

Laboratório de Fundamentos em TIC

Estruturas de Pastas e Programação Shell

Prof. Gabriel Resende Machado



gabrielmachado@unifeso.edu.br



<https://www.linkedin.com/in/machadogabriel>



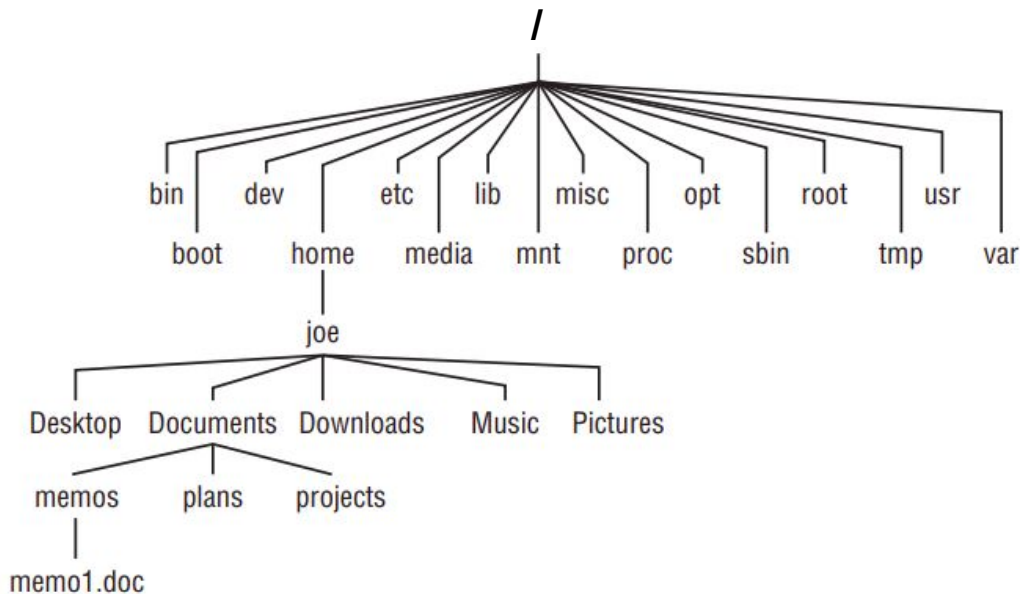
<https://github.com/UNIFESO-Gabriel/fundamentos-em-tic>

No Linux, Tudo é Arquivo

- No Linux, **tudo é considerado um arquivo** uma das filosofias fundamentais do Unix e, por extensão, do Linux. Ela se origina do desejo de manter o sistema simples, coerente e flexível. Benefícios:
 - **Abstração Uniforme:** seja um arquivo de texto, um dispositivo de *hardware*, um processo ou um diretório, todos podem ser manipulados usando as mesmas chamadas de sistema e comandos;
 - **Consistência:** o usuário pode usar comandos como *cat*, *ls*, *cp*, *mv*, *rm*, etc., para manipular não apenas arquivos de texto, mas também dispositivos e processos;
 - **Interoperabilidade:** o usuário pode redirecionar a saída de um programa para outro usando pipes, graças ao fato de que eles tratam a entrada e a saída como arquivos;
 - **Simplicidade:** A interface de programação de aplicativos (API) do sistema é simplificada, pois os desenvolvedores só precisam entender como manipular arquivos. Eles não precisam aprender diferentes conjuntos de comandos para interagir com diferentes.

