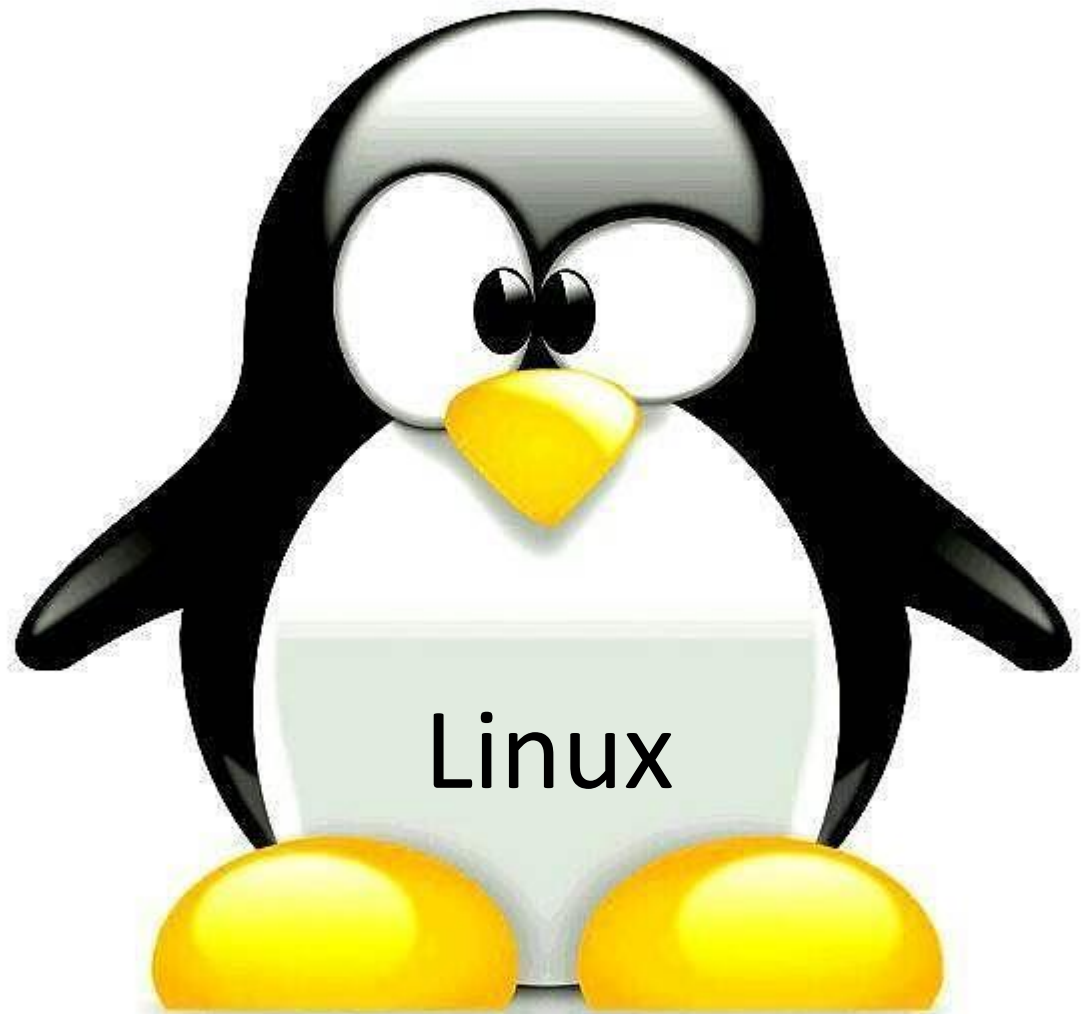


Comandos para
trabalhar com
Ficheiros



Listar

O comando `ls` lista o conteúdo de uma directoria.

Mostra os nomes dos objectos, se usarmos uma listagem longa com a opção `-l` vai listar também outros campos.

Com a opção `-R` podemos listar em modo recursivo, na prática, a directoria actual e as que se encontram abaixo.

