

# Comandos Linux

## Índice

Globing e Quoting .....	3
globing e quoting .....	3
;	3
&& .....	3
.....	3
*	3
?	4
[0-9] .....	4
[a-z] ou [A-Z] .....	4
{...,...} .....	4
\ .....	4
“...” ou ‘...’ .....	5
A .....	5
apropos .....	5
apt .....	5
B .....	6
builtin .....	6
bash .....	6
C .....	6
cat .....	6
cd .....	6
cp .....	7
chmod .....	7
D .....	7
dpkg .....	7
E .....	8
echo .....	8
env .....	8
exit .....	8
export .....	8
F .....	8
find .....	9
Find e -exec .....	10
G .....	11
H .....	11
hash .....	11

head.....	11
help.....	12
history.....	12
I.....	12
J.....	12
K.....	12
L.....	12
less.....	12
Atalhos e Comandos dentro do less.....	12
Dicas de Sequência de Comandos.....	14
login.....	14
logout.....	14
ls.....	14
M.....	16
mkdir.....	16
more.....	16
Atalhos e Comandos dentro do more.....	16
Dicas de Sequência de Comandos.....	17
mv.....	17
N.....	17
O.....	17
P.....	18
passwd.....	18
pwd.....	18
Q.....	18
R.....	18
rm.....	18
rmdir.....	18
S.....	18
set.....	18
T.....	19
tail.....	19
touch.....	19
U.....	19
V.....	19
X.....	19
W.....	19
which.....	19

# Globing e Quoting

## globing e quoting

;

;  
; → ; entre dois comandos faz o shell fazer o comando1 depois o comando2.

```
comando1;comando2
```

&&

&& → (E) entre dois comandos faz o shell executar o comando2 se o comando1 ter exit status = 0.

```
comando1 && comando2
```

||

|| → (OU) entre dois comandos faz o shell executar o comando2 se o comando1 dar exit status != 0.

```
comando1 || comando2
```

\*

\* → antes ou depois de um termo expandirá a busca/seleção para qualquer termo a partir do asteriscos ou antes dele, dependendo da limitação de texto antes ou depois. Exemplos:

```
cat *.txt
```

saída: mostraria todo o conteúdo de cada arquivo .txt de uma pasta.

```
cat text*.txt
```

saída: mostraria todo o conteúdo de todos os arquivos que tem o nome com text(AlgumaCoisa).txt

?

? → antes ou depois de um termo irá expandir a buscar/selecionar o nº de caracteres correspondentes ao nº de ? digitados.

## [0-9]

[123] → antes ou depois de um termo irá expandir para buscar/selecionar arquivos onde o espaço que está substituindo sejam os números 1 ou 2 ou 3.

[1239] → antes ou depois de um termo irá expandir para buscar/selecionar arquivos onde o espaço que está substituindo sejam os números 1 ou 2 ou 3 ou 9.

[0-9] → antes ou depois de um termo irá expandir para buscar/selecionar arquivos onde o espaço que está substituindo sejam os números 0 ao 9.

**NOTE** ! nega o comando. EX: [!0-9] → retornaria coisas que não são nºs.

## [a-z] ou [A-Z]

[a-z] → antes ou depois de um termo irá expandir para buscar/selecionar arquivos onde o espaço que está substituindo sejam as letras de “a” a “z” minúsculas. OBS: ! nega o comando. EX: [!a-z] -> retornaria coisas que não são letras

[A-Z] → antes ou depois de um termo irá expandir para buscar/selecionar arquivos onde o espaço que está substituindo sejam as letras de “A” a “Z” maiúsculas.

## {...,...}

**comando {primeiro\*,segundo?}** → vai buscar/selecionar resultados correspondentes ao primeiro\* e/ou ao segundo? ao mesmo tempo.

\

\qualquerCoisa → ao usar o \ antes de qualquer coisa isso indica para o bash que não queremos que essa coisa seja interpretada como normalmente seria.