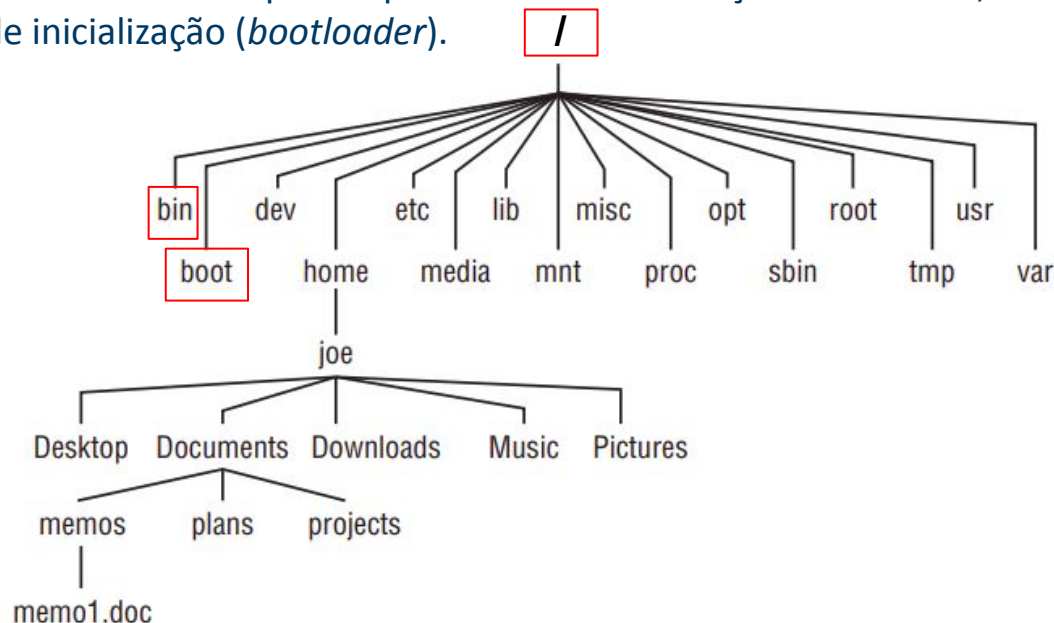
A Organização dos Diretórios no Linux I

- A estrutura de arquivos no Linux é hierárquica e baseada em um sistema de diretórios (também conhecido como pastas);
- A raiz do sistema de arquivos é representada por uma barra invertida (/). A partir daí, todos os outros diretórios e arquivos são organizados em uma estrutura de árvore.
- Essa estrutura de diretórios permite uma organização lógica e eficiente dos arquivos no sistema, facilitando a administração, a manutenção e a segurança do sistema.



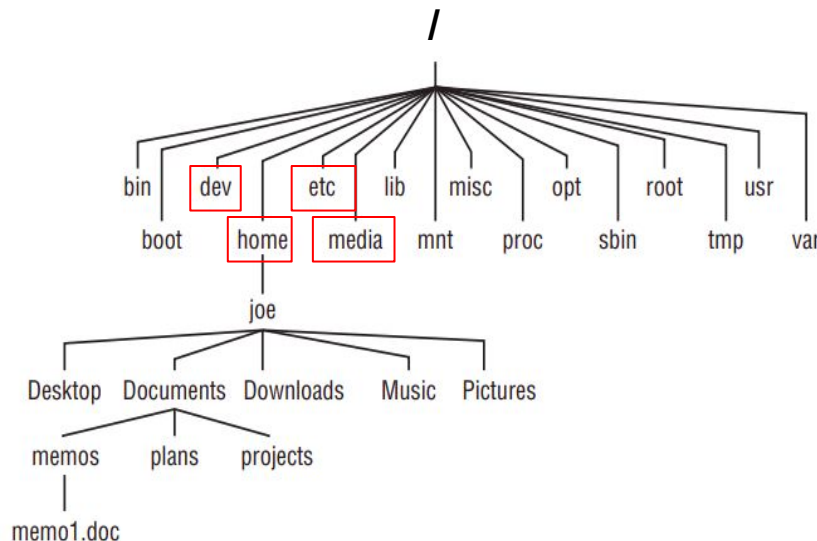
A Organização dos Diretórios no Linux II

- **Diretório Raíz:** é o diretório raiz do sistema de arquivos. Todos os outros diretórios e arquivos são filhos deste diretório;
- **/bin (Binários):** Contém arquivos executáveis essenciais para o sistema operacional, como comandos básicos do Shell;
- **/boot :** Contém arquivos necessários para o processo de inicialização do sistema, como o kernel do Linux e o carregador de inicialização (*bootloader*).



