CTRL+F1 até F2 >> modo texto do sistema (TTYs)

CTRL+F7 >> modo gráfico do sistema

;

; --> ; 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

globing e quoting

*

*** -->** 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. Exemplo:

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.

> **OBS: ! 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 primeio* 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

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 sistema.

B

builtin

builtin comando --> pesquisar mais sobre depois

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 --> não entendi direito

C

cat

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 nomeArquino --> 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 utilado 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 excolhido, 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/* diretórioDestino --> copia todos os arquivos de diretórioOrigem para diretórioDestino. Não copia diretórios.

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

cp -b diretorio1/* diretorio2 --> sobresescreve 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, incluido a própria pasta, para diretorioDestino.

chmod

chmod+x nomeArquivo --> permissão de execução para nomeArquivo

D

dpkg

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

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

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

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

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/aqrquivo --> 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 diretorio /caminho

find -type d --> procura e exibe todas as pastas a 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. -name também aceita ? , * e [] , para isso utiliza entre aspas a expressão e suporta arquivos com . . **Obs: Ele 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 . . **Obs: Ele diferencia maiúsculas e minúsculas.**

find -name “arquivo*” --> procura e exibe todos os arquivos ou pastas que commecem com arquivo. Obs: 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. Obs: opções extras: **-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. Obs: **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. Obs: **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. Obs: **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. Obs 1: Ao usar mais de um OU(-o) ou/e E(-a) juntos na mesma expressão utilizar parenteses para ter certeza da precendência do E/OU. Obs 2: 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

-exec --> indica que após ele virá um conjunto de instruções para se executar em todos os arquivos resultantes do find. Obs: 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**. **Obs: {}** indica que para cada caminho que o find encontrar será executado o que está depois de **-exec**. **Obs2: "" 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

ver $HISTFILE

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, (aperte q para sair).

- comandos dentro do less
  ^
  | *--> navega para linha de cima*

  |
  v *--> navega para linha de baixo*
  e *ou* ***j*** --> navega uma linha por vez pra baixo
  y ou **k** --> navega um linha por vez pra cima
  Espaço ou **f** --> navega por páginas. Próxima página
  b --> navega por páginas. Página anterior.
  Home ou **g** --> retorna para início do documento
  End ou **Shift + G** --> vai para o fim do documento
  p + [0-100] --> p depois número inteiro entre 0 e 100 pula para a porcentagem correspondente.
  h --> ajuda

!comando --> com ! logo em seguida digitar um comando do terminal, podemos usar comandos do shell dentro do more.
v --> abre o editor padrão do sistema na linha atual.
-I --> ignora todos os cases
Ctrl + l --> redesenha a tela
r --> redesenha a tela.
:n --> próximo arquivo
:p --> arquivo anterior
q --> sair
? --> pesquisa reversa (começa pesquisa do final). Comandos de navegação são os mesmos que os da / .
/ --> pesquisar
n --> próxima incidência
N --> incidência anterior
m + letra --> marcar posição
' + letraMarcada --> vai para posição marcada pela letraMarcada
'' -->volta na posição anterior
&/ --> pesquisar (só retorna as linhas onde encontrou a palavra).
&+Enter --> desativa filtros

Sequência de comandos
-I &

locate

locate nomeArquivo --> pesquisar o nomeArquivo em todas as pastas do sistema (também pesquisa pastas).

Obs1: Cada distro tem o seu locate, uma instalam por padrão outras não, se quiser utilizar deve procurar no google “install locate nomeDistro”, pois em cada distro ele pode estar em um pacote diferente.

Obs2: O locate armazena o local de cada arquivo em um banco de dados, que não é atualizado a cada exclusão/inclusão de arquivos. Então se quiser fazer uma pesquisa atualizada uitlizar:

> **sudo updatedb**
> **locate nomeArquivo**

locate -e --> pesquisar o nomeArquivo em todas as pastas do sistema, menos arquivos excluídos. O banco ainda não é atualizado.

locate ‘*/nomeArquivo*’ --> pesquisar todos arquivos que contenham /nomeArquivo* antes, depois, ou no meio do nome do arquivo.

Obs3: o locate aceita o globing, mas tomar cuidado que os arquivos não começam com nomeArquivos e sim com /nomeArquivos.

