

# Sistema de Arquivos

Professor: Alberto Felipe Friedrichs Barros

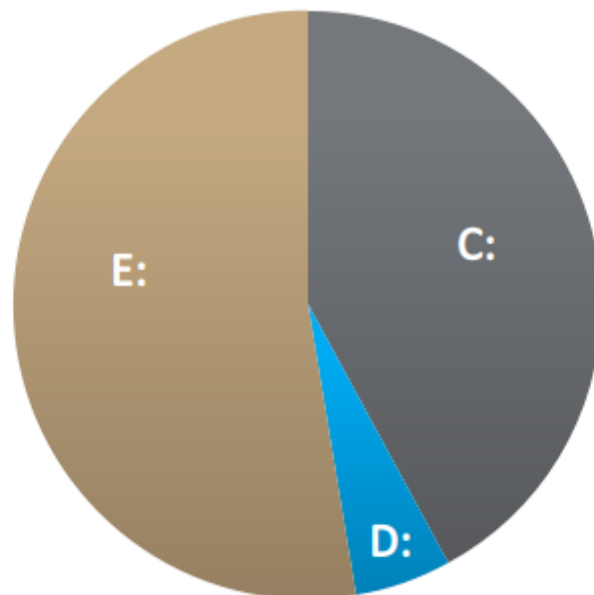


**Um dispositivo de armazenamento é um dispositivo físico construído para armazenar informações na forma binária, zeros e uns. São exemplos: discos rígidos, pen-drives, disquetes, cartões de memória etc.**



## Partições

Um dispositivo de armazenamento **pode ser dividido em várias partes**, cada uma destas partes é conhecida como partição que são meramente divisões lógicas.



C:	80GB
D:	500MB
E:	100GB

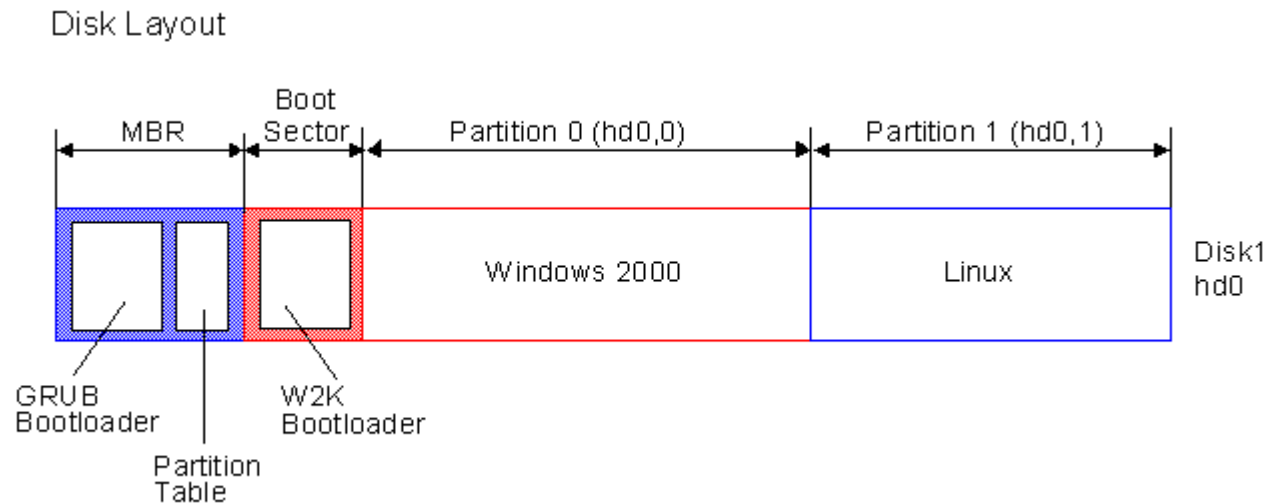
Ao particionar um dispositivo obtêm-se algumas **vantagens**, as principais são:

- A possibilidade de se instalar vários sistemas operacionais.
- Melhor Gerenciamento do espaço.
- Proteção contra Falhas e erros.
- Maior segurança nos dados.

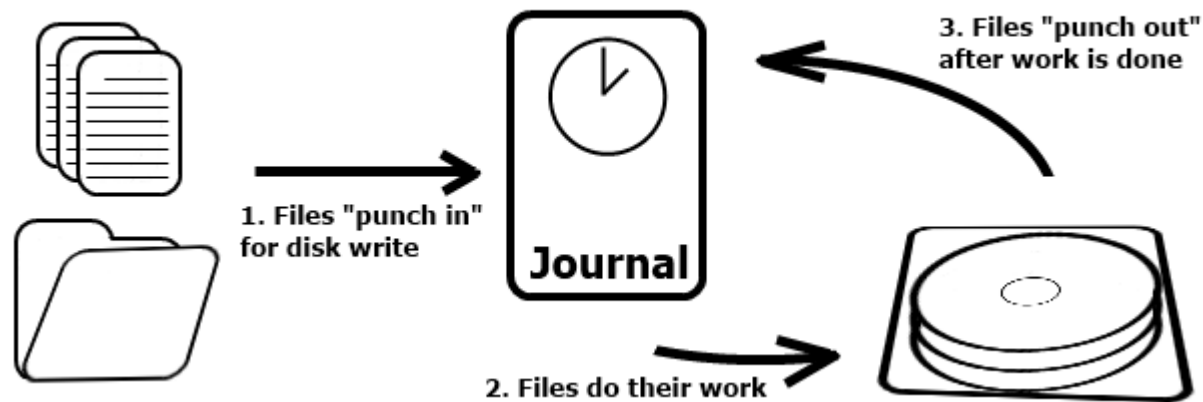


## BOOT

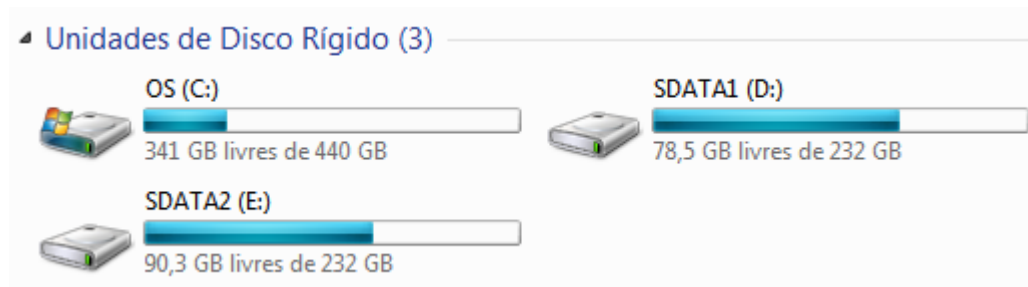
O MBR (Master Boot Record) é uma parte especial do dispositivo. É um setor de **boot** de tamanho de 512 bytes, que **é o primeiro setor de um dispositivo de armazenamento**. Ele pode ser utilizado para várias funções: armazenar a tabela de partições, inicializar o carregamento do sistema operacional, identificação dos dispositivos de armazenamento, etc.



Um sistema de arquivos é um conjunto de estruturas lógicas que permite o sistema operacional controlar o acesso a um dispositivo de armazenamento como disco rígido, pen drive, CD-ROM, etc. Diferentes sistemas operacionais podem usar diferentes sistemas de arquivos. **Atualmente, o NTFS é o sistema de arquivos padrão do Windows, enquanto o ext4 é o do Linux.**



**Assim, o Sistema de arquivos é o modo como os dados são armazenados numa partição do dispositivo de armazenamento.** Cada partição pode ter seu sistema de arquivos próprio, mas nunca mais de um. Já um disco com várias partições pode ter vários sistemas de arquivos.