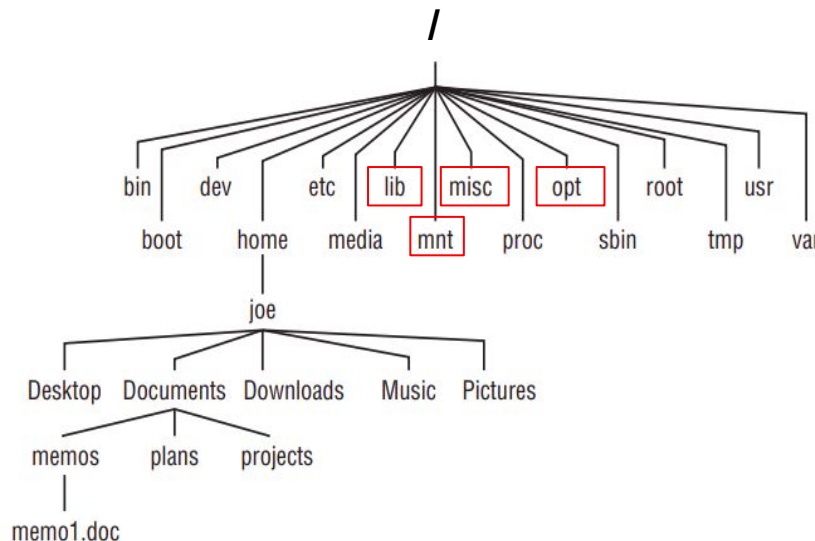
A Organização dos Diretórios no Linux III

- **/dev (devices):** contém arquivos de dispositivo que representam dispositivos físicos e virtuais no sistema. Esses arquivos de dispositivo permitem que os programas e os usuários interajam com os dispositivos físicos e virtuais, como discos rígidos, unidades de CD/DVD, teclados, mouse e outros dispositivos de entrada/saída;
- **/etc (et cetera - configurações):** contém arquivos de configuração do sistema, como configurações de rede, usuários, senhas e serviços;
- **/home (diretórios pessoais):** contém diretórios pessoais para cada usuário do sistema. Cada usuário tem seu próprio diretório dentro de `/home`, onde podem armazenar seus arquivos pessoais;
- **/media (Mídias removíveis):** é um ponto de montagem para dispositivos de mídia removíveis, como unidades USB, discos externos e discos ópticos.



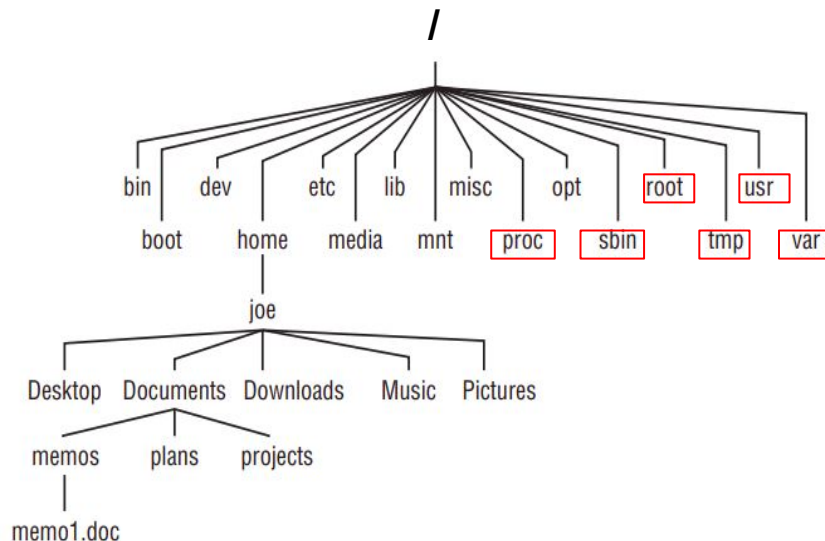
A Organização dos Diretórios no Linux IV

- **/lib (libraries):** Contém arquivos de biblioteca compartilhados pelos programas do sistema;
- **/mnt (mount - pontos de montagem temporários):** é usado para montar sistemas de arquivos temporariamente, como discos externos ou partições não padrão;
- **/opt (optional - software opcional):** contém software adicional instalado pelo usuário, como aplicativos de terceiros, como o Google Chrome;
- **/misc (miscellaneous):** diretório para propósitos diversos. O diretório */misc* no sistema de arquivos do Linux não existe por padrão. Ele não é um diretório reservado ou especialmente significativo no sistema de arquivos do Linux.



