /dev/null
```

Interface

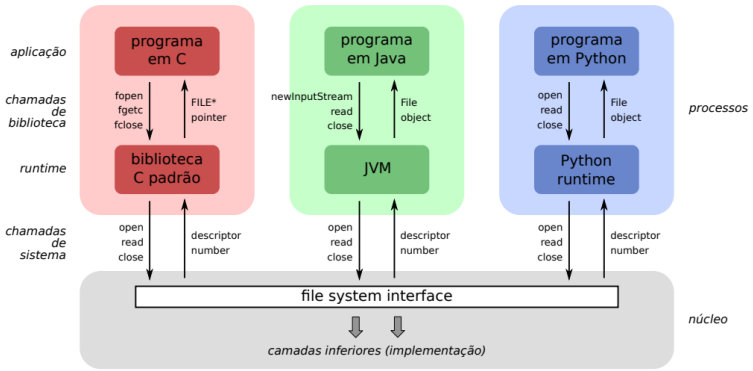


Figura 3: Interface de Acesso (Maziero, 2019).

Funções de Bibliotecas

Operação	C (padrão C99)	Java (classe File)
Abrir arquivo	`fd = fopen(...)`	`obj = File(...)`
Ler dados	`fread(fd, ...)`	`obj.read()`
Escrever dados	`fwrite(fd, ...)`	`obj.write()`
Fechar arquivo	`fclose(fd)`	`obj.close()`
Remover arquivo	`remove(...)`	`obj.delete()`
Criar diretório	`mkdir(...)`	`obj.mkdir()`

Chamadas de Sistema

Operação	Linux	Windows
Abrir arquivo	`OPEN`	`NtOpenFile`
Ler dados	`READ`	`NtReadRequestData`
Escrever dados	`WRITE`	`NtWriteRequestData`
Fechar arquivo	`CLOSE`	`NtClose`
Remover arquivo	`UNLINK`	`NtDeleteFile`
Criar diretório	`MKDIR`	`NtCreateDirectoryObject`

- **Descritor de alto nível:** Provido pela biblioteca ou *runtime*.
  - C: variável dinâmica do tipo `FILE*`;
  - Java: objetos da classe `File`.

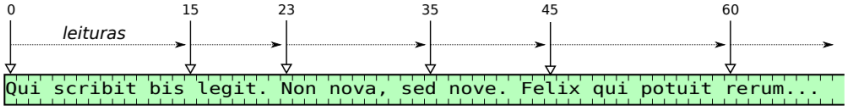


Figura 4: Interface de Acesso (Maziero, 2019).

Acesso Sequencial

- Os dados são lidos ou escritos na ordem em que aparecem no arquivo.
- Usado quando o processamento é feito do início ao fim, sem saltos.
- Exemplo: leitura de um `.txt` linha a linha ou processamento de logs.
- Funções típicas: `fgetc()`, `fgets()`, `fread()`, `fwrite()`.

Acesso Aleatório (Direto)

- Permite mover o ponteiro de leitura/gravação para qualquer posição do arquivo.
- Ideal para atualizar ou consultar registros específicos.
- Usado em bancos de dados, manipulação de mídias e índices.
- Funções típicas: `fseek()`, `ftell()`, `rewind()`.

O acesso sequencial é simples e eficiente para leitura completa, enquanto o aleatório é mais flexível e usado em estruturas de dados persistentes.

- A memória RAM é espelhada em um arquivo no disco.
- Quando há *page fault* o processo acessa a página de forma aleatória.