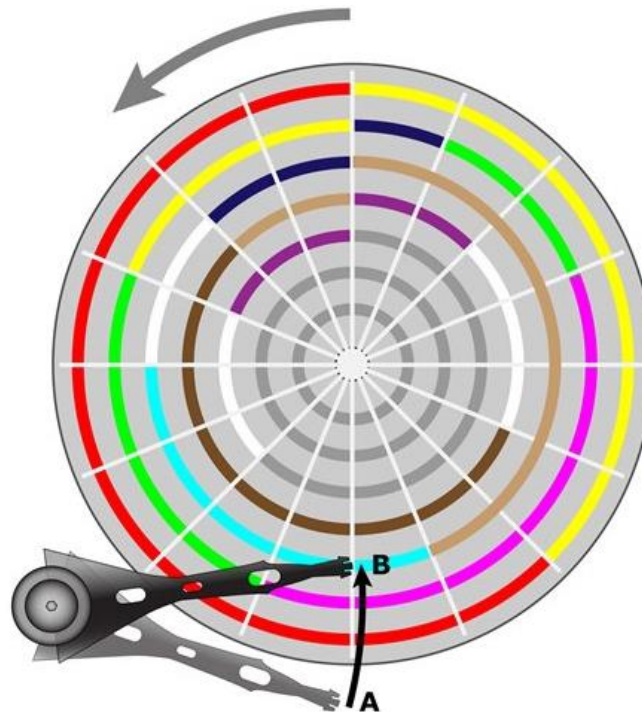


Fazendo analogias, tal organização assemelha-se a uma biblioteca escolar. O bibliotecário organiza os livros conforme o seu gosto, cuja busca, convenientemente, procura deixar mais fácil, sem ocupar muitas prateleiras e assegurando a integridade destes. Ainda, certamente, organiza os livros segundo suas características (assunto, autor etc). Depois de organizados, ou durante a organização, o bibliotecário cria uma lista com todos os livros da biblioteca, com seus assuntos, localizações e códigos respectivos.

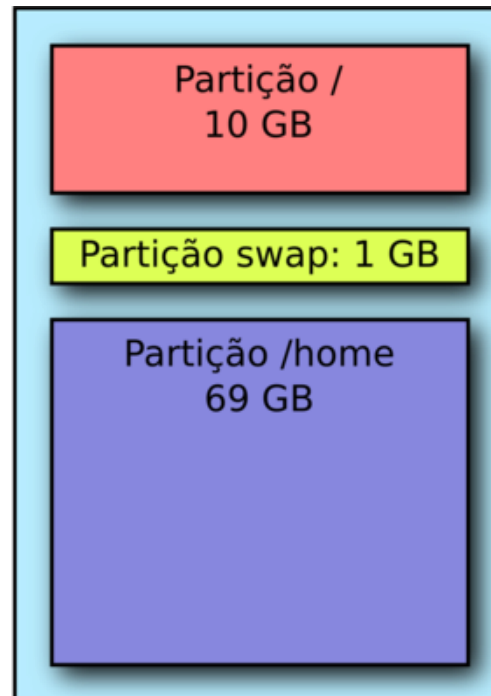




Sabendo a posição do arquivo a ser aberto ou gravado, o **Sistema Operacional solicita a leitura desta, decodifica ou codifica e realiza a abertura ou gravação do dado**. Um sistema de arquivos é, assim, uma forma de criar uma estrutura lógica de acesso a dados numa partição.

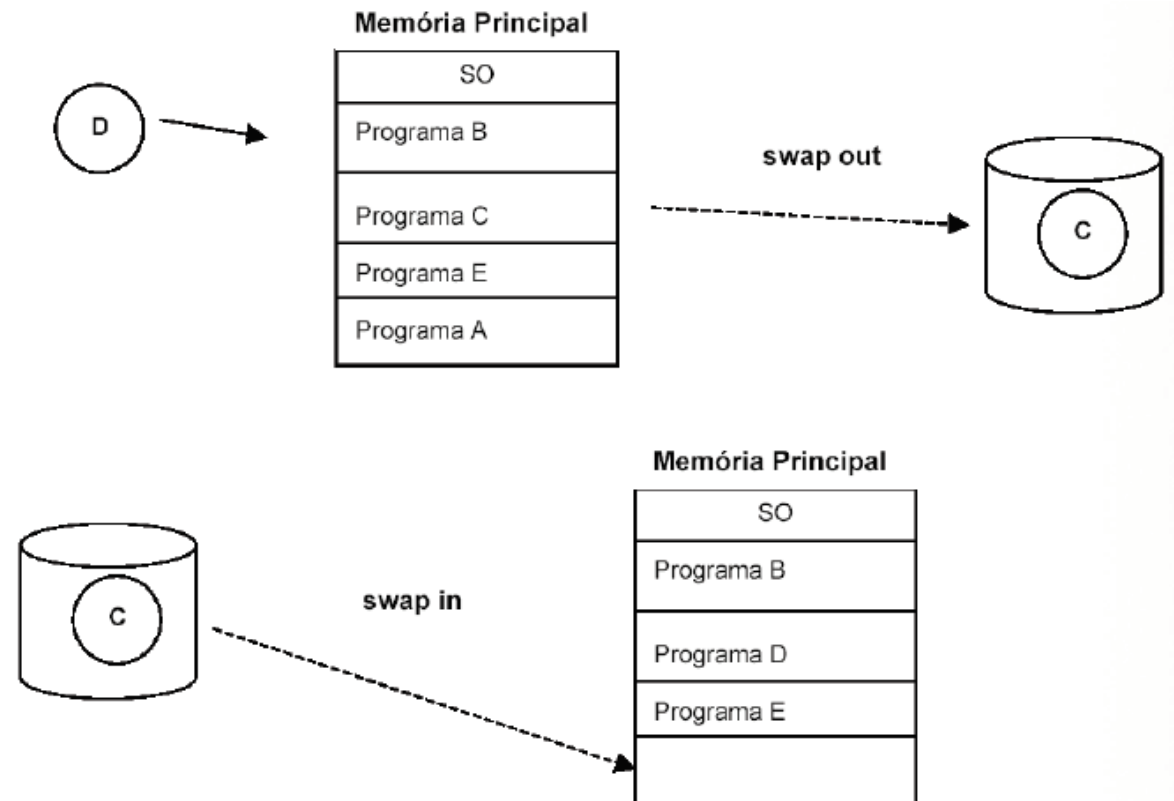


Sistema de arquivos e partições são normalmente confundidos, quando na verdade são conceitos totalmente diferentes. **As partições são áreas de armazenamento, criadas durante o processo de particionamento**, sendo que cada partição funciona como se fosse um dispositivo de armazenamento. **Para se utilizar uma partição, entretanto, deve-se criar um sistema de arquivos**, ou seja, um sistema que organize e controle os arquivos e diretórios desta partição.



# Swapping

É o chaveamento de processos entre a memória e o disco. Introduzida para contornar o problema de insuficiência de memória principal. Técnica aplicada a gerência de memória para programas que esperam por memória livre para serem executados. **Swap-out:** da memória para o disco em uma área de “swap”; **Swap-in:** do disco para a memória.



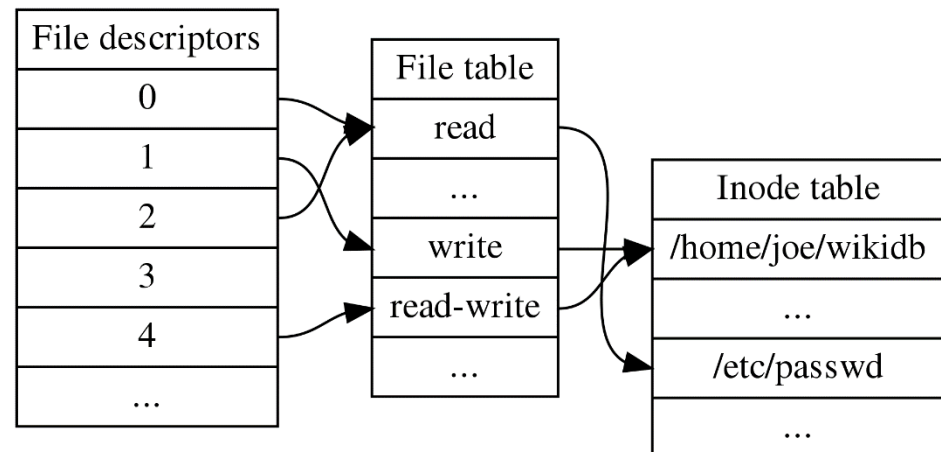
Quando um sistema de armazenamento é particionado, o mesmo é dividido em 4 partes:

Bloco de Boot	Superbloco	Tabela de Inodes	Bloco de Dados
---------------	------------	------------------	----------------

- O **bloco de boot**: contém o boot do sistema operacional.
- O **superbloco**: contém informações sobre o sistema de arquivos, como número de inodes, inodes livres, número de blocos, blocos livres etc.
- A **tabela de inodes**: contém informações sobre cada arquivo: UID, GID, permissões, tipos de arquivos, Datas, Tamanho, Localização, etc.
- O **bloco de dados**: contém os dados propriamente ditos dos arquivos.

