

Curso de Shell Script - Módulo 1 (Comandos básicos)

Geofisicando (Rodolfo Dirack)

Vídeos:

https://www.youtube.com/watch?v=HRRfgufskaw&list=PLLCFxfe9wkl-k0w-c_1i4sdZPUYt0Yc2P

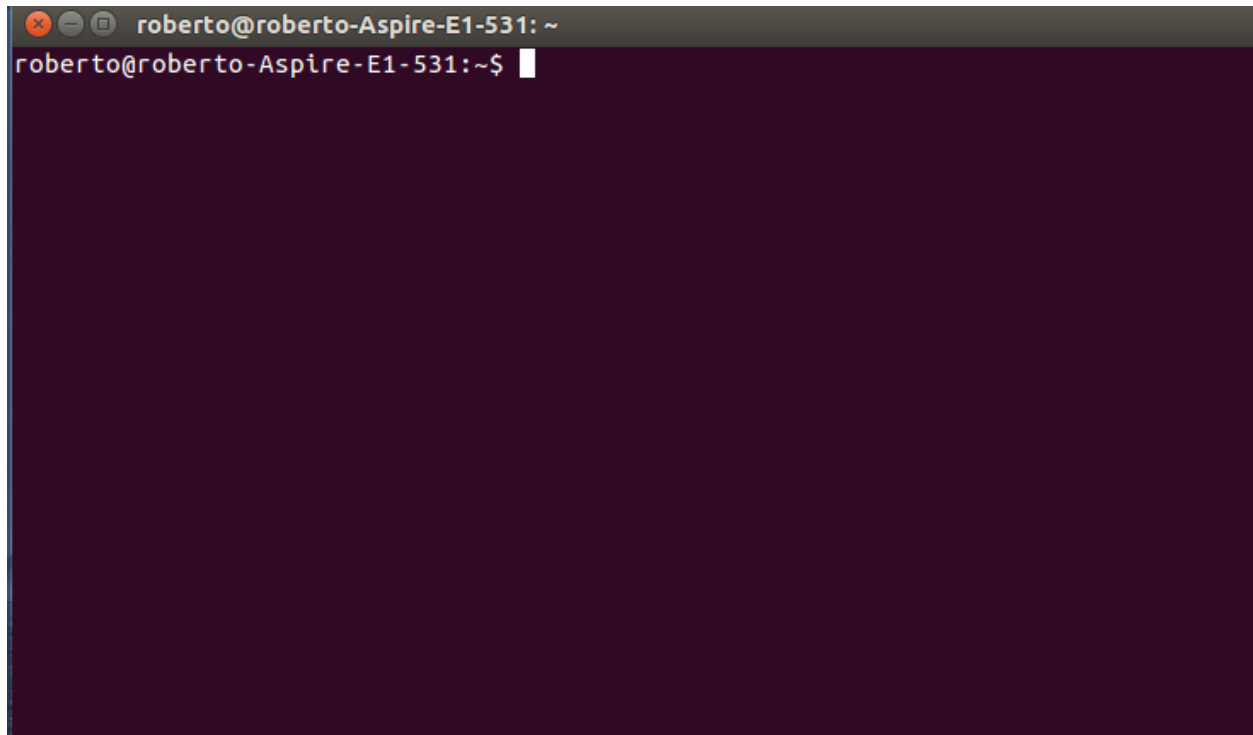
Resumo feito por Roberto Pinheiro

<https://github.com/betopinheiro1005>

Aula 01 - Terminal

Abrindo o terminal no Linux Ubuntu

- No Linux Ubuntu, para abrir um terminal pressione **<Ctrl><Alt><T>**



Fechando o terminal

Para fechar a tela do terminal, use o comando:

exit

Aula 02 - Comandos clear, reset e Ctrl + L

Estes comandos limpam a tela do terminal.

clear

O comando **clear** ou pressionar **<Ctrl><L>** limpa a tela.

- Na realidade, o comando só rola a tela para que a informação existente não apareça.

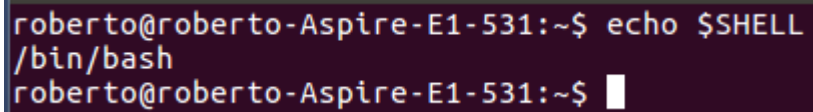
reset

O comando **reset**, de fato limpa a toda a tela.

Aula 03 - O que é shell e o que é shell script

Exibindo o nome do shell

echo \$SHELL



```
roberto@roberto-Aspire-E1-531:~$ echo $SHELL
/bin/bash
roberto@roberto-Aspire-E1-531:~$
```

bash é o nome do shell.

/bin/bash é o endereço do shell

- Existem vários tipos de shell

O que é o Shell

- Shell é o interpretador de comandos no terminal, além disso, o Shell verifica se não há erros de sintaxe na digitação dos comandos. Caso não aja erros o Shell executa o comando.

- O shell é uma interface que se comunica com a parte central do sistema: o **kernel**

O que é Shell Script

- O Shell entende uma linguagem específica chamada **Shell Script** que é uma linguagem com os comandos que iremos estudar durante o curso.

- Podemos utilizar essa linguagem como uma linguagem de programação.

- Shell Script é uma sequência de comandos que executa alguma tarefa.

Aula 04 - Comando pwd

- O comando **pwd** obtém o endereço do diretório atual no terminal.

pwd

```
roberto@roberto-Aspire-E1-531:~$ pwd
/home/roberto
roberto@roberto-Aspire-E1-531:~$ █
```

~ representa a pasta do usuário (**/home/betopinheiro**):

cd /
pwd

```
roberto@roberto-Aspire-E1-531:~$ cd /
roberto@roberto-Aspire-E1-531:/$ pwd
/
roberto@roberto-Aspire-E1-531:/$ █
```

/ é o diretório raiz, ou seja, contém todos os outros diretórios.

Aula 05 - Comando cd

- O comando **cd** (change directory) altera o diretório corrente no terminal.

cd ~ ou **cd \$HOME**
pwd

Vai para a pasta do usuário.

```
roberto@roberto-Aspire-E1-531:/$ cd $HOME
roberto@roberto-Aspire-E1-531:~$ pwd
/home/roberto
roberto@roberto-Aspire-E1-531:~$
```

cd .. ou **cd -**

Volta uma pasta (sobe um nível).

```
roberto@roberto-Aspire-E1-531:~$ cd ..
roberto@roberto-Aspire-E1-531:/home$ pwd
/home
roberto@roberto-Aspire-E1-531:/home$
```

cd /

```
roberto@roberto-Aspire-E1-531:/home$ cd /
roberto@roberto-Aspire-E1-531:/$ pwd
/
roberto@roberto-Aspire-E1-531:/$
```

Vai para o diretório raiz.

Aula 06 - Comando ls

- O comando **ls** exibe o conteúdo de um diretório no terminal.

ls

```
roberto@roberto-Aspire-E1-531:/$ ls
bin    dev    initrd.img    lib64    mnt    root    snap    tmp    vmlinuz
boot   etc    initrd.img.old    lost+found    opt    run    srv    usr    vmlinuz.old
cdrom  home  lib           media    proc   sbin    sys    var
roberto@roberto-Aspire-E1-531:/$
```

- Cada tipo de arquivo é exibido com uma cor diferente.

- em vermelho: arquivos compactados ou pacotes deb (debian)
- em roxo: arquivos de som
- em azul: diretórios
- em preto: arquivos comuns
- em azul claro: links
- em verde: arquivos executáveis (comandos de script do shell)

Exibindo o inode dos arquivos

ls -i

```
roberto@roberto-Aspire-E1-531:/$ ls -i
262145 bin          15 initrd.img.old      1 proc    1966081 tmp
3145729 boot        4325377 lib           3801089 root    2359297 usr
1310721 cdrom      2228225 lib64          2 run     2883585 var
2 dev           11 lost+found    5898241 sbin     16 vmlinuz
3670017 etc        2097153 media       6160385 snap    14 vmlinuz.old
393217 home     1048577 mnt          2490369 srv
12 initrd.img  2621441 opt           1 sys
roberto@roberto-Aspire-E1-531:/$
```

inode é um número de identificação de cada arquivo.

Listando arquivos com o caractere coringa (*)

cd /sbin

ls n*

```
roberto@roberto-Aspire-E1-531:/sbin$ ls n*
nameif ntfsclose ntfscp ntfslabel ntfsresize ntfsundelete
roberto@roberto-Aspire-E1-531:/sbin$
```

Listando também os arquivos ocultos

```
cd ~
```

```
ls -a
```

```
roberto@roberto-Aspire-E1-531:/sbin$ cd ~
roberto@roberto-Aspire-E1-531:~$ ls -a
.                .dmrc            .local           snap
..               Documentos      Modelos          .sudo_as_admin_successful
Área de Trabalho Downloads        .mozilla         Vídeos
.bash_history    examples.desktop Música            .wget-hsts
.bash_logout     flameshot.deb   .profile         .windows-serial
.bashrc          .gconf          Público          .wine
.cache           .gnupg          Release.key      .Xauthority
.compiz          .ICEauthority  Release.key.1    .xsession-errors
.config          Imagens        .script.sh.swp  .xsession-errors.old
roberto@roberto-Aspire-E1-531:~$
```

- No Linux arquivos ocultos começam com ponto.

- Em geral, são representados dessa maneira para que o usuário não altere esses arquivos (exemplo: arquivos de configuração ou de sistema).

Lista longa

- Exibe as permissões de cada arquivo.

```
ls -l
```

```
roberto@roberto-Aspire-E1-531:~$ ls -l
total 592
drwxr-xr-x 2 roberto roberto 4096 Jan 12 13:59 Área de Trabalho
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Documentos
drwxr-xr-x 3 roberto roberto 4096 Jan 12 07:56 Downloads
-rw-r--r-- 1 roberto roberto 8980 Jan 11 23:26 examples.desktop
-rw-rw-r-- 1 roberto roberto 548224 Dez  8 05:05 flameshot.deb
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Imagens
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Modelos
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Música
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Público
-rw-rw-r-- 1 roberto roberto 3122 Mar 28 2017 Release.key
-rw-rw-r-- 1 roberto roberto 3122 Mar 28 2017 Release.key.1
drwx----- 6 roberto roberto 4096 Jan 12 13:39 snap
drwxr-xr-x 2 roberto roberto 4096 Jan 11 23:44 Vídeos
roberto@roberto-Aspire-E1-531:~$
```

Listando os arquivos em uma única coluna

ls -l

```
roberto@roberto-Aspire-E1-531:~$ ls -l
Área de Trabalho
Documentos
Downloads
examples.desktop
flameshot.deb
Imagens
Modelos
Música
Público
Release.key
Release.key.1
snap
Vídeos
roberto@roberto-Aspire-E1-531:~$
```


Aula 07 - Comando echo

- O comando **echo** exibe mensagens na tela do terminal.

```
echo "Olá mundo shell! :)"
```

```
roberto@roberto-Aspire-E1-531:~$ echo "Olá mundo shell! :)"
Olá mundo shell! :)
roberto@roberto-Aspire-E1-531:~$
```

- Também pode ser utilizado para exibir variáveis.

```
echo $HOME
```

```
roberto@roberto-Aspire-E1-531:~$ echo $HOME
/home/roberto
roberto@roberto-Aspire-E1-531:~$
```

```
NOME="Rodolfo Dirack"
```

```
echo $NOME
```

```
roberto@roberto-Aspire-E1-531:~$ NOME="Rodolfo Dirack"
roberto@roberto-Aspire-E1-531:~$ echo $NOME
Rodolfo Dirack
roberto@roberto-Aspire-E1-531:~$
```

Interpretando caracteres especiais

```
echo -e "Rodolfo Dirack\n1\n2\n3"
```

```
roberto@roberto-Aspire-E1-531:~$ echo -e "Rodolfo Dirack\n1\n2\n3"
Rodolfo Dirack
1
2
3
roberto@roberto-Aspire-E1-531:~$
```

Direcionando um texto para um arquivo

```
echo "Roberto Pinheiro" >> usuarios.txt
```

```
echo "Rodolfo Dirack" >> usuarios.txt
```

```
cat usuarios.txt
```

```
roberto@roberto-Aspire-E1-531:~$ echo "Roberto Pinheiro" >> usuarios.txt
roberto@roberto-Aspire-E1-531:~$ echo "Rodolfo Dirack" >> usuarios.txt
roberto@roberto-Aspire-E1-531:~$ cat usuarios.txt
Roberto Pinheiro
Rodolfo Dirack
roberto@roberto-Aspire-E1-531:~$
```

>>

Se o arquivo não existe ele será criado, se existe o texto será adicionado no final do arquivo.

Aula 08 - Comandos chmod e ls -l

- O comando **chmod** permite alterar permissões de arquivos no Linux.

- Na pasta do usuário, crie 2 arquivos de texto vazios:

```
touch teste.txt teste2.txt
ls *.txt
```

```
roberto@roberto-Aspire-E1-531:~$ touch teste.txt teste2.txt
roberto@roberto-Aspire-E1-531:~$ ls *.txt
teste2.txt teste.txt usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

- O comando **ls -l** exibe a listagem longa (com as permissões) dos arquivos da pasta

```
ls -l *.txt
```

```
roberto@roberto-Aspire-E1-531:~$ ls -l *.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 14:55 teste2.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 14:55 teste.txt
-rw-rw-r-- 1 roberto roberto 32 Jan 12 14:48 usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

Permissões de arquivo

1. **r**: permissão de leitura
2. **w**: permissão de escrita
3. **x**: permissão de execução

- Crie um shell script com o seguinte conteúdo:

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

```
echo "Olá mundo shell! :)"
```

```
vi script.sh
```

<Insert>

```
#!/bin/bash
echo "Olá mundo shell! :)"
~
~
```

<Esc> :wq

Permissão de execução para o arquivo

ls -l script.sh

```
roberto@roberto-Aspire-E1-531:~$ ls -l script.sh
-rw-rw-r-- 1 roberto roberto 41 Jan 12 19:04 script.sh
roberto@roberto-Aspire-E1-531:~$
```

1. **Primeiro grupo de permissões:** para o usuário (usuário proprietário do arquivo).
2. **Segundo grupo de permissões:** para um grupo específico de usuários.
3. **Terceiro grupo de permissões:** para todos os demais usuários

./script.sh

```
roberto@roberto-Aspire-E1-531:~$ ./script.sh
bash: ./script.sh: Permissão negada
roberto@roberto-Aspire-E1-531:~$
```

Alterando as permissões do usuário proprietário do arquivo (u)

chmod u=rwx script.sh

ls -l script.sh

```
roberto@roberto-Aspire-E1-531:~$ chmod u=rwx script.sh
roberto@roberto-Aspire-E1-531:~$ ls -l script.sh
-rwxrw-r-- 1 roberto roberto 41 Jan 12 19:04 script.sh
roberto@roberto-Aspire-E1-531:~$
```

- Agora o usuário (proprietário do arquivo → betopinheiro) tem as três permissões (rwx) e portanto pode executar o shell script.