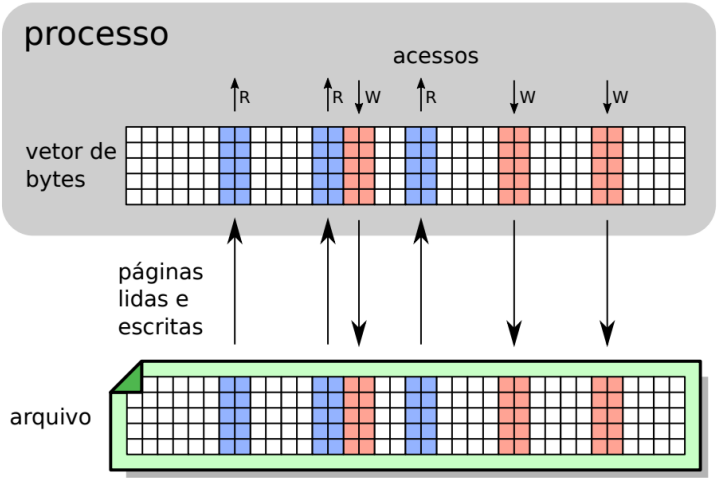


Figura 5: Interface de Acesso (Maziero, 2019).

O controle de acesso é realizado pelo kernel!

- **Usuários:**
  - `user`: o proprietário do arquivo;
  - `group`: grupo de usuários associado ao arquivo;
  - `other`: os demais usuários do sistema.
- **Permissões:**
  - `read` — leitura do arquivo;
  - `write` — escrita (modificação) do arquivo;
  - `execute` — execução (para programas ou scripts).

```
1        $ls -l
```

Gerência

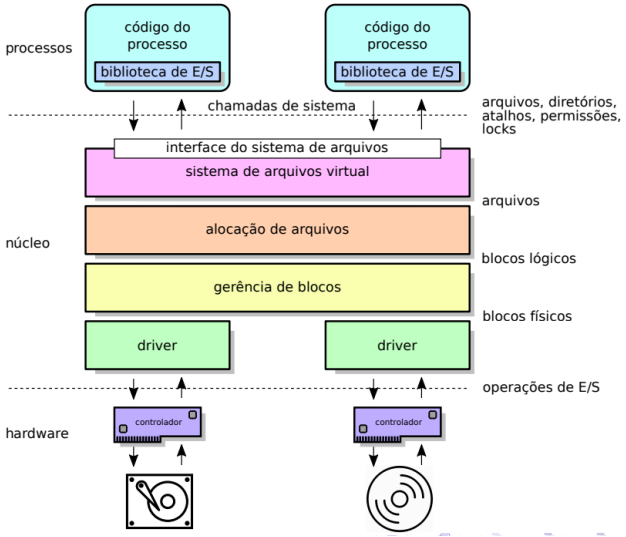

processos
código do processo
biblioteca de E/S
código do processo
biblioteca de E/S
chamadas de sistema
arquivos, diretórios, atalhos, permissões, locks
interface do sistema de arquivos
sistema de arquivos virtual
arquivos
núcleo
alocação de arquivos
blocos lógicos
gerência de blocos
blocos físicos
driver
driver
operações de E/S
hardware
controlador
controlador

Figura 6: Gerência de arquivos (Maziero, 2019).

- **Disco:**
  - Vetor de blocos com tamanho típico de `512 bytes` ou `4096 bytes`;
  - Estruturado em partições;
  - MBR (Master Boot Record): contém a tabela de partições e o código de inicialização.
- **Partição:**
  - Cada uma das áreas independentes do disco;
  - Possui um VBR (Volume Boot Record) no início;
  - É organizada com um filesystem específico (ex.: `ext4`, `NTFS`, `FAT32`).

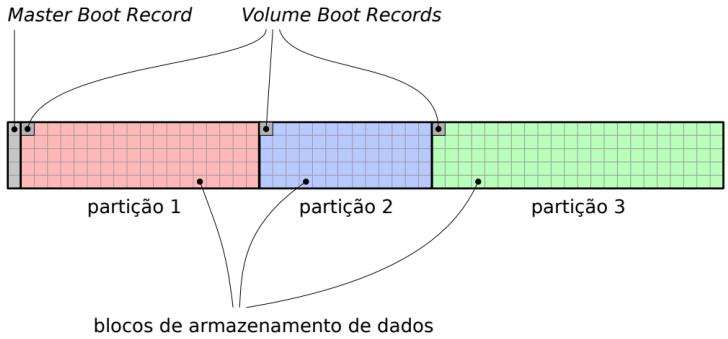


Figura 7: Interface de Acesso (Maziero, 2019).