Nalguns sistemas existe ainda uma ferramenta `tree` (tradução de árvore) onde mostra em formato mais fácil de ler a estrutura de directorias.



Listar

ls lista só os nomes

ls -l listagem longa com mais informação que iremos ver à frente, campo a campo

ls -R lista os nomes de forma recursiva, ou seja, nas directorias mostra o seu conteúdo

```
[root@localhost ~]# ls
abc bench.py def ghi hello.c
[root@localhost ~]# ls -l
total 20
drwxr-xr-x 2 root root 61 Apr 1 21:21 abc
-rw-r--r-- 1 root root 113 Sep 9 2018 bench.py
drwxr-xr-x 2 root root 67 Apr 1 21:21 def
drwxr-xr-x 2 root root 68 Apr 1 21:21 ghi
-rw-r--r-- 1 root root 185 Sep 9 2018 hello.c
[root@localhost ~]# ls -R
.:
abc bench.py def ghi hello.c

./abc:
ola.txt

./def:
vou_ser_feliz

./ghi:
vou_vencer.txt
[root@localhost ~]#
```

Listar – comando ls

1º carácter
indica o tipo
de objecto, d
para
directoria e -
para
ficheiros

```
[root@localhost ~]# ls -l
total 20
drwxr-xr-x 2 root root 61 Apr  1 21:21 abc
-rw-r--r-- 1 root root 113 Sep  9 2018 bench.py
drwxr-xr-x 2 root root 67 Apr  1 21:21 def
drwxr-xr-x 2 root root 68 Apr  1 21:21 ghi
-rw-r--r-- 1 root root 185 Sep  9 2018 hello.c
```

2º a 11º
carácter
indicam as
permissões

Nº de links

O nome do
utilizador

O grupo a que
pertence o
utilizador

dimensão

Data / hora

Nome do ficheiro
ou directoria

Listar – comando ls

1º carácter
indica o tipo
de objecto, d
para
directoria e -
para
ficheiros

```
[root@localhost ~]# ls -l
total 20
drwxr-xr-x 2 root root 61 Apr  1 21:21 abc
-rw-r--r-- 1 root root 113 Sep  9 2018 bench.py
drwxr-xr-x 2 root root 67 Apr  1 21:21 def
drwxr-xr-x 2 root root 68 Apr  1 21:21 ghi
-rw-r--r-- 1 root root 185 Sep  9 2018 hello.c
```

2º a 11º
carácter
indicam as
permissões

Nº de links

O nome do
utilizador

O grupo a que
pertence o
utilizador

dimensão

Data / hora

Nome do ficheiro
ou directoria

Listar – comando tree

Todos os nomes que comecem com o simbolo ponto (.) não aparecem “normalmente” com o comando ls nem na listagem longa ls -l

Para os vermos
teremos que usar a
opção -a ou adicionar
a opção à listagem
longa.

Ou seja ls -a ou ls -la

```
[amatias@localhost Desktop]$ ls
teste1.sh teste.sh
[amatias@localhost Desktop]$ ls -l
total 8
-rwxr-xr-x 1 amatias amatias 471 May  9 17:54 teste1.sh
-rwxr-xr-x 1 amatias amatias 509 May  9 17:53 teste.sh
[amatias@localhost Desktop]$ ls -la
total 12
drwxr-xr-x. 3 amatias amatias  94 May  9 17:56 .
drwx----- 25 amatias amatias 4096 May  9 15:47 ..
-rw-rw-r--  1 amatias amatias   0 May  9 17:55 .estouescondido
drwxrwxr-x  2 amatias amatias   6 May  9 17:56 .SouaPontoPontoEscondida
-rwxr-xr-x  1 amatias amatias 471 May  9 17:54 teste1.sh
-rwxr-xr-x  1 amatias amatias 509 May  9 17:53 teste.sh
```

Listar – comando tree

O comando `- tree -` apresenta uma árvore invertida desde o local onde nos encontramos com todas as directorias abaixo e os respectivos ficheiros como se fossem folhas dessa árvore