./script.sh

```
roberto@roberto-Aspire-E1-531:~$ ./script.sh
Olá mundo shell! :)
roberto@roberto-Aspire-E1-531:~$
```

Alterando as permissões do grupo (g)

```
chmod g=rw script.sh
```

```
ls -l script.sh
```

```
roberto@roberto-Aspire-E1-531:~$ chmod g=rw script.sh
roberto@roberto-Aspire-E1-531:~$ ls -l script.sh
-rwxrw-r-- 1 roberto roberto 41 Jan 12 19:04 script.sh
roberto@roberto-Aspire-E1-531:~$
```

- Agora o grupo tem permissão apenas para leitura e escrita.

Alterando as permissões para os demais usuários (o)

```
chmod o=r script.sh
```

```
ls -l script.sh
```

```
roberto@roberto-Aspire-E1-531:~$ chmod o=r script.sh
roberto@roberto-Aspire-E1-531:~$ ls -l script.sh
-rwxrw-r-- 1 roberto roberto 41 Jan 12 19:04 script.sh
roberto@roberto-Aspire-E1-531:~$
```

- Agora os demais usuários tem permissão apenas para leitura.

Resumindo

1. O usuário proprietário do arquivo pode ler, escrever ou executar o script.
2. O grupo pode ler ou escrever o arquivo, mas não pode executá-lo.
3. Todos os demais só podem ler o arquivo.

Aula 09 - Comandos basename e dirname

- Esses comandos são muito úteis dentro de um script.

Criando uma variável

```
ARQUIVO="/home/betopinheiro/teste.txt"
```

```
echo $ARQUIVO
```

```
roberto@roberto-Aspire-E1-531:~$ ARQUIVO="/home/betopinheiro/teste.txt"
roberto@roberto-Aspire-E1-531:~$ echo $ARQUIVO
/home/betopinheiro/teste.txt
roberto@roberto-Aspire-E1-531:~$
```

comando dirname

- O comando **dirname** exibe apenas o diretório do arquivo (ou seja: o caminho do arquivo).

```
dirname $ARQUIVO
```

```
roberto@roberto-Aspire-E1-531:~$ dirname $ARQUIVO
/home/betopinheiro
roberto@roberto-Aspire-E1-531:~$
```

comando basename

- O comando **basename** exibe apenas o nome do arquivo.

```
basename $ARQUIVO
```

```
roberto@roberto-Aspire-E1-531:~$ basename $ARQUIVO
teste.txt
roberto@roberto-Aspire-E1-531:~$
```

Exibindo apenas o nome do arquivo, sem a extensão

```
basename $ARQUIVO .txt
```

```
roberto@roberto-Aspire-E1-531:~$ basename $ARQUIVO .txt
teste
roberto@roberto-Aspire-E1-531:~$
```

Aula 10 - Comando history

- O comando **history** exibe o histórico de comandos.

Exibindo o histórico de comandos

history

```
roberto@roberto-Aspire-E1-531:~$ history
 1  echo $SHELL
 2  pwd
 3  cd /
 4  pwd
 5  cd $HOME
 6  pwd
 7  cd ..
 8  pwd
 9  cd /
10  ls
11  ls -i
12  cd /sbin
13  ls n*
14  cd ~
15  ls -a
16  ls -l
17  ls -1
18  echo "Olá mundo shell! :)"
19  echo $HOME
20  NOME="Rodolfo Dirack"
21  echo $NOME
22  echo -e "Rodolfo Dirack\n1\n2\n3"
23  echo "Roberto Pinheiro" >> usuarios.txt
24  echo "Rodolfo Dirack" >> usuarios.txt
25  cat usuarios.txt
26  touch teste.txt teste2.txt
27  ls *.txt
28  ls -l *.txt
29  vi script.sh
30  ls -l script.sh
31  ./script.sh
32  chmod u=rwx script.sh
33  ./script.sh
34  chmod g=rw script.sh
35  ls -l script.sh
36  chmod o=r script.sh
37  ls -l script.sh
38  ARQUIVO="/home/betopinheiro/teste.txt"
39  echo $ARQUIVO
40  dirname $ARQUIVO
41  basename $ARQUIVO
42  basename $ARQUIVO .txt
```

- O comando history armazena até 2000 comandos.

Executando um comando existente no history

Executando um comando específico do histórico

!28

```
roberto@roberto-Aspire-E1-531:~$ !28
ls -l *.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 19:36 teste2.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 19:36 teste.txt
-rw-rw-r-- 1 roberto roberto 32 Jan 12 19:35 usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

Executando o último comando do histórico

!!

```
roberto@roberto-Aspire-E1-531:~$ !!
ls -l *.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 19:36 teste2.txt
-rw-rw-r-- 1 roberto roberto  0 Jan 12 19:36 teste.txt
-rw-rw-r-- 1 roberto roberto 32 Jan 12 19:35 usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

Executando um último comando específico

!cat

```
roberto@roberto-Aspire-E1-531:~$ !cat
cat usuarios.txt
Roberto Pinheiro
Rodolfo Dirack
roberto@roberto-Aspire-E1-531:~$
```

- Executa o último comando **cat** encontrado no histórico de comandos.

Exibindo um determinado número de linhas de comandos do histórico

history | tail -5

```
roberto@roberto-Aspire-E1-531:~$ history | tail -5
 43  history
 44  ls -l *.txt
 45  clear
 46  cat usuarios.txt
 47  history | tail -5
roberto@roberto-Aspire-E1-531:~$
```


Direcionando a saída do comando para um arquivo

```
history | tail -5 >> historico.txt
```

```
cat historico.txt
```

```
roberto@roberto-Aspire-E1-531:~$ history | tail -5 >> historico.txt
roberto@roberto-Aspire-E1-531:~$ cat historico.txt
 44  ls -l *.txt
 45  clear
 46  cat usuarios.txt
 47  history | tail -5
 48  history | tail -5 >> historico.txt
roberto@roberto-Aspire-E1-531:~$
```

Limpando o histórico

```
history -c
```

Aula 11 - Comandos mkdir e mktemp

- Estes comandos permitem criar diretórios e arquivos temporários.

Criando um diretório

mkdir

O comando **mkdir** cria um diretório.

```
mkdir testes
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mkdir testes
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  flameshot.deb  Música          script.sh       teste.txt
Documentos        historico.txt  Público         snap            usuarios.txt
Downloads         Imagens       Release.key     teste2.txt      Vídeos
Examples.desktop  Modelos       Release.key.1   testes
roberto@roberto-Aspire-E1-531:~$
```

- Foi criado um diretório chamado "**testes**" vazio.

Criando um diretório dentro de outro

```
mkdir -p ./testes/testes1
```

```
cd testes
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mkdir -p ./testes/testes1
roberto@roberto-Aspire-E1-531:~$ cd testes
roberto@roberto-Aspire-E1-531:~/testes$ ls
testes1
roberto@roberto-Aspire-E1-531:~/testes$
```

ou:

```
ls testes
```

```
roberto@roberto-Aspire-E1-531:~$ ls testes
testes1
roberto@roberto-Aspire-E1-531:~$
```

Criando um arquivo temporário

mktemp

- O comando **mktemp** cria arquivos temporários

- Por padrão ele vai gerar os arquivos dentro de uma pasta chamada **/tmp** (pasta de arquivos temporários do shell)

mktemp

```
roberto@roberto-Aspire-E1-531:~$ mktemp  
/tmp/tmp.BcLmbPfdon  
roberto@roberto-Aspire-E1-531:~$
```

- O comando criou um arquivo com um nome aleatório (**tmp.BcLmbPfdon**) dentro da pasta **/tmp**

cd /tmp
ls tmp.*

```
roberto@roberto-Aspire-E1-531:~$ cd /tmp  
roberto@roberto-Aspire-E1-531:/tmp$ ls tmp.*  
tmp.BcLmbPfdon  tmp.tyshf13KkK  
roberto@roberto-Aspire-E1-531:/tmp$
```

Criando um arquivo temporário dentro da pasta atual

cd ~
cd testes/testes1
mktemp ./tmp.XXXX
ls

```
roberto@roberto-Aspire-E1-531:~$ cd testes/testes1  
roberto@roberto-Aspire-E1-531:~/testes/testes1$ mktemp ./tmp.XXXX  
./tmp.8Qde  
roberto@roberto-Aspire-E1-531:~/testes/testes1$ ls  
tmp.8Qde  
roberto@roberto-Aspire-E1-531:~/testes/testes1$
```

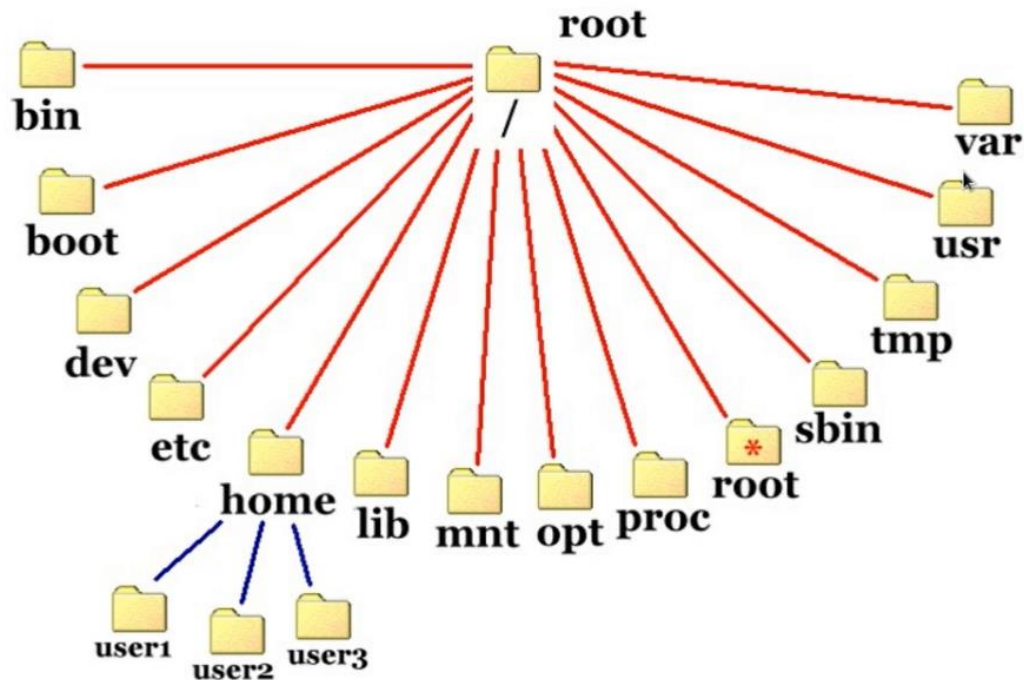
tmp.XXXX

XXXX é um placeholder, ou seja, o **mktemp** ao criar o arquivo temporário vai substituir XXXX por 4 caracteres **aleatórios** (para que os nomes dos arquivos não se repitam).

- Foi criado um arquivo temporário chamado **tmp.8Qde** dentro da pasta atual

Aula 12 - Árvore de diretórios no Linux e revisão dos comandos

Árvore de diretórios no Linux



- No diretório **raiz** se for digitado apenas o comando **cd**, o usuário é direcionado para a pasta **home**:

```
cd /  
pwd
```

```
roberto@roberto-Aspire-E1-531:~/testes/testes1$ cd /  
roberto@roberto-Aspire-E1-531:/$ pwd  
/  
roberto@roberto-Aspire-E1-531:/$
```

```
cd  
pwd
```

```
roberto@roberto-Aspire-E1-531:/$ cd  
roberto@roberto-Aspire-E1-531:~$ pwd  
/home/roberto  
roberto@roberto-Aspire-E1-531:~$
```

Aula 13 - Diretórios ponto e ponto ponto no Linux

`ls -a`

Exibe todos os arquivos do diretório, inclusive os ocultos (que iniciam com .)

```
betopinheiro@DESKTOP-GFBTDQD:~/testes$ ls -a
.  ..  script.sh  testes
```

Diretório atual

O diretório `.` é uma referência para o diretório atual.

