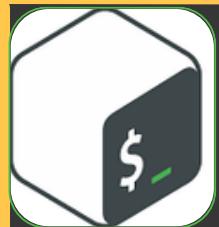


Linha de Comando e Navegação Básica no Terminal Bash





O Terminal Bash (Bourne Again Shell)

O Terminal **Bash** (Bourne Again Shell) é um dos interpretadores de linha de comando mais utilizados no **Linux**. Ele permite que os usuários interajam com o sistema operacional por meio de comandos textuais, facilitando tarefas como navegação de diretórios, manipulação de arquivos e execução de programas.

Abertura do Terminal

No Linux, o terminal pode ser aberto por meio de combinações de teclas como:

Ctrl + Alt + T



Comandos Básicos de Navegação

A navegação dentro do sistema de arquivos é uma das primeiras habilidades que um usuário precisa dominar ao utilizar o **Bash**.

Aqui estão alguns comandos essenciais:

- **pwd** (Print Working Directory) - Exibe o diretório atual.
- **ls** (List) - Lista arquivos e diretórios do local atual.
- **cd [diretório]** (Change Directory) - Muda para outro diretório.
 - Exemplo: cd /home/user/Documents leva ao diretório "Documents" do usuário.
 - cd .. sobe um nível na hierarquia de diretórios.
 - cd ~ retorna ao diretório principal do usuário.
- **clear** - Limpa a tela do terminal para facilitar a leitura.



Manipulação de Arquivos e Diretórios

A linha de comando também permite criar, mover e excluir **arquivos e diretórios** com rapidez:

- **touch** [nome_do_arquivo] - Cria um novo arquivo vazio.
- **mkdir** [nome_do_diretório] - Cria um novo diretório.
- **rm** [nome_do_arquivo] - Remove um arquivo.
- **rm -r** [nome_do_diretório] - Remove um diretório e seu conteúdo.
- **mv** [arquivo_origem] [destino] - Move ou renomeia arquivos e diretórios.
- **cp** [arquivo_origem] [destino] - Copia arquivos para outro local.



Manipulação de Arquivos e Diretórios

Exemplo:

cria um diretório chamado "projeto" no diretório atual.

mkdir projeto

Para criar múltiplos diretórios ao mesmo tempo:

mkdir dir1 dir2 dir3

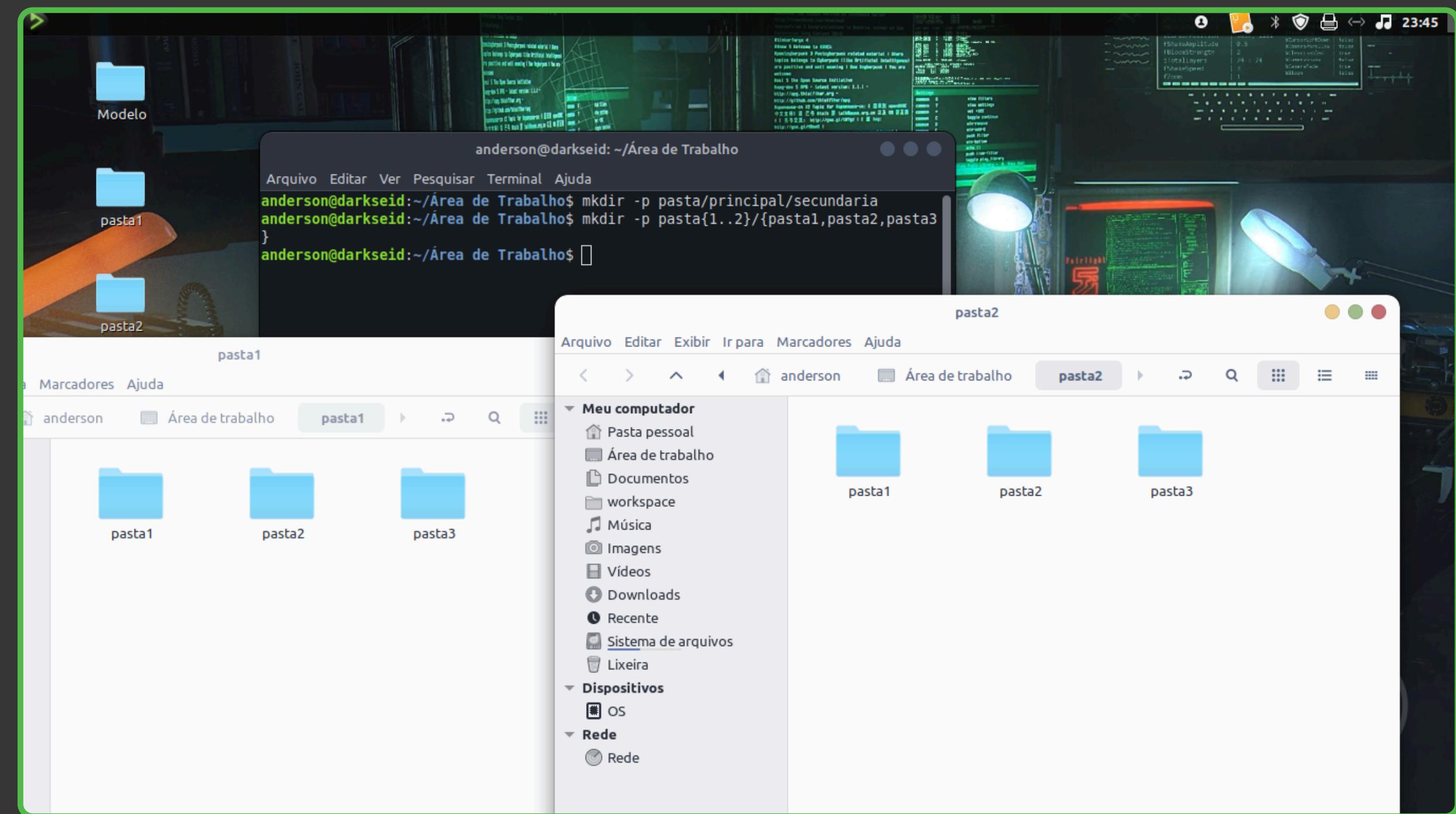


Manipulação de Arquivos e Diretórios

Exemplo:

Criando múltiplos diretórios aninhados com padrões numéricos e nomes diferentes:

mkdir -p pasta{1..2}/{pasta1,pasta2,pasta3}

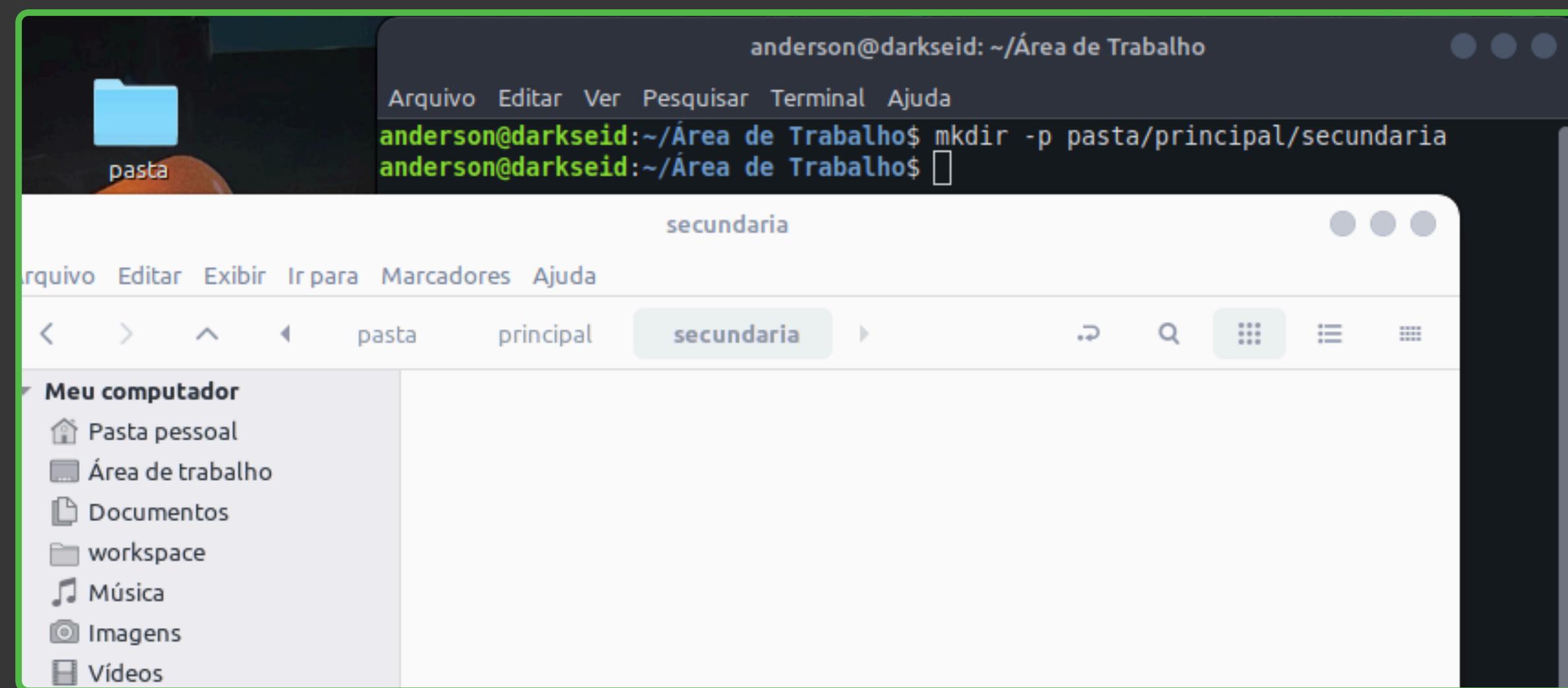


Manipulação de Arquivos e Diretórios

Exemplo:

Para criar diretórios aninhados em uma única **linha** de comando:

mkdir -p pasta/principal/secundaria



Manipulação de Arquivos e Diretórios

Exemplo:

Para criar diretórios aninhados em uma única **linha** de comando:

- **rm [nome_do_arquivo]** - Remove um arquivo.
- **rm -r [nome_do_diretório]** - Remove um diretório e seu conteúdo
- **mv [arquivo_origem] [destino]** - Move ou renomeia arquivos e diretórios.
- **cp [arquivo_origem] [destino]** - Copia arquivos para outro local.



Manipulação de Arquivos e Diretórios

Ajuda e Manual de Comandos

Para aprender mais sobre um comando específico, o Bash oferece recursos de ajuda:

- **man [comando]** - Exibe o manual completo do comando.
- **[comando] --help** - Exibe uma descrição curta e opções do comando.



Introdução ao Shell Script

Shell Script é uma linguagem de script utilizada para automatizar tarefas no sistema operacional Unix/Linux. Ele consiste em um conjunto de comandos executados sequencialmente por um interpretador de linha de comando, como o Bash (Bourne Again Shell), que é o mais popular entre os shells.

O que é Shell Script?

Shell Script é essencialmente um arquivo de texto contendo comandos que podem ser executados no terminal. Esses scripts são utilizados para automatizar processos repetitivos, configurar sistemas, administrar servidores e realizar diversas outras tarefas de forma eficiente.



Benefícios do uso de Shell Script

- **Automatização:** Reduz a necessidade de intervenção manual em tarefas repetitivas.
- **Eficiência:** Permite a execução de vários comandos de forma sequencial e automática.
- **Facilidade de aprendizado:** Usa comandos já conhecidos do terminal.
- **Portabilidade:** Scripts podem ser usados em diferentes sistemas Unix/Linux sem grandes modificações.

Como criar um Shell Script?

1. Criar um arquivo de texto com extensão **.sh**.
2. Adicionar a linha **#!/bin/bash** no início do arquivo, indicando o interpretador a ser usado.
3. Escrever os comandos desejados dentro do arquivo.
4. Dar permissão de execução ao script com o comando:



Como criar um Shell Script?

- Dar permissão de execução ao script com o comando:

```
chmod +x nome_do_script.sh
```

- Executar o script com:

```
./nome_do_script.sh
```

Exemplo de um Shell Script

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

```
# Exibe uma mensagem
echo "Olá, bem-vindo ao mundo do Shell Script!"
```



Imprimindo no Terminal

comando para imprimir no terminal é **printf**. Ele usa os mesmos argumentos que o comando printf na linguagem de programação C. Por exemplo:

```
❯ imprimir.sh
1  #!/bin/bash
2  #Filename: imprimir.sh
3
4  printf "Hello world \n"
```



Formatação e Impressão dos Dados

Os comandos printf seguem este formato:

```
Σ 1  imprimir.sh
  2  #!/bin/bash
  3
  4  valores="10,5"
  5  printf "valor R\$ %s \n" $valores
```



Formatação e Impressão dos Dados

Variáveis

Você cria variáveis assim:

```
 imprimir.sh
1  #!/bin/bash
2  #Filename: imprimir.sh
3
4  nome="João"
5  idade=25
6
7  echo "Nome: $nome"
8  echo "Idade: $idade"
9
10 printf "Nome: %s \"$nome\"\n"
11 printf "Idade %s \"$idade\"\n"
```



Formatação e Impressão dos Dados

read

Comando read (entrada do usuário)

```
imprimir.sh
1  #!/bin/bash
2
3  echo "Digite seu nome:"
4  read nome
5  echo "Olá, $nome!"
```



Formatação e Impressão dos Dados

IF

Condisional **if**

```
imprimir.sh
1  #!/bin/bash
2
3  echo "Digite sua idade:"
4  read idade
5
6  if [ $idade -ge 18 ]; then
7      echo "Você é maior de idade."
8  else
9      echo "Você é menor de idade."
10 fi
```



