Comandos para a administración de ficheiros en Linux

Sumario

Introdución ao sistema de ficheiros de linux

Para unha descrición completa do sistema de arquivos podemos poñer na nosa Shell o comando:

```
$ man hier
```

Así e todo, podemos ver na seguinte táboa un resumo da organización dos directorios nun sistema linux:

Directorio	Descrición
/bin/	Comandos e programas binarios esenciais (cp, mv, ls, rm, etc.),
/boot/	Ficheiros utilizados durante o arranque do sistema (núcleo e discos RAM)
/dev/	Dispositivos esenciais, discos duros, terminais, son, video, lectores dvd/cd, etc
/etc/	Ficheiros de configuración utilizados en todo o sistema e que son específicos do ordenador
/etc/opt/	Ficheiros de configuración utilizados por programas aloxados dentro de /opt/
/etc/X11/	Ficheiros de configuración para o sistema X Window (Opcional)
/etc/sgml/	Ficheiros de configuración para SGML (Opcional)
/etc/xml/	Ficheiros de configuración para XML (Opcional)
/home/	Directorios de inicio dos usuarios (Opcional)
/lib/	Bibliotecas compartidas esenciais para os binarios de /bin/, /sbin/ e o núcleo do sistema.
/mnt/	Sistemas de ficheiros montados temporalmente.

Directorio	Descrición
/media/	Puntos de montaxe para dispositivos de medios como unidades lectoras de discos compactos.
/opt/	Paquetes de aplicacións estáticas.
/proc/	Sistema de ficheiros virtual que documenta sucesos e estados do núcleo. Contén principalmente ficheiros de texto.
/root/	Directorio de inicio do usuario root (super-usuario) (Opcional)
/sbin/	Comandos/programas binarios de administración de sistema.
/tmp/	Ficheiros temporais
/srv/	Datos específicos de sitio servidos polo sistema.
/usr/	Xerarquía secundaria para datos compartidos de só lectura (*Unix system resources*). Este directorio pode ser compartido por múltiples ordenadores e non debe conter datos específicos do ordenador que os comparte.
/usr/bin/	Comandos/programas binarios.
/usr/include/	Ficheiros de inclusión estándar (cabeceiras de cabecera utilizados para desenvolvemento).
/usr/lib/	Bibliotecas compartidas.
/usr/share/	Datos compartidos independentes da arquitectura do sistema. Imaxes, ficheiros de texto, etc.
/usr/src/	Códigos fonte (Opcional)
/usr/X11R6/	Sistema X Window, versión 11, lanzamento 6 (Opcional)
/usr/local/	Xerarquía terciaria para datos compartidos de só lectura específicos do ordenador que os comparte.
/var/	Ficheiros variables, como son *logs*, bases de datos, directorio raíz de servidores HTTP e FTP, colas de correo, ficheiros temporais, etc.
/var/cache/	Cache da datos de aplicacións.
/var/crash/	Depósito de información referente a caidas do sistema (Opcional)
/var/games/	Datos variables de aplicacións para xogos (Opcional)
/var/lib/	Información de estado variable. Algúns servidores como MySQL e PostgreSQL almacenan as súas bases de datos en directorios subordinados deste.
/var/lock/	Ficheiros de bloqueo.
/var/log/	Ficheiros e directorios de rexistro de sistemas (*logs*).
/var/mail/	Buzóns de correo de usuarios (Opcional)
/var/opt/	Datos variables de /opt/.
/var/spool/	Colas de datos de aplicacións.
/var/tmp/	Ficheiros temporales preservados entre reinicios.

pwd

O comando pwd nos indica o directorio de traballo, é dicir, o directorio no que nos atopamos nese momento.

cd

O comando cd permítenos cambiar o directorio de traballo.

- Direccións absolutas:

```
usuario@1[~]$ cd /usr/lib/apache
usuario@1[apache]$
```