## “...” ou ‘...’

**echo ‘parametro’** → ao usa ‘ ou “ no início e final de um parametro indica para o bash que o que está entre ‘ ou “ é um único parametro e não um comando.

# A

## apropos

**apropos palavraBuscada** → Busca um comando/manual com determinada palavra-chave.

## apt

**sudo apt autoremove** → remove bibliotecas que não estão sendo utilizadas por outros programas (mesma coisa que **sudo apt-get autoremove**).

**sudo apt install nomepacote** → instalar pacote (mesma coisa que **sudo apt-get install nomePacote**).

**sudo apt search nomePacote** → buscar pacote (mesma coisa que **sudo apt-cache search**).

**sudo apt remove nomepacote** → remove pacote (mesma coisa que **sudo apt-get remove nomePacote**).

**sudo apt remove --purge nomePacote** → remove pacote e seus arquivos de configurações.

**sudo apt remove --purge nomePacote -y** → remove pacote e seus arquivos de configurações concordando com alterações.

**sudo apt upgrade** → atualizar pacotes do sistema (mesma coisa que **sudo apt-get upgrade**)

**sudo apt-get autoremove** → remove bibliotecas que não estão sendo utilizadas por outros programas.

**sudo apt-get dist-upgrade** → atualiza somente o kernel do sistema.

**apt-cache search nomePacote** → buscar pacote.

**sudo apt-get install nomePacote** → instalar pacote.

**sudo apt-get install --only-upgrade nomePacote** → atualizar somente o pacote escolhido.

**sudo apt-get -f install nomePacote** → instala pacote com dependencias, corrige dependencias.

**sudo apt-get remove nomePacote** → remover pacote.

**sudo apt-get update** → atualizar lista dos repositórios.

**sudo apt-get upgrade** → atualizar pacotes do sistema.

# B

## builtin

**builtin comando** →

## bash

**bash** → abre um bash. Pode abrir um bash dentro de outro bash. Dica: abra um bash dentro de outro bash para fazer testes.

**bash -c** →

# C

## cat

### IMPORTANT

*Objetivo do cat*

O objetivo do cat é servir como concatenador de arquivos.

**cat nomeArquivo** → exibe conteúdo de nomeArquivo.

**cat nomeArquivo1 nomeArquivo2** → exibe e concatena nomeArquivo1 e nomeArquivo2.

**cat -n nomeArquivo** → exibe conteúdo do nomeArquivo e exibe o número de linhas do documento.

**cat -q nomeArquivo** → exibe o conteúdo do nomeArquivo sem informações sobre o arquivo. Exibe somente o texto, não exibe o título.

**cat -n 5 nomeArquivo** → exibe o conteúdo das 5 primeiras linhas. Obs: Também pode ser utilizado qualquer outro número no lugar do 5.

**cat /etc/os-release** → exibe informações sobre a versão do sistema operacional

**cat /etc/lsb-release** → exibe informações sobre a versão do sistema operacional debian-like

**cat /etc/redhat-release** → exibe informações sobre a versão do sistema operacional redhat-like.

## cd

**cd** → muda do diretório atual para a home do usuário.

**cd ~/pasta** → muda do diretório atual para /home/usuario/pasta. É um atalho para usar a /home do usuário.

**cd .** → muda para o diretório atual.

**cd ..** → muda para o diretório anterior.

**cd endereco/da/pasta/desejada** → muda de pasta.

**cd -** → muda para a pasta anterior em que o usuário acessou no terminal.

## cp

**cp nomeArquivo nomeArquivo(copia)** → cria uma cópia do nomeArquivo com nome escolhido, no caso seria, de nomeArquivo(copia)

**cp NomeArquivo diretorio1/subdiretorio1** → cria uma cópia de nomeArquivo chamado nomeArquivo em subdiretorio1.

**cp -i nomeArquivo nomeArquivo(copia)** → antes de copiar arquivos exibe um pergunta para confirmar a operação.

**cp diretorioOrigem/\* diretorioDestino** → copia todos os arquivos de diretorioOrigem para diretorioDestino. Não copia diretórios.