`cd .`
`pwd`

```
roberto@roberto-Aspire-E1-531:~$ ls -a
.          .gnupg          .script.sh.swo
..         historico.txt   .script.sh.swp
Área de Trabalho .ICEauthority  snap
.bash_history  Imagens        .sudo_as_admin_successful
.bash_logout  .local         teste2.txt
.bashrc       Modelos        testes
.cache        .mozilla       teste.txt
.compiz       Música         usuarios.txt
.config       .profile       Vídeos
.dnrc         Público        .wget-hsts
Documentos    Release.key    .windows-serial
Downloads     Release.key.1  .wine
examples.desktop script.sh      .Xauthority
flameshot.deb .script.sh.swm .xsession-errors
.gconf        .script.sh.swn .xsession-errors.old
roberto@roberto-Aspire-E1-531:~$
```

Executando um shell script no diretório atual

`cat script.sh`

```
roberto@roberto-Aspire-E1-531:~$ cat script.sh
#!/bin/bash

echo "Olá mundo shell! :)"
roberto@roberto-Aspire-E1-531:~$
```

`./script.sh`

```
roberto@roberto-Aspire-E1-531:~$ ./script.sh
Olá mundo shell! :)
roberto@roberto-Aspire-E1-531:~$
```

- Equivale a digitar o comando:

`/home/roberto/script.sh`

```
roberto@roberto-Aspire-E1-531:~$ /home/roberto/script.sh
Olá mundo shell! :)
roberto@roberto-Aspire-E1-531:~$
```

Um diretório acima do atual

`cd ..`

```
roberto@roberto-Aspire-E1-531:~$ cd ..
roberto@roberto-Aspire-E1-531:/home$ pwd
/home
roberto@roberto-Aspire-E1-531:/home$
```

`..` se refere a um diretório acima

- Todas as pastas possuem os diretórios `.` e `..`

Aula 14 - Comando touch

O comando **touch** permite criar arquivos em branco e arquivos ocultos.

Criando um arquivo oculto

```
cd ~/testes  
touch .oculto.txt  
ls
```

```
roberto@roberto-Aspire-E1-531:/home$ cd ~/testes  
roberto@roberto-Aspire-E1-531:~/testes$ touch .oculto.txt  
roberto@roberto-Aspire-E1-531:~/testes$ ls  
testes1  
roberto@roberto-Aspire-E1-531:~/testes$ █
```

- Com o comando **ls**, arquivos ocultos não são exibidos.

Listando todos os arquivos, inclusive os ocultos

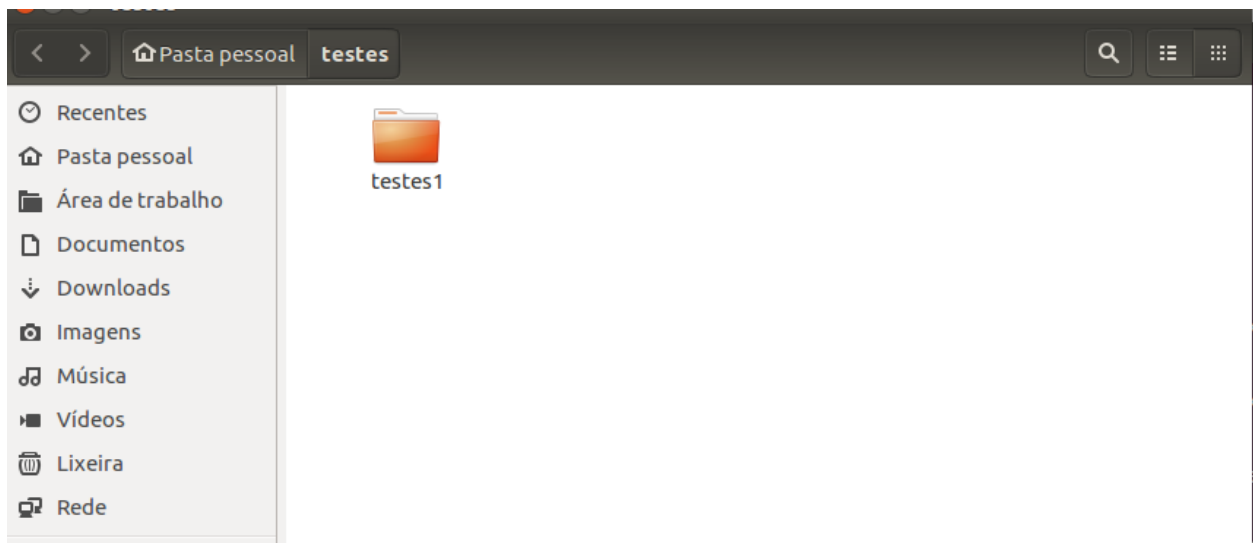
- Para exibir todos os arquivos, inclusive os ocultos, deve-se utilizar o comando:

```
ls -a
```

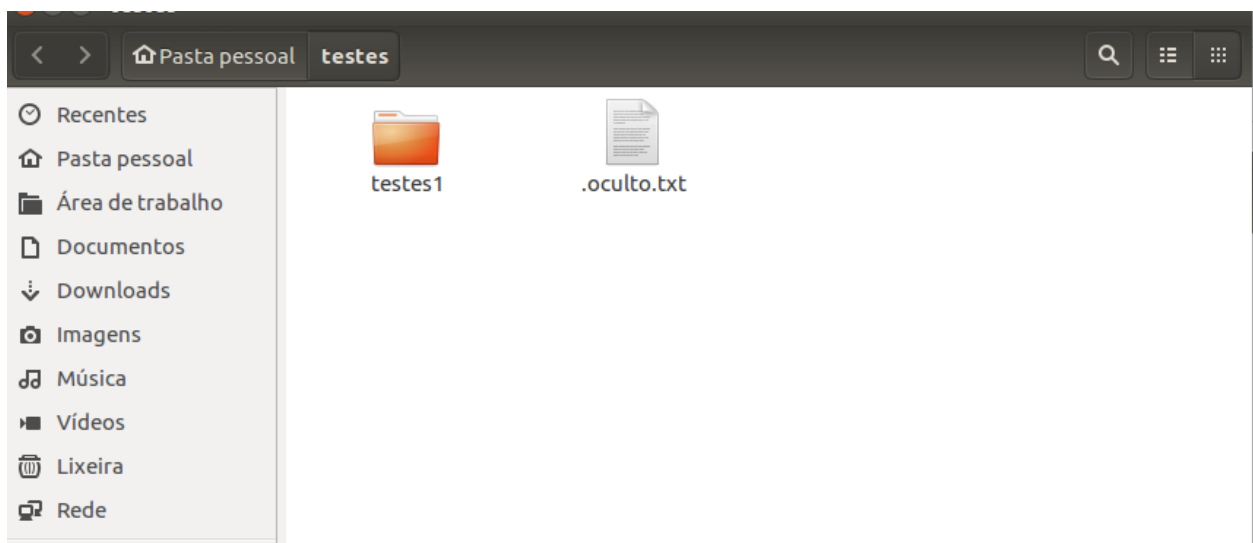
```
roberto@roberto-Aspire-E1-531:~/testes$ ls -a  
.  ..  .oculto.txt  testes1  
roberto@roberto-Aspire-E1-531:~/testes$ █
```

Abrindo o navegador do Linux (Nautilus)

nautilus . &



- Pressione as teclas <Ctrl><h>



- O arquivo oculto passa a ser exibido.

- Se for pressionado novamente as teclas <Ctrl><h> o arquivo oculto não será exibido. E assim sucessivamente.

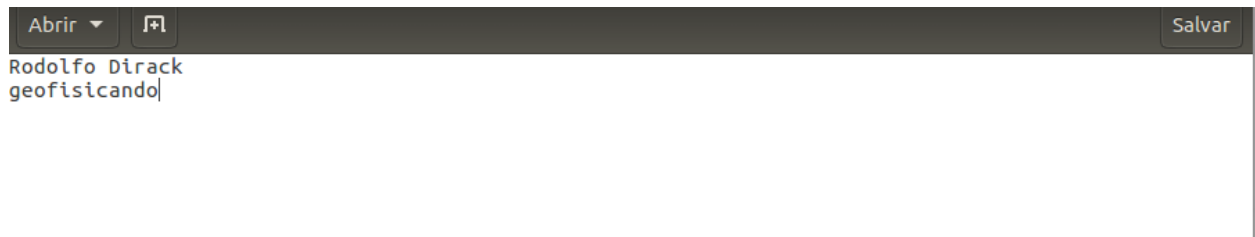
<Ctrl><h> habilita ou desabilita a exibição de arquivos ocultos.

- Abra o arquivo no editor **gedit**:

gedit .oculto.txt &

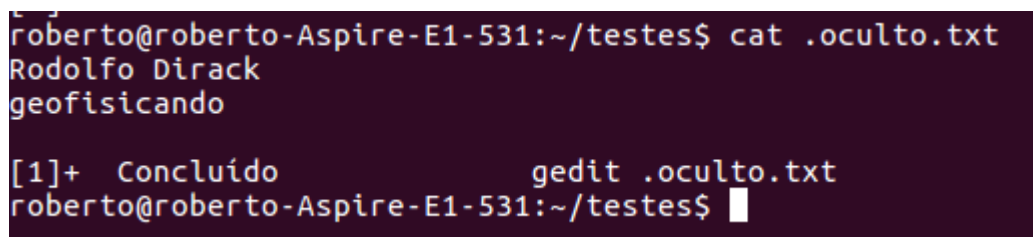
& é usado para não vincular o terminal com o editor de texto

Rodolfo Dirack
geofisicando



- Salve o arquivo e saia do editor.

cat .oculto.txt



Aula 15 - Comando mv

- O comando **mv** permite mover arquivos e diretórios pela linha de comando.

- Dentro da pasta de usuário, crie uma subpasta chamada **scripts**:

```
mkdir scripts
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mkdir scripts
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  flameshot.deb  Música          scripts        testes
Documentos        historico.txt  Público         script.sh      teste.txt
Downloads          Imagens       Release.key     snap           usuarios.txt
examples.desktop  Modelos       Release.key.1   teste2.txt     Vídeos
roberto@roberto-Aspire-E1-531:~$
```

Movendo arquivos

- Mova o arquivo **script.sh** para dentro dela, executando o comando:

```
mv script.sh ~/scripts
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  flameshot.deb  Música          scripts        teste.txt
Documentos        historico.txt  Público         snap           usuarios.txt
Downloads          Imagens       Release.key     teste2.txt     Vídeos
examples.desktop  Modelos       Release.key.1   testes
roberto@roberto-Aspire-E1-531:~$
```

```
cd scripts
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mv script.sh ~/scripts
roberto@roberto-Aspire-E1-531:~$ cd scripts
roberto@roberto-Aspire-E1-531:~/scripts$ ls
script.sh
roberto@roberto-Aspire-E1-531:~/scripts$
```

Movendo arquivos com a extensão txt para a pasta testes

```
ls *.txt
```

```
roberto@roberto-Aspire-E1-531:~$ ls *.txt
historico.txt  teste2.txt  teste.txt  usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

```
mv *.txt ~/testes
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mv *.txt ~/testes
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  examples.desktop  Modelos  Release.key  snap
Documentos        flameshot.deb     Música   Release.key.1 testes
Downloads         Imagens          Público  scripts      Vídeos
roberto@roberto-Aspire-E1-531:~$
```

```
cd testes
ls
```

```
roberto@roberto-Aspire-E1-531:~$ cd testes
roberto@roberto-Aspire-E1-531:~/testes$ ls
historico.txt  teste2.txt  testes1  teste.txt  usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes$
```

- Os arquivos `historico.txt`, `usuarios.txt`, `teste.txt` e `teste2.txt` foram movidos da pasta `~` para a pasta `~/testes`

Movendo pastas

```
cd ~
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes$ cd ~
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  examples.desktop  Modelos  Release.key  snap
Documentos        flameshot.deb     Música   Release.key.1 testes
Downloads         Imagens          Público  scripts      Vídeos
roberto@roberto-Aspire-E1-531:~$
```

- Mova a pasta **scripts** existente dentro da pasta de usuário para a pasta **testes**:

```
mv scripts/ testes/  
ls
```

```
roberto@roberto-Aspire-E1-531:~$ mv scripts/ testes/  
roberto@roberto-Aspire-E1-531:~$ ls  
Área de Trabalho  examples.desktop  Modelos  Release.key  testes  
Documentos        flameshot.deb     Música   Release.key.1 Vídeos  
Downloads         Imagens          Público  snap  
roberto@roberto-Aspire-E1-531:~$
```

```
cd testes  
ls
```

```
roberto@roberto-Aspire-E1-531:~$ cd testes  
roberto@roberto-Aspire-E1-531:~/testes$ ls  
historico.txt  scripts  teste2.txt  testes1  teste.txt  usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes$
```

Renomeando arquivo

- O comando **mv** também serve para renomear arquivos e/ou diretórios.

Renomeando arquivo teste.txt para teste1.txt

```
mv teste.txt teste1.txt
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes$ mv teste.txt teste1.txt  
roberto@roberto-Aspire-E1-531:~/testes$ ls  
historico.txt  scripts  teste1.txt  teste2.txt  testes1  usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes$
```

Renomeando diretório

Renomeando diretório testes1 para arquivos_txt

```
mv testes1/ arquivos_txt/
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes$ mv testes1/ arquivos_txt/
roberto@roberto-Aspire-E1-531:~/testes$ ls
arquivos_txt historico.txt scripts teste1.txt teste2.txt usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes$
```

- Mova os arquivos com extensão **txt** para a pasta **arquivos_txt**:

```
mv *.txt ./arquivos_txt
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes$ mv *.txt ./arquivos_txt
roberto@roberto-Aspire-E1-531:~/testes$ ls
arquivos_txt scripts
roberto@roberto-Aspire-E1-531:~/testes$
```

```
ls ./arquivos_txt
```

```
roberto@roberto-Aspire-E1-531:~/testes$ ls ./arquivos_txt
historico.txt teste1.txt teste2.txt tmp.8Qde usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes$
```

- Agora os arquivos com a extensão **txt** estão dentro da pasta **arquivos_txt**

Aula 16 - Comando cp

- O comando **cp** permite copiar arquivos e diretórios.

Copiando arquivos

- A partir do diretório de usuário, copie o arquivo **script.sh** existente dentro da pasta **~/testes/scripts** para a pasta atual (**~**):