- Direccións relativas:

```
usuario@1[Documents]$ cd ../Desktop
usuario@1[Desktop]$
```

ls - touch - mkdir - ln

O comando ls permítenos visualiza-los directorios e ficheiros que hai dentro dun directorio (por defecto no directorio de traballo).

```
# Listado normal:
usuario@1[Documents]$ ls

# Listado recursivo:
usuario@1[Documents]$ ls -F -R

# Listado completo con características dos ficheiros e directorios:
usuario@1[Documents]$ ls -l

# Mostra o tamaño en formato humano:
usuario@1[Documents]$ ls -sh

# Mostra os números de i-nodo dos ficheiros e directorios:
usuario@1[Documents]$ ls -i

# Mostra os ficheiros e directorios ocultos (emprezan por .)
usuario@1[Documents]$ ls -a

# Listado completo con características do arquivo "datos1", se existe:
usuario@1[Documents]$ ls -l datos1

# Listado completo con características dos arquivos que comezan
# por "datos" e teñen un caracter mais ao final (só un caracter máis):
usuario@1[Documents]$ ls -l datos?

# Listado completo con caracteríticas dos arquivos que comezan
# por "datos" e teñen cero ou máis caracteres ao final (os que sexan):
usuario@1[Documents]$ ls -l datos*
```

En linux existen catro tipos de ficheiros:

- **Ficheiros normais (-):**

  - Son ficheiros con datos, xa sexan de texto ou binarios. Teñen un nome e poden ter unha extensión, aínda que linux non asocia tipos de ficheiros nin aplicacións segundo as extensións dos mesmos.
  - Podemos crear un ficheiro baleiro có comando touch.

```
$ touch test1
$ ls -il test1
1954793     -rw-r--r-- 1     usuario     usuario     0 Sep 1 09:35 test1
```

- **Directorios (d):**

  - Os directorios en linux son ficheiros dun tipo especial, que como información conteñen unha lista de entradas cós números de i-nodo e nome dos ficheiros e directorios que conteñen (*táboa de entradas de directorio*).
  -Para crear un directorio, usaremos o comando mkdir:

```
$ mkdir dir3
$ ls -il
total 16
1954886 drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:42 dir1/
1954889 drwxr-xr-x 2 usuario usuario 4096 Sep 1 10:55 dir2/
1954893 drwxr-xr-x 2 usuario usuario 4096 Sep 1 11:01 dir3/
1954888 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
1954793 -rw-r--r-- 1 usuario usuario 6 Sep 1 09:51 test4
$
```

- **Dispositivos:** Os dispositivos represéntanse en linux como ficheiros, que serán de tipo **b** (como os discos duros) se son dispositivos de bloque e **c** se son de carácter (como a impresora e o teclado).
- **Enlaces:** Os enlaces son ficheiros aparentemente independentes que en realidade acceden ao mesmo elemento do sistema de ficheiros. Podemos distinguir dous tipos de enlaces:

  - Enlaces débiles ou simbólicos: Un enlace simbólico é un "acceso directo" a outro ficheiro. Ao crear un enlace simbólico créase un novo ficheiro dun tipo especial (**l**) que como datos ten a ruta do ficheiro de destino. Exemplo de creación de enlace simbólico (comando ln coa opción -s):

```
$ ls -l
total 0
$ touch arquivo
$ ls -l
total 0
-rw-r--r-- 1 root root 0 sep 21 20:20 arquivo
$ ln -s arquivo arquivo-simbolico
$ ls -l
total 0
-rw-r--r-- 1 root root 0 sep 21 20:20 arquivo
lrwxrwxrwx 1 root root 7 sep 21 20:20 arquivo-simbolico -> arquivo
```

  · Si borramos **arquivo** o enlace **arquivo-simbolico** non funcionará.
  · Un enlace simbólico é como un acceso directo en Windows.

  - Enlaces fortes ou duros: Os enlaces duros son distintas rutas na árbore que apuntan ao mesmo i-nodo. Non aparecen marcados de ningún xeito especial, pero accedamos a un ou a outro en realidade estaremos accedendo ao mesmo ficheiro. Non se permite crear enlaces fortes a directorios. Exemplo de creación de enlace duro (con ln sen o -s):

```
$  ln arquivo arquivo-forte
$ ls -li
total 0
261743 -rw-r--r-- 2 root root 0 sep 21 20:32 arquivo
261743 -rw-r--r-- 2 root root 0 sep 21 20:32 arquivo-forte
261744 lrwxrwxrwx 1 root root 7 sep 21 20:32 arquivo-simbolico -> arquivo
```

  · Os cambios feitos nun arquivo quedarán reflectidos no outro.
  · Se borramos un arquivo o outro permanecerá.
  · Pódense crear enlaces simbólicos pero NON duros a directorios.
  · Os enlaces simbólicos, a diferencia dos duros, **permiten "traspasar" as fronteiras dos sistemas de arquivos**. Esto débese a que os enlaces duros "apuntan" ó arquivo orixinal a través do seu i-nodo. O i-nodo é un elemento que só ten sentido dentro dun mesmo sistema de arquivos. Coma os enlaces simbólicos están baseados en arquivos que almacenan a ruta hacia o arquivo apuntando, éstes sí poden "apuntar" a arquivos noutros sistemas de arquivos, como por exemplo un sistema de arquivo en rede, ou un sistema de arquivos noutro dispositivo.

find

O comando find permite a busca de arquivos:

Parámetros:

-name nome (nome do arquivo a buscar).