**cp -u diretorioOrigem/\* bkpDiretorioDestino** → copia só os arquivos que foram modificados de diretorioOrigem para bkpDiretorioDestino

**cp -b diretorio1/\* diretorio2** → sobrescreve os arquivos de diretorio1, mas antes cria cópias de backup.

**cp -r diretorioOrigem diretorioDestino** → copia diretorio. Copia o conteúdo diretorioOrigem para o diretorioDestino recursivamente.

**cp -r diretorioOrigem diretorioDestino/** → cria uma pasta diretorioOrigem em diretorioDestino, se existir, e copia todo o conteúdo de diretorioOrigem, incluído a própria pasta, para diretorioDestino.

## chmod

**chmod+x nomeArquivo** → permissão de execução para nomeArquivo. Equivalente a **chmod 777 nomeArquivo**.

## D

### dpkg

**sudo dpkg -i nomeArquivo.deb** → instala pacote deb , não instala dependências.

**dpkg --get-selections** → exibe lista simplificada de pacotes que estão e já estiveram no sistema e seu estado.

**dpkg -l** → lista pacotes que estão ou já estiveram no sistema (mesma coisa que **dpkg --list**).

**dpkg -l | grep nomePacote** → encontra o pacote especificado na lista de pacotes que estão ou já estiveram no sistema.

**dpkg --list** → lista pacotes que estão ou já estiveram no sistema.

**sudo dpkg -r nomeArquivo.deb** → desinstala pacote deb , não desinstala dependências.

**sudo dpkg --remove nomeArquivo.deb** → desinstala pacote deb , não desinstala dependências (mesma coisa que **sudo dpkg -r nomeArquivo.deb**).

**sudo dpkg -P nomeArquivo.deb** → remove pacotes deb e arquivos de configurações.

**dpkg --purge nomeArquivo.deb** → remove pacotes deb e arquivos de configurações (mesma coisa que **sudo dpkg -P nomeArquivo.deb**).

## E

### echo

**echo mensagem** → Exibe uma mensagem na tela.

### env

**env** → lista todas as variáveis de ambiente.

**env nomeVariavel1=valorVariavel1 nomeVariavelN=valorVariavelN comando** → cria um novo ambiente com as variáveis temporárias e executa o comando ou script. Pode-se criar variáveis com nomes diferentes ou até mesmo iguais a variáveis já existentes, pois o escopo desse comando só vale para o novo ambiente criado. Acabou o comando, acabou o ambiente, fim das variáveis e seus valores.

### exit

**exit** → sair de um shell do usuário. Finaliza o shell.

### export

**export nomeVariavel** → Define atributo de exportação para variáveis shell. Transforma uma variável de shell em de variável de ambiente.

**export nomeVariavel -n** → variável deixa de ser de ambiente. Variável ainda existe, mas como variável de shell.

## F



# find

**find** → exibe arquivos que estão localizados a partir do diretório atual.

**find -print** → exibe arquivos que estão localizados a partir do diretório atual.

**find /caminho/absoluto/diretório/arquivo** → exibe arquivos que estão localizados a partir do diretório indicado pelo caminho absoluto.

**find /caminho -type f** → procurar e exibe todos os arquivos a partir do diretório /caminho

**find -type d** → procura e exibe todas as pastas a partir do diretório atual.

**find .** → exibe arquivos que estão localizados a partir do diretório atual. Utiliza caminho absoluto.

**find -iname nomeArquivoOuDiretorio** → procura e depois exibe todos os resultados que contêm nomeArquivoOuDiretorio, a partir da pasta atual. -iname também aceita ?, \* e [], para isso utiliza entre aspas a expressão e suporta arquivos com . (ponto).