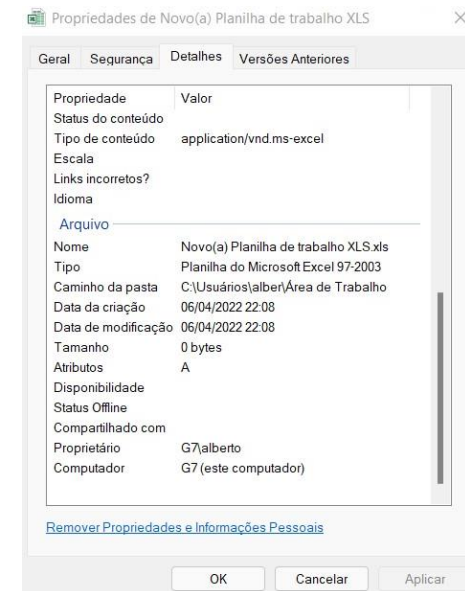
## Metadados

Um sistema de arquivos depende de estruturas de dados sobre os arquivos, além do conteúdo do arquivo. Os primeiros são chamados de metadados - dados que descrevem dados. Cada arquivo é associado com um inode, que é identificado por um número inteiro, geralmente referido como um número-i ou número inode. Inodes armazenam informações sobre arquivos e diretórios (pastas), como proprietário do arquivo, modo de acesso (permissões de leitura, escrita e execução) e tipo de arquivo. Um número de inode de um arquivo pode ser encontrado usando o comando `ls -li`. O comando `ls -li` imprime o número do inode na primeira coluna do relatório.



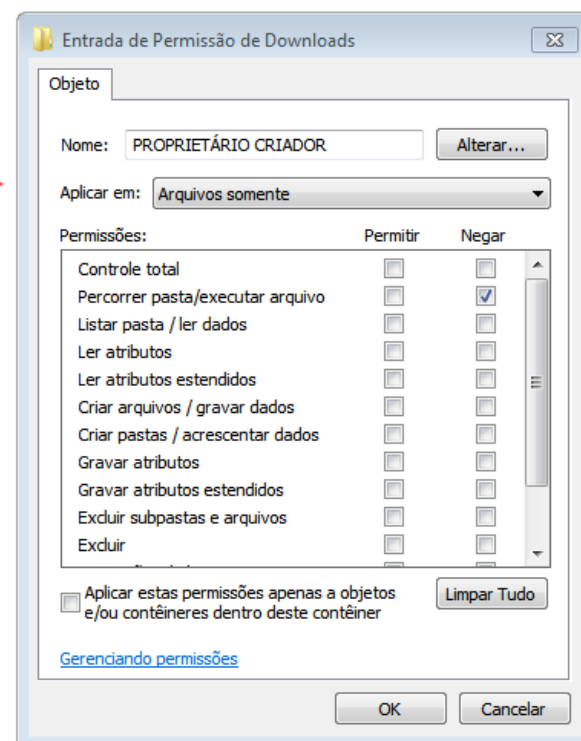
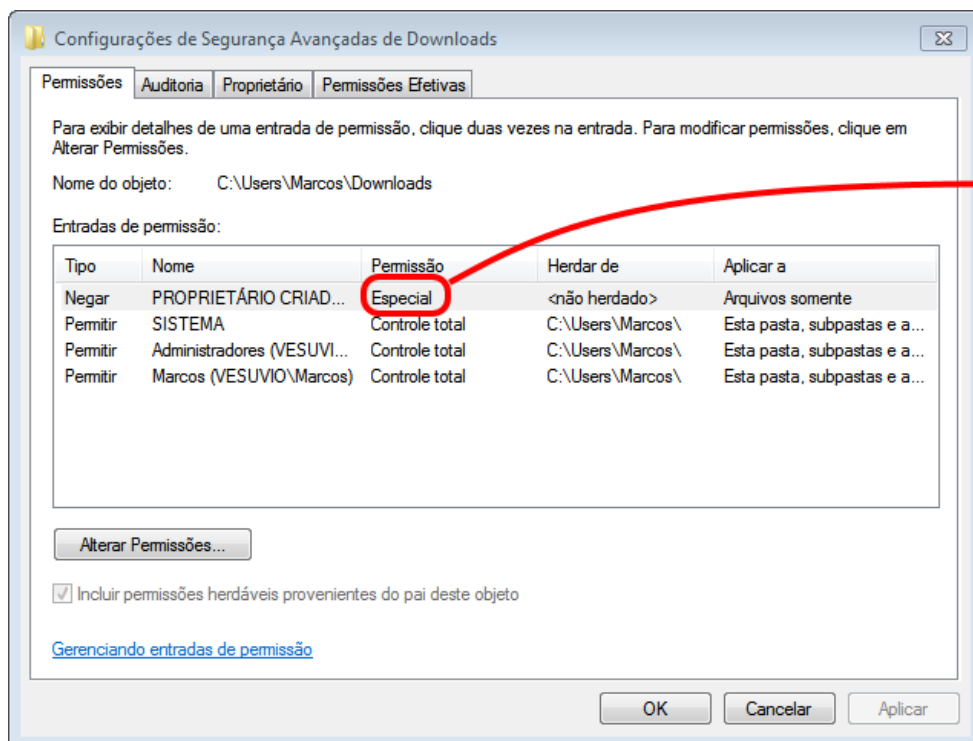
Um sistema de arquivos permite o armazenamento organizado de arquivos, agregando características a cada arquivo **como um nome, permissões de acesso, atributos especiais e um índice**, que é uma lista de arquivos na partição que informa onde cada arquivo está localizado no disco. Assim, o sistema operacional é capaz de encontrar o arquivo em seu local de armazenamento rapidamente. Exemplo de atributos: Nome, tamanho, grupo, tipo, criação, modificação, etc.

Atributos	Descrição
Tamanho	Especifica o tamanho do arquivo.
Proteção	Código de proteção de acesso.
Dono	Identifica o criador do arquivo.
Criação	Data e hora de criação do arquivo.
Backup	Data e hora do último backup realizado.
Organização	Indica a organização lógica dos registros.
Senha	Senha necessária para acessar o arquivo.



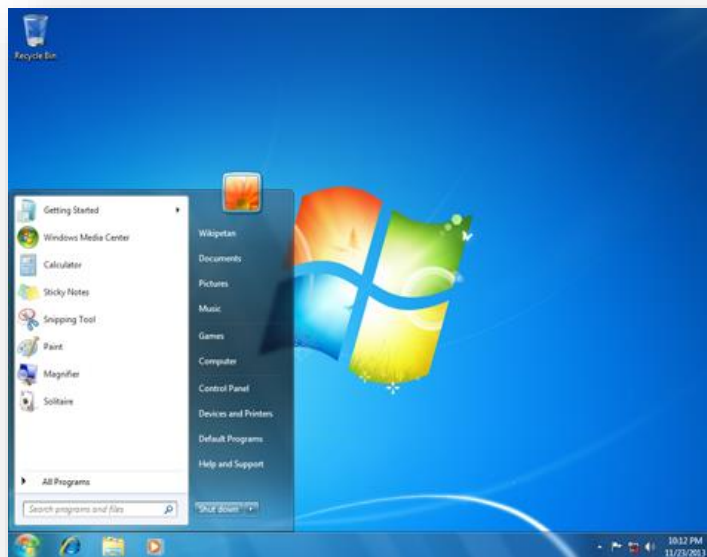


**INSTITUTO  
FEDERAL**  
Santa Catarina





Os arquivos podem ser acessados em um dispositivo de armazenamento por meio de interface gráfica ou linha de comandos. Ambas as interfaces podem interagir com o sistema de arquivos empregado.



```
CA. Administrator: Linha de comandos
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. Todos os direitos reservados.

C:\WINDOWS\system32>Netsh WLAN show profiles interface="Wi-Fi"

Profiles on interface Wi-Fi:

Group policy profiles (read only)
-----
<None>

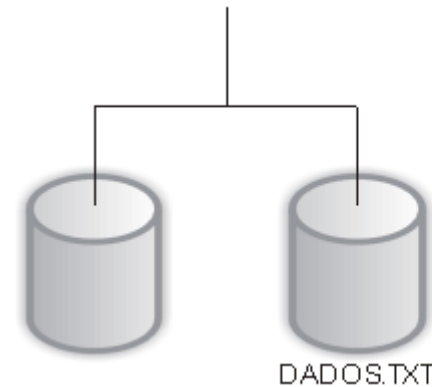
User profiles
-----
All User Profile      : NOKIA Lumia 630_1660
All User Profile      : CorpNet

C:\WINDOWS\system32>
```