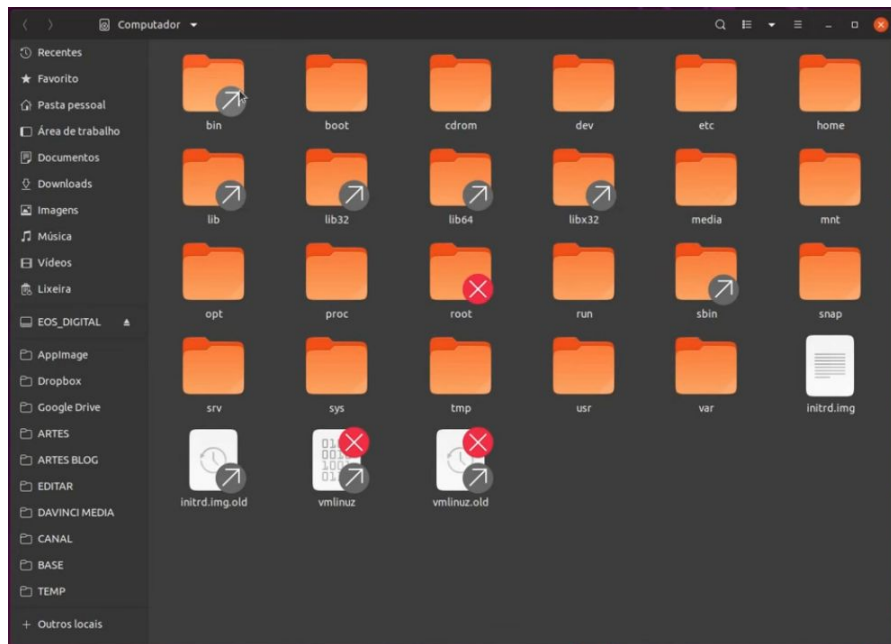
A Organização dos Diretórios no Linux IV

- **/proc (processes):** este diretório contém um sistema de arquivos virtual que fornece informações sobre os processos em execução no sistema;
- **/sbin (system binaries):** este diretório contém arquivos executáveis binários que são usados pelo administrador do sistema;
- **/root (usuário raiz):** este diretório é o diretório inicial do usuário *root*, que é a conta de administrador em uma distro Linux;
- **/usr:** este diretório contém a maioria do software instalável pelo usuário no sistema;
- **/tmp:** este diretório é usado para armazenar arquivos temporários. É uma pasta virtual, *i.e.* armazenada na memória do computador até seu desligamento;
- **/var:** este diretório contém arquivos de dados variáveis, como logs, diretórios de *spool* (armazenam dados temporariamente antes que eles sejam processados ou enviados para outro local) e caches temporários.



Indicadores Gráficos dos Diretórios

- A seta nos diretórios indica que aquele diretório é um **link simbólico** para um outro diretório (uma espécie de atalho). Por exemplo, a seta na pasta `/bin` aponta para o diretório `/usr/bin`;
- O 'X' nos arquivos e diretórios indica que o acesso, leitura e escrita àquele recurso é proibido para o usuário atual do sistema.



Sistema de Permissões

- As permissões no Linux são um recurso de segurança que permite **controlar quem pode acessar, ler, escrever e executar arquivos e diretórios no sistema**.
- As permissões são definidas **para cada arquivo e diretório** e são representadas por um conjunto de *bits* que indicam as permissões de acesso para o proprietário do arquivo ou diretório, o grupo ao qual o arquivo ou diretório pertence e outros usuários no sistema.
- As permissões são divididas em três grupos:
 - **Proprietário (User)**: as permissões para o proprietário do arquivo ou diretório;
 - **Grupo (Group)**: as permissões para os membros do grupo ao qual o arquivo ou diretório pertence;
 - **Outros (Others)**: as permissões para outros usuários no sistema.
- Cada grupo de permissões é representado por três *bits*:
 - **r (read)**: permite que o usuário leia o conteúdo do arquivo ou diretório;
 - **w (write)**: permite que o usuário modifique o conteúdo do arquivo ou diretório;
 - **x (execution)**: permite que o usuário execute o arquivo ou acesse o conteúdo do diretório.