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. **Dica:** pode-se usar também a combinação de teclas <Ctrl>+d, 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:
(não é padrão)
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:
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 unica coluna com bastante detalhes sobre os arquivos.

ls -l --> mesma função do **ls –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 nomeDiretotio -v --> criar uma pasta e no final exibe uma mensagem que a pasta foi craiada.

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

mv

more nomeArquivo --> mostra conteúdo do arquivo, (aperte q para sair).

- comandos dentro do more
  Enter --> navega linha por linha
  Espaço --> navega por páginas. Próxima página
  b --> navega por páginas. Página anterior.
  q --> sair
  h --> ajuda
  / --> pesquisar
  n --> próxima incidência. Quando alcançar última incidência irá sair do more
  v --> abre editor vi na linha atual do arquivo
  !comando --> com ! logo em seguida digitar um comando do terminal, podemos usar comandos do shell dentro do more.

Sequência de comandos
-I &

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

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.

r**mdir -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.

touch -c nomeArquivo --> se arquivo não existir não cria ele.

type

type comando --> eixbe informações sobre o comando, que tipo é o comando

type -a comando --> exibe todas as informações do comando: localizações, tipo (alias, comando interno do shell (builtin), funções, etc).

type -t comando --> exibe somente o do tipo do comando.

U

V

X

W

which

which comandoPesquisado --> encontra onde está localizado o arquivo executável no PATH, ele ignora funções do shell

which -a comandoPesquisado --> encontra todos os locais onde o comando pode estar localizado, ignora funções do shell.

Whereis

whereis comando --> pesquisa todos os locais dos arquivos existentes relativos ao comando, como manuais, bibliotecas .h, etc.

whatis

whatis comando -->

wc

wc nomeArquivoTexto.txt --> retorna uma linha com informações sobre o arquivo que contêm na ordem:

nºdeLinhas nºdePalavras nºdeBytes nomeArquivoTexto.tx

Exemplo:
170 933 6111 arquivoTexto

wc -L nomeArquivoTexto.txt --> retorna uma linha com o número de carcteres da linha mais longa do nomeArquivoTexto.txt

nºdeCarcteres nomeArquivoTexto.tx

Exemplo:
223 arquivoTexto

Y

Z

pwd

pwd --> Mostra o diretório que você está.

/etc/skel --> skel é o pdrão pra a criação de uma nova pastahome para algum usuario;
 Caso você queira alguma configuração padrão para todos os novos usuários, você
deve alterar o /etc/skel.

tail nomeArquivo -->

head nomeArquivo -->

---------- REDE ----------
nmcli d - lista placas de rede
nmtui   - interface " gráfica" de gerenciador de rede
systemctl restart network --> reiniciar rede do computador

SHELL BASH

COMANDOS SCRIPT

exit

exit numero --> script fica com status de saida com o numero indicado.

Atalhos

Ctrl+A ou **Home** --> Começo da linha.

Ctrl+E ou **End** --> Final da linha.

Ctrl+T --> troca duas últimas letras do que foi digitado.

Ctrl+L --> limpar a tela do terminal.

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 vi

Ctrl+R --> Abre uma pesquisa que busca, depois digite um termo a ser buscado. Com o termo buscado digitado aperte **Ctrl+R** para buscar para trás. **Ctrl+C** cancela o comando.

Execução

Como executar

./nomeArquivo

ou

bash nomeArquivo

Variáveis

O escopo das variáveis de shell e ambiente são válidos somente no terminal aberto. Fecho o terminal, perdeu o valor das variáveis. Abriu um segundo terminal, as variáveis criadas no primeiro terminal não serão pertecerão ao escopo do segundo terminal.

Regras Para Nomeação

- Nome da variável não pode começar com número.
- Variáveis com valores que são strings utilizar “ ou ‘ no início do valor e “ ou ‘ no final do valor da variável.
- Variáveis de Ambiente (export) --> Nomes de variáveis são escritos com todas as letras em CAIXA ALTA.

- Variáveis de shell e temporárias --> Nomes de variáveis são escritos com todas as letras em caixa baixa.
- Usar _ (underline) no lugar do espaço ao criar o nome conmposto de variáveis.
- Sempre referenciar uma variável usar o nome da variável entre {}. Exemplo: Minha idade é ${IDADE}anos.