```
cp ./testes/scripts/script.sh .
```

. representa o diretório atual

```
roberto@roberto-Aspire-E1-531:~$ cp ./testes/scripts/script.sh .
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  examples.desktop  Modelos  Release.key  snap
Documentos        flameshot.deb     Música   Release.key.1  testes
Downloads         Imagens          Público  script.sh      Vídeos
roberto@roberto-Aspire-E1-531:~$
```

Copiando diretórios

- Para copiar um diretório é necessário utilizar a opção **-r**:

- Utilizando essa opção, é feita a cópia do diretório com todos os seus arquivos.

Exemplo:

- Copie o diretório **arquivos_txt** existente dentro de **~/testes** para a pasta de usuário:

```
cp -r ~/testes/arquivos_txt ~
```

ls

```
roberto@roberto-Aspire-E1-531:~$ cp -r ~/testes/arquivos_txt ~
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  Downloads  Imagens  Público  script.sh  Vídeos
arquivos_txt      examples.desktop  Modelos  Release.key  snap
Documentos        flameshot.deb     Música   Release.key.1  testes
roberto@roberto-Aspire-E1-531:~$
```

Aula 17 - Comando cat

- Dentro da pasta de testes (~/.testes), crie uma pasta chamada testeCat:

```
mkdir testeCat; cd testeCat
```

O ponto-e-vírgula (;) permite executar mais de um comando simultaneamente no shell.

- Dentro dessa pasta, crie dois arquivos de texto: **arq1.txt** e **arq2.txt**:

```
echo "arquivo 1" >> arq1.txt  
echo "arquivo 2" >> arq2.txt
```

Exibindo o conteúdo dos arquivos

```
cat arq1.txt  
cat arq2.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat arq1.txt  
arquivo 1  
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat arq2.txt  
arquivo 2  
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```

Inserindo mais linhas nos arquivos

```
echo "este é o primeiro arquivo" >> arq1.txt  
echo "terceira linha" >> arq1.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ echo "este é o primeiro arquivo  
" >> arq1.txt  
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ echo "terceira linha" >> arq1.t  
xt  
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```

```
cat arq1.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat arq1.txt  
arquivo 1  
este é o primeiro arquivo  
terceira linha  
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```

```
echo "*****" >> arq2.txt
```

cat arq2.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat arq2.txt
arquivo 2
*****
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```

Concatenando arquivos em um novo arquivo

cat arq2.txt arq1.txt >> arq3.txt

cat arq3.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat arq3.txt
arquivo 2
*****
arquivo 1
este é o primeiro arquivo
terceira linha
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```

Exibindo o número de linhas de um arquivo

cat -n arq3.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cat -n arq3.txt
 1 arquivo 2
 2 *****
 3 arquivo 1
 4 este é o primeiro arquivo
 5 terceira linha
roberto@roberto-Aspire-E1-531:~/testes/testeCat$
```


Aula 18 - Comandos rm e rmdir

O comando `rm` remove arquivos.

O comando `rmdir` remove diretórios.

Removendo arquivos

ls

```
roberto@roberto-Aspire-E1-531:~/testes/testeCat$ cd ~
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  Downloads      Imagens  Público  script.sh  Vídeos
arquivos_txt      examples.desktop Modelos   Release.key snap
Documentos        flameshot.deb  Música   Release.key.1 testes
roberto@roberto-Aspire-E1-531:~$
```

- No diretório de usuário (~) remova o arquivo `script.sh`:

`rm script.sh`

ls

```
roberto@roberto-Aspire-E1-531:~$ rm script.sh
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  Downloads      Imagens  Público  snap
arquivos_txt      examples.desktop Modelos   Release.key testes
Documentos        flameshot.deb  Música   Release.key.1 Vídeos
roberto@roberto-Aspire-E1-531:~$
```

Remoção de arquivos mediante confirmação

Para solicitar confirmação de remoção utiliza-se a opção **-i**

`ls ~/testes/arquivos_txt`

```
roberto@roberto-Aspire-E1-531:~$ ls ~/testes/arquivos_txt
historico.txt teste1.txt teste2.txt tmp.8Qde usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

`rm -i ~/testes/arquivos_txt/tmp.8Qde`

`ls`

```
roberto@roberto-Aspire-E1-531:~$ rm -i ~/testes/arquivos_txt/tmp.8Qde
rm: remover arquivo comum vazio '/home/roberto/testes/arquivos_txt/tmp.8Qde'? n
roberto@roberto-Aspire-E1-531:~$ ls ~/testes/arquivos_txt
historico.txt teste1.txt teste2.txt tmp.8Qde usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

digitando **n** na confirmação, o arquivo não é removido.

`rm -i ~/testes/arquivos_txt/tmp.8Qde`

```
roberto@roberto-Aspire-E1-531:~$ rm -i ~/testes/arquivos_txt/tmp.8Qde
rm: remover arquivo comum vazio '/home/roberto/testes/arquivos_txt/tmp.8Qde'? y
```

`ls`

```
roberto@roberto-Aspire-E1-531:~$ ls ~/testes/arquivos_txt
historico.txt teste1.txt teste2.txt usuarios.txt
roberto@roberto-Aspire-E1-531:~$
```

digitando **y** na confirmação, o arquivo é removido.

Removendo diretórios

- O comando **rmdir** sem nenhuma opção remove um diretório apenas se ele estiver vazio.

```
rmdir arquivos_txt
```

```
roberto@roberto-Aspire-E1-531:~$ rmdir arquivos_txt
rmdir: falhou em remover 'arquivos_txt': Diretório não vazio
roberto@roberto-Aspire-E1-531:~$
```

- Para remover um diretório contendo arquivos deve-se usar **rmdir -r**. Neste caso, será removido o diretório com todos os seus arquivos.

- Para solicitar confirmação de exclusão de cada um dos arquivos existentes dentro de um diretório utiliza-se **rm -ir**

Removendo arquivos de um diretório com confirmação

No diretório de usuário, execute o comando:

```
rm -i arquivos_txt/*.txt
```

```
roberto@roberto-Aspire-E1-531:~$ rm -i arquivos_txt/*.txt
rm: remover arquivo comum 'arquivos_txt/historico.txt'? y
rm: remover arquivo comum vazio 'arquivos_txt/teste1.txt'? y
rm: remover arquivo comum vazio 'arquivos_txt/teste2.txt'? y
rm: remover arquivo comum 'arquivos_txt/usuarios.txt'? y
roberto@roberto-Aspire-E1-531:~$
```

```
ls
```

```
ls arquivos_txt
```

```
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  Downloads      Imagens  Público  snap
arquivos_txt      examples.desktop  Modelos  Release.key  testes
Documentos        flameshot.deb    Música   Release.key.1  Vídeos
roberto@roberto-Aspire-E1-531:~$ ls arquivos_txt
tmp.8Qde
roberto@roberto-Aspire-E1-531:~$
```

Observe que ainda existe um arquivo dentro da pasta.

Removendo o diretório

```
rm -r arquivos_txt
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~$ rm -r arquivos_txt
roberto@roberto-Aspire-E1-531:~$ ls
Área de Trabalho  examples.desktop  Modelos  Release.key  testes
Documentos        flameshot.deb     Música   Release.key.1 Vídeos
Downloads         Imagens          Público  snap
```

- O diretório `arquivos_txt` com o seu arquivo `tmp.8Qde` foram removidos.

Aula 19 - O que é um Pipe - Como redirecionar saídas dos comandos

O pipe (|) é utilizado para redirecionar a saída de um comando para outro comando.

Utilizando a calculadora do Linux

`bc -l`

a opção `-l` permite utilizar números decimais.

```
roberto@roberto-Aspire-E1-531:~$ bc -l
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
1+2.2
3.2
```

Redirecionando o cálculo que se deseja fazer para a calculadora

`echo "1 + 2.2" | bc -l`

```
roberto@roberto-Aspire-E1-531:~$ echo "1 + 2.2" | bc -l
3.2
roberto@roberto-Aspire-E1-531:~$
```

Redirecionamento com >> e >

>> e > são utilizados para redirecionar a saída.

Redirecionando com >>

- Se o arquivo não existe, ele será criado com o conteúdo informado.
- Se o arquivo já existe, será adicionada uma nova linha, com o conteúdo informado, no final do arquivo.

exemplo:

```
echo "Conteúdo da primeira linha" >> redirecionamento.txt
echo "Conteúdo da segunda linha" >> redirecionamento.txt
echo "Conteúdo da terceira linha" >> redirecionamento.txt
```

```
cat redirecionamento.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Conteúdo da primeira
linha" >> redirecionamento.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Conteúdo da segunda l
inha" >> redirecionamento.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Conteúdo da terceira
linha" >> redirecionamento.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat redirecionamento.txt
Conteúdo da primeira linha
Conteúdo da segunda linha
Conteúdo da terceira linha
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Redirecionando com >

- Se o arquivo não existe, ele será criado com o conteúdo informado.
- Se o arquivo já existe, seu conteúdo será substituído pelo conteúdo informado.

```
echo "conteúdo do arquivo" > redirecionamento.txt
```

```
cat redirecionamento.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "conteúdo do arquivo"
> redirecionamento.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat redirecionamento.txt
conteúdo do arquivo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Redirecionando com <

< é utilizado para redirecionar a entrada.

```
cat < redirecionamento.txt
```

Redireciona o conteúdo do arquivo `redirecionamento.txt` para o comando `cat` (para que seja exibido).

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat < redirecionamento.txt
conteúdo do arquivo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ █
```

Combinando redirecionamentos

```
echo "2+3" | bc -l >> redirecionamento.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "2+3" | bc -l >> redir
ecionamento.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat redirecionamento.txt
conteúdo do arquivo
5
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ █
```

Aula 20 - Comando cut

- O comando **cut** filtra saídas do comando **cat** e de arquivos **csv**

Filtrando saídas do comando cat

`echo "Dirack" | cut -c1`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Dirack" | cut -c1
D
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

`echo "Dirack" | cut -c2-4`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Dirack" | cut -c2-4
ira
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Selecionando colunas de um arquivo csv

`echo -e "1,2,3\n4,5,6\n7,8,9" >> matriz.csv`

`cat matriz.csv`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "1,2,3\n4,5,6\n7,8,9" >> matriz.csv
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat matriz.csv
1,2,3
4,5,6
7,8,9
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Selecionando a primeira colunas do arquivo

`cat matriz.csv | cut -c1`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat matriz.csv | cut -c1
1
4
7
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```


Selecionando a segunda coluna do arquivo

```
cat matriz.csv | cut -d"," -f2
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat matriz.csv | cut -d"," -f2
2
5
8
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Selecionando a segunda e terceira coluna do arquivo

```
cat matriz.csv | cut -d"," -f2-3
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat matriz.csv | cut -d"," -f2-3
2,3
5,6
8,9
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 21 - Comando head

O comando **head** exibe as primeiras linhas de um arquivo.

Exemplo:

- Para exibir as duas primeiras linhas do arquivo matriz.csv, execute o comando:

head -n 2 matriz.csv

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head -n 2 matriz.csv
1,2,3
4,5,6
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Por padrão, o comando head sem a flag -n exibe as dez primeiras linhas de um arquivo.

Exemplo:

history >> historico.txt

head historico.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ history >> historico.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head historico.txt
 1 history
 2 rm historico.txt
 3 ls
 4 cd ~
 5 ls
 6 clear
 7 ls
 8 head Release.key
 9 clear
10 head -n 5 Release.key Release.key.1
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Usando o comando head para exibir linhas de mais de um arquivo

`head -n 2 historico.txt matriz.csv`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head -n 2 historico.txt matriz.csv
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head -n 2 historico.txt <==
1 history
2 rm historico.txt

roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head -n 2 matriz.csv <==
1,2,3
4,5,6
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Redirecionamento

O comando **head** aceita redirecionamento.

Exemplo:

`echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27" | head -n 3`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27" | head -n 3
10 11 12
13 14 15
16 17 18
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 22 - Comando tail

O comando **tail** exibe as últimas linhas de um arquivo.

ls

```
echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27\n28 29 30\n31 32 33\n34 35 36"  
>> matriz.csv
```

cat matriz.csv

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls  
historico.txt  redirecionamento.txt  teste2.txt  
matriz.csv    teste1.txt             usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27\n28 29 30\n31 32 33\n34 35 36" >> matriz.csv  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat matriz.csv  
1,2,3  
4,5,6  
7,8,9  
10 11 12  
13 14 15  
16 17 18  
19 20 21  
22 23 24  
25 26 27  
28 29 30  
31 32 33  
34 35 36  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

head -n 5 matriz.csv

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head -n 5 matriz.csv  
1,2,3  
4,5,6  
7,8,9  
10 11 12  
13 14 15  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- O comando **tail** sem a flag **-n** exibe as dez últimas linhas do arquivo.

head matriz.csv

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ head matriz.csv
1,2,3
4,5,6
7,8,9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 26 27
28 29 30
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Redirecionamento

O comando **tail** também aceita redirecionamento.