Não existe em todas as distribuições Linux, mas nas que existe dá muito jeito 😊

```
sana@sana-HP-ProBook-4530s:~$ tree
.
├── Desktop
├── Documents
│   └── MumbleAutomaticCertificateBackup.p12
├── Downloads
│   ├── dummy.pdf
│   ├── edited_dummy.pdf
│   ├── edit_pdf.docx
│   └── TouchpadIndicator-master
│       ├── convenience.js
│       ├── COPYING
│       ├── extension.js
│       ├── icons
│       │   ├── input-touchpad-symbolic.svg
│       │   ├── my-touchpad-disabled-dark.svg
│       │   ├── my-touchpad-normal-dark.svg
│       │   └── touchpad-disabled-symbolic.svg
│       ├── lib.js
│       ├── locale
│       │   ├── de
│       │   │   └── LC_MESSAGES
│       │   │       ├── de.po
│       │   │       └── touchpad-indicator@orangeshirt.mo
│       └── es
```

Criação de ficheiros

Existem várias formas de criar ficheiros, por exemplo através dos editores de texto já falados como o nano e o vi que criam ficheiros com texto, embora estejam mais direccionados para editar programas.



Mas caso queiramos criar só um ficheiro, mesmo sem conteúdo poderemos usar o comando touch.

Touch pode ainda ser usado só para mudar a data/hora dos ficheiros com a hora actual (esta situação é muito usada por programadores para actualizar a data/hora do Makefile).

Criação de ficheiros

A sintaxe para criar ficheiros é touch, onde cria o ficheiros com 0 bytes.

Na imagem podemos ver a criação do ficheiro1 com : touch ficheiro1

No caso de querermos criar vários poderemos indicar o seu nome, separado por espaços. Tal como abaixo onde foram criados o ficheiro2 e texto3.

```
[root@localhost ~]# ls -l
total 0
[root@localhost ~]# touch ficheiro1
[root@localhost ~]# ls -l
total 0
-rw-r--r-- 1 root root 0 Mar 31 17:51 ficheiro1
[root@localhost ~]# touch ficheiro2 texto3
[root@localhost ~]# ls -l
total 0
-rw-r--r-- 1 root root 0 Mar 31 17:51 ficheiro1
-rw-r--r-- 1 root root 0 Mar 31 17:51 ficheiro2
-rw-r--r-- 1 root root 0 Mar 31 17:51 texto3
[root@localhost ~]#
```

Ler o conteúdo de ficheiros

O comando “cat hello.c” mostra o ficheiro, como se pode ver na imagem abaixo.

```
[root@localhost ~]# cat hello.c
/* This C source can be compiled with:
   gcc -o hello hello.c
 */
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello World\n");
    return 0;
}
```



Ler o conteúdo de ficheiros

O comando “cat hello.c” mostra o ficheiro, como se pode ver na imagem abaixo.

```
[root@localhost ~]# cat hello.c
/* This C source can be compiled with:
   gcc -o hello hello.c
 */
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello World\n");
    return 0;
}
```


Se o ficheiro for grande poderemos usar o comando “more hello.c” que nos mostra por páginas, parando no final de cada uma para poder ser lido confortavelmente.



Criação de ficheiros

O comando “touch hello.c” muda a data do ficheiro, como se pode ver na imagem abaixo, sem mexer no seu conteúdo.

```
[root@localhost ~]# ls -l
total 8
-rw-r--r-- 1 root root 113 Sep  9  2018 bench.py
-rw-r--r-- 1 root root 185 Sep  9  2018 hello.c
[root@localhost ~]# touch hello.c
[root@localhost ~]# ls -l
total 8
-rw-r--r-- 1 root root 113 Sep  9  2018 bench.py
-rw-r--r-- 1 root root 185 Apr  1 22:40 hello.c
[root@localhost ~]#
```



Criação de ficheiro com comando cat

Também podemos usar o comando cat que serve para ver o conteúdo de um ficheiro mas poderemos usá-lo para escrever num ficheiro através do carácter > que indica que se deve enviar para dentro do ficheiro indicado a seguir o que aparecer no ecrã.