Para criar variáveis:

nomeVariavel=valor

(dica: sem espaço antes e depois do = **)**

Quando uma variável é criada ela é uma varável de shell, ou seja, a variável só é visível para o shell, outros scripts externos não "exergam" as variáveis. Para tornar as variáveis de shell em variávei de ambiente temos que exportá-las, ou seja, a variável fica visível para outros scripts externos.

Como usar:

echo O valor da variavel e **$nomeVariavel**

(**$depoisNomeDaVariavel** --> retorna o valor da variavel)

Exportar variável:

O export transforma uma variável de bash em variável de ambiente. Depois podemos mudar o valor da variável no bash normalmente e o valor dela irá mudar também no ambiente.

O export só exporta os valores das variáveis e novas variáveis para os processos filhos, não funciona no processo pai.

export nomeVariavel
export nomeVariavel=valorVariavel

ou

export nomeVariavel1 nomeVariavel2 nomeVariavelN

Remover Propriedade de AMBIENTE da variável:

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

Remover(excluir) variável
unset nomeVariavel --> remove variável definitivamente, ou seja, a variavel é excluída do AMBIENTE e do shell.

Comandos Úteis

env

env --> lista todas as variáveis de ambiente

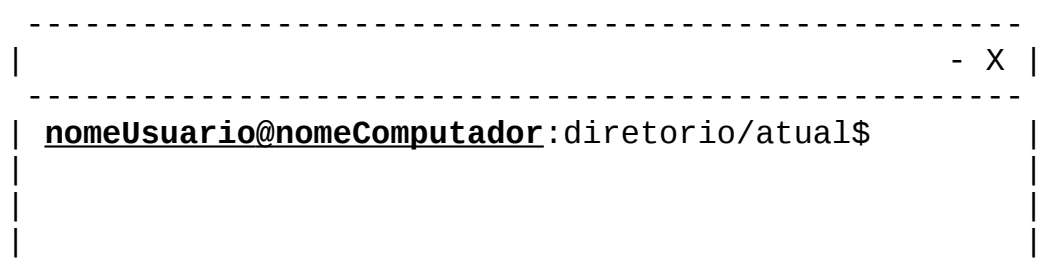
**env nomeVariavel1=valorVariavel1 nomeVariavelN=valorVariavelN comando --> ** cria um novo ambiente com as variáveis temporárias e executa o comando ou script. Pode-se criar variáveis com nomes diferentes ou até mesmo iguais a variáveis já existentes, pois o escopo desse comando só vale para o novo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

Variáveis de Ambiente Especiais

Existem variáveis de ambiente que são reservadas e tem uma função especifica no terminal/sistema. Por exemplo:
$PWD é reservada para indicar o diretório atual.

Podemos alterar o valor dessas variáveis, como se fosse uma trapaça. Em alguns casos como do PWD fica esquisito, mas existem outras variáveis em que essa “trapaça” pode ser útil.

PS1 --> **prompt 1 . Ela define o que está escrito na área antes do** : no terminal. Por padrão no LinuxMint temos



Se definirmos como:
PS1=’\u@\w:’
\u = indica o usuário
@ = indica que estamos no diretório
\w = indica qual o diretório estamos.
: = :

Então temos:

PS2 --> é um indicador de próxima linha, ele serve para quando executamos um comando e ele é muito grande e, por isso, desejamos quebrar ele em duas linhas

```
 ------------------------------------------------------
|                                                  - X |
 ------------------------------------------------------
| nomeUsuario@nomeComputador:~$ “ Essa e uma mensagem |
|> de exemplo quebrando a linha                       |
|                                                     |
|                                                     |
 -----------------------------------------------------
```

PS2=’nova linha>’

Então temos:

```
 ------------------------------------------------------
|                                                  - X |
 ------------------------------------------------------
| nomeUsuario@nomeComputador:~$ “ Essa e uma mensagem |
|nova linha> de exemplo quebrando a linha             |
|                                                     |
|                                                     |
 -----------------------------------------------------
```

$HOME --> indica o diretório padrão(home) do usuário. Por padrão a home do usuária fica em /home/nomeUsuario. A home do root fica em /root

$LOGNAME --> indica o nome do usuário do sistema

$UID --> indica o user ID. Número do usuário. SOMENTE LEITURA

$EDITOR --> indica o editor padrão.