Exemplo:

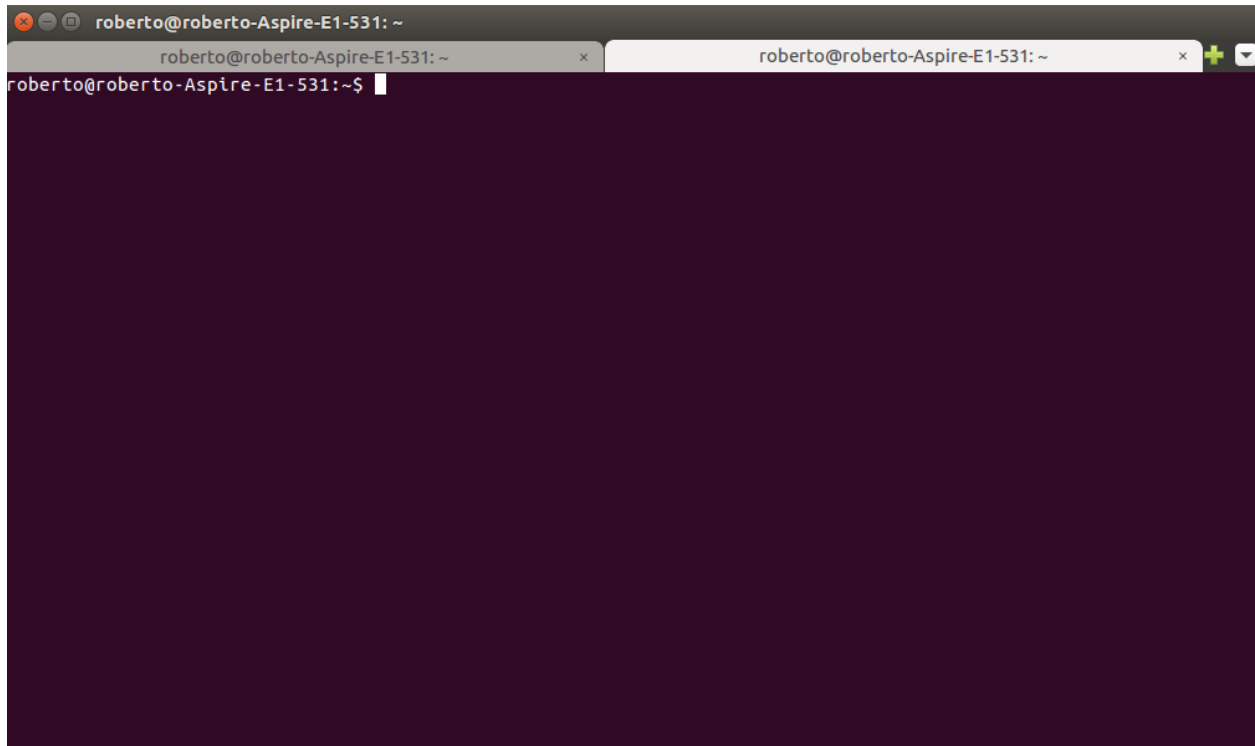
echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27" | tail -n 3

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "10 11 12\n13 14 15\n16 17 18\n19 20 21\n22 23 24\n25 26 27" | tail -n 3
19 20 21
22 23 24
25 26 27
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 23 - Comando tail -f

O comando **tail -f** permite monitorar arquivos.

- Abra duas abas do terminal, pressionando **<Ctrl><Alt><t>** para abrir a primeira tela do terminal e em seguida **<Ctrl><Shift><t>** para abrir uma nova aba do terminal.



- Para alternar entre as abas pressione **<Alt><1>** (para ir para a primeira aba) ou **<Alt><2>** (para ir para a segunda aba).

Na aba 1 (do lado esquerdo), vá para o diretório **~/testes/arquivos_txt** e execute o comando:

```
echo "Primeira linha" >> tail.txt; cat tail.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Primeira linha" >> tail.txt; cat tail.txt
Primeira linha
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Monitorando o arquivo tail.txt

- Na aba 2, vá para a pasta **~/testes/arquivos_txt** e execute o comando:

```
tail -f tail.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ tail -f tail.txt
Primeira linha
```

- Observe que será exibido o prompt piscando indicando que o arquivo **tail.txt** está sendo monitorado.

- Na **aba 1**, acrescente uma nova linha ao arquivo **tail.txt**, executando o comando:

```
echo "Segunda linha" >> tail.txt
```

- Na **aba 2**, será exibido:

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ tail -f tail.txt
Primeira linha
Segunda linha
```

- Na **aba 1**, execute o comando:

```
sleep 5; echo "Nova linha" >> tail.txt
```

- Voltando para a **aba 2** é possível monitorar que 5 segundos a entrada do comando, a nova linha será acrescentada ao arquivo **tail.txt**.

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ tail -f tail.txt
Primeira linha
Segunda linha
Nova linha
```

- Repita o mesmo comando na **aba 1**:

```
sleep 5; echo "Nova linha" >> tail.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Segunda linha" >> tail.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sleep 5; echo "Nova linha" >> tail.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sleep 5; echo "Nova linha" >> tail.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Voltando para a **aba 2** é possível monitorar que 5 segundos a entrada do comando, a nova linha será acrescentada ao arquivo **tail.txt**.

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ tail -f tail.txt
Primeira linha
Segunda linha
Nova linha
Nova linha
```

- Na **aba 2** para sair da monitoração do arquivo **tail.txt** pressione <Ctrl><c>

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ tail -f tail.txt
Primeira linha
Segunda linha
Nova linha
Nova linha
^C
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 24 - Comando seq

O comando **seq** exibe sequências de números e laços de repetição.

Exibindo uma sequência de números

O comando a seguir, exibe uma sequência de números de 0 a 10, com passo 2:

```
seq 0 2 10
```

```
roberto@roberto-Aspire-E1-531:~$ seq 0 2 10
0
2
4
6
8
10
roberto@roberto-Aspire-E1-531:~$
```

- Execute o comando:

```
for n in $(seq 0 1 5)
do
echo "valor de $n"
done
```

```
roberto@roberto-Aspire-E1-531:~$ for n in $(seq 0 1 5)
> do
> echo "valor de $n"
> done
valor de 0
valor de 1
valor de 2
valor de 3
valor de 4
valor de 5
```

Armazenando o resultado numa variável

```
lista=$(seq 0 1 5)
echo $lista
```

```
roberto@roberto-Aspire-E1-531:~$ lista=$(seq 0 1 5)
roberto@roberto-Aspire-E1-531:~$ echo $lista
0 1 2 3 4 5
roberto@roberto-Aspire-E1-531:~$
```


Usando redireccionamiento

```
seq 1 1 3 >> numeros.txt
```

```
cat numeros.txt
```

```
roberto@roberto-Aspire-E1-531:~$ seq 1 1 3 >> numeros.txt
roberto@roberto-Aspire-E1-531:~$ cat numeros.txt
1
2
3
roberto@roberto-Aspire-E1-531:~$ █
```

Aula 25 - Comando wc

O comando **wc** exibe número de linhas, palavras e caracteres de arquivo.

Exibindo o número de linhas, de palavras e de caracteres de um arquivo

```
echo -e "Linha 1\nLinha 2\nLinha 3" > texto.txt
```

```
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "Linha 1\nLinha 2\nLinha 3" > texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
Linha 1
Linha 2
Linha 3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
wc texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc texto.txt
 3  6 24 texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Ou seja, o arquivo **texto.txt** possui 3 linhas, 6 palavras e 24 caracteres.

Exibindo as informações separadamente

Número de linhas

```
wc -l texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc -l texto.txt
3 texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Número de palavras

```
wc -w texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc -w texto.txt
6 texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Número de caracteres

`wc -m texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc -m texto.txt
24 texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Redirecionamento

`wc -m texto.txt | cut -d" " -f1`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc -m texto.txt | cut -d" "
-f1
24
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

`wc -m texto.txt | cut -d" " -f2`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ wc -m texto.txt | cut -d" "
-f2
texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 26 - Comando tee

O comando **tee** gera arquivos de log das saídas de um pipe.

Criando um arquivo de log

```
echo "1,2,3" | tee log.txt | wc
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "1,2,3" | tee log.txt  
| wc  
      1      1      6  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
cat log.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat log.txt  
1,2,3  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
echo "5,6,7" | tee log.txt | wc
```

```
cat log.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "5,6,7" | tee log.txt  
| wc  
      1      1      6  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat log.txt  
5,6,7  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Como o arquivo **log.txt** já existia ele foi sobrescrito.

Adicionando linhas ao arquivo de log ao invés de sobrescrevê-lo

- Para isso usa-se a flag **-a**:

```
echo "1,2,3" | tee -a log.txt | wc
```

```
cat log.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "1,2,3" | tee -a log.t
xt | wc
      1      1      6
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat log.txt
5,6,7
1,2,3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
echo "Nova linha" | tee -a log.txt | wc
```

```
cat log.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "Nova linha" | tee -a
log.txt | wc
      1      2     11
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat log.txt
5,6,7
1,2,3
Nova linha
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 27 - Comando date (Parte 1)

- O comando **date** exibe data e hora local em formato específico.

date

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date
Sex Jan 14 20:22:07 -03 2022
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo a zona de tempo UTC

UTC é a zona de tempo padrão, sem correção.

date -u

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date -u
Sex Jan 14 23:24:52 UTC 2022
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo a data de ontem, de hoje e de amanhã

date -d yesterday; date -d today; date -d tomorrow;

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date -d yesterday; date -d
today; date -d tomorrow;
Qui Jan 13 20:32:26 -03 2022
Sex Jan 14 20:32:26 -03 2022
Sáb Jan 15 20:32:26 -03 2022
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo datas específicas

Dois dias atrás

date -d "2 days ago"

Cinco semanas adiante

```
date -d "5 weeks"
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date -d "5 weeks"
Sex Feb 18 20:37:09 -03 2022
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Usando date com descritores de formato

```
date +%d/%m/%Y
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date +%d/%m/%Y
14/01/2022
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
date +%H:%M:%S
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date +%H:%M:%S
20:45:21
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo o nome do dia

Abreviado

```
date +%a
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date +%a
Sex
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Completo

```
date +%A
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ date +%A
sexta
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 28 - Comando date (Parte 2)

Descritores de formato

Representando o caractere literal %

"%" é um metacaractere. É um caractere especial para o comando **date** que indica que será passado um descritor de formato.

Para utilizar "%" na forma literal, usar "%%":

date +%d%%m%%Y

```
roberto@roberto-Aspire-E1-531:~$ date +%d%%m%%Y
15%01%2022
roberto@roberto-Aspire-E1-531:~$
```

Utilizando tabulação

Utiliza-se **%t**

date +%d%t%m%t%Y

```
roberto@roberto-Aspire-E1-531:~$ date +%d%t%m%t%Y
15      01      2022
roberto@roberto-Aspire-E1-531:~$
```

Utilizando quebra de linha

Utiliza-se **%n**

date +%d%n%m%n%Y

```
roberto@roberto-Aspire-E1-531:~$ date +%d%n%m%n%Y
15
01
2022
roberto@roberto-Aspire-E1-531:~$
```


Exibindo o nome do mês

Abreviado

date +%b

```
roberto@roberto-Aspire-E1-531:~$ date +%b
Jan
roberto@roberto-Aspire-E1-531:~$
```

Completo

date +%B

```
roberto@roberto-Aspire-E1-531:~$ date +%B
janeiro
roberto@roberto-Aspire-E1-531:~$
```

Exibindo o número do dia do ano

date +%j

```
roberto@roberto-Aspire-E1-531:~$ date +%j
015
roberto@roberto-Aspire-E1-531:~$
```

Exibindo o número de segundos desde 01 de janeiro de 1970

date +%s

```
roberto@roberto-Aspire-E1-531:~$ date +%s
1642246231
roberto@roberto-Aspire-E1-531:~$
```

Aula 29 - Comando rev

- O comando **rev** reverte sequências de caracteres

rev

```
roberto@roberto-Aspire-E1-531:~$ rev
rodolfo dirack
kcarid oflodor
1,2,3,4,5,6
6,5,4,3,2,1
```

- Para sair: **<Ctrl><c>**

Obtendo o conteúdo do último campo

Os campos são: nome, idade, email, senha

echo "rodolfo,27,rodolfo@gmail.com,123456" > users.txt

cat users.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "rodolfo,27,rodolfo@gmail.com,123456" > users.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat users.txt
rodolfo,27,rodolfo@gmail.com,123456
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Obtendo o valor do campo senha

cat users.txt | cut -d"," -f4

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat users.txt | cut -d"," -f4
123456
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- O campo senha é sempre o último. Mas se for inserido um novo campo neste arquivo, o comando anterior não mais exibirá a senha e sim o email.

echo "rodolfo,dirack,27,rodolfo@gmail.com,123456" > users.txt

`cat users.txt | cut -d"," -f4`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat users.txt | cut -d"," -f4
rodolfo@gmail.com
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Para resolver esse problema podemos utilizar o comando "**rev**"

`cat users.txt | rev | cut -d"," -f1 | rev`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat users.txt | rev | cut -d"," -f
1 | rev
123456
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Dessa forma mesmo se colunas forem acrescentadas ou removidas, a senha será exibida.

Aula 30 - Comando paste

- O comando **paste** concatena arquivos por colunas.

Concatenando arquivos por linhas

- O comando **cat** concatena arquivos por linhas.

```
echo "roberto,pinheiro,60,roberto@gmail.com,789012" > users2.txt
```

```
cat users.txt users2.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat users.txt users2.txt
rodolfo,dirack,27,rodolfo@gmail.com,123456
roberto,pinheiro,60,roberto@gmail.com,789012
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Concatenando arquivos por colunas

- O comando **paste** concatena arquivos por colunas.

```
paste users.txt users2.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ paste users.txt users2.txt
rodolfo,dirack,27,rodolfo@gmail.com,123456      roberto,pinheiro,60,roberto@gmail.com,789012
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Alterando o caractere de separação de colunas

```
paste users.txt users2.txt -d:
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ paste users.txt users2.txt -d:
rodolfo,dirack,27,rodolfo@gmail.com,123456:roberto,pinheiro,60,roberto@gmail.com,789012
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Na variável de ambiente **\$PATH**, a separação é feita dessa forma:

```
echo $PATH
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- O delimitador pode ser "\n". Nesse caso o comando paste se comportará da mesma forma que o comando cat, ou seja, exibirá a concatenação por linhas.

```
paste users.txt users2.txt -d"\n"
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ paste users.txt users2.txt -d"\n"
rodolfo,dirack,27,rodolfo@gmail.com,123456
roberto,pinheiro,60,roberto@gmail.com,789012
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Armazenando a concatenação em um novo arquivo

```
paste users.txt users2.txt -d"\n" > cadastro.txt
```

```
ls
```

```
cat cadastro.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ paste users.txt users2.txt -d"\n" > cadastro.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls
cadastro.txt  log.txt      redirecionamento.txt  teste1.txt  texto.txt  users.txt
historico.txt matriz.csv   tail.txt              teste2.txt  users2.txt usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat cadastro.txt
rodolfo,dirack,27,rodolfo@gmail.com,123456
roberto,pinheiro,60,roberto@gmail.com,789012
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 31 - Comando cal

- O comando **cal** exibe o calendário na tela do terminal.