```
1        # fdisk /dev/<DISCO>
```

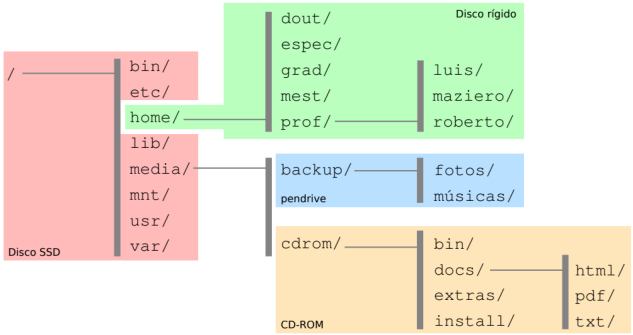


Figura 8: Montagem de volumes (Maziero, 2019).

```
1          $ df
```

Políticas de Cache

Discos usam blocos físicos de 512 bytes ou 4.096 bytes

Read-Through - LEIA E GUARDE!

- **O cache é consultado** a cada operação de leitura;
- Se o bloco **não estiver no cache**, ele é lido diretamente do disco;
- **Blocos lidos** são armazenados no cache para futuras leituras.

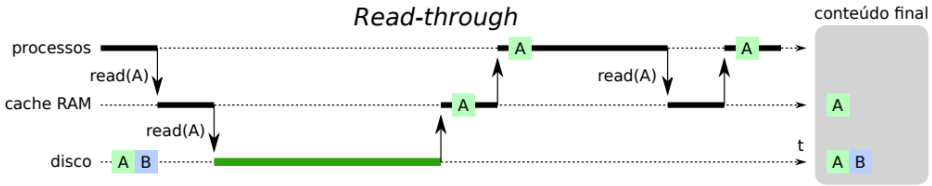


Figura 9: Read-Through (Maziero, 2019).

Melhora o desempenho ao abrir um processo.

Read-Ahead - LEIA ANTES!

- Ao ler um bloco do disco, o sistema **traz mais blocos do que o requerido**;
- **Blocos adicionais** são lidos quando o disco está ocioso;
- Essa estratégia é **benéfica em acessos sequenciais** e quando há **boa localidade** de referência.

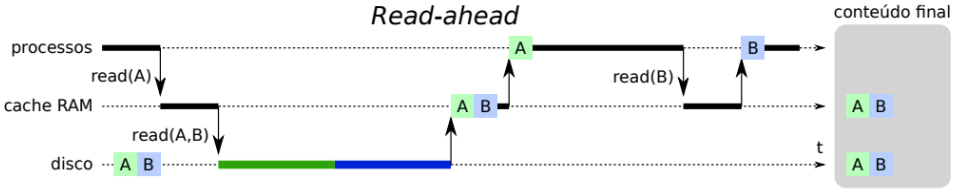


Figura 10: Read-Ahead (Maziero, 2019).

Write-Through - ESCREVA E GUARDE!

- As **escritas são encaminhadas diretamente** ao driver do dispositivo;
- O **processo solicitante é suspenso** até a conclusão da operação;
- Uma **cópia dos dados é mantida em cache** para futuras leituras;
- Essa estratégia é **usada ao escrever metadados** dos arquivos.