Formatação e Impressão dos Dados

IF

Formas de criar condições, Quando você quer comparar números inteiros, usa os operadores aritméticos:

Operador	Significado	Exemplo
-eq	igual	[5 -eq 5]
-ne	diferente	[5 -ne 3]
-gt	maior que	[5 -gt 3]
-lt	menor que	[3 -lt 5]
-ge	maior ou igual	[5 -ge 5]
-le	menor ou igual	[3 -le 5]



Formatação e Impressão dos Dados

IF: Comparações de strings

Para comparar textos:

Operador	Significado	Exemplo
=	igual	["\$a" = "hello"]
!=	diferente	["\$a" != "hello"]
-z	string vazia	[-z "\$a"]
-n	string não vazia	[-n "\$a"]



Formatação e Impressão dos Dados

IF: Comparações de strings

Para comparar textos:

 **imprimir.sh**

```
1  #!/bin/bash
2
3  nome="João"
4
5  if [ "$nome" = "João" ]; then
6      echo "Olá, João!"
7  else
8      echo "Você não é o João."
9  fi
```



Formatação e Impressão dos Dados

IF: Testes de arquivos

Você pode verificar se arquivos ou diretórios existem, ou têm certas permissões:

Teste	Significado
-e	existe (arquivo ou diretório)
-f	é um arquivo normal
-d	é um diretório
-r	tem permissão de leitura
-w	tem permissão de escrita
-x	tem permissão de execução

```
imprimir.sh
#!/bin/bash
if [ -f "meuarquivo.txt" ]; then
    echo "Arquivo existe"
else
    echo "Arquivo não encontrado"
fi
```



Formatação e Impressão dos Dados

IF:Condições compostas (AND e OR)

lógico (**&&** ou **-a**) Ambas as condições devem ser verdadeiras.

imprimir.sh

```
1  #!/bin/bash
2
3  if [ "$idade" -ge 18 ] && [ "$idade" -lt 60 ]; then
4      echo "Adulto"
5  fi
```

Ou (menos usado):

imprimir.sh

```
1  #!/bin/bash
2
3  if [ "$idade" -ge 18 -a "$idade" -lt 60 ]; then
4      echo "Adulto"
5  fi
```



Formatação e Impressão dos Dados

IF:Condições compostas (AND e OR)

lógico (|| ou -o) Basta uma das condições ser verdadeira.

imprimir.sh

```
1  #!/bin/bash
2
3  if [ "$usuario" = "admin" ] || [ "$usuario" = "root" ]; then
4      echo "Acesso total"
5  fi
```

imprimir.sh

```
1  #!/bin/bash
2  usuario="admin"
3  if [ "$usuario" = "admin" -o "$usuario" = "root" ]; then
4      echo "Acesso total"
5  fi
```



Formatação e Impressão dos Dados

if com [[...]] (avançado e mais seguro)

- Você pode usar [[...]] no lugar de [...]. Ele permite:
- Operadores como && e || diretamente dentro do colchete
- Uso de curingas (== *.txt, por exemplo)
- Evita erros com espaços ou variáveis vazias

```
imprimir.sh
#!/bin/bash
if [[ "$nome" == Jo* && "$idade" -gt 10 ]]; then
    echo "Nome começa com Jo e tem mais de 10 anos"
fi
```



Formatação e Impressão dos Dados

if com [[...]] (avançado e mais seguro)

- Você pode usar [[...]] no lugar de [...]. Ele permite:
- Operadores como && e || diretamente dentro do colchete
- Uso de curingas (== *.txt, por exemplo)
- Evita erros com espaços ou variáveis vazias



imprimir.sh

```
1  #!/bin/bash
2
3  if [[ "$nome" == Jo* && "$idade" -gt 10 ]]; then
4      echo "Nome começa com Jo e tem mais de 10 anos"
5  fi
```

Dicas importantes

- Sempre coloque variáveis entre aspas ("\$variavel"), para evitar erros com espaços ou valores vazios.
- Use [[...]] para scripts mais robustos.
- Deixe seu código comentado para facilitar manutenção e entendimento.



Formatação e Impressão dos Dados

if com [[...]] (avançado e mais seguro)

```
imprimir.sh
1 #!/bin/bash
2
3 echo "Digite sua idade:"
4 read idade
5
6 if [ "$idade" -lt 12 ]; then
7     echo "Você é uma criança"
8 elif [ "$idade" -lt 18 ]; then
9     echo "Você é um adolescente"
10 elif [ "$idade" -lt 60 ]; then
11     echo "Você é um adulto"
12 else
13     echo "Você é um idoso"
14 fi
```