$HISTFILE --> indica o local do arquivo onde fica gravado o histórico dos comandos.

echo $HISTFILE --> exibe o local do arquivo do histórico.

$?

$? --> mostra o status do último comando executado.
Resultados esperados:
0 – ok . comando/programa funcionou corretamente.
Outro código:

PATH

$PATH --> lista todos os locais onde estão os binários (dos comandos)

echo $PATH --> exibe todos os locais onde estão os binários (dos comandos)
Ele pesquisa os os locais (direitórios) dos camandos da esquerda para direita, do primeiro para o último, um por um. Podemos

redefinir o $PATH de uma forma para ele pesquisar em uma pasta primeiro ou por último:

nomeUsuario@nomeComputador: echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/bin:/usr/games:/usr/local/games

nomeUsuario@nomeComputador: PATH=$PATH:/home/nomeUsuario

Apartir daqui /home/nomeUsuario está no final da lista do PATH

nomeUsuario@nomeComputador: echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/bin:/usr/games:/usr/local/games:/home/nomeUsuario

Agora vamos fazer /home/nomeUsuario primeiro na lista de locais de busca do PATH

nomeUsuario@nomeComputador: PATH=/home/nomeUsuario:$PATH

Agora temos:

nomeUsuario@nomeComputador: echo $PATH

/home/nomeUsuario:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/bin:/usr/games:/usr/local/games:/home/nomeUsuario

$MANPATH

$MANPATH --> Essa variável diz para o man, onde ele deve procurar os manuais.

Outras Observações:

O shell sempre vai dar preferência em comandos builtin, depois programas, na hora na execução de um comando. Por exemplo existe dois pwd:

pwd é um comando interno do shell (shell builtin)
pwd é /bin/pwd

quando executamos pwd no terminal o shell irá buscar pelo pwd builtin

\ --> simula o enter, caso queira quebrar a linha ao digitar alguma coisa

help/man/Consultar Informações

help

Para usar o **comando + help** ou **parecido** irá depender do tipo de comando que quer urilizar junto com o help. Cada tipo de comando tem uma sintaxe diferente:

- ***Builtin* do *shell***

  help + comando

- ***Programas externos ao shell***

  Em geral programas externos ao shell aceitam alguma dessas variações do help:

  - **comando --help** --> mais usada
  - **comando -h**
  - **comando -?**

man

O man(ual) online está presente na maioria dos comandos.

man + comando --> exibe um manual com informações do comando.

man man --> manual do man

man -k palavraChave --> faz uma busca pela palavraChave em todos os presentes manuais do sistema

Atalhos de navegação

Dentro do man de algum comando utiliza-se os atalhos:

barra de espaço *--> navega para próxima página*

^
| *--> navega para linha de cima*

|
v *--> navega para linha de baixo*

/ --> pesquisar o termo digitado
e ainda usamos:
n - próxima incidência
N *- incidência anterior*

Onde encontrar a documentação

Para saber o local (diretórios) em que está o(s) manual(ais) de uma comando usar:

whereis comando

Depois procure arquivos que teham nomes que fazem referência a documentação de manuais.

Onde o man encontra a documentação

ver **$MANPATH**

Se a variável $MANPATH é vazia ele procura um arquivo de configuração e este arquivo está em **less /etc/manpath.config .** No /etc/manpath.config o man encontra instruções de como encontrar os manuais dos comandos

Existe um comando que mostra, de forma resumida, onde o man deve procurar os manuais:

> **manpath**
/usr/local/man:/usr/local/share/man:/usr/share/man

info

info --> exibe uma documentação mais “navegavel” pelo terminal de vários comandos.

info comando --> exibe informações sobre o comando

/usr/share/doc e HowTo

Na pasta /usr/share/doc existe mais documentação sobre os programas. Para encontrar a documentação de um programa basta dar

cd /usr/share/doc/nomePrograma

howTo

Utilizando o comando
apt-cache search howto

podemos procurar por howtos (manuais) instaláveis que vão ser instalado no /usr/share/doc

No site https://www.tldp.org/ (The Linux Documention Project) é possível encontrar uma vasta documentação e consequentemente vários howtos.

ARQUIVOS e DIRETÓRIOS

Nomeação de Arquivos – Regras

Obs1: A nomeação de arquivos é caSe SenSitiVE.