Sistema de Permissões - CHMOD

- **chmod (change mode)** é um comando no Linux que é usado para alterar as **permissões de acesso a arquivos e diretórios**. As permissões de acesso controlam quem pode ler, escrever e executar um arquivo ou diretório;
- Os números de permissão são calculados somando os **valores fixos** das permissões individuais:
 - r (*read*): 4
 - w (*write*): 2
 - x (*execution*): 1
- Por exemplo, para o comando **chmod 755 arquivo.txt**, o número 7 representa as permissões do proprietário (rwx), o número 5 representa as permissões do grupo e outros usuários (r-x).
 - Além disso, as permissões também podem ser definidas usando a notação simbólica, que é mais fácil de entender e usar. Para definir as mesmas permissões do exemplo anterior usando a notação simbólica **chmod u=rwx,g=rx,o=rx arquivo.txt**, onde:
 - Proprietário (*User*): $rwx = 4 + 2 + 1 = 7$
 - Grupo (*Group*): $r-x = 4 + 0 + 1 = 5$
 - Outros (*Others*): $r-x = 4 + 0 + 1 = 5$

Introdução à Programação Shell

- Um *script* Shell é um arquivo de texto que contém uma série de comandos e instruções para serem executados por um Shell, como o Bash (*Bourne Again Shell*) na distro Ubuntu;
- Os *scripts* Shell são usados para automatizar tarefas repetitivas, executar tarefas complexas ou realizar tarefas que seriam demoradas e/ou difíceis de fazer manualmente;
- Esses *scripts* podem conter uma variedade de comandos, incluindo comandos de sistema, comandos de arquivo, comandos de rede, comandos de processo e muito mais. Eles também podem incluir estruturas de controle de fluxo, como *loops* e condicionais, para controlar o fluxo de execução do *script*.

```
#!/bin/bash
# Script criado em 01/01/2020

clear
while : ; do

total=$(top | free '/Men:/ { print $1 }')
usada=$(top | free '/Men:/ { print $2 }')

echo " $total de uso da Men: ${usada}..."
#(Mensagem).....#

echo "Uso da memória RAM"

sleep 10
clear
done
```

Script Bash que monitora o uso da memória RAM do sistema.

Introdução à Programação Shell


Localização do interpretador Shell
a ser usado (bash).



```
#!/bin/bash
```

```
#Isso é um comentário!
```

Comentários são ignorados pelo
interpretador de comandos.



```
echo "Digite o comando do programa"
```

```
mkdir um-novo-diretorio
```


```
touch um_novo_diretorio/um-novo-arquivo
```

```
rm *.txt
```

```
valor=$(ping -w 2 192.168.0.130)
```

```
echo valor
```

Os blocos de
comandos são
executados
sequencialmente.



Introdução à Programação Shell

- Crie um novo diretório dentro de /home, e inclua o arquivo **meu-script.sh**;
- Dê ao arquivo permissão de execução;
- Inclua os seguintes comandos:

```
#!/bin/bash
echo "Olá! Meu nome de usuário é:"
whoami
echo "Eu estou no diretório:"
pwd
echo "Os dados dessa sessão são:"
uptime
```

Execute o arquivo,
escrevendo seu identificador
completo ou apenas o
comando `./meu-script.sh`
a partir do diretório atual.



Introdução à Programação Shell

- Declarando e utilizando variáveis:
 - É possível utilizar variáveis para armazenar valores de comandos.

Comando `read` para entrada de valores do usuário.

```
#!/bin/bash  
  
echo "Digite um nome"  
read nome  
saudacao="Olá! $nome"  
echo "Programa executado com sucesso."  
echo "$saudacao"
```

O símbolo `$` indica que estamos acessando o conteúdo da variável `nome`.

Importante: Não use espaços na atribuição, ou o nome da variável será considerado comando!

```
#!/bin/bash  
  
echo "Arquivos do diretório"  
lista_de_arquivos=$(ls -h)  
echo "$lista_de_arquivos"
```

`$(comando)` indica que estamos considerando a saída do comando

Introdução à Programação Shell

- Declarando e utilizando parâmetros:
 - Parâmetros podem ser passados na inicialização do *script* Shell.