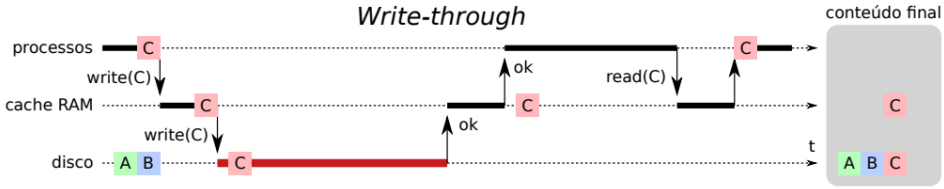


Figura 11: Write-Through (Maziero, 2019).

Write-Back - ESCREVA E GUARDE DEPOIS!

- As **escritas são feitas apenas no cache**;
- O **processo é liberado imediatamente** após a escrita em cache;
- A **gravação efetiva no disco** ocorre posteriormente, de forma assíncrona;
- Essa técnica **melhora o desempenho de escrita**;
- Porém, há **risco de perda de dados** em caso de queda de energia ou falha do sistema.

Arquivos — © 2025 ALVES, M.

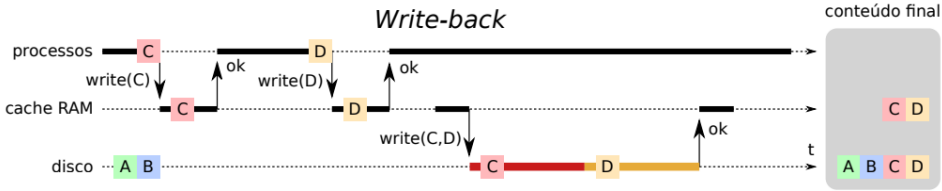


Figura 12: Write-Through (Maziero, 2019).

Quais das políticas?

Para dados, o OS usa *write-back*. Mais rápido!
Para metadados, o OS usa *write- through*. Mais seguro!

Alocação de Arquivos

- Um **arquivo** é definido por:
  - **Conteúdo:** vetor de bytes.
  - **Metadados:**
    - **Atributos:** nome, data(s), permissões, etc.;
    - **Controles:** localização dos dados no disco, entre outros.
- **FCB – File Control Block:**
  - Um descritor para cada arquivo armazenado;
  - Contém os metadados do arquivo;
  - Também deve ser armazenado no disco.
- **Diretório:** tabela composta pelos **FCBs** dos arquivos.

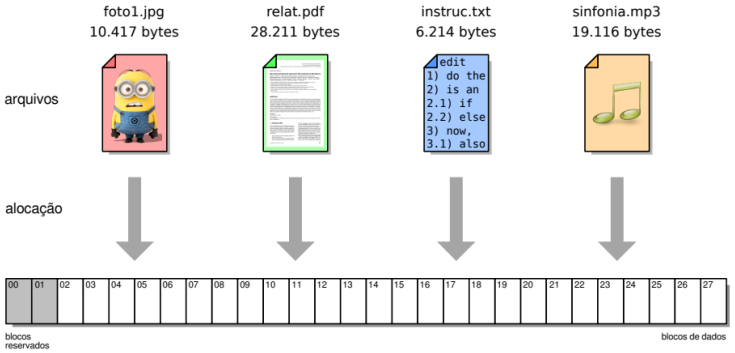


Figura 13: Alocação (Maziero, 2019).

- **Critérios de avaliação:**
  - **Rapidez** na leitura e escrita de arquivos;
  - **Robustez** em relação a erros no disco;
  - **Flexibilidade** na alocação e modificação de arquivos.

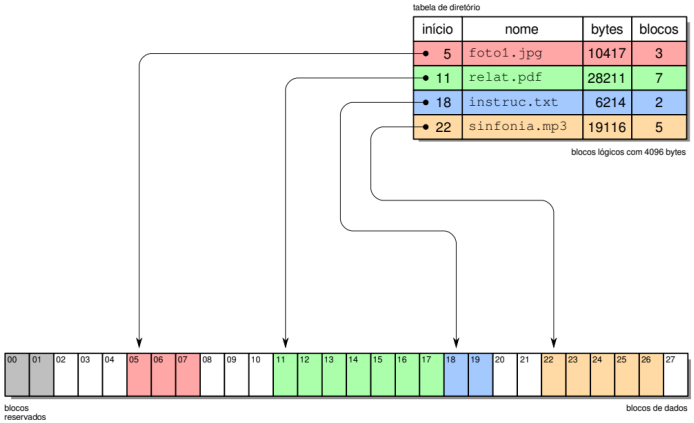


Figura 14: Alocação (Maziero, 2019).