- não pode ter / no nome do arquivos
- Como a nomeação é case sensitive, evite criar arquivos que tenho o mesmo nome, onde a única diferença seja o case das letras.
- Nomes de pastas costumam ser em letras minúsculas.
- De preferência utilizar os caracteres 0-9, a-z, A-Z, +, -, _ , . , na nomeação de arquivos.
- evitar criar arquivos com espaço no nome.
- não usar +, -, _ , no começo do nome de um arquivo.

Comandos uteis

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 - --> muda para a pasta anterior em que o usuário acessou no terminal.

Compressão de Arquivos

tar

tar -cf arquivo.tar * --> cria um arquivo .tar (comprimido) que contém todos os arquivos da pasta, dentro dela mesma.

tar -cvf arquivo.tar * --> cria um arquivo .tar (comprimido) que contém todos os arquivos da pasta, dentro dela mesma.,e exibe quais arquivos estão sendo comprimidos. Não comprime arquivos ocultos.

tar -cvf arquivo.tar pastaDestino --> cria um arquivo .tar (comprimido) que contém todos os arquivos da pastaDestino, dentro dela mesma.,e exibe quais arquivos estão sendo comprimidos. Comprime arquivos ocultos.

tar -xf arquivo.tar --> descompacta o arquivo arquivo.tar dentro da pasta.

tar -xvf arquivo.tar --> descompacta o arquivo arquivo.tar dentro da pasta, e exibe quais arquivos estão dentro do arquivos .tar

tar -tf arquivo.tar --> lista arquivos que estão compactados dentro de arquivo.tar.

tar -tvf arquivo.tar --> lista arquivos que estão compactados dentro de arquivo.tar com mais detalhes.

tar -uf PastaArquivo.tar arquivo1 arquivo2 --> atualiza o arquivo PastaArquivo.tar com as modificações, se ocorrerem, nos arquivos arquivo1 e arquivo2. Ao atualizar ele adiciona ao arquivo tar o arquivo modificado, não sobrescreve os arquivos.

tar -uvf PastaArquivo.tar arquivo1 arquivo2 --> atualiza o arquivo PastaArquivo.tar com as modificações, se ocorrerem, nos arquivos arquivo1 e arquivo2, e complementa com o função de verbose.

tar -czvf arquivo.tar.gz arquivo1 arquivo2 --> cria um arquivo .tar (comprimido) que contém os arquivos arquivo1 e arquivo2, e exibe quais arquivos estão sendo comprimidos. Também comprime o arquivo .tar no formato gzip (arquivo reduz de tamanho em relação ao .tar). Utilizando o verbose

tar -cjvf arquivo.tar.bz2 arquivo1 arquivo2 --> cria um arquivo .tar (comprimido) que contém os arquivos arquivo1 e arquivo2, e exibe quais arquivos estão sendo comprimidos. Também comprime o arquivo .tar no formato bzip2 (arquivo reduz de tamanho em relação ao .tar). Utilizando o verbose

tar -xzvf arquivo.tar.bz --> descompacta o arquivo arquivo.tar dentro da pasta, e exibe quais arquivos estão dentro do arquivos .tar

gzip

O gzip compactar apenas arquivo individuais e descompacta apenas arquivos individuais. Ao compactar e descompactar ele exclui o arquivo fonte.

gzip nomeArquivo --> compacta o arquivo com o gzip.

gzip -1 nomeArquivo --> compacta o arquivo com o gzip da forma mais rápida, mas perde qualidade de compactação. Aceita de -1 até -9, -9 é mais lento, mas a compactação é melhor.

gzip -v nomeArquivo --> compacta o arquivo com o gzip na opção de verbose.

gzip -l nomeArquivo --> lista arquivo que está compactado em nomeArquivo

gzip -d nomeArquivo --> descompacta o arquivo com gzip.

gunzip nomeArquivo.gz --> descompacta o arquivo com gzip.

gzip -c nomeArquivo --> envia saída do comando direto para o terminal. Também pode ser usado quando queremos enviar a saída do comando para alguma instrução. Não apaga o arquivo de origem.

gzip -c nomeArquivo > novoNomeArquivo.gz --> envia saída do comando bzip2 -c nomeArquivo para o arquivo nomeArquivo.gz. Esse é um modo de, ao criar um arquivo .bz, sem excluir o original.