Exibindo o calendário do mês atual

cal

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cal
      Janeiro 2022
Su Mo Tu We Th Fr Sa
    1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo o calendário do mês atual, o mês anterior e o mês próximo

cal -3

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cal -3
      Dezembro 2021      Janeiro 2022      Fevereiro 2022
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
    1  2  3  4          1  2  3  4  5          1  2  3  4  5
 5  6  7  8  9 10 11  2  3  4  5  6  7  8  6  7  8  9 10 11 12
12 13 14 15 16 17 18  9 10 11 12 13 14 15 13 14 15 16 17 18 19
19 20 21 22 23 24 25 16 17 18 19 20 21 22 20 21 22 23 24 25 26
26 27 28 29 30 31     23 24 25 26 27 28 29 27 28
                        30 31
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Exibe o mês atual, um mês antes e um mês depois.

Exibindo o calendário completo do ano atual

cal -y

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cal -y
2022
    Janeiro                Fevereiro                Março
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
      1      1 2 3 4 5      1 2 3 4 5      1 2 3 4 5
  2 3 4 5 6 7 8    6 7 8 9 10 11 12    6 7 8 9 10 11 12
 9 10 11 12 13 14 15 13 14 15 16 17 18 19    13 14 15 16 17 18 19
16 17 18 19 20 21 22 20 21 22 23 24 25 26    20 21 22 23 24 25 26
23 24 25 26 27 28 29 27 28                27 28 29 30 31
30 31

    Abril                Maio                Junho
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
      1 2      1 2 3 4 5 6 7      1 2 3 4
  3 4 5 6 7 8 9    8 9 10 11 12 13 14    5 6 7 8 9 10 11
10 11 12 13 14 15 16 15 16 17 18 19 20 21    12 13 14 15 16 17 18
17 18 19 20 21 22 23 22 23 24 25 26 27 28    19 20 21 22 23 24 25
24 25 26 27 28 29 30 29 30 31                26 27 28 29 30

    Julho                Agosto                Setembro
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
      1 2      1 2 3 4 5 6      1 2 3
  3 4 5 6 7 8 9    7 8 9 10 11 12 13    4 5 6 7 8 9 10
10 11 12 13 14 15 16 14 15 16 17 18 19 20    11 12 13 14 15 16 17
17 18 19 20 21 22 23 21 22 23 24 25 26 27    18 19 20 21 22 23 24
24 25 26 27 28 29 30 28 29 30 31                25 26 27 28 29 30
31

    Outubro                Novembro                Dezembro
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
      1      1 2 3 4 5      1 2 3
  2 3 4 5 6 7 8    6 7 8 9 10 11 12    4 5 6 7 8 9 10
 9 10 11 12 13 14 15 13 14 15 16 17 18 19    11 12 13 14 15 16 17
16 17 18 19 20 21 22 20 21 22 23 24 25 26    18 19 20 21 22 23 24
23 24 25 26 27 28 29 27 28 29 30                25 26 27 28 29 30 31
30 31
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ █
```

Exibindo o calendário de um ano específico

cal -y 2023

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cal -y 2023
2023
    Janeiro                     Fevereiro                     Março
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
 1  2  3  4  5  6  7      1  2  3  4      1  2  3  4
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    5  6  7  8  9 10 11
15 16 17 18 19 20 21    12 13 14 15 16 17 18    12 13 14 15 16 17 18
22 23 24 25 26 27 28    19 20 21 22 23 24 25    19 20 21 22 23 24 25
29 30 31                26 27 28                26 27 28 29 30 31

    Abril                      Maio                      Junho
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
                1      1  2  3  4  5  6      1  2  3
 2  3  4  5  6  7  8    7  8  9 10 11 12 13    4  5  6  7  8  9 10
 9 10 11 12 13 14 15    14 15 16 17 18 19 20    11 12 13 14 15 16 17
16 17 18 19 20 21 22    21 22 23 24 25 26 27    18 19 20 21 22 23 24
23 24 25 26 27 28 29    28 29 30 31            25 26 27 28 29 30
30

    Julho                      Agosto                      Setembro
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
                1      1  2  3  4  5      1  2
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    3  4  5  6  7  8  9
 9 10 11 12 13 14 15    13 14 15 16 17 18 19    10 11 12 13 14 15 16
16 17 18 19 20 21 22    20 21 22 23 24 25 26    17 18 19 20 21 22 23
23 24 25 26 27 28 29    27 28 29 30 31          24 25 26 27 28 29 30
30 31

    Outubro                    Novembro                    Dezembro
Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá Do Se Te Qu Qu Se Sá
 1  2  3  4  5  6  7      1  2  3  4      1  2
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    3  4  5  6  7  8  9
15 16 17 18 19 20 21    12 13 14 15 16 17 18    10 11 12 13 14 15 16
22 23 24 25 26 27 28    19 20 21 22 23 24 25    17 18 19 20 21 22 23
29 30 31                26 27 28 29 30          24 25 26 27 28 29 30
31

roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo um mês do ano atual

cal -m 5

```
roberto@roberto-Aspire-E1-531:~$ cal -m 5
    Maio 2022
Do Se Te Qu Qu Se Sá
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

roberto@roberto-Aspire-E1-531:~$
```


Consultando as opções do comando cal

man cal

```
CAT(1) User Commands
NAME
  cat - concatenate files and print on the standard output

SYNOPSIS
  cat [OPTION]... [FILE]...

DESCRIPTION
  Concatenate FILE(s) to standard output.

  With no FILE, or when FILE is -, read standard input.

  -A, --show-all
      equivalent to -vET

  -b, --number-nonblank
      number nonempty output lines, overrides -n

  -e
      equivalent to -vE

  -E, --show-ends
      display $ at end of each line

  -n, --number
      number all output lines

  -s, --squeeze-blank
      suppress repeated empty output lines

  -t
      equivalent to -vT

  -T, --show-tabs
      display TAB characters as ^I

  -u
      (ignored)

  -v, --show-nonprinting
      use ^ and M- notation, except for LFD and TAB

  --help display this help and exit
```

```
--version
    output version information and exit

EXAMPLES
  cat f - g
    Output f's contents, then standard input, then g's contents.

  cat
    Copy standard input to standard output.

AUTHOR
  Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS
  GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
  Report cat translation bugs to <http://translationproject.org/team/>

COPYRIGHT
  Copyright © 2016 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
  This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO
  tac(1)

  Full documentation at: <http://www.gnu.org/software/coreutils/cat>
  or available locally via: info '(coreutils) cat invocation'

GNU coreutils 8.25 February 2017 CAT(1)
Manual page cat(1) line 28/69 (END) (press h for help or q to quit)
```

- Pressione <q> para sair do manual;

Aula 32 - Comando sort (Parte 1) - Organizar lista em ordem alfabética e numérica

- O comando **sort** organiza lista em ordem alfabética e numérica.

Organizando em ordem alfabética

```
echo -e "um\ndois\ntrês" > numeros.txt
cat numeros.txt
sort numeros.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "um\ndois\ntrês" > numeros.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat numeros.txt
um
dois
três
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort numeros.txt
dois
três
um
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo ocorrências únicas

```
echo "um" >> numeros.txt
sort numeros.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort numeros.txt
dois
três
um
um
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

"um" é exibido duas vezes.

- Para que não sejam exibidas opções duplicatas, utiliza-se a flag **-u**:

```
sort -u numeros.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -u numeros.txt
dois
três
um
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 33 - Comando sort (Parte 2) - Organizar lista por coluna e na ordem inversa

Redirecionando a saída de um comando sort para um novo arquivo

O redirecionamento pode ser feito com a flag **-o**:

```
rm numeros2.txt
```

```
echo -e "2\n1\n4\n3" >> numeros2.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "2\n1\n4\n3" >> numeros2.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat numeros2.txt
2
1
4
3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
paste numeros.txt numeros2.txt >> numeros3.txt
```

```
cat numeros3.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ paste numeros.txt numeros2.txt >> numeros3.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat numeros3.txt
um      2
dois    1
três    4
um      3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Ordenando pela coluna 2

```
sort -k2 numeros3.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -k2 numeros3.txt
dois    1
um      2
um      3
três    4
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Ordenando pela coluna 1

```
sort -k1 numeros3.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -k1 numeros3.txt
dois      1
três      4
um        2
um        3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Ordenando em ordem decrescente

Por padrão, a ordenação é feita pela primeira coluna:

```
sort -r numeros3.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -r numeros3.txt
um        3
um        2
três      4
dois      1
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Ordenando em ordem decrescente pela segunda coluna

```
sort -r numeros3.txt -k2
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -r numeros3.txt -k2
três      4
um        3
um        2
dois      1
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 34 - Comando sort (Parte 3) - Organizar lista grande de arquivos por coluna

cat lista1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat lista1.txt
creStackedTrace-m0-1.rsfc
creStackedTrace-m0-2.rsfc
creStackedTrace-m0-3.rsfc
creStackedTrace-m0-4.rsfc
creStackedTrace-m0-5.rsfc
creStackedTrace-m0-6.rsfc
creStackedTrace-m0-7.rsfc
creStackedTrace-m0-8.rsfc
creStackedTrace-m0-9.rsfc
creStackedTrace-m0-10.rsfc
creStackedTrace-m0-11.rsfc
creStackedTrace-m0-12.rsfc
creStackedTrace-m0-13.rsfc
creStackedTrace-m0-14.rsfc
creStackedTrace-m0-15.rsfc
creStackedTrace-m0-16.rsfc
creStackedTrace-m0-17.rsfc
creStackedTrace-m0-18.rsfc
creStackedTrace-m0-19.rsfc
creStackedTrace-m0-20.rsfc
creStackedTrace-m0-21.rsfc
creStackedTrace-m0-22.rsfc
creStackedTrace-m0-23.rsfc
creStackedTrace-m0-24.rsfc
creStackedTrace-m0-25.rsfc
creStackedTrace-m0-26.rsfc
creStackedTrace-m0-27.rsfc
creStackedTrace-m0-28.rsfc
creStackedTrace-m0-29.rsfc
creStackedTrace-m0-30.rsfc
```

sort lista1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort lista1.txt
creStackedTrace-m0-10.rsf
creStackedTrace-m0-11.rsf
creStackedTrace-m0-12.rsf
creStackedTrace-m0-13.rsf
creStackedTrace-m0-14.rsf
creStackedTrace-m0-15.rsf
creStackedTrace-m0-16.rsf
creStackedTrace-m0-17.rsf
creStackedTrace-m0-18.rsf
creStackedTrace-m0-19.rsf
creStackedTrace-m0-1.rsf
creStackedTrace-m0-20.rsf
creStackedTrace-m0-21.rsf
creStackedTrace-m0-22.rsf
creStackedTrace-m0-23.rsf
creStackedTrace-m0-24.rsf
creStackedTrace-m0-25.rsf
creStackedTrace-m0-26.rsf
creStackedTrace-m0-27.rsf
creStackedTrace-m0-28.rsf
creStackedTrace-m0-29.rsf
creStackedTrace-m0-2.rsf
creStackedTrace-m0-30.rsf
creStackedTrace-m0-3.rsf
creStackedTrace-m0-4.rsf
creStackedTrace-m0-5.rsf
creStackedTrace-m0-6.rsf
creStackedTrace-m0-7.rsf
creStackedTrace-m0-8.rsf
creStackedTrace-m0-9.rsf
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- A ordenação não é feita como o desejado. Para isso execute o comando:

```
sort -t"-" -k3 -n lista1.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sort -t"-" -k3 -n lista1.txt
creStackedTrace-m0-1.rsf
creStackedTrace-m0-2.rsf
creStackedTrace-m0-3.rsf
creStackedTrace-m0-4.rsf
creStackedTrace-m0-5.rsf
creStackedTrace-m0-6.rsf
creStackedTrace-m0-7.rsf
creStackedTrace-m0-8.rsf
creStackedTrace-m0-9.rsf
creStackedTrace-m0-10.rsf
creStackedTrace-m0-11.rsf
creStackedTrace-m0-12.rsf
creStackedTrace-m0-13.rsf
creStackedTrace-m0-14.rsf
creStackedTrace-m0-15.rsf
creStackedTrace-m0-16.rsf
creStackedTrace-m0-17.rsf
creStackedTrace-m0-18.rsf
creStackedTrace-m0-19.rsf
creStackedTrace-m0-20.rsf
creStackedTrace-m0-21.rsf
creStackedTrace-m0-22.rsf
creStackedTrace-m0-23.rsf
creStackedTrace-m0-24.rsf
creStackedTrace-m0-25.rsf
creStackedTrace-m0-26.rsf
creStackedTrace-m0-27.rsf
creStackedTrace-m0-28.rsf
creStackedTrace-m0-29.rsf
creStackedTrace-m0-30.rsf
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- A flag **-t** indica qual será o separador
- A flag **-k** indica qual é a coluna
- A flag **-n** indica ordenação numérica

Aula 35 - Comando xargs

- O comando **xargs** executa comandos, baseado em uma lista de argumentos, em loop sem usar laços.

Separando os argumentos

De dois em dois

```
echo "1 2 3 4 5 6" | xargs -n 2
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "1 2 3 4 5 6" | xargs -n 2
1 2
3 4
5 6
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

De três em três

```
echo "1 2 3 4 5 6" | xargs -n 3
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "1 2 3 4 5 6" | xargs -n 3
1 2 3
4 5 6
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Lendo a partir de um arquivo

- Isso pode ser feito utilizando a flag **-a**