Alocação usada em CDROM

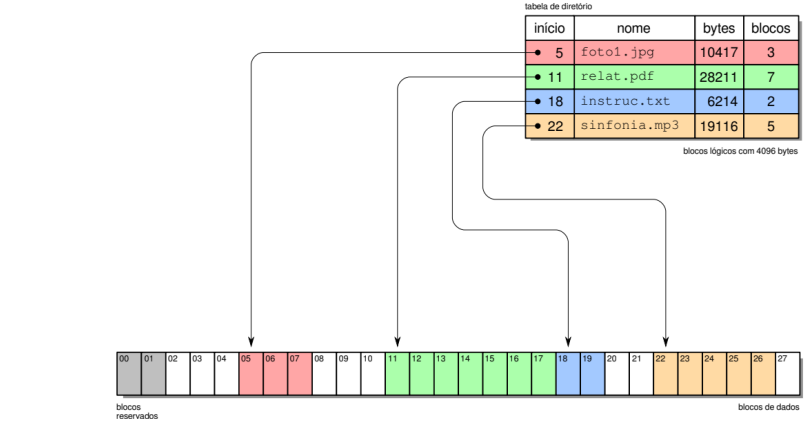


Figura 15: Alocação (Maziero, 2019).

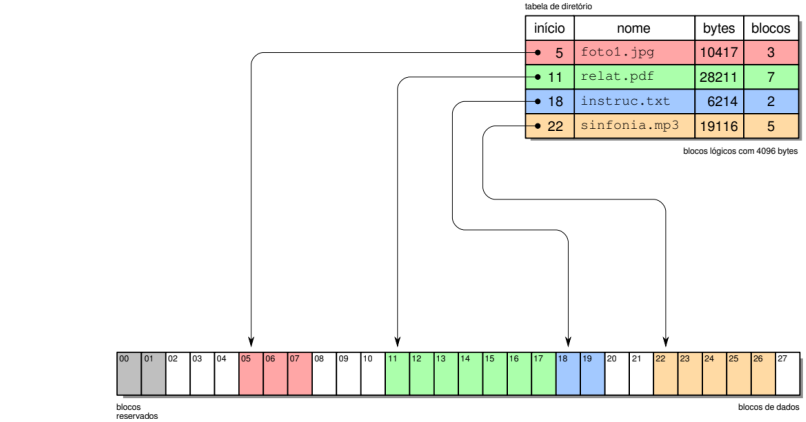


Figura 16: Alocação (Maziero, 2019).

- Acessos sequencial e direto aos dados são rápidos.
- Boa robustez a falhas de disco.
- Baixa flexibilidade.
- Forte risco de fragmentação externa

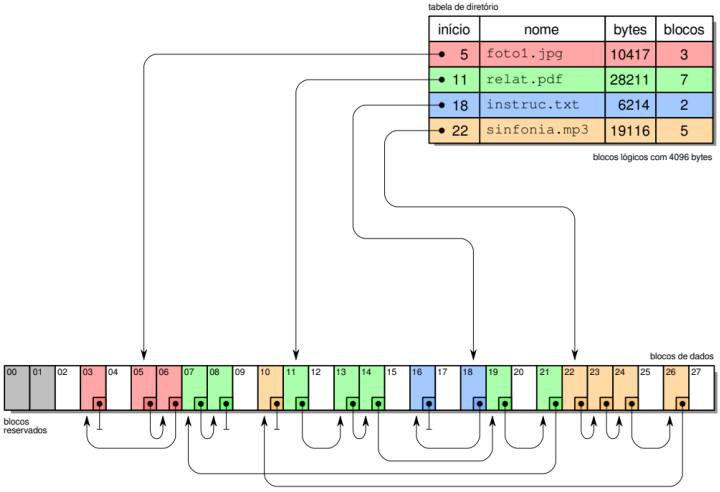


Figura 17: Alocação (Maziero, 2019).