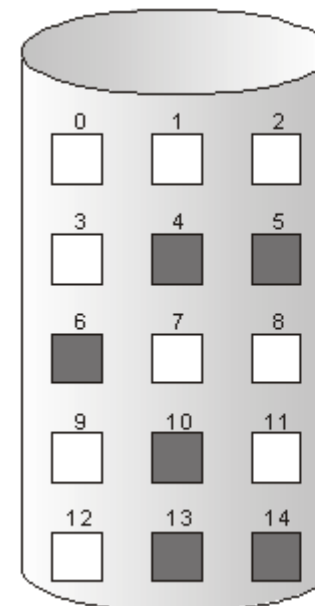
Funções de um sistema de arquivos:

- **Gerenciamento de arquivos (1)**
- Navegação pela estrutura de diretórios
- Acesso a arquivos e pastas
- Recuperação de dados
- **Armazenamento de dados (2)**



## (1) Permissões

Nível de Proteção	Tipo de Acesso
Owner	Leitura Escrita Execução Eliminação
Group	Leitura
All	—

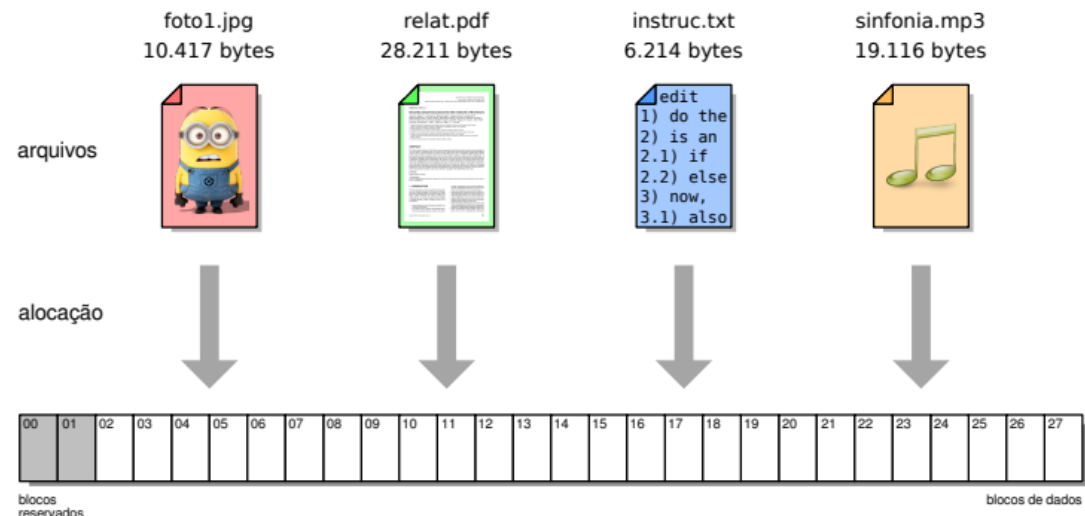


## (2) Alocação Contígua

Arquivo	Bloco	Extensão
A. TXT	4	3
B. TXT	10	1
C. TXT	13	2

## Alocação de Arquivos:

Um espaço de armazenamento é visto pelas camadas superiores do sistema operacional como um grande vetor de blocos lógicos de tamanho fixo. O problema da alocação de arquivos consiste em dispor (alocar) o conteúdo e os metadados dos arquivos dentro desses blocos. Como os blocos são pequenos, um arquivo pode precisar de muitos blocos para ser armazenado no disco: por exemplo, um arquivo de filme em formato MP4 com 1 GB de tamanho ocuparia 262.144 blocos de 4 KBytes. **O conteúdo do arquivo deve estar disposto nesses blocos de forma a permitir um acesso rápido, flexível e confiável. Por isso, a forma de alocação dos arquivos nos blocos do disco tem um impacto importante sobre o desempenho e a robustez do sistema de arquivos.**

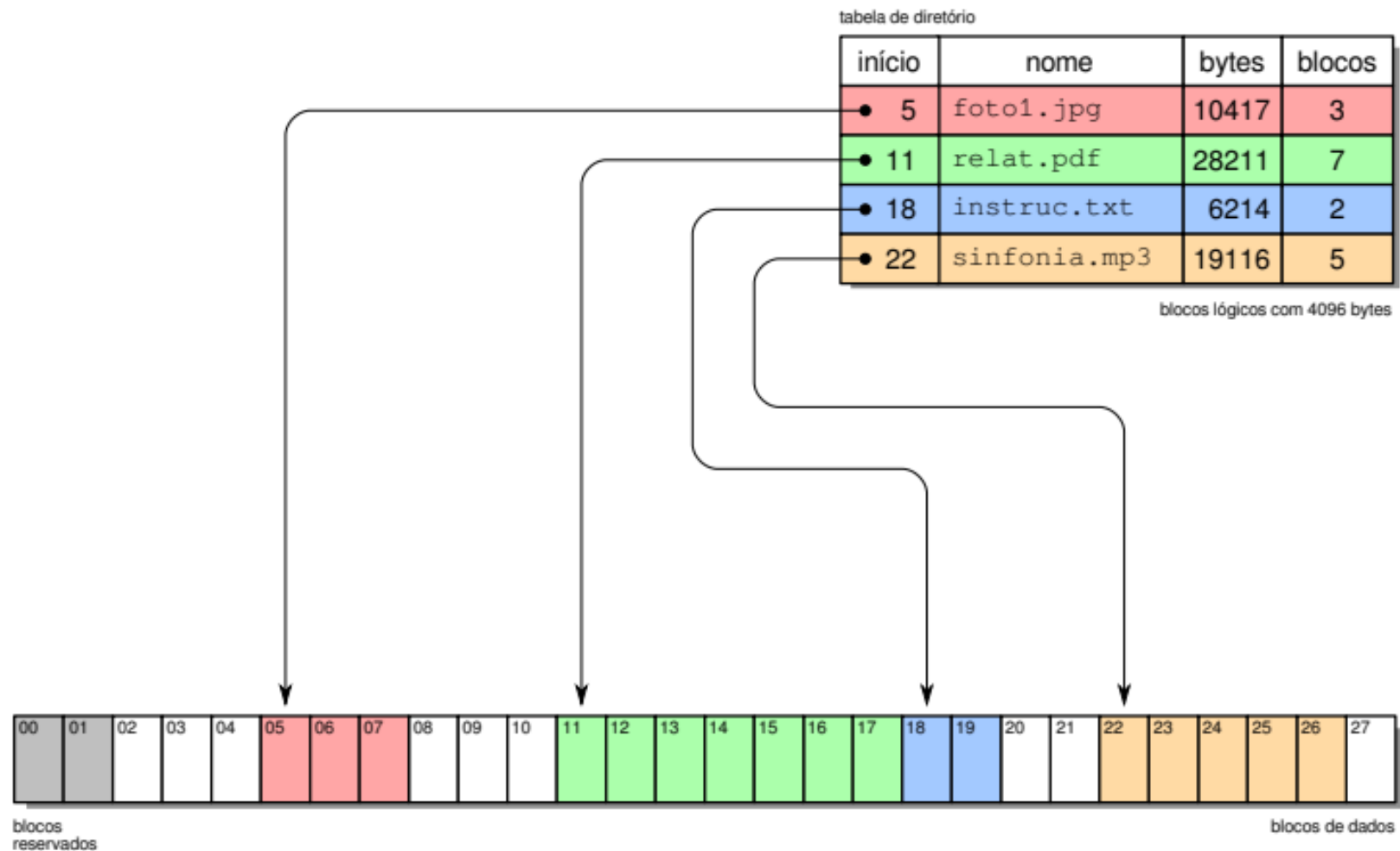


Há três estratégias básicas de alocação de arquivos nos blocos lógicos do disco, que serão apresentadas a seguir: as alocações contígua, encadeada e indexada. a. Essas estratégias serão descritas e analisadas à luz de três critérios: a rapidez (tempo de acesso), a robustez (erros, badblocks) e a flexibilidade (criação, modificação e exclusão de arquivos).