bzip2

O bzip2 compactar apenas arquivos individuais e descompacta apenas arquivos individuais. Ao compactar e descompactar ele exclui o arquivo-fonte. Arquivos executáveis (scripts) que são compactados ou descompactados no Windows perdem a parte do "cabeçalho" que descreve que o arquivo tem permissão de execução.

bzi2 nomeArquivo --> compacta o arquivo com o bzip2.

bzip2 -1 nomeArquivo --> compacta o arquivo com o gzip da forma mais rápida, mas perde qualidade de compactação. Aceita de -1 até -9, -9 é mais lento, mas a compactação é melhor.

bzip2 -v nomeArquivo --> compacta o arquivo com o bzip na opção de verbose.

bzip2 -l nomeArquivo --> lista arquivo que está compactado em nomeArquivo

bzip2 -d nomeArquivo --> descompacta o arquivo com bzip2.

bunzip2 nomeArquivo.bz2 --> descompacta o arquivo com bzip2.

bzip2 -c nomeArquivo --> envia saída do comando direto para o terminal. Também pode ser usado quando queremos enviar a saída do comando para alguma instrução. Não apaga o arquivo de origem.

bzip2 -c nomeArquivo > novoNomeArquivo.bz2 --> envia saída do comando bzip2 -c nomeArquivo para o arquivo nomeArquivo.bz2. Esse é um modo de, ao criar um arquivo .bz2, sem excluir o original.

zip

Zip por padrão não adiciona a compacta o conteúdo das pastas recursivamente. Ao utilizar o zip para compacta não precisar utilizar a extensão .zip.

zip nomeArquivoDestino.zip arquivo1 arquivo2 --> compacta arquivo1 e arquivo2 em um arquivo .zip com nome de nomeArquivoDestino.zip.

zip nomeArquivo --> compacta o arquivo com o zip.

zip nomePasta.zip * --> compacta todos os arquivos da pasta para nomePasta.zip.

Obs: Não compacta recursivamente.

zip -r nomePasta.zip * --> compacta todos os arquivos recursivamente da pasta para nomePasta.zip.

zip -rv nomePasta.zip * --> compacta todos os arquivos recursivamente da pasta para nomePasta.zip com o verbose.

Compactar uma Pasta por completa --> voltar um diretório no mínimo e utilizar o comando:

zip -r nomeDiretorio.zip nomeDiretorio

unzip -l nomeArquivo.zip --> lista arquivos que estão compactados em nomeArquivo

unzip nomeArquivo --> descompacta o arquivo com zip na pasta atual.

unzip -q nomeArquivo --> descompacta o arquivo com o zip na pasta atual sem jogar informações no terminal.

unzip -v nomeArquivo --> descompacta o arquivo com o zip na pasta atual na opção de verbose.

unzip -d caminho/Diretorio nomeArquivo.zip --> descompacta o arquivo com o zip em um diretório específico. [rever comando e descrição]

unzip -d caminho/Diretorio nomeArquivo1 nomeArquivo2 diretorio/nomeArquivo3 --> descompacta somente arquivos internos específicos de um arquivo.zip. [rever comando e descrição]

Também podemos usar padrõe do shell e globing para descompactare compactar os arquivos com o zip:

unzip -d caminho/Diretorio nomeArquivo{1,2,3}.txt --> descompacta somente arquivos internos específicos (nomeArquivo1, nomeArquivo2, nomeArquivo3) de um arquivo.zip.

find -name “*.txt” | zip -@ nomeArquivo.zip --> procura todos os arquivos .txt e depois compacta todos no arquivo.zip

Manipulação de Arquivos de Textos

wc

wc --> ver wc

more

more --> ver more

less

less --> ver less

tail

tail --> ver tail

cut

considere o texto abaixo

```
>>> notasfiscais.posicional
0001guilherme 100,00
0002paulo      50,15
0003ana       110,24
0004carlos    300,00
0005guilherme 110,20
7658lucia     125.46
```

cut -c 1,4 notasfiscais.pososcional --> retorna uma lista com os 1ºs e 4ºs caracteres de cada linha

```
 --------------------------
|01                        |
|02                        |
|03                        |
|04                        |
|05                        |
|06                        |
|07                        |
|7658                      |
 --------------------------
```