- Mais flexibilidade na criação de arquivos.
- Elimina a fragmentação externa.
- Acesso sequencial é usualmente rápido.
- Acesso direto é lento (percorrer a lista de blocos).
- Pouco robusto: blocos corrompidos (perde a referência (ponteiro) da cadeia)

- FAT - File Allocation Table
- Tabela de ponteiros armazenada nos blocos iniciais
- Mantida em cache na memória RAM

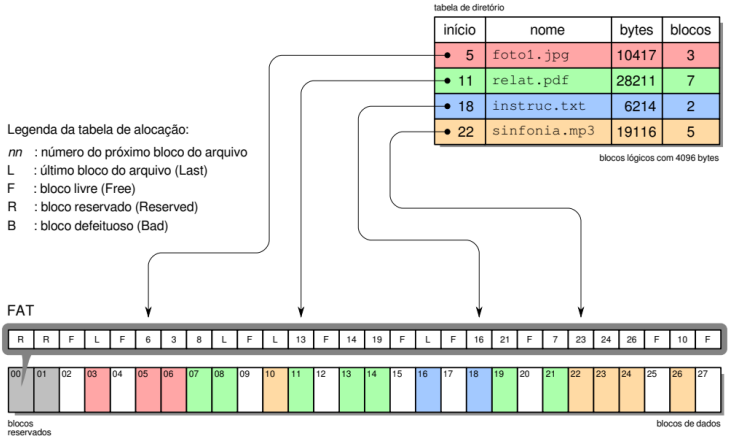


Figura 18: Alocação (Maziero, 2019).

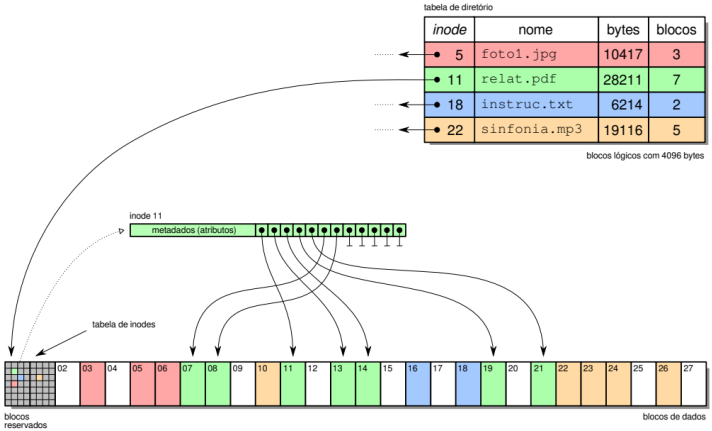


Figura 19: Alocação (Maziero, 2019).

- Index node (inode): estrutura com índice e metadados.
- Struct com tamanho máximo.
- Rápida para acessos sequenciais e diretos.
- Robusta para erros em blocos de dados.
- Flexível.
- Inodes podem ser perdidos (A tabela de Inodes é replicada).

Tamanho do Inode limita o tamanho do arquivo.

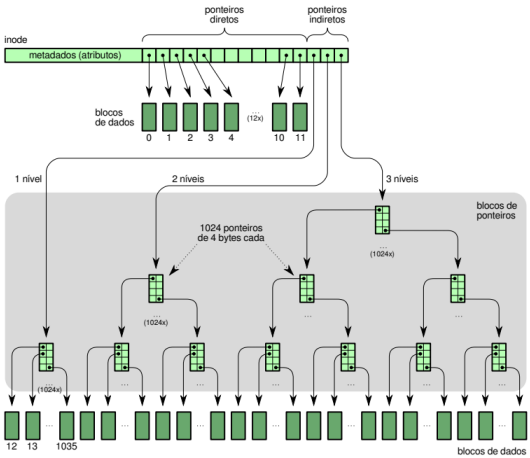


Figura 20: Alocação (Maziero, 2019).