**TIP** | -iname não diferencia maiúsculas e minúsculas.

**find -name nomeArquivoOuDiretorio** → procura e depois exibe todos os resultados que são exatamente nomeArquivoOuDiretorio, a partir da pasta atual. -name também aceita ?, \* e [], para isso utiliza entre aspas a expressão e suporta arquivos com .(ponto).

**TIP** | -name diferencia maiúsculas e minúsculas.

**find -name "arquivo\*"** → procura e exibe todos os arquivos ou pastas que comecem com arquivo.

**TIP** | É obrigatório o uso de " " nesses casos.

**find -type d -name "nomeDiretorio"** → procura e retorna somente diretórios que contenham o termo nomeDiretorio.

**find -type f -name "nomeArquivo"** → procura e retorna somente arquivos que contenham o termo nomeArquivo.

**find -type l -name "nomeLinkSimbolico"** → procura e retorna somente links simbólicos que contenham o termo nomeDiretorio.

**find /caminho -user nomeUsuario** → procura em /caminho os arquivos e pastas que o usuário nomeUsuario é dono.

**find /caminho -group nomeGrupo** → procura em /caminho os arquivos e pastas que o grupo nomeGrupo é dono.

**find -size +10k** → procura e retorna arquivos maiores que 10kb.

*Opções extras:*

**TIP**

-00[k|M|G], arquivos menores que 00 kb ou Mb ou Gb.  
+00[k|M|G], arquivos maiores que 00 kb ou Mb ou Gb.

**find -atime -7** → procura arquivos que foram acessados no período dos últimos 7 dias.

**TIP**

**find -atime n**, onde n = nº de dias do período que deseja procurar.

**find -mtime -7** → procura arquivos que foram modificados no período dos últimos 7 dias.

**TIP**

**find -mtime n**, onde n = nº de dias do período que deseja procurar.

**find -ctime -7** → procura arquivos que foram modificados, mudadas as permissões dos arquivos, incluindo grupos, arquivos mudou de diretório, no período dos últimos 7 dias.

**TIP**

**find -ctime n**, onde n = nº de dias do período que deseja procurar.

**find -inum 4474556** → procura arquivos que tem o inode especificado.

**find -name "\*condicao1" find -name "\*condicao2"** → procura arquivos que satisfaçam as duas condições de busca. Isso equivale ao um E: **find -name "\*condicao1" find -a -name "\*condicao2"**.

**find -name "\*condicao1" -o find -name "\*condicao2"** → procura arquivos que satisfaçam a condição 1 OU a condição 2.

**find \( -name "\*condicao1" -o find -name "\*condicao2" \) -a -name "\*condicao3"** → procura arquivos que satisfaçam a (condição 1 OU a condição 2) E depois a condição 3.

*Precedência*

**TIP**

Ao usar mais de um OU(-o) ou/e E(-a) juntos na mesma expressão utilizar parenteses para ter certeza da precedência do E/OU.

*Escape dos ( )*

**TIP**

Quando usar ( ) utilizar a \ antes, então fica assim: \(\) .

**find -name "\*condicao1\*" ! -name "\*condicao2\*"** → procura arquivos que satisfazem a sentença inteira, ou seja, retorna arquivos onde a condicao1 é verdadeira e a condicao2 é falsa. ! é equivalente ao NÃO (negação), ele inverte o valor lógico da sentença.

## Find e -exec

## IMPORTANT

*-exec*

**-exec** → indica que após ele virá um conjunto de instruções para se executar em todos os arquivos resultantes do find.

## WARNING

tomar cuidado ao utilizar, pois pode ser usado até o rm como comando no final da busca e todos os arquivo que resultarem da busca do find serão excluídos.

**find -name "\*condicao1\*" -exec ls -l "{}" \;** → para cada saída do find será apresentado como ls -l.

## NOTE

*Uso do {}*

{ } indica que para cada caminho que o find encontrar será executado o que está depois de -exec.

## NOTE

*Uso do " " e \*

"" e \ indica escape do bash, pois { } e ; são caracteres especiais do bash.