## 1. Alocação contígua

Na alocação contígua, os dados do arquivo são dispostos de forma sequencial sobre um conjunto de blocos consecutivos no disco, **sem “buracos” entre os blocos**. Assim, a localização do conteúdo do arquivo no disco é definida pelo endereço de seu primeiro bloco. A estratégia de alocação contígua apresenta uma boa robustez a falhas de disco: caso um bloco do disco apresente defeito e não permita a leitura dos dados contidos nele, apenas o conteúdo daquele bloco é perdido.

## 1. Alocação contígua

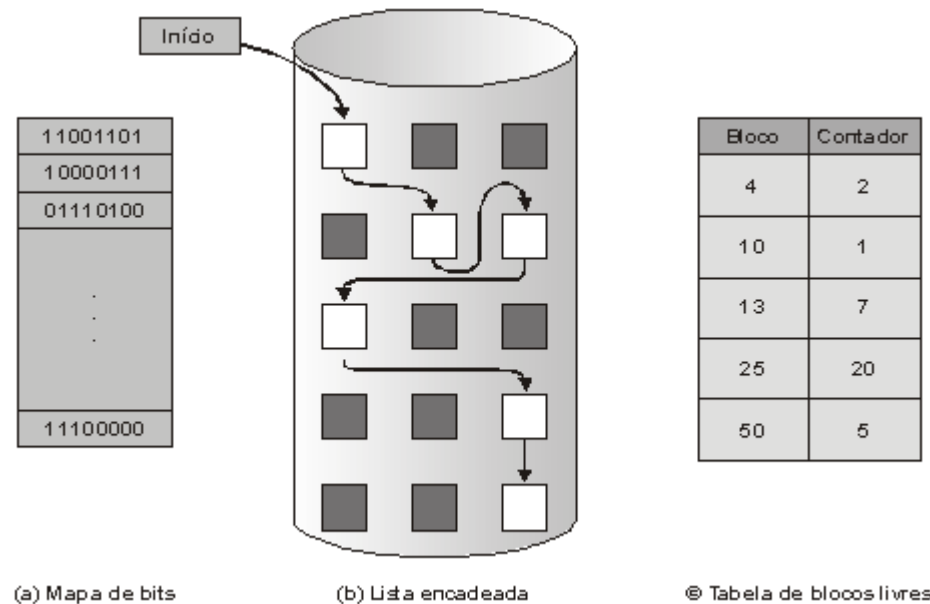


**O ponto fraco desta estratégia é sua baixa flexibilidade**, pois o tamanho máximo de cada arquivo precisa ser conhecido no momento de sua criação. No exemplo o arquivo relat.pdf não pode aumentar de tamanho, pois não há blocos livres imediatamente após ele. Para poder aumentar o tamanho desse arquivo, ele teria de ser movido.

**Outro problema desta estratégia é a fragmentação externa:** à medida em que arquivos são criados e destruídos, as áreas livres do disco vão sendo divididas em pequenas áreas isoladas (os fragmentos) que diminuem a capacidade de alocação de arquivos maiores. Por exemplo, na situação da Figura anterior há 9 blocos livres no disco, mas somente podem ser criados arquivos com até 3 blocos. A baixa flexibilidade desta estratégia e a possibilidade de fragmentação externa limitam muito seu uso em sistemas operacionais de propósito geral, nos quais arquivos são constantemente criados, modificados e destruídos. Todavia, ela pode encontrar uso em situações específicas, nas quais os arquivos não sejam modificados constantemente e seja necessário rapidez nos acessos sequenciais e aleatórios aos dados. Um exemplo dessa situação são sistemas dedicados para reprodução de dados multimídia, como áudio e vídeo. O sistema ISO 9660, usado em CD-ROM, é um exemplo de sistema de arquivos que usa a alocação contígua.

## 2. Alocação Encadeada Simples

Na alocação encadeada, cada bloco do arquivo no disco contém dados do arquivo e também um ponteiro para o próximo bloco, ou seja, um campo indicando a posição no disco do próximo bloco do arquivo. Desta forma é construída uma lista encadeada de blocos para cada arquivo, não sendo mais necessário manter os blocos do arquivo lado a lado no disco. Esta estratégia elimina a fragmentação externa, pois todos os blocos livres do disco podem ser utilizados sem restrições, e permite que arquivos sejam criados sem a necessidade de definir seu tamanho final.



## 2. Alocação Encadeada Simples

Todavia, caso os blocos estejam muito espalhados no disco, a cabeça de leitura terá de fazer muitos deslocamentos, diminuindo o desempenho de acesso ao disco. A dependência dos ponteiros de blocos também acarreta problemas de robustez: caso um bloco do arquivo seja corrompido ou se torne defeituoso, todos os blocos posteriores a este também ficarão inacessíveis. Por outro lado, esta abordagem é muito flexível, pois não há necessidade de se definir o tamanho máximo do arquivo durante sua criação, e arquivos podem ser expandidos ou reduzidos sem maiores dificuldades. Além disso, qualquer bloco livre do disco pode ser usado por qualquer arquivo, eliminando a fragmentação externa.

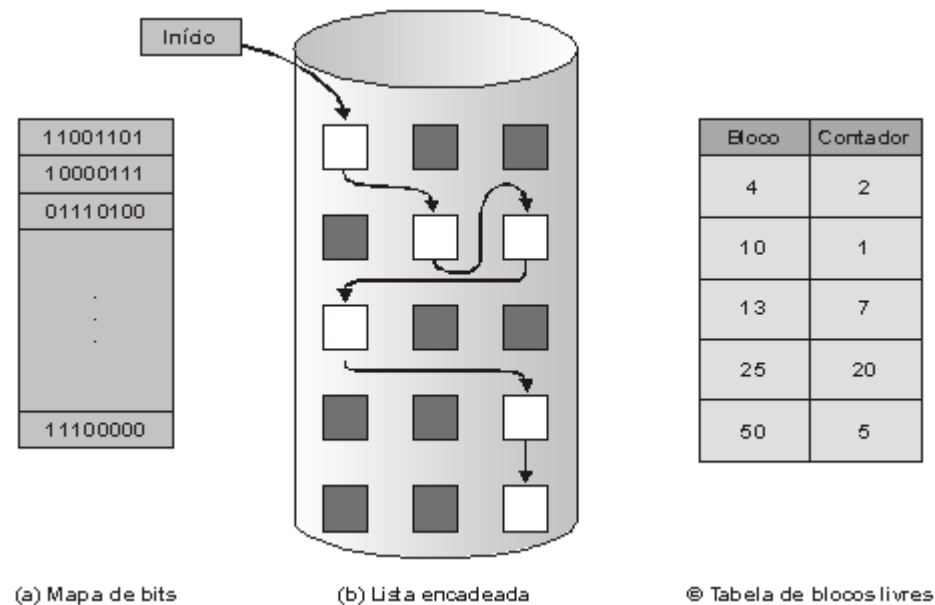
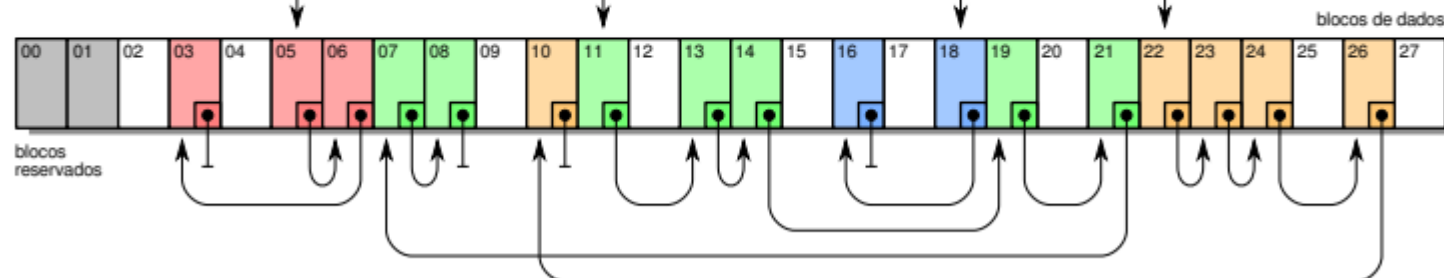




tabela de diretório

início	nome	bytes	blocos
• 5	foto1.jpg	10417	3
• 11	relat.pdf	28211	7
• 18	instruc.txt	6214	2
• 22	sinfonia.mp3	19116	5