Parâmetros 1 e 2 digitados na inicialização do programa.

```
#!/bin/bash  
  
if [ $1 -lt $2 ]; then  
    echo "O valor " $1 "é menor que o valor" $2  
fi
```

Saída do terminal.


```
$ script 3 5  
O valor 3 é menor que o valor 5
```

Introdução à Programação Shell

- Expressões Lógicas:

Expressão	Significado
string1 = string2	string1 e string2 são idênticas;
string1 != string2	string1 e string2 são diferentes;
a -eq b	a possui o mesmo valor que b;
a -ne b	a não igual a b;
a -gt b	a é maior que b;
a -ge b	a é maior ou igual a b;
a -lt b	a é menor que b;
a -le b	a é menor ou igual a b;

a e b inteiros



Introdução à Programação Shell

- Estruturas de Controle - Decisão:

Utilize o comando **if** para incluir desvios condicionais no código escrito.

o **else** (senão) é
opcional.

```
#!/bin/bash
echo "Digite um número a ser avaliado:"
read numero;
if [ "$numero" -ge 0 ];
then
    echo "O número $numero é positivo!"
else
    echo "O número $numero é negativo!"
fi
```

IMPORTANTE! sempre deixe espaços ao redor dos caracteres [e]. Isso é necessário porque os mesmos são um atalho para o comando *test* do Linux.

- Não se esqueça de fechar o **if** com o comando **fi**.
- O comando **if** deve ser complementado com a condição **then** (então).

Introdução à Programação Shell

- Estruturas de Controle - Repetição:

É possível criar loop condicionais para avaliar uma condição.

do e done
indicam o início
e o final do
bloco de
comandos.

```
#!/bin/bash
echo "Exemplo de um laço utilizando for"
for i in {0..10};
do
    echo "$i"
done
```

Os símbolos { e } indicam um
conjunto de itens.

O operador .. (dois pontos) indica
uma sequência do valor inicial ao
final.

Introdução à Programação Shell

- Estruturas de Controle - Repetição:

É possível criar loop condicionais para avaliar uma condição.

O comando **while** indica que o bloco de comandos se repete enquanto a condição for verdadeira.

```
#!/bin/bash
echo "Informe uma opção, ou digite -1 para sair"
read opcao;
while [ $opcao != "-1" ];
do
    echo "A opção digitada foi $opcao"
    echo "Digite uma nova opção!"
    read opcao;
done
```

Introdução à Programação Shell - Exercício I

1. Escreva um *script* que pergunte que horas são. Dependendo da hora digitada pelo usuário imprima a mensagem "Bom dia", "Boa tarde" ou "Boa noite". Faça a leitura da hora utilizando o comando "read". Considere que o usuário deverá digitar um valor entre 0 e 23.
2. Escreva o mesmo *script* do exercício anterior, porém recebendo a hora digitada como um parâmetro para o programa.
3. O comando **date +%H** imprime a hora atual do sistema. Utilize este comando para alterar o exercício anterior e obter a hora automaticamente, isto é, sem entrada do usuário.

Introdução à Programação Shell - Exercício II

O seguinte script em shell opera sobre arquivos terminados em “.bin” no diretório onde é executado, porém não funciona em arquivos com espaços no título:

```
#!/bin/bash

for nome_de_arquivo in *.bin
do
    nome_sem_extensao=$(basename $nome_de_arquivo .bin)
    mv $nome_de_arquivo $nome_sem_extensao.pdf
done
```

- Pesquise sobre o comando **basename**. O que esse comando faz?
- O que o *script* faz quando é executado?
- Altere o script para que ao invés de “.bin” e “.pdf”, receba como parâmetro duas strings fornecidas pelo usuário, funcionando assim para qualquer tipo de arquivo;
- Altere o *script* para que possa ser executado em arquivos com espaços no título.

Laboratório de Fundamentos em TIC

Estruturas de Pastas e Programação Shell

Prof. Gabriel Resende Machado



gabrielmachado@unifeso.edu.br



<https://www.linkedin.com/in/machadogabriel>



<https://github.com/UNIFESO-Gabriel/fundamentos-em-tic>