**find -name "\*condicao1\*" -exec echo "{}" foi encontrado ";"** → exibe “foi encontrado” logo após cada saída do find.

**find name "\*condicao1\*" -exec rm -i "{}" \;** → remove interativamente cada arquivo que o find encontrar que satisfazem a condicao1.

# G

# H

## hash

**hash** → exibe lista de comandos de programas/binários “cacheados” na memória.

**hash -r** → esquece local de todos os programas já “cacheados”

**hash -d nomeComando nomeComando2** → esquece local já “cacheado” do nomeComando e nomeComando2.

## head

**head nomeArquivo** → exibe primeiras linhas do arquivo.

**head -n 5 nomeArquivo1 nomeArquivo2** → exibe as 5 primeiras linhas do nomeArquivo1 e do nomeArquivo2.

# help

**help comando** → exibe uma descrição, muitas vezes breve, do que o comando pode fazer junto com instruções de execução do comando.

## IMPORTANT

O **help comando** funciona se o comando for um built-in do Shell. Built-ins são os comandos incorporados ao Shell. Para saber se o comando é um built-in use **type comando**.

# history

**history** → lista o histórico dos comandos no terminal.

**history #Nº** → list os últimos Nº comandos no terminal.

# I

# J










# K

# L

# less

**less nomeArquivo** -→ mostra conteúdo do arquivo

## Atalhos e Comandos dentro do less

Comando/Atalho	Propósito
	navega para linha de cima
	navega para linha de baixo.
 ou 	navega uma linha por vez pra baixo.
 ou 	navega um linha por vez pra cima.
 ou 	navega por páginas. Próxima página.
	navega por páginas. Página anterior.

Comando/Atalho	Propósito						
<code>Home</code> ou <code>B</code>	retorna para início do documento.						
<code>End</code> ou <code>Shift + G</code>	vai para o fim do documento.						
<code>P</code> + <code>0-100</code>	<code>P</code> depois número inteiro entre 0 e 100 pula para a porcentagem correspondente.						
<code>H</code>	ajuda.						
<code>!</code> comando	com <code>!</code> logo em seguida digitar um comando do terminal, podemos usar comandos do shell dentro do less.						
<code>V</code>	abre o editor padrão do sistema na linha atual.						
<code>-</code> + <code>I</code>	ignora todos os cases nas buscas						
<code>Ctrl</code> + <code>L</code>	redesenha a tela.						
<code>R</code>	edesenha a tela.						
<code>:</code> + <code>N</code>	próximo arquivo.						
<code>:</code> + <code>p</code>	arquivo anterior.						
<code>Q</code>	sair.						
<code>?</code>	pesquisa reversa (começa pesquisa do final). Comandos de navegação são os mesmos que os da <code>/</code> .						
<code>/</code>  <i>Table 1. Atalhos de Navegação da Pesquisa</i> <table border="1"> <thead> <tr> <th>Atalho</th><th>Propósito</th></tr> </thead> <tbody> <tr> <td><code>N</code></td><td>Próxima incidência</td></tr> <tr> <td><code>Shift</code> + <code>N</code></td><td>Incidência anterior</td></tr> </tbody> </table>	Atalho	Propósito	<code>N</code>	Próxima incidência	<code>Shift</code> + <code>N</code>	Incidência anterior	pesquisar
Atalho	Propósito						
<code>N</code>	Próxima incidência						
<code>Shift</code> + <code>N</code>	Incidência anterior						
<code>M</code> + <code>letra</code>	Marcar posição						
<code>'</code> + <code>letraMarcada</code>	vai para posição marcada pela letraMarcada						
<code>''</code>	Volta na posição anterior marcada com <code>'</code> + <code>letraMarcada</code>						

Comando/Atalho	Propósito
<div>⌘ + /</div> <div> <b>TIP</b> <div>⌘ + Enter :Desativa filtros.</div> </div>	Pesquisar (só retorna as linhas onde encontrou a palavra).