```
xargs -a lista1.txt -n 2
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ xargs -a lista1.txt -n 2
creStackedTrace-m0-1.rsfcrcStackedTrace-m0-2.rsfcrc
creStackedTrace-m0-3.rsfcrcStackedTrace-m0-4.rsfcrc
creStackedTrace-m0-5.rsfcrcStackedTrace-m0-6.rsfcrc
creStackedTrace-m0-7.rsfcrcStackedTrace-m0-8.rsfcrc
creStackedTrace-m0-9.rsfcrcStackedTrace-m0-10.rsfcrc
creStackedTrace-m0-11.rsfcrcStackedTrace-m0-12.rsfcrc
creStackedTrace-m0-13.rsfcrcStackedTrace-m0-14.rsfcrc
creStackedTrace-m0-15.rsfcrcStackedTrace-m0-16.rsfcrc
creStackedTrace-m0-17.rsfcrcStackedTrace-m0-18.rsfcrc
creStackedTrace-m0-19.rsfcrcStackedTrace-m0-20.rsfcrc
creStackedTrace-m0-21.rsfcrcStackedTrace-m0-22.rsfcrc
creStackedTrace-m0-23.rsfcrcStackedTrace-m0-24.rsfcrc
creStackedTrace-m0-25.rsfcrcStackedTrace-m0-26.rsfcrc
creStackedTrace-m0-27.rsfcrcStackedTrace-m0-28.rsfcrc
creStackedTrace-m0-29.rsfcrcStackedTrace-m0-30.rsfcrc
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```


Executando uma sequência de comandos em loop

Executando uma sequência de comandos com confirmação

```
echo "um dois tres" | xargs -n 1 -p mkdir  
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "um dois tres" | xargs -n 1 -p mkdir  
mkdir um ?...y  
mkdir dois ?...n  
mkdir tres ?...y  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls  
cadastro.txt  lista1.txt  matriz.csv  numeros3.txt  redirecionamento.txt  teste1.txt  texto.txt  um  users.txt  
historico.txt  log.txt    numeros2.txt  numeros.txt  tail.txt            teste2.txt  tres  users2.txt  usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Foram criados os diretórios "um" e "tres"

Removendo os diretórios criados

```
echo "um tres" | xargs -n 1 -p rm -r
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "um tres" | xargs -n 1 -p rm -r  
rm -r um ?...y  
rm -r tres ?...y  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls  
cadastro.txt  lista1.txt  matriz.csv  numeros3.txt  redirecionamento.txt  teste1.txt  texto.txt  users.txt  
historico.txt  log.txt    numeros2.txt  numeros.txt  tail.txt            teste2.txt  users2.txt  usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Forma mais complexa de criar arquivos de texto em loop

```
echo -e "um\ndois\ntres" | xargs \  
> -I % -p bash -c 'touch %.txt'
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "um\ndois\ntres" | xargs \  
> -I % -p bash -c 'touch %.txt'  
bash -c touch um.txt ?...y  
bash -c touch dois.txt ?...y  
bash -c touch tres.txt ?...n  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls  
cadastro.txt  historico.txt  log.txt  numeros2.txt  numeros.txt  tail.txt  teste2.txt  um.txt  users.txt  
dois.txt      lista1.txt    matriz.csv  numeros3.txt  redirecionamento.txt  teste1.txt  texto.txt  users2.txt  usuarios.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Foram criados os arquivos: um.txt e dois.txt.

Forma mais complexa de remover arquivos de texto em loop

```
echo -e "um\ndois" | xargs \  
> -I % -p bash -c 'rm %.txt'
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo -e "um\ndoIs" | xargs \
> -I % -p bash -c 'rm %.txt'
bash -c rm um.txt ?...y
bash -c rm dois.txt ?...y
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls
cadastro.txt  lista1.txt  matriz.csv  numeros3.txt  redirecionamento.txt  teste1.txt  texto.txt  users.txt
historico.txt log.txt     numeros2.txt numeros.txt   tail.txt        teste2.txt  users2.txt usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 36 - Comando sed (Parte 1) - Editor de fluxo de texto do terminal

- O comando **sed** é um editor de texto do terminal. Ele edita os textos a partir de expressões regulares (regex).

Exemplo de expressão regular

ls *[0-9].txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls *[0-9].txt
lista1.txt  numeros2.txt  numeros3.txt  teste1.txt  teste2.txt  users2.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Editando um arquivo de texto

ls

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ ls
cadastro.txt  matriz.csv  redirecionamento.txt  texto.txt
historico.txt  numeros2.txt  tail.txt              users2.txt
lista1.txt     numeros3.txt  teste1.txt            users.txt
log.txt        numeros.txt  teste2.txt            usuarios.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

cat texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
Linha 1
Linha 2
Linha 3
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Edite o arquivo **texto.txt** com o editor **gedit**:

gedit texto.txt

- Apague o texto existente e digite o seguinte texto:

Testo qualche

Olá tudo bem galera?
Este é um texto
de teste

Ass.
Rodolfo



Testo qualche

Olá tudo bem galera?
Este é um texto
de teste|

Ass.
Rodolfo

Clique no botão **Salvar** e saia do editor pressionando **<Ctrl><q>**.

cat texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
Testo qualche

Olá tudo bem galera?
Este é um texto
de teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Selecionando linhas do arquivo

Primeira linha do arquivo de texto

sed -n '1p' texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '1p' texto.txt
Testo qualche
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Buscando uma linha através de um palavra

sed -n '/galera/p' texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '/galera/p' texto.txt
Olá tudo bem galera?
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Imprimindo um intervalo de linhas

`sed -n '1,3p' texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '1,3p' texto.txt
Testo qualche
Olá tudo bem galera?
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Caracteres especiais do comando sed

`^` representa o início da linha

`$` representa o final da linha

Exibindo linha que termina com ?

`sed -n '/?$/p' texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '/?$/p' texto.txt
Olá tudo bem galera?
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo linhas que terminam com a letra e

`sed -n '/e$/p' texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '/e$/p' texto.txt
Testo qualche
de teste
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo todas as linhas que começam com a letra O

`sed -n '/^O/p' texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -n '/^O/p' texto.txt
Olá tudo bem galera?
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 37 - Comando sed (Parte 2) - Substituir uma string por outra

- Por padrão o comando **sed** não registra uma substituição no arquivo (apenas exibe). Para se fazer uma substituição que fique registrada no arquivo, utiliza-se a flag **-i**

Corrigindo a primeira linha de texto

```
sed -i 's/Testo/Texto/; s/qualque/qualquer/' texto.txt
```

```
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i 's/Testo/Texto/; s/qualque/qualquer/' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
Texto qualquer

Olá tudo bem galera?
Este é um texto
de teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Comentando uma parte do código

No Linux, **#** é o caractere que indica comentário

- Para comentar a primeira linha do arquivo:

```
sed -i '1s/^/#/' texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i '1s/^/#/' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Este é um texto
de teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Para comentar, também, as linhas 3 a 5 do arquivo:

```
sed -i '3,5s/^/#/' texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i '3,5s/^/#/' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

#Olá tudo bem galera?
#Este é um texto
#de teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 38 - Comando sed (Parte 3) - Deletar linhas e strings de arquivo de texto

Faça as seguintes alterações no arquivo `texto.txt`:

```
#Texto qualquerLIXO
```

```
Olá tudo bem galera?LIXO  
Esse é  
BUGapenas BUGum BUGteste
```

```
***LINHA LIXO***
```

```
***BUG***
```

```
Ass.  
RodolfoBUG
```

`cat texto.txt`

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ gedit texto.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt  
#Texto qualquerLIXO  
  
Olá tudo bem galera?LIXO  
Esse é  
BUGapenas BUGum BUGteste  
  
***LINHA LIXO***  
  
***BUG***  
Ass.  
RodolfoBUG  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```


Corrigindo o texto

Apagando as ocorrências da palavra LIXO

```
sed -i 's/LIXO//' texto.txt  
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i 's/LIXO//' texto.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt  
#Texto qualquer  
  
Olá tudo bem galera?  
Esse é  
BUGapenas BUGum BUGteste  
  
***LINHA ***  
  
***BUG***  
Ass.  
RodolfoBUG  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Apagando as ocorrências da palavra BUG

- Por padrão o comando **sed** substitui apenas a primeira ocorrência, depois ele passa para a próxima linha. Para substituir todas as ocorrências da palavra em todas as linhas, utiliza-se o modificador **g**.

```
sed -i 's/BUG//g' texto.txt  
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i 's/BUG//g' texto.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt  
#Texto qualquer  
  
Olá tudo bem galera?  
Esse é  
apenas um teste  
  
***LINHA ***  
  
*****  
Ass.  
Rodolfo  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Deletando as linhas que iniciam com asterisco

```
sed -i '/^*/d' texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i '/^*/d' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
apenas um teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Deletando a linha 6 (linha em branco)

- Visualizando antes de deletar:

```
sed '6d' texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed '6d' texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
apenas um teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Deletando:

```
sed -i '6d' texto.txt
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i '6d' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
apenas um teste

Ass.
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Deletando a linha que termina com ponto

- Verificando:

```
sed '/\.$/d' texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed '/\.$/d' texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
apenas um teste

Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Deletando:

```
sed -i '/\.$/d' texto.txt
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ sed -i '/\.$/d' texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
apenas um teste

Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 39 - Comando grep (Parte 1) - Pesquisar strings em arquivo de texto

grep - Globally search a regular expression and print

- Altere o arquivo **texto.txt** para:

#Texto qualquer

Olá tudo bem galera?

Esse é

BUGapenas **BUG**um **BUG**teste

Rodolfo

cat texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
BUGapenas BUGum BUGteste

Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Fazendo a busca da palavra a ser deletada

grep -n **BUG** texto.txt

A flag **-n** exibe o número da linha

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -n BUG texto.txt
5:BUGapenas BUGum BUGteste
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo todas as linhas que não contém a palavra BUG

- Com a flag -v é exibido as linhas que não contém a palavra.

```
grep -v BUG texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -v BUG texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
Rodolfo
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo o número de linhas que contém a palavra BUG

- Com a flag -c é exibido o número de linhas que contém a palavra.

```
grep -c BUG texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -c BUG texto.txt
1
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Acrescentando uma nova linha com a palavra BUG no final do arquivo

```
echo "BUG" >> texto.txt
```

```
cat texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo "BUG" >> texto.txt
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ cat texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
BUGapenas BUGum BUGteste

Rodolfo
BUG
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

```
grep -v BUG texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -c BUG texto.txt
2
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

- Agora são duas linhas que contém a palavra "BUG"

grep -n BUG texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -n BUG texto.txt
5:BUGapenas BUGum BUGteste
8:BUG
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo todas as ocorrências de uma palavra

grep -o BUG texto.txt

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -o BUG texto.txt
BUG
BUG
BUG
BUG
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Aula 40 - Comando grep (Parte 2) - Exibir o contexto de pesquisa

Exibindo o contexto de uma pesquisa

Exibindo a linha com a ocorrência da busca e n linhas antes

- A flag **-B** (before) exibe o número de linhas que também deve ser exibido antes da linha com a ocorrência.

```
grep -B 2 galera texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -B 2 galera texto.txt
#Texto qualquer

Olá tudo bem galera?
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo a linha com a ocorrência da busca e n linhas depois

- A flag **-A** (after) exibe o número de linhas que também deve ser exibido antes da linha com a ocorrência.

```
grep -A 2 galera texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -A 2 galera texto.txt
Olá tudo bem galera?
Esse é
BUGapenas BUGum BUGteste
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Exibindo a linha com a ocorrência da busca e n linhas antes e depois

- A flag **-C** (context) exibe o número de linhas que também deve ser exibido antes e depois da linha com a ocorrência.

```
grep -C 2 galera texto.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -C 2 galera texto.txt
#Texto qualquer

Olá tudo bem galera?
Esse é
BUGapenas BUGum BUGteste
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$
```

Retornando o resultado de uma busca

- A flag **-q** retorna 0 se a busca encontrar alguma ocorrência, ou retorna 1 se não encontrar nenhuma ocorrência. O resultado fica armazenado na variável \$?

```
grep -q galera texto.txt  
echo $?
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -q galera texto.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo $?  
0  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ █
```

```
grep -q galerax texto.txt  
echo $?
```

```
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ grep -q galerax texto.txt  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ echo $?  
1  
roberto@roberto-Aspire-E1-531:~/testes/arquivos_txt$ █
```


Aula 41 - Comando tree

- O comando **tree** exibe uma lista em formato de árvore de diretórios a partir do diretório atual.