-type f dir (f->arquivos dir->directorios)

Exemplo: busca arquivos de tamaño superior a 300MB

```
# Arquivos que teñen un tamaño maior de 300k
$ find /  -type f -size +300k

# Arquivos ou directorios con nome sources.list
$ find / -name sources.list

# Arquivos ou directorios creados polo usuario patricia
$ find /home -user patricia

# Arquivos ou directorio con nome que termine en "list"
```

```
$ find /etc -name *list

# Arquivos ou directorios dentro do directorio /tmp modificados fai máis de 60 días:
$ find /tmp -mtime +60

# Arquivos ou directorios existentes no teu directorio home que foron modificados nas últimas 24 horas.
$ find $HOME -mtime 0

# Arquivos ou directorios dentro do directorio de traballo que teñen de permisos exactamente igual a rw-rw-r--:
$ find . -perm 664

# Arquivos ou directorios dentro do directorio de traballo que teñen de permisos rw-rw-r--, como mínimo:
$ find . -perm -664

# Arquivos ou directorios dentro do directorio de traballo que teñen de permisos usuario=rw- ou/e grupo=rw- ou/e outros=r--, como mí
$ find . -perm /664

# Arquivos ou directorios dentro do directorio de traballo creados polo usuario con UID 1000:
$ find . -uid 1000

# Arquivos ou directorios dentro do directorio de traballo creados por un usuario do grupo con GID 1000:
$ find . -gid 1000
```

locate

Permítenos localizar un arquivo no arbol de directorios.

```
$  locate resolv.conf
/etc/resolv.conf
/etc/resolvconf/resolv.conf.d
/etc/resolvconf/resolv.conf.d/base
/etc/resolvconf/resolv.conf.d/head
/etc/resolvconf/resolv.conf.d/original
/usr/share/man/man5/resolv.conf.5.gz

# Para buscar os arquivos que se chamen exactamente "resolv.conf"
$ locate -b '\resolv.conf'
/etc/resolv.conf

# Para actualizar os arquivos recentemente indexados
# antes de buscar, primeiro, empregar o comando:
$ updatedb
```

which

O comando which devolve a ruta dos ficheiros executables dun comando dado. Esta busca a fai sobre os directorios que están especificados na variable **PATH**.

En caso de non atopalo retornase unha mensaxe de erro especificando que o comando non foi atopado.

Para saber os directorios contidos en **PATH** débese escribir na consola:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/games/bin:/usr/local/sbin:/usr/sbin:/sbin
```

Para configurar o PATH podemos axudarnos do seguinte enlace.

Vexamos un exemplo da utilización do comando **which**:

```
$ which -a ls
/bin/ls
```

cp

Copia arquivos e directorios. O xeito de empregalo sería así: **cp orixe destino**

- Copiar un arquivo existente no directorio no que nos atopamos, *test1*, noutro arquivo chamado *test2* e que se garde nese mesmo directorio:

```
$ cp test1 test2
$ ls -il
total 0
1954793 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test1
1954794 -rw-r--r-- 1 usuario usuario 0 Sep 1 09:39 test2
$
```

Como se pode ver, o novo arquivo presenta un número de inodo novo.

- Copiar un arquivo existente no directorio no que nos atopamos, *test1*, noutro directorio chamado *dir1* existente no directorio que nos atopamos:

```
$ cp test1 dir1
$ ls -il dir1
total 0
1954887 -rw-r--r-- 1 usuario usuario 0 Sep 6 09:42 test1
$
```

O arquivo copiado sería: *./dir1/test1*.

- Pódese empregar o parámetro **-p** para que os parámetros de acceso e os datos de tempos de modificación do arquivo orixinal:

```
$ cp -p test1 test3
$ ls -il
total 4
1954886 drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:42 dir1/
1954793 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test1
1954794 -rw-r--r-- 1 usuario usuario 0 Sep 1 09:39 test2
1954888 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
$
```

Os datos de data de modificación dos arquivos orixe (*test1*) e (*test3*) coinciden.

- O parámetro **-R** é moi potente pois permítenos copiar ***recursivamente*** o contido dun directorio:

```
$ cp -R dir1 dir2
$ ls -l
total 8
drwxr-xr-x 2 usuario usuario 4096 Sep 6 09:42 dir1/
drwxr-xr-x 2 usuario usuario 4096 Sep 6 09:45 dir2/
-rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test1
-rw-r--r-- 1 usuario usuario 0 Sep 6 09:39 test2
-rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
$
```

Agora *dir2* é unha copia completa de *dir1*.

- Tamén podemos empregar o carácter ***** có comando **cp**:

```
$ cp -f test* dir2
$ ls -al dir2
total 12
drwxr-xr-x 2 usuario usuario 4096 Sep 6 10:55 ./
drwxr-xr-x 4 usuario usuario 4096 Sep 6 10:46 ../
-rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test1
-rw-r--r-- 1 usuario usuario 0 Sep 6 10:55 test2
-rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
$
```

  Có parámetro **-f** fórzase a sobreescritura dos arquivos existentes

- Se precisamos ter dúas copias do mesmo arquivo, pero sen ter fisicamente dous arquivos separados, podemos empregar unha copia física e varias copias virtuais, chamadas ***links***.

    Existen dous tipos de *links* en Linux:

- *Symbolic*, ou *soft links*.
- *Hard links*.

Un *hard link* crea un arquivo separado que contén información sobre o arquivo orixinal e onde está gardado.

```
$ cp -l test1 test4
$ ls -il
total 16
1954886 drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:42 dir1/
1954889 drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:45 dir2/
1954793 -rw-r--r-- 2 usuario usuario 0 Sep 1 09:51 test1
1954794 -rw-r--r-- 1 usuario usuario 0 Sep 1 09:39 test2
1954888 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
1954793 -rw-r--r-- 2 usuario usuario 0 Sep 1 09:51 test4
$
```

O parámetro **-l** crea un *hard link* para o arquivo *test1* chamado *test4*. Mirando o listado dos arquivos pódese ver que o número de **inodos** de *test1* e *test4* son o mesmo (2), indican que, en realidade, son ambos o mesmo arquivo.

¡Ollo!

Non se poden crear *hard links* entre arquivos gardados en distintos puntos de montaxe. Nese caso, só se poderán crear *soft links*.

- Có parámetro **-s** crearanse *symbolic*, ou *soft links*:

```
$ cp -s test1 test5
$ ls -il test*
total 16
1954793 -rw-r--r-- 2 usuario usuario 6 Sep 1 09:51 test1
1954794 -rw-r--r-- 1 usuario usuario 0 Sep 1 09:39 test2
1954888 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
1954793 -rw-r--r-- 2 usuario usuario 6 Sep 1 09:51 test4
1954891 lrwxrwxrwx 1 usuario usuario 5 Sep 1 09:56 test5 -> test1
$
```

Como se ve o novo arquivo *test5* ten un número de **inodo** distinto que o arquivo *test1*, indicando que o sistema Linux o trata como un arquivo distinto. Tamén se ve que o tamaño do arquivo *test5* é distinto do tamaño do arquivo *test1*. Por último, na zona onde aparece o nome do arquivo, aparece a relación existente entre os dous arquivos.

- Non se permite crear enlaces fortes entre directorios.

INTERÉSACHE

Tamén se pode empregar o comando **ln**. Por defecto o comando **ln** crea *hard links*. Se ti queres crear un *soft link* emprega o parámetro **-s**.

mv

El comando **mv** emprégase para mover arquivos e directorios:

```
$ mv test2 test6
$ ls -il test*
1954793 -rw-r--r-- 2 usaurio usaurio 6 Sep 1 09:51 test1
1954888 -rw-r--r-- 1 usaurio usaurio 0 Dec 25 2008 test3
1954793 -rw-r--r-- 2 usaurio usaurio 6 Sep 1 09:51 test4
1954891 lrwxrwxrwx 1 usaurio usaurio 5 Sep 1 09:56 test5 -> test1
```

```
1954794 -rw-r--r-- 1 usaurio usaurio 0 Sep 1 09:39 test6
$
```

Vese que ó mover un arquivo o nome do arquivo cambia pero ten o mesmo inodo e data de creación polo que o arquivo en si non cambiou.

- O problema é mover un arquivo que teña *soft links*:

```
$ mv test1 test8
$ ls -il test*
total 16
1954888 -rw-r--r-- 1 usuario usuario 0 Dec 25 2008 test3
1954793 -rw-r--r-- 2 usuario usuario 6 Sep 1 09:51 test4
1954891 lrwxrwxrwx 1 usuario usuario 5 Sep 1 09:56 test5 -> test1
1954794 -rw-r--r-- 1 usuario usuario 0 Sep 1 09:39 test6
1954793 -rw-r--r-- 2 usuario usuario 6 Sep 1 09:51 test8
[rich@test2 clsc]$ mv test8 test1
```

O arquivo *test4* que ten o *hard link* emprega o mesmo número de inodo polo que non hai problema. Pero fixádevos que agora, **o arquivo *test5* apunta agora a un arquivo non válido** (Por iso aparecerá en vermello).

- Tamén se pode empregar o comando *mv* para mover directorios:

```
$ mv dir2 dir4
```

Vese que o único que cambiou é o nome do directorio e nada mais.

rm

O comando rm nos permite **borrar** arquivos.

- En Linux, borrar denomínase *remover*. O xeito mais básico de chamar ao comando **rm** é o seguinte:

```
$ rm -i test6
rm: ¿Borrar o arquivo regular baleiro `test6?? (s/n) s
$ ls -l
total 8
drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:42 dir1/
drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:45 dir4/
-rw-r--r-- 1 usuario usuario 6 Sep 1 09:51 test3
-rw-r--r-- 2 usuario usuario 0 Dec 25 2008 test4
lrwxrwxrwx 1 usuario usuario 5 Sep 1 09:56 test5 -> test1
-rw-r--r-- 2 usuario usuario 6 Sep 1 09:51 test8
$
```

¡Ollo!

Notar que o comando pregunta se se está seguro de eliminar o arquivo. Hai que darse conta de que no *bash shell* non existe ningún tipo de papeleira, polo que se eliminamos un arquivo é para sempre.

- Agora, eliminaremos un arquivo que ten un **enlace asociado**, para facer esta práctica primeiramente eliminamos o arquivo *soft link* **test5** e logo creamos (tal e como se veu antes) un arquivo que conteña a palabra *Ola* chamado **test1**, un *hard link* de dito arquivo chamado **test2** e un *soft link* do arquivo **test1** chamado **test5**. Unha vez feito isto podemos xa eliminar o arquivo **test1** e ver que pasa:

```
$ rm test1
$ ls -l
total 16
drwxr-xr-x 2 usuario usuario 4096 Sep 1 09:42 dir1/
drwxr-xr-x 2 usuario  usuario 4096 Sep 1 09:45 dir4/
-rw-r--r-- 1 usuario  usuario 0 Dec 25 2008 test2
-rw-r--r-- 1 usuario  usuario 6 Sep 1 09:51 test3
```

```
lrwxrwxrwx 1 usuario  usuario 5 Sep 1 09:56 test5 -> test1
$ cat test2
ola
$ cat test5
cat: test5: Non existe o ficheiro ou directorio
```

Removeuse o arquivo **test1**, que tiña asociado tanto un *hard link* có arquivo **test2** como un *soft link* có arquivo **test5**. Como se ve, os dous arquivos seguen aparecendo unha vez eliminado o **test1**. Cando se mira o contido do arquivo **test2**, que era o *hard link*, este aínda mostra o contido do arquivo. Pero cando se mira o contido do arquivo **test5**, que era un *soft link*, vese que está totalmente baleiro.

rmdir

Borrar directorios pode ser difícil, pero hai un motivo para que isto sexa así. Existen moitas posibilidades de que algo moi malo ocorra cando alguén se pon a borrar directorios. O *shell bash* trata de protexernos de catástrofes accidentais todo o posible. O comando básico para eliminar directorios é **rmdir**.

- O seguinte comando elimina o directorio **dir3**, que é un directorio baleiro:

```
$ rmdir dir3
$
```

- Por defecto, o comando **rmdir** só elimina directorios baleiros. Se intentamos eliminar o directorio **test1**, que non é un directorio baleiro, pasará o seguinte:

```
$ rmdir dir1
rmdir: Non se puido eliminar 'dir1': O directorio non está baleiro
$
```

Como o directorio **dir1** ten un arquivo, o comando **rmdir** non o elimina.

- Pódense eliminar directorios baleiros empregando o comando **rm**, se o intentamos empregar sen parámetros para eliminar o directorio **dir1** ocorre o seguinte:

```
$ rm dir1
rm: non se pode borrar 'dir1': É un directorio
$
```

- Sen embargo, se realmente queres eliminar un directorio, podes empregar o parámetro **-r** para eliminar recursivamente os arquivos do directorio, e o directorio en si:

```
$ rm -r dir1
$
```

Se vos pide confirmación para eliminar emprega o parámetro **-rf**.

stat

O comando *stat* nos da unha completa información dun arquivo no sistema de arquivos: Para facer o seguinte exemplo crearemos un arquivo (**test10**)de texto con varias liñas escritas alternadas con liñas en branco. Logo executaremos o seguinte comando:

```
$ stat test10
File: test10
Size: 40                        Blocks: 8               IO Block: 4096      arquivo regular
Device: 801h/2049d              Inode: 347187            Links: 1
Access: (0644/-rw-r--r--)       Uid: ( 1000/usuario)     Gid: ( 1000/ usuario)
Acess: 2008-11-05 23:21:01.000000000 +0100
Modify: 2008-11-05 23:21:01.000000000 +0100
Change: 2008-11-05 23:21:01.000000000 +0100
$
```

file

O comando *file* permítenos coñecer de que tipo é un arquivo determinado. Distingue entre tres tipos distintos:

- **Arquivos de texto:** Arquivos que conteñen caracteres imprimibles.
- **Arquivos executables:** Arquivos que se poden executar no sistema.
- **Arquivos de datos:** Arquivos que conteñen caracteres binarios non imprimibles, pero que non se poden executar no sistema.

```
$ file test10
test10: ASCII text
$
```

cat

O comando *cat* nos permite ver os datos existentes no interior dun arquivo de texto:

```
$ cat test10
ola

IES San Clemente

apuntes de linux
$
```

Este comando ten varios parámetros interesantes:

- O comando **-n** fai que se mostren o número das liñas nos arquivos de texto:

```
$ cat -n test10
    1 ola
    2
    3 IES San Clemente
    4
    5 apuntes de linux
$
```

- Se queremos que numere só as liñas que teñen texto empregarase o parámetro **-b**:

```
$ cat -b test10
    1 ola

    2 IES San Clemente

    3 apuntes de linux
$
```

Se o arquivo é moi longo o comando mostrará todo o contido sen parar, para resolver este "problema" tense o comando *more*.

more

El comando more saca por pantalla o contido dos arquivos de texto, pero para despois de mostrar cada páxina.

O comando *more* permite un movemento moi rudimentario ao traveso do arquivo de texto. Para ter mais liberdade de movementos e mais opcións o mellor é empregar o comando *less*.

less

Este comando é, simplemente, unha versión avanzada do comando *more* (***less is more***).

O comando *less* nos permite avanzar e retroceder ao longo do arquivo de texto.

O comando *less* tamén ten a propiedade de ser capaz de mostrar o contido dun arquivo antes de que o sistema termine de leelo... esta característica non a ten nin *more* nin *cat*. Polo demais, tamén se comporta como *more* no sentido de que, por defecto, mostra os arquivos de páxina en páxina.

Fixarse que o comando *less* nos da información adicional, como pode ser o número total de liñas existente no arquivo, e o rango de liñas mostrado por pantalla.

O comando *less* soporta moitas opcións á hora da execución como as flechas de arriba e abaixo, avance e retroceso de páxina,... (para mais información o mellor é acudir á axuda).

tail

O comando *tail* nos devolve o último grupo de liñas do arquivo. Por defecto mostra as últimas 10 liñas, pero pódese cambiar isto empregando parámetros (**-n nºdeliñas**).

- **Seguimento de arquivos:**

  tail ten unha opción especial, **-f** (do inglés *follow*, seguir), que permite facer seguimento a un arquivo. En lugar de mostrar as últimas liñas e terminar, *tail* mostrará as últimas liñas e seguirá lendo do archivo; conforme se lle engadan novas liñas, *tail* as imprimirá. Esta función é particularmente útil para arquivos de rexistro.

- Para pechar *tail* cando estea facendo seguimento, chega con interrumpilo con Ctrl+C.

head

Este comando mostra, por defecto, as 10 primeiras liñas do arquivo que se pase como parámetro.

e2fsck

e2fsck é empregado para chequear un sistema de arquivos ext2.

Exemplo:

```
umount /
e2fsck /dev/...
```