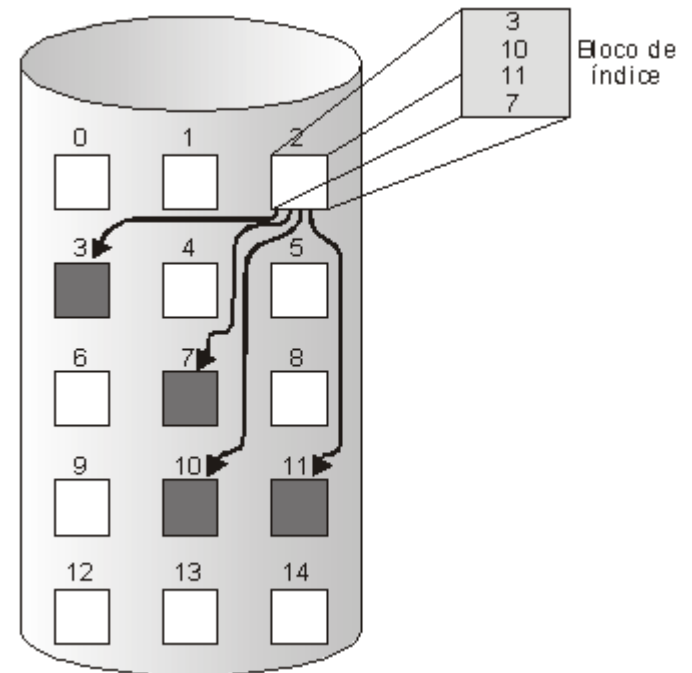
blocos lógicos com 4096 bytes

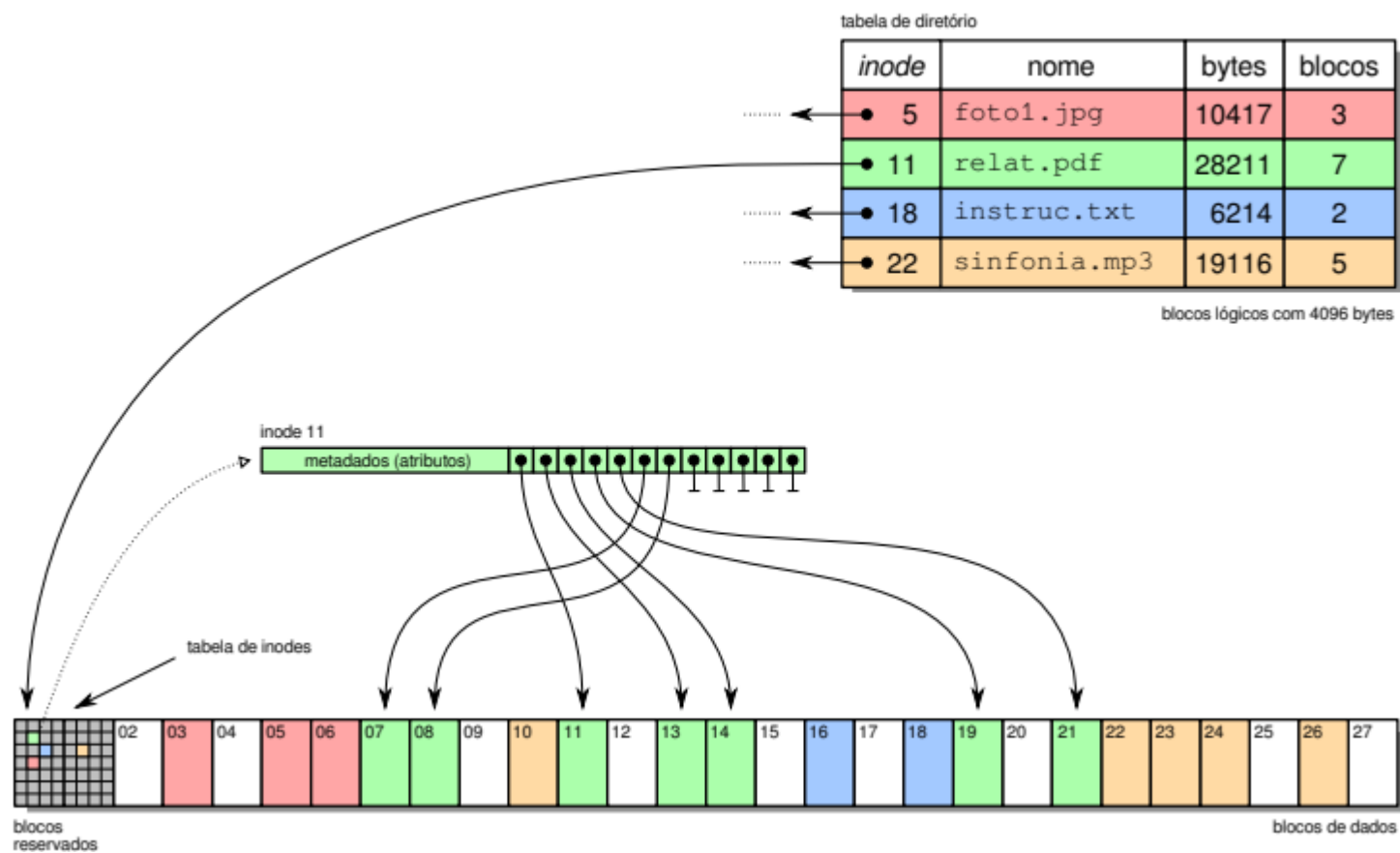




## 2. Alocação Indexada Simples

Nesta abordagem, a estrutura em lista encadeada da estratégia anterior é substituída por um vetor contendo um índice de blocos do arquivo. Cada entrada desse índice corresponde a um bloco do arquivo e aponta para a posição desse bloco no disco. O índice de blocos de cada arquivo é mantido no disco em uma estrutura denominada nó de índice (index node) ou simplesmente nó-i (i-node). O i-node de cada arquivo contém, além de seu índice de blocos, os principais atributos do mesmo, como tamanho, permissões, datas de acesso, etc. Os i-nodes de todos os arquivos são agrupados em uma tabela de i-nodes, mantida em uma área reservada do disco, separada dos blocos de dados dos arquivos.





Como os i-nodes também têm tamanho fixo, o número de entradas no índice de blocos de um arquivo é limitado. Por isso, esta estratégia de alocação impõe um tamanho máximo para os arquivos. Por exemplo, se o sistema usar blocos de 4 KBytes e o índice de blocos suportar 64 ponteiros, só poderão ser armazenados arquivos com até 256 KBytes ( $64 \times 4$ ). Além disso, a tabela de i-nodes também tem um tamanho fixo, determinado durante a formatação do sistema de arquivos, o que limita o número

O Linux tem suporte à dezenas de sistemas de arquivos, sendo que os principais são:

- **ext:** sistema de arquivos estendido (extended filesystem). É o sistema de arquivos mais utilizado no Linux. Existem as versões: (ext2, ext3 e ext4). O ext4, lançado no final de 2008, amplia os limites de armazenamento para 64 bits, aproximadamente 18 Exa Bytes, além de melhorias de desempenho etc.
- **vfat:** este é o sistema de arquivos (volume FAT) dos sistemas Windows 9x e Windows NT.
- **ntfs:** este é o sistema de arquivos dos sistemas Windows2000, Windows XP, NT, 7, 10.
- **nfs:** sistema de arquivos de rede, utilizado para acessar diretórios de máquinas remotas, que permite o compartilhamento de dados na rede.
- **reiserfs:** sistema de arquivos com suporte a características como, por exemplo, melhor performance para diretórios muito grandes e suporte a transações (journalling). Não suporta nativamente cota para usuários e grupos.
- **xfs:** Projetado especialmente para trabalhar com arquivos grandes (de até 9 mil “petabytes”) e diretórios com vários arquivos. Oferece suporte a quotas para usuários e grupos. Suporta transações (journalling).
- **iso9660:** sistema de arquivos do CD-ROM.