- Alocação indexada é ruim para arquivos muito grandes.
- Muitos metadados;
- Ineficiência e custo de gestão;

Diretórios

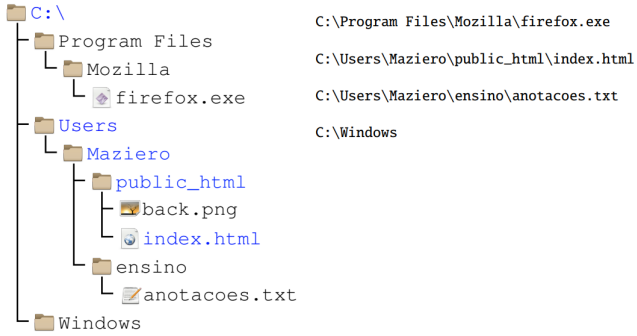


Figura 21: Diretórios Windows (Maziero, 2019).

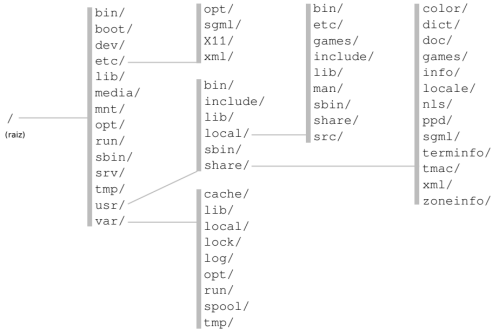
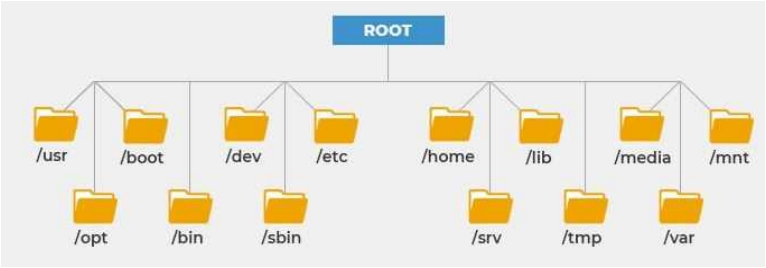
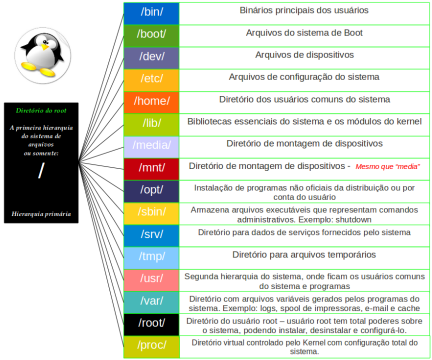


Figura 22: Diretórios Windows (Maziero, 2019).


ROOT
/usr
/boot
/dev
/etc
/home
/lib
/media
/mnt
/opt
/bin
/sbin
/srv
/tmp
/var


Diretório do root
A primeira hierarquia do sistema de arquivos em somente:
/
Hierarquia primária
/bin/
Binários principais dos usuários
/boot/
Arquivos do sistema de Boot
/dev/
Arquivos de dispositivos
/etc/
Arquivos de configuração do sistema
/home/
Diretório dos usuários comuns do sistema
/lib/
Bibliotecas essenciais do sistema e os módulos do kernel
/media/
Diretório de montagem de dispositivos
/mnt/
Diretório de montagem de dispositivos - Mesmo que "media"
/opt/
Instalação de programas não oficiais da distribuição ou por conta do usuário
/sbin/
Armazena arquivos executáveis que representam comandos administrativos. Exemplo: shutdown
/srv/
Diretório para dados de serviços fornecidos pelo sistema
/tmp/
Diretório para arquivos temporários
/usr/
Segunda hierarquia do sistema, onde ficam os usuários comuns do sistema e programas
/var/
Diretório com arquivos variáveis gerados pelos programas do sistema. Exemplo: logs, spool de impressoras, e-mail e cache
/root/
Diretório do usuário root – usuário root tem total poderes sobre o sistema, podendo instalar, desinstalar e configurá-lo.