## Dicas de Sequência de Comandos

- -I & : dica está incompleta.

## login

**login** → Começa uma nova sessão no sistema, abre um novo shell. Será pedido nome do usuário e senha.

## logout

**logout** → finaliza a sessão do usuário, incluindo todos os shells abertos na sessão. Conselhável utilizar o exit.

<b>TIP</b>	<i>Atalho</i>
	Pode-se usar também a combinação de teclas <b>Ctrl</b> + <b>D</b> , mas não é sempre que vai estar configurada.

## ls

**ls** → Exibe os arquivos do diretório em uma lista com colunas.

**ls --color** → Saida da listagem das pastas fica colorida.

**ls --color=auto** → exibe os arquivos do diretório em uma lista com colunas com cores:

Table 2. Sigificado das cores (Não é um padrão)

Cor	Sigificado
Branco	Arquivos
Azul	Pastas
Azul-claro	link
Verde	Executável
Vermelho	Pacotes

**ls \*** → Exibe todas, menos arquivos ocultos, as pasta e um nível de subpasta de cada pasta do diretório atual.

**ls .\*** → Exibe todos arquivos ocultos e as pasta ocultas e um nível de subpasta de cada pasta do diretório atual.

**ls -a** → lista pasta e arquivos, incluindo `.`, `..` e ocultas.

**ls -F** → lista arquivos e pastas com identificadores visuais (para monitores de uma cor) ao final do nome do arquivo:

Table 3. Significado dos Caracteres

Caracter	Significado
Sem caracter	Arquivo normal
/	Diretório
*	Executável
@	Link

**ls -p** → Lista somente pasta com o identificador visual `/` no final do nome. Outros tipos de arquivos não tem caracteres de identificação visual.

**ls --color=never** → Não exibe cor na listagem de pasta e arquivos.

**ls -r** → inverter a ordem de exibição dos arquivos. Será de z-a no lugar de a-z.

**ls --sort=size** → Exibe os arquivos do diretório em uma lista ordenada pelo tamanho, do maior para o menor.

**ls -S** → ver `ls --sort=size`.

**ls --sort=time** → exibe os arquivos do diretório em uma lista ordenada pelo tempo de criação do arquivo, do mais recente para o mais antigo.

**ls -t** → ver `ls --sort=time`

**ls --sort=extension** → exibe os arquivos do diretório em uma lista ordenada pelo tipo de extensão do arquivo., do que tem extensão para o que não tem em ordem alfabética.

**ls -X** → ver `ls --sort=extension`.

**ls --format=long** → exibe os arquivos do diretório em uma lista de uma única coluna com bastante detalhes sobre os arquivos.

**ls -l** → mesma função do `ls -l --format=long`.

**ls -R** → `ls` recursivo. Exibe o conteúdo dos diretórios recursivamente. O `ls` vai listar todas as pastas mães e filhas.

**ls -ld nomeDiretório** → exibe informações completas somente do nomeDiretório.

**ls -d** → exibe lista somente dos diretórios.

# M

## mkdir

**mkdir nomeDiretorio** → criar um um diretório na pasta atual

**mkdir diretorio1/subdiretorio/nomeDiretorio** → cria diretorio na pasta direto1/subdiretorio

**mkdir -p diretorio1/subdiretorio1/subdiretorio2** → força a criação de pasta, mesmo se ,por exemplo, subdiretorio1 e subdiretorio2 não existirem.