Instalando o tree no sistema

sudo apt-get install tree

```
roberto@roberto-Aspire-E1-531:~$ sudo apt-get install tree
[sudo] senha para roberto:
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os NOVOS pacotes a seguir serão instalados:
  tree
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 1 não atualizados.
É preciso baixar 40,6 kB de arquivos.
Depois desta operação, 138 kB adicionais de espaço em disco serão usados.
Obter:1 http://br.archive.ubuntu.com/ubuntu xenial/universe amd64 tree amd64 1.7.0-3 [40,6 kB]
Baixados 40,6 kB em 0s (120 kB/s)
A seleccionar pacote anteriormente não seleccionado tree.
(Lendo banco de dados ... 232958 ficheiros e directórios actualmente instalados.)
A preparar para desempacotar .../tree_1.7.0-3_amd64.deb ...
A descompactar tree (1.7.0-3) ...
A processar 'triggers' para man-db (2.7.5-1) ...
Configurando tree (1.7.0-3) ...
roberto@roberto-Aspire-E1-531:~$
```

tree

```
roberto@roberto-Aspire-E1-531:~/testes$ tree
.
├── arquivos_txt
│   ├── cadastro.txt
│   ├── historico.txt
│   ├── lista1.txt
│   ├── log.txt
│   ├── matriz.csv
│   ├── numeros2.txt
│   ├── numeros3.txt
│   ├── numeros.txt
│   ├── redirecionamento.txt
│   ├── tail.txt
│   ├── teste1.txt
│   ├── teste2.txt
│   ├── texto.txt
│   ├── users2.txt
│   ├── users.txt
│   └── usuarios.txt
├── scripts
│   └── script.sh
└── testeCat
    ├── arq1.txt
    ├── arq2.txt
    └── arq3.txt

3 directories, 20 files
roberto@roberto-Aspire-E1-531:~/testes$
```

Exibindo um diretório específico

```
tree arquivos_txt
```

```
roberto@roberto-Aspire-E1-531:~/testes$ tree arquivos_txt
arquivos_txt
├── cadastro.txt
├── historico.txt
├── lista1.txt
├── log.txt
├── matriz.csv
├── numeros2.txt
├── numeros3.txt
├── numeros.txt
├── redirecionamento.txt
├── tail.txt
├── teste1.txt
├── teste2.txt
├── texto.txt
├── users2.txt
├── users.txt
└── usuarios.txt

0 directories, 16 files
roberto@roberto-Aspire-E1-531:~/testes$
```

Criando pastas dentro de um diretório de testes

```
mkdir -p testeTree/teste1/teste2/teste3
```

```
ls
```

```
cd testeTree
```

```
tree
```

```
roberto@roberto-Aspire-E1-531:~/testes$ mkdir -p testeTree/teste1/teste2/teste3
roberto@roberto-Aspire-E1-531:~/testes$ ls
arquivos_txt  scripts  testeCat  testeTree
roberto@roberto-Aspire-E1-531:~/testes$ cd testeTree
roberto@roberto-Aspire-E1-531:~/testes/testeTree$ tree
.
├── teste1
│   ├── teste2
│   │   └── teste3
└──
```

```
3 directories, 0 files
roberto@roberto-Aspire-E1-531:~/testes/testeTree$
```

Inserindo arquivos dentro dos diretórios criados

```
touch teste1/arq1.txt
touch teste1/teste2/arq2.txt
touch teste1/teste2/teste3/arq3.txt
```

Exibindo a árvore de diretórios

tree

```
roberto@roberto-Aspire-E1-531:~/testes/testeTree$ tree
.
├── teste1
│   ├── arq1.txt
│   └── teste2
│       ├── arq2.txt
│       └── teste3
│           └── arq3.txt
3 directories, 3 files
```

Armazenando o resultado dentro de um arquivo de texto

Para isso utiliza-se a flag **-o**

```
tree -o lista.txt
ls
cat lista.txt
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeTree$ tree -o lista.txt
roberto@roberto-Aspire-E1-531:~/testes/testeTree$ ls
lista.txt  teste1
roberto@roberto-Aspire-E1-531:~/testes/testeTree$ cat lista.txt
.
├── lista.txt
└── teste1
    ├── arq1.txt
    └── teste2
        ├── arq2.txt
        └── teste3
            └── arq3.txt
3 directories, 4 files
roberto@roberto-Aspire-E1-531:~/testes/testeTree$
```

Aula 42 - Comando ln

- O comando **ln** permite criar links simbólicos e hardlinks em shell.

mkdir -p testeLink/pasta1/pasta2/arquivos

```
roberto@roberto-Aspire-E1-531:~/testes$ mkdir -p testeLink/pasta1/pasta2/arquivos
roberto@roberto-Aspire-E1-531:~/testes$ ls
arquivos_txt  scripts  testeCat  testeLink  testeTree
roberto@roberto-Aspire-E1-531:~/testes$
```

- Dentro da pasta **testeLink** crie um arquivo chamado **arq1.txt**:

echo "Arquivo qualquer" >> pasta1/pasta2/arquivos/arq1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ echo "Arquivo qualquer" >> pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ tree
├── pasta1
│   └── pasta2
│       └── arquivos
│           └── arq1.txt
3 directories, 1 file
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Criando um link simbólico para o arquivo criado

A flag **-s** indica que é um link simbólico (soft link)

ln -s pasta1/pasta2/arquivos/arq1.txt linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ln -s pasta1/pasta2/arquivos/arq1.txt linkSimbolico
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls
linkSimbolico  pasta1
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Exibindo as permissões do link criado

ls -l linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l linkSimbolico
lrwxrwxrwx 1 roberto roberto 31 Jan 16 13:39 linkSimbolico -> pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Agora **linkSimbolico** aponta para o arquivo criado **arq1.txt** (é um atalho para ele)

Verificando o inode do link criado

ls -li linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -li linkSimbolico
787414 linkSimbolico
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Verificando o inode do arquivo criado

ls -li pasta1/pasta2/arquivos/arq1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -li pasta1/pasta2/arquivos/arq1.txt
787413 pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Observe que o inode dos arquivos é **diferente**.

- **linkSimbolico** aponta para **arq1.txt**:

cat pasta1/pasta2/arquivos/arq1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ cat pasta1/pasta2/arquivos/arq1.txt
Arquivo qualquer
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

cat linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ cat linkSimbolico
Arquivo qualquer
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Alterando o conteúdo de **arq1.txt** isso irá refletir em **linkSimbolico**:

- Com **gedit**, altere o conteúdo de **arq1.txt** para:

Arquivo qualquer **alterado**

- Salve e feche o editor.

cat pasta1/pasta2/arquivos/arq1.txt

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ gedit pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ cat pasta1/pasta2/arquivos/arq1.txt
Arquivo qualquer alterado
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

cat linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ cat linkSimbolico
Arquivo qualquer alterado
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Exibindo o tipo de arquivo

O comando **file** exibe o tipo de arquivo.

file linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ file linkSimbolico
linkSimbolico: symbolic link to pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Removendo o arquivo original

- Se o arquivo original (**arq1.txt**) for removido quebra-se o vínculo com o link simbólico e perde-se a informação.

rm pasta1/pasta2/arquivos/arq1.txt

ls -l linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ rm pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l linkSimbolico
lrwxrwxrwx 1 roberto roberto 31 Jan 16 13:39 linkSimbolico -> pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

file linkSimbolico

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ file linkSimbolico
linkSimbolico: broken symbolic link to pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Removendo o link quebrado

rm linkSimbolico

Criando um hardlink

echo "Hard Link" >> pasta1/pasta2/arquivos/arq1.txt

In pasta1/pasta2/arquivos/arq1.txt hardLink

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ln pasta1/pasta2/arquivos/arq1.txt hardLink
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls
hardLink pasta1
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

ls -l hardLink

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l hardLink
-rw-rw-r-- 2 roberto roberto 10 Jan 16 14:13 hardLink
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Verificando o inode dos arquivos

ls -li pasta1/pasta2/arquivos/arq1.txt

ls -li hardLink

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -li pasta1/pasta2/arquivos/arq1.txt
787413 pasta1/pasta2/arquivos/arq1.txt
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -li hardLink
787413 hardLink
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Observe que o inode dos arquivos é exatamente o **mesmo**.

Importante:

- Enquanto o inode do link simbólico era diferente, o inode do hardlink é igual.
- Enquanto o soft link é um atalho para o arquivo, o hard link é um ponteiro para o inode.

Removendo o arquivo original

- Se o arquivo original (**arq1.txt**) for removido a informação não será preservada.

rm pasta1/pasta2/arquivos/arq1.txt

ls -l hardLink

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l hardLink
-rw-rw-r-- 1 roberto roberto 10 Jan 16 14:13 hardLink
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

cat hardLink

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ cat hardLink
Hard Link
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- No hard link, mesmo que o arquivo original for removido, a informação original não é perdida.

Aula 43 - Utilizando links simbólicos como referência à uma biblioteca

- Dentro da pasta `~/testes/testeLink` crie uma subpasta chamada `lib`:

```
mkdir lib  
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ mkdir lib  
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls  
hardLink  lib  pasta1  
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Dentro da pasta `lib`, com o editor `gedit` crie um arquivo chamado `lib1.0.sh`:

```
gedit lib1.0.sh
```

```
echo "Versão 1.0"
```

```
echo "Versão 1.0"|
```

Dando permissão de execução para o arquivo criado

```
chmod +x lib1.0.sh
```

Executando o arquivo

```
./lib1.0.sh
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$ chmod +x lib1.0.sh  
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$ ./lib1.0.sh  
Versão 1.0  
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$
```

- Dentro da pasta **lib**, com o editor **gedit** crie um arquivo chamado **lib2.0.sh**:

```
gedit lib2.0.sh
```

```
echo "Versão 2.0"
```



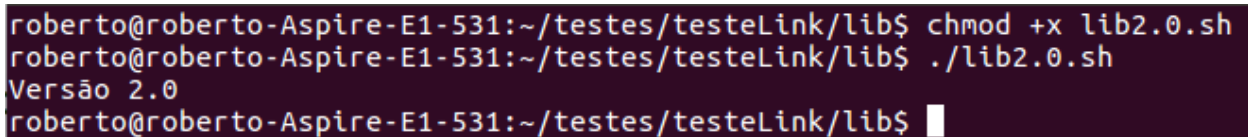
```
echo "Versão 2.0"
```

Dando permissão de execução para o arquivo criado

```
chmod +x lib2.0.sh
```

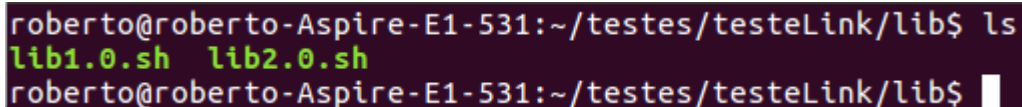
Executando o arquivo

```
./lib2.0.sh
```



```
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$ chmod +x lib2.0.sh
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$ ./lib2.0.sh
Versão 2.0
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$
```

```
ls
```



```
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$ ls
lib1.0.sh  lib2.0.sh
roberto@roberto-Aspire-E1-531:~/testes/testeLink/lib$
```

Programa principal

Vá para a pasta `~/testes/testeLink`

```
cd ~/testes/testeLink  
gedit main.sh
```

```
./lib/lib1.0.sh
```

```
./lib/lib1.0.sh
```

- Salve o arquivo e saia do editor.

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls  
hardLink  lib  main.sh  pasta1  
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Dando permissão de execução para o arquivo criado

```
chmod +x main.sh
```

Executando o arquivo

```
./main.sh
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ chmod +x main.sh  
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ./main.sh  
Versão 1.0  
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Utilizando link simbólico

- Altere o conteúdo do arquivo `main.sh` para:

```
gedit main.sh
```

```
./link_lib
```

```
./link_lib
```

Usando versão 1.0

- Dentro da pasta `~/testes/TesteLink` crie o link simbólico `link_db`:

```
ln -s ./lib/lib1.0.sh link_lib
```

```
ls
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ln -s ./lib/lib1.0.sh link_lib
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls
hardLink  lib  link_lib  main.sh  pasta1
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

```
ls -l link_lib
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l link_lib
lrwxrwxrwx 1 roberto roberto 15 Jan 16 22:02 link_lib -> ./lib/lib1.0.sh
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Executando o programa principal (main.sh)

```
./main.sh
```

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ./main.sh
Versão 1.0
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Usando a versão 2.0

`ln -s ./lib/lib2.0.sh link_lib`

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ln -s ./lib/lib2.0.sh link_lib
ln: falhou ao criar link simbólico 'link_lib': Arquivo existe
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Com a flag `-f` (force) é forçado a alteração no link simbólico.

`ln -s -f ./lib/lib2.0.sh link_lib`

`ls -l link_lib`

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ln -s -f ./lib/lib2.0.sh link_lib
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ls -l link_lib
lrwxrwxrwx 1 roberto roberto 15 Jan 16 22:15 link_lib -> ./lib/lib2.0.sh
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

- Perceba que agora link_lib aponta para a versão 2.0

Executando o programa principal (main.sh)

`./main.sh`

```
roberto@roberto-Aspire-E1-531:~/testes/testeLink$ ./main.sh
Versão 2.0
roberto@roberto-Aspire-E1-531:~/testes/testeLink$
```

Aula 44 - Comando main

- O comando **main** é utilizado para obter a documentação dos comandos do Linux.

Exibindo a documentação do comando ls

man ls

```
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default).  Sort
  entries alphabetically if none of -cftuvSUX nor --sort is specified.

  Mandatory arguments to long options are mandatory for short options too.

  -a, --all
      do not ignore entries starting with .

  -A, --almost-all
      do not list implied . and ..

  --author
      with -l, print the author of each file

  -b, --escape
      print C-style escapes for nongraphic characters

  --block-size=SIZE
      scale sizes by SIZE before printing them;  e.g., '--block-size=M'
      prints sizes in units of 1,048,576 bytes; see SIZE format below

  -B, --ignore-backups
      do not list implied entries ending with ~

  -c
      with -lt: sort by, and show, ctime (time of last modification of
      file status information); with -l: show ctime and sort by name;
      otherwise: sort by ctime, newest first

  -C
      list entries by columns
```

.
.
.
.

- Para se mover na documentação pode-se usar as setas direcionadoras: seta para baixo ou seta para cima.

- Para sair da página pressione a tecla <q>

man man

- A documentação dos comandos está distribuída em seções.

- 1 Executable programs or shell commands
- 2 System calls (functions provided by the kernel)
- 3 Library calls (functions within program libraries)
- 4 Special files (usually found in /dev)
- 5 File formats and conventions eg /etc/passwd
- 6 Games
- 7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
- 8 System administration commands (usually only for root)
- 9 Kernel routines [Non standard]

Uma breve descrição dos comandos

É obtida utilizando-se a flag **-f**:

Exemplos:

man -f ls

```
roberto@roberto-Aspire-E1-531:~$ man -f ls
ls (1) - list directory contents
roberto@roberto-Aspire-E1-531:~$
```

man -f pwd

```
roberto@roberto-Aspire-E1-531:~$ man -f pwd
pwd (1) - print name of current/working directory
roberto@roberto-Aspire-E1-531:~$
```

Exibindo o local da documentação do comando

- Para exibir o local da documentação de um comando utiliza-se a flag **-w**:

man -w pwd

```
roberto@roberto-Aspire-E1-531:~$ man -w pwd
/usr/share/man/man1/pwd.1.gz
roberto@roberto-Aspire-E1-531:~$
```