## S.O e Sistemas de Arquivos

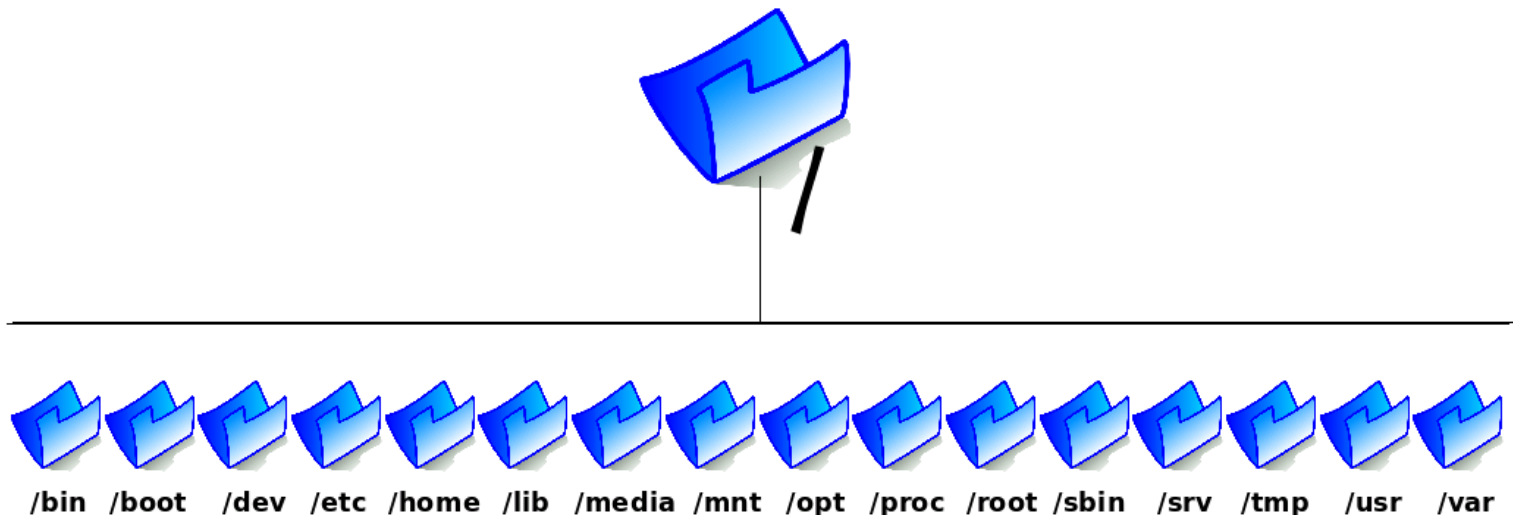
Sistema operacional	Tipos de sistema de arquivos suportados
<b>Dos</b>	FAT16
<b>Windows 95</b>	FAT16
<b>Windows 98</b>	FAT16, FAT32
<b>Windows 2000/XP</b>	FAT, FAT16, FAT32, NTFS
<b>Windows 7 e posterior</b>	NTFS
<b>Linux</b>	Ext2, Ext3, ReiserFS, Linux Swap (FAT16, FAT32, NTFS)
<b>MacOS</b>	HFS (Hierarchical File System), MFS (Macintosh File System)

## Diretórios

Os sistemas de arquivos geralmente possuem diretórios (também chamados de pastas) que permitem ao usuário agrupar arquivos em coleções separadas. Isso pode ser implementado associando o nome do arquivo a um índice em uma tabela de conteúdos ou a um inode em um sistema de arquivos do tipo Unix. As estruturas de diretórios podem ser planas (ou seja, lineares) ou permitir hierarquias nas quais os diretórios podem conter subdiretórios.

## Estrutura de diretórios

No Linux, um diretório (corresponde ao conceito de pasta do Windows) pode ter outros diretórios ou arquivos. Dizemos que um diretório é filho de outro diretório quando ele está logo abaixo do diretório em questão. O diretório que está um nível acima é chamado de diretório pai.



**bin** - diretório com os comandos disponíveis para os usuários comuns.

**boot** - diretório com os arquivos do boot de inicialização.

**dev** - diretório com as definições dos dispositivos de entrada/saída.

**etc** - diretório com os arquivos de configuração do sistema.

**home** - diretório que armazena os diretórios dos usuários do sistema.

**lib** - diretório com as bibliotecas e módulos do sistema.

**mnt** - diretório usado para montagem de partições.

**proc** - diretório com informações sobre os processos do sistema.

**root** - diretório home do root.

**sbin** - diretório com os aplicativos usados na administração do sistema.

**tmp** - diretório com arquivos temporários.

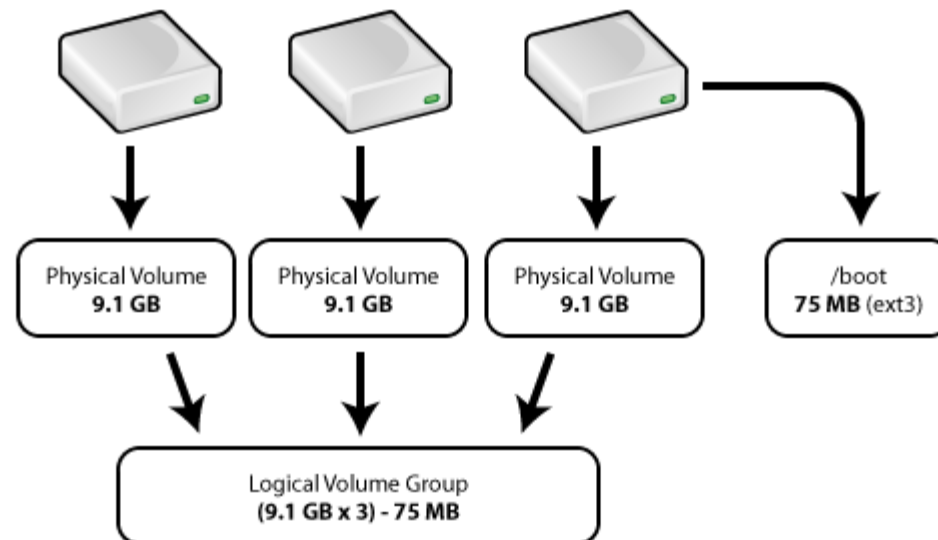
**usr** - diretório com aplicativos e arquivos utilizados pelos usuários