cut -c 1-4 notasfiscais.pososcional –> retorna uma lista com os caracteres do 1 ao 4 de cada linha

```
 -------------------------
|0001                     |
|0002                     |
|0003                     |
|0004                     |
|0500                     |
|0006                     |
|7658l                    |
 -------------------------
```

cut -c 1-4,15-20 notasfiscais.pososcional –> retorna uma lista com os caracteres do 1 ao 4 e do 15 ao 20 de cada linha

```
 -------------------------
|0001100,00               |
|0002 50,00               |
|0003110,24               |
|0004300,29               |
|0500110,20               |
|7658125,46               |
 -------------------------
```

cut -c 1-4,15-20 –output-delimiter=”:” notasfiscais.pososcional –> retorna uma lista com os caracteres do 1 ao 4 e do 15 ao 20 de cada linha com utilização de um delimitador. O delimitador no caso é : .

```
 -------------------------
|0001:100,00              |
|0002: 50,00              |
|0003:110,24              |
|0004:300,29              |
|0500:110,20              |
|7658:125,46              |
 -------------------------
```

Redirecionamento de Entrada e Saída

stdin – entrada padrão – standard input

stout – saída padrão – standard output

stderr – saída padrão de erro – Standard error output

0> - saía de entrada
1> -saída padrão
2> - saída de erro

Saída

>

> --> indica saída

comando > arquivoDeSaída --> envia para o arquivoDeSaída a saída do comando.S

ls > listaDoLs.txt --> envia para o arquivo listaDoLs a saída do comando ls.Obs: somente se saída do ls for diferente de erro.

>>

>> --> adiciona algumaCoisa ao final do arquivo.
echo “linha adicional” >> nomeArq.tx --> adiciona uma linha no final do nomeArquivo, sem apagar conteúdo original, só acrescenta.

Canais 0> , 1> , 2>

0> --> canal de entrada
1> --> canal se saída padrão
2> --> canal de saída de erro

Exemplos de Uso

Obs: /6xvlxmv54546 não existe

- **cd /6xvlxmv54546 2> saidaDoCd**

  >>> Conteúdo do arquivo saidaDoCd

  bash: cd: /6xvlxmv54546: Arquivo ou diretório inexistente

- **cd /6xvlxmv54546 2>&1 > saidaDoCd** --> joga a saída do erro do cd para a saída padrão, no caso o terminal, e também joga a saída do erro no arquivo saidaDoCd.

  >>> Saída do terminal

  ```
  ------------------------------------------------------------
  |                                                        - X |
  ------------------------------------------------------------
  |  user@pc:~/Pasta$ cd /6xvlxmv54546 2>&1 > saidaDoCd       |
  |  bash: cd: /6xvlxmv54546: Arquivo ou diretório inexistente |
  |                                                           |
  |                                                           |
  ------------------------------------------------------------
  ```

  >>> Conteúdo do arquivo saidaDoCd

bash: cd: /6xvlxmv54546: Arquivo ou diretório inexistente

Entrada

< --> símbolo de entrada, indica a entrada

Exemplo de Uso

-Compactar arquivos que estão em uma lista em um arquivo de texto.

1) find -type f -name *.png > arquivos_a_compactar.txt --> procura por todos os arquivos que tenham a extensão .png e depois joga a lista desses arquivos no arquivos de texto arquivos_a_compactar.

2) zip -@ arquivos_compactados.zip < arquivos_a_compactar.txt --> zip compacta arquivos que estão listados em arquivos_a_compactar.txt em um arquivo zip chamado arquivos_compactados.zp. -@ é utilizado para instruir ao zip para esperar a inserção dos arquivos que irão ser compactados. Com a utilização do < em vez da entrada padrão, terminal, será usada como entrada padrão o arquivos_a_compactar.txt.

| (Pipe)

Considere o exemplo a seguir:

- **find -type f -name *.png | zip -@ arquivos_compactados.zip**

O | (pipe) está separando os dois comandos, isso significa que todos os comandos que estiverem a esquerda do | devem ser considerados a entrada padrão para o comando a direita do |.

Podemos utilizar vários | seguidos, assim sempre o que tiver a esquerda será a entrada e a direita a saída.

Exemplo:

- **find -type f -name *.png | zip -@ arquivos_compactados.zip | wc**

- **ls -la | less** --> resultado do ls -la é jogado para o edidor less, dessa forma, fica mais fácil visualizar e até pesquisar em um lista grande de arquivos.