**mkdir nomeDiretorio -v** → criar uma pasta e no final exibe uma mensagem que a pasta foi criada.

**mkdir -p diretorio1/subdiretorio{1,2,3}/pastaAno200{1,2,3,4}** → exemplo de criação de diretórios onde cria da seguinte maneira:

```
skinparam Legend {
    FontSize 17
}
legend
diretorio1
|_ subdiretorio1
|_ pastaAno2001
|_ pastaAno2002
|_ pastaAno2003
|_ pastaAno2004
|_ subdiretorio2
|_ pastaAno2001
|_ pastaAno2002
|_ pastaAno2003
|_ pastaAno2004
|_ subdiretorio3
|_ pastaAno2001
|_ pastaAno2002
|_ pastaAno2003
|_ pastaAno2004
end legend
```

## more

**more nomeArquivo** → mostra conteúdo do arquivo.

### Atalhos e Comandos dentro do more

Comando/Atalho	Propósito
<input type="text" value="Enter"/>	Navega linha por linha



Comando/Atalho	Propósito				
Espaço	navega por páginas. Próxima página.				
B	navega por páginas. Página anterior.				
H	ajuda.				
! comando	com ! logo em seguida digitar um comando do terminal, podemos usar comandos do shell dentro do more.				
Q	sair.				
/	pesquisar				
<p><i>Table 4. Atalhos de Navegação da Pesquisa</i></p> <table> <tr> <th>Atalho</th><th>Propósito</th></tr> <tr> <td>N</td><td>Próxima incidência</td></tr> </table>		Atalho	Propósito	N	Próxima incidência
Atalho	Propósito				
N	Próxima incidência				
V	abre editor vi na linha atual do arquivo				

## Dicas de Sequência de Comandos

- -I & : dica está incompleta.

## mv

**mv Nomearquivo diretorio/Destino** → move arquivo ou pasta para diretorio/Destino.

**mv nomeArquivo novoNomeArquivo** → renomeia nomeArquivo (ou pasta) para novoNomeArquivo. Funciona se novoNomeArquivo não existir, se novoNomeArquivo existir irá sobreescrever o novoNomeArquivo.

**mv -vi Nomearquivo diretorio/Destino** → forma mais segura de renomear um arquivo ou pasta. Irá perguntar se caso sobreescreva o arquivo.

**mv -n nomeArquivo novoNomeArquivo** → forma segura de renomear arquivo ou pasta. Se arquivo ou pasta existir não sobreescreve.

## N

## O

# P

## passwd

**passwd** → permite que usuário mude a sua própria senha.

**passwd nomeUsuario** → permite que superusuário, no caso pode ser o usuário root, mudar a senha do usuário especificado.

## pwd

**pwd** → exibe o caminho atual em que está localizado.

# Q

# R

## rm

**rm nomeArquivo** → remove/exclui nomeArquivo

**rm -fr nomePasta** → exclui pasta e subpastas recursivamente e forçadamente sem pergutar.

## rmdir

**rmdir nomeDiretorio** → apaga pasta nomeDiretorio, somente se ela estiver vazia.

**rmdir diretorio1/subdiretorio/nomeDiretorio** → apaga somente pasta nomeDiretorio, se estiver vazia.

**rmdir -p diretorio1/subdiretorio1/subdiretorio2/nomeDiretorio** → força o apagamento das que estão vazias.

**rmdir -v diretorio1** → remove diretorio1 e exibe uma mensagem relatando a remoção de diretorio1

# S

## set

**set -o emacs** → configura atalhos do terminal para atalhos de emacs, já é o padrão.

**set -o vi** → configura atalhos do terminal para atalhos de emacs

# T

## tail

**tail nomeArquivo** → imprime as 10 últimas linhas de um arquivo.

**tail n +30 nomeArquivo** → imprime a partir da linha 30 do nomeArquivo. o número de linhas para mostrar pode variar.

**tail n 30 nomeArquivo** → imprime as últimas 30 linhas do nomeArquivo. o número de linhas para mostrar pode variar.

**tail -f nomeArquivo** → lê sempre as últimas movimentações do arquivo.

## touch

**touch nomeArquivo** → cria um arquivo vazio. Touch em um arquivo existente atualiza a data de última modificação e acesso do arquivo.

**touch nomeDiretorio** → Touch em um diretório já existente atualiza a data de última modificação e acesso da pasta.

# U

# V

# X

# W

## which

**which comando** → exibe caminho do executável.