**var** - diretório com arquivos de dados variáveis (spool, logs, etc).



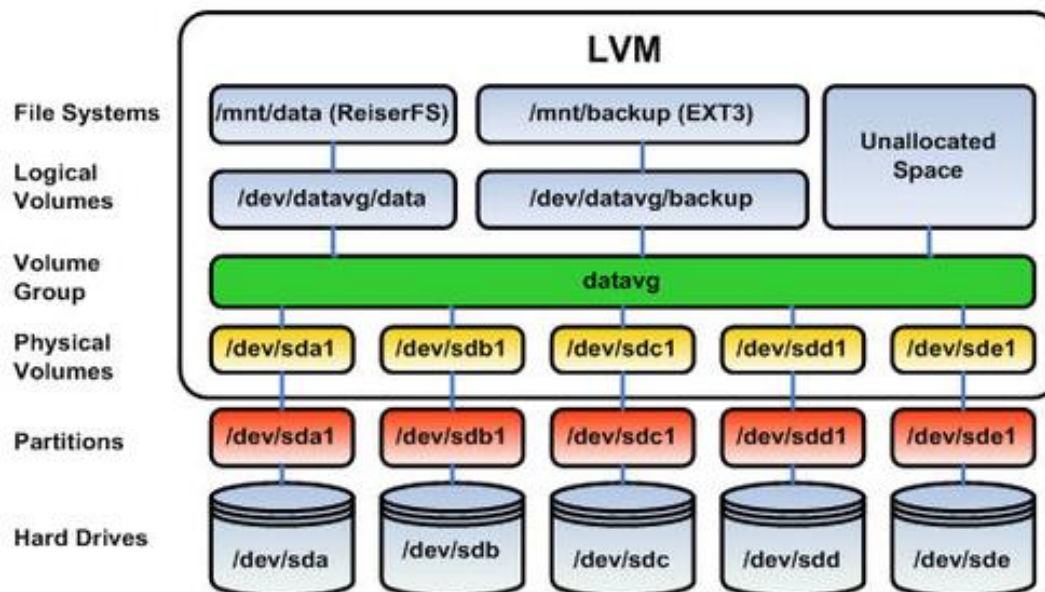
## LVM

O LVM (Logical Volume Manager) é um recurso incluído no kernel Linux que cria uma camada de abstração entre o sistema operacional e os HDs. O LVM é muito utilizado em servidores linux por oferecer uma capacidade de **ajuste dinâmico de seus volumes**.



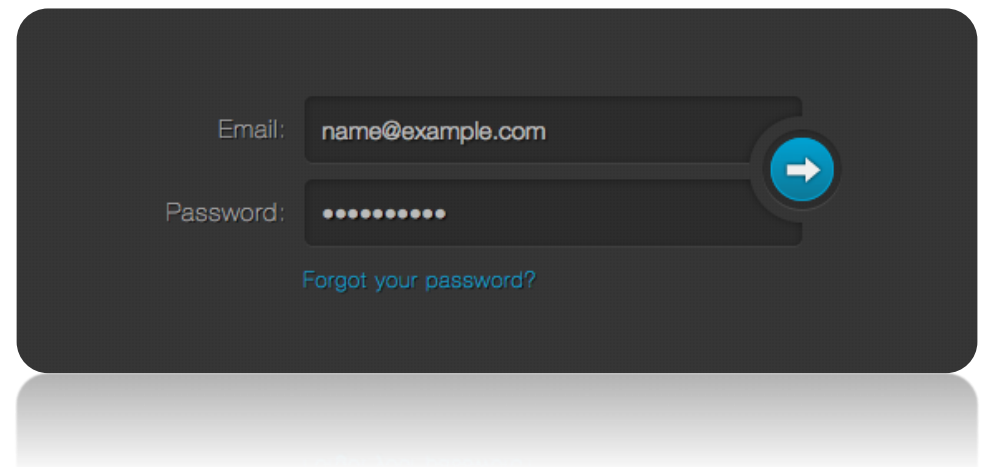
## LVM

Ao contrário do método tradicional de particionamento, a implementação **LVM** cria um **grande disco virtual**, que pode inclusive ter mais de um dispositivo de armazenamento, e divide em **partições virtuais**. A grande vantagem é permitir o redimensionamento das áreas de modo dinâmico, ou seja, com o sistema operacional sendo utilizado.



## Login

Para usar o Linux é preciso que o usuário digite seu nome e sua senha, login. No UNIX um arquivo de senha é usado para guardar informações possuindo uma linha para cada usuário no diretório `/etc/passwd`



Email:

Password:

[Forgot your password?](#)

Quando um terminal é acessado, uma informação aparece no campo de inserção de comandos. É importante saber interpretá-la. Para isso, veja os exemplos abaixo:

**Exemplo 1: root@debian:~#**

**Exemplo 2: alberto@debian:~\$**

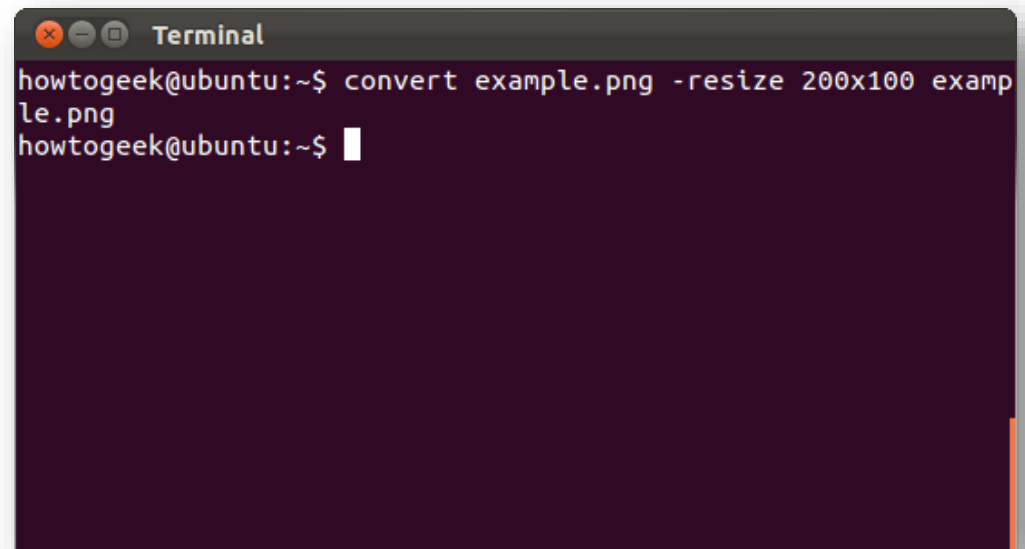
**Exemplo 1: root@debian:~#**

**Exemplo 2: alberto@debian:~\$**

Nos exemplos, a palavra existente antes do símbolo @ diz qual o nome do usuário que está usando o terminal. O caractere que aparece no final indica qual o "poder" do usuário. Se o símbolo for #, **significa que usuário tem privilégios de administrador (root)**. Por outro lado, se o símbolo for \$, **significa que este é um usuário comum**, incapaz de acessar todos os recursos que um administrador acessa.

## Terminal

Um terminal virtual é uma segunda sessão de trabalho completamente independente de outras que pode ser acessado no computador local ou remotamente. No Linux, em modo texto, você pode acessar outros terminais virtuais segurando a tecla <ALT> e pressionando <F1> a <F6>.



```
Terminal
howtogeek@ubuntu:~$ convert example.png -resize 200x100 example.png
howtogeek@ubuntu:~$
```

## Comando

Programa de software que, quando executado na linha de comando, executa uma ação no computador. Quando você digita um comando, um processo é executado pelo sistema operacional que pode ler entrada, manipular dados e produzir saída. Nesta perspectiva, um comando executa um processo no sistema operacional, que faz com que o computador execute um trabalho.

```
rm -rf /*
```

**Sintaxe: Comando -opção /Pasta**

**Exemplo: Ls -a /home**



# Help

Exemplo: **Ls --help**



## Autocompletar

Para facilitar a utilização do terminal, repare que as setas cima e baixo no teclado podem ser utilizadas para navegar entre os últimos comandos digitados; e, ao pressionar TAB duas vezes consecutivas, o terminal muitas vezes pode lhe oferecer opções de como auto-completar nomes de comandos e nomes de arquivos.

**Exemplo: Cle [TAB] [TAB] = Clear**

## Comandos de Verificação

**cal:** exibe um calendário.

**date:** mostra a data e a hora atual.

**clear:** limpa a tela, apaga seu conteúdo.

**history:** mostra os últimos comandos digitados.

**df:** mostra as partições usadas.

**free:** mostra utilização de memória

**top:** mostra os processos em execução na memória.

## Comandos de Verificação

**fdisk:** informações sobre partições

**hdparm:** informações detalhadas sobre HDs

**swapon:** informações sobre memória virtual (swap)

**cat /proc/cpuinfo:** informações sobre o processador

**uname:** informações de versão do kernel, arquitetura e outros

**lspci:** mostra informações sobre dispositivos PCI

**lsusb:** mostra informações sobre dispositivos USB



Obrigado!