Podemos ver que inicialmente não temos o ficheiro ola.txt, o comando cat que o cria. Ficamos então a ver que o ficheiro tem 15 caracteres e no fim, o conteúdo do ficheiro com o cat ola.txt

```
[root@localhost ~]# ls -l
total 8
-rw-r--r-- 1 root root 113 Sep  9 2018 bench.py
-rw-r--r-- 1 root root 185 Apr  1 22:40 hello.c
[root@localhost ~]# cat > ola.txt
Bom dia mundo!
[root@localhost ~]# ls -l
total 12
-rw-r--r-- 1 root root 113 Sep  9 2018 bench.py
-rw-r--r-- 1 root root 185 Apr  1 22:40 hello.c
-rw-r--r-- 1 root root  15 Apr  1 22:45 ola.txt
[root@localhost ~]# cat ola.txt
Bom dia mundo!
[root@localhost ~]#
```



Remover um ficheiro

O comando rm (remove) um ou mais ficheiros.

Poderemos usar com a opção `-i` em que nos obriga a confirmar que queremos mesmo remover para superar erros.

Podemos ainda usar a opção `-r` em que remove directorias com conteúdo, na prática é como uma remoção recursiva em que remove a directoria e respectivas directorias e ficheiros abaixo.

```
[root@localhost ~]# ls -l
total 24
drwxr-xr-x 2 root root 61 Apr 1 23:30 abc
-rw-r--r-- 1 root root 113 Sep 9 2018 bench.py
drwxr-xr-x 2 root root 67 Apr 1 23:30 def
drwxr-xr-x 2 root root 68 Apr 1 23:30 ghi
-rw-r--r-- 1 root root 185 Apr 1 22:40 hello.c
-rw-r--r-- 1 root root 15 Apr 1 22:45 ola.txt
[root@localhost ~]# ls -R
.:
abc bench.py def ghi hello.c ola.txt
```

```
./abc:
ola.txt
```

```
./def:
vou_ser_feliz
```

```
./ghi:
vou_vencer.txt
[root@localhost ~]# rm hello.c
[root@localhost ~]# rm -r ghi
[root@localhost ~]# ls -R
.:
abc bench.py def ola.txt
```

```
./abc:
ola.txt
```

```
./def:
vou_ser_feliz
[root@localhost ~]#
```



Copiar um ficheiro

O comando cp, copia ficheiros.

Para usar indicamos qual o ficheiro origem, a ser copiado, e qual o nome do ficheiro destino que vai ficar cópia do primeiro.

cp ficheiro_origem ficheiro_destino

```
[root@localhost ~]# ls -l
```

```
total 20
```

```
drwxr-xr-x 2 root root 61 Apr 1 23:30 abc
```

```
-rw-r--r-- 1 root root 113 Sep 9 2018 bench.py
```

```
drwxr-xr-x 2 root root 67 Apr 1 23:30 def
```

```
drwxr-xr-x 2 root root 68 Apr 1 23:41 ghi
```

```
-rw-r--r-- 1 root root 15 Apr 1 22:45 ola.txt
```

```
[root@localhost ~]# cp ola.txt um_ficheiro.x
```

```
[root@localhost ~]# ls -l
```

```
total 24
```

```
drwxr-xr-x 2 root root 61 Apr 1 23:30 abc
```

```
-rw-r--r-- 1 root root 113 Sep 9 2018 bench.py
```

```
drwxr-xr-x 2 root root 67 Apr 1 23:30 def
```

```
drwxr-xr-x 2 root root 68 Apr 1 23:41 ghi
```

```
-rw-r--r-- 1 root root 15 Apr 1 22:45 ola.txt
```

```
-rw-r--r-- 1 root root 15 Apr 1 23:45 um_ficheiro.x
```

```
[root@localhost ~]#
```

Mover um ficheiro

O comando mv, move ficheiros, em que se estiverem na mesma directoria, poderemos dizer que só a alteração do nome do ficheiro.