/proc/
Diretório virtual controlado pelo Kernel com configuração total do sistema.

Referência direta

Somente o nome do arquivo.

```
1 materiais.pdf
2 uma-bela-foto.jpg
```

Referência absoluta

Caminho inicia no diretório raiz.

```
1 \Windows\system32\drivers\etc\hosts.lm
2 /home/maziero/bin/scripts/../../docs/proj1.pdf
```

Referência relativa

Caminho inicia no diretório atual.

```
1 imagens\satelite\brasil\geral.jpg
2 ../../../share/icons/128x128/calculator.svg
```

O diretório não tem arquivo, ele referencia arquivos!

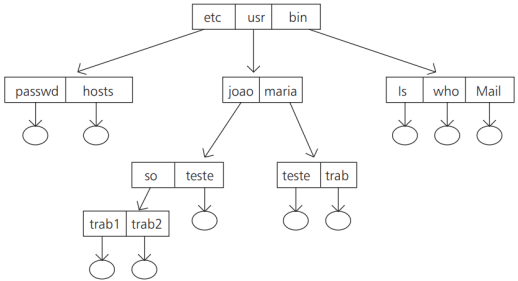


Figura 23: Diretorório organizado em árvore (OLIVEIRA; CARISSIMI; TOSCANI, 2001).

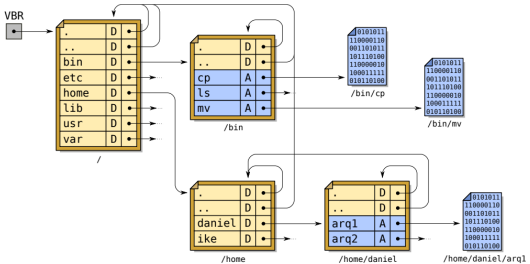


Figura 24: Implementação de Diretórios (Maziero, 2019).

Capítulo 8 (OLIVEIRA; CARISSIMI; TOSCANI, 2001)

Arquivos — © 2025 ALVES, M.

Bibliografia

Bibliografia Básica I

OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. **Sistemas Operacionais**. 2. ed. Porto Alegre: Sagra-Luzzatto, 2001.

STALLINGS, William. **Operating Systems: Internals and Design Principles**. 6. ed. Upper Saddle River, NJ: Prentice-Hall, 2009.

TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.

Bibliografia Complementar I

SILBERSCHATZ, Abraham; GALVIN, Peter; GAGNE, Greg. **Sistemas Operacionais: Conceitos e Aplicações**. Rio de Janeiro: LTC, 2001.

TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas Operacionais: Projeto e Implementação**. 2. ed. Porto Alegre: Bookman, 2000.

MAZIERO, Carlos Alberto. **Sistemas Operacionais: Conceitos e Mecanismos [recurso eletrônico]**. Curitiba: DINF - UFPR, 2019. ISBN 978-85-7335-340-2.

Estes slides estão protegidos por uma licença Creative Commons

Este modelo foi adaptado de Maxime Chupin.

Marisangila Alves, MSc
marisangila.alves@udesc.br
marisangila.com.br

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

UDESC
Universidade do Estado de Santa Catarina

2025/2

Sistemas Operacionais

Gerenciamento de Sistemas de Arquivos