Podemos também usar para mover para outro local, principalmente para uma directoria.

```
[root@localhost ~]# ls -R
.:
abc bench.py ola.txt um_ficheiro.x
./abc:
ficheiro
[root@localhost ~]# mv ola.txt boa_tarde.txt
[root@localhost ~]# ls -R
.:
abc bench.py boa_tarde.txt um_ficheiro.x
./abc:
ficheiro
[root@localhost ~]# mv boa_tarde.txt abc
[root@localhost ~]# ls -R
.:
abc bench.py um_ficheiro.x
./abc:
boa_tarde.txt ficheiro
[root@localhost ~]#
```



Procurar uma palavra dentro de um ficheiro

```
amatias@pinguim $ cat numeros.txt
```

1

2

3

...

10

11

Olá

Bom dia



```
amatias@pinguim $ grep 1 numeros.txt
```

1

10

11

O comando `grep` pesquisa a palavra indicada dentro de um ficheiro, mostrando a linha onde essa palavra está.

Neste caso `grep 1` vai mostrar as linhas com 1, 10 e 11

Informação

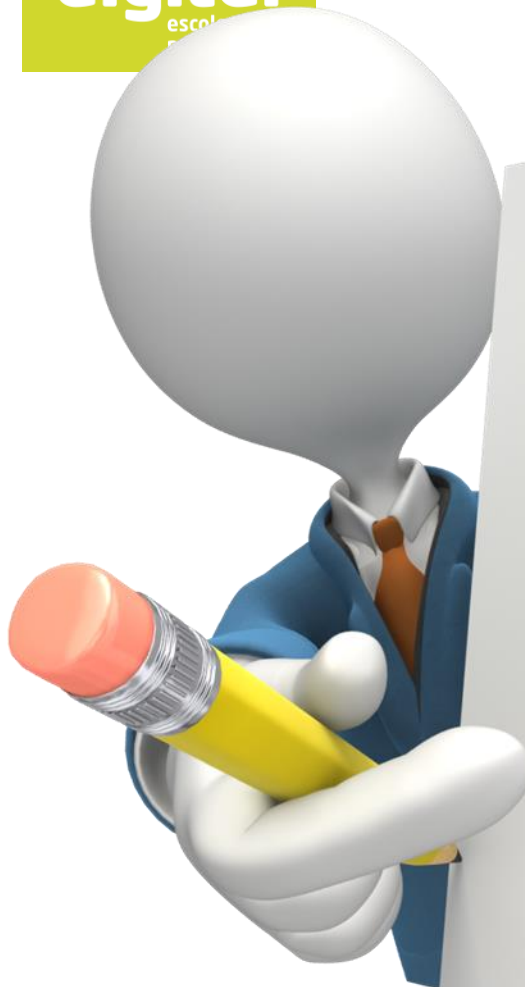
```
[amatias@centos7 ~]$ cd /etc
[amatias@centos7 etc]$ cat hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
[amatias@centos7 etc]$ file hosts
hosts: ASCII text
[amatias@centos7 etc]$ stat hosts
  File: 'hosts'
  Size: 158          Blocks: 8          IO Block: 4096  regular file
Device: fd00h/64768d Inode: 17277741  Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)  Gid: (  0/   root)
Context: system_u:object_r:net_conf_t:s0
Access: 2020-07-13 12:49:34.329998765 +0100
Modify: 2013-06-07 15:31:32.000000000 +0100
Change: 2020-05-29 16:20:22.221997459 +0100
 Birth: -
[amatias@centos7 etc]$
```

O comando **file** mostra informação genérica sobre o ficheiro, já o comando **stat** mostra mais informação.



Quem entende a piada?





Perguntas

1. Para obter a listagem longa dos ficheiros e directorias, qual a opção que devo usar?
2. Ao executar o comando touch num ficheiro que existe, o que acontece?
3. Para duplicar o ficheiro com nome 1.x que comando devo executar? Qual o último argumento que indico?
4. Posso mover um ficheiro para outra localização/directoria? Como?
5. Se remover uma directoria com `rm -r directoria`, os ficheiros que lá estão dentro, onde vão ficar?