 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGs DAW DEPT. INF.
---	--	--

ÍNDEX

0. El Shell o terminal	3
1. Ordres bàsiques per a començar.....	4
exit:	4
who [am i]:.....	4
write:.....	4
mesg:	5
date:	5
cal:	5
uname:	5
passwd:	6
clear:	6
script:.....	6
2. Ordres del sistema d'arxius	8
cd:	8
ls:	8
pwd:	10
mkdir:	10
rmdir:	10
3. Editor de text: nano.	10
4. Ús d'arxius i permisos	11
cat:	11
more:	11
head:	11
tail:	11
cp:	11
mv:	12
rm:	12
file:	12
chmod:	13
umask:	14
whereis:	14
id:	14
su:	15
sudo:	15
newgrp:	16



Generalitat de Catalunya
Departament d'Ensenyament
Institut Joaquim Mir
Vilanova i la Geltrú

M01. SISTEMES INFORMÀTICS
UF1. INSTAL·LACIÓ, CONFIGURACIÓ
I EXPLOTACIÓ DEL SISTEMA
INFORMÀTIC


CFGS
DAW

DEPT.
INF.



Codi doc: MO-0044
Versió núm.: 3

Aquesta còpia pot ser obsoleta un cop impresa. Comprova que coincideixi amb la versió disponible a la intranet del centre.

 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGS DAW DEPT. INF.
---	--	--

0. El Shell o terminal

GNU/Linux era, originàriament, un sistema operatiu basat en text, i d'això hereta moltes de les propietats actuals que té un GNU/Linux modern.

El *shell* de Linux (o terminal) és el programa que ens permet treballar amb ordres basades en text i és important conèixer-lo en profunditat ja que la majoria de tasques que ha de realitzar un administrador de sistemes les pot fer treballant amb ordres del terminal.

A GNU/Linux existeixen un gran nombre de “programes terminal”, dels quals els més importants o coneguts són els següents:

bash	GNU B ourne A gain S hell	El més popular , una extensió del <i>shell</i> Bourne . Està per defecte en la majoria de distribucions actuals.
bsh	GNU B ourne S hell	No s'utilitza massa actualment, al haver estat substituït per bash.
tcsh		Variant del C Shell, popular en alguns cercles, i molt similar a bash, però amb algunes diferències en la forma d'operar.
csh	C Shell	No massa utilitzat.
ksh	GNU Korn Shell	Barreja i extensió de bsh i csh. Té un petit cercles d'usuaris molt adeptes.
zsh	Z Shell	Evolució del ksh.

Existeixen una gran nombre de *shells* més, però tenen molt poca rellevància. Més endavant ja veurem com canviar el *shell* per defecte, que a Ubuntu és bash.

Si ens trobem en un Linux amb entorn de text (p. ex. Ubuntu Server), no caldrà fer res. Un cop haguem fet *login*, el propi Linux ens deixa a mans del terminal, que serà la única eina d'interacció entre usuari i el sistema.

Si ens trobem en un entorn gràfic (p. ex. Ubuntu Desktop) caldrà executar algun programa, com el terminal, el xterm o el konsole. A Ubuntu serà terminal.



1. Ordres bàsiques per a començar

exit:

Finalitza la sessió de feina i torna a aparèixer el missatge de *login* per a que es connecti un altre usuari. És aconsellable desconnectar sempre per qüestions de seguretat. Els terminals no estan assignats de manera fixa a cada usuari, per tant podem entrar amb la mateixa identitat a terminals diferents. Si estem treballant amb un escriptori gràfic tenim 7 terminals diferents. Els 6 primers són textuals i el 7 és el gràfic. Per a entrar en un d'aquests terminals farem CTRL + ALT + Fn (a on Fn seran les tecles F1 fins a F7).

who [am i]:

Ens informa dels usuaris que estan connectats actualment al sistema (1a columna), el terminal en el que es troben (2a columna) i la data i la hora a la que va entrar (3a columna). Si fem `who am i` ens informa de nosaltres mateixos.

```
armand@ubuntu:~$ who
linux    tty1      2015-09-27 12:02
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
david    pts/3      2015-09-27 12:05 (192.168.0.101)
armand@ubuntu:~$ who am i
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
armand@ubuntu:~$
```


write:

Es fa servir per a comunicar-nos amb els altres usuaris que estan connectats al sistema en aquest moment. Escrivim el missatge i per a finalitzar-lo fem CTRL+D.

Exemple:

```
armand@ubuntu:~$ who
linux    tty1      2015-09-27 12:02
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
david    pts/3      2015-09-27 12:05 (192.168.0.101)
armand@ubuntu:~$ who am i
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
armand@ubuntu:~$ write david
hola bon dia!!!
armand@ubuntu:~$
```

```
Bitwise xterm - jmir.bscp - david@192.168.0.100:22 - david@ubuntu: ~
david@ubuntu:~$
Message from armand@ubuntu on pts/1 at 12:15 ...
hola bon dia!!!
EOF
david@ubuntu:~$
```

 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGS DAW DEPT. INF.
---	--	--

mesg:

Activa o desactiva la recepció de missatges. Per a activar els missatges farem `$mesg y` i per a desactivar-los `$mesg n`

date:

Informa de l'hora i la data actuals. Fent servir `%` precedit de `+` podem donar format a la data amb els següents modificadors:

- `%d`: Dia
- `%m`: Mes
- `%y`: Any
- `%w`: Dia de la setmana.

Exemple:

```
armand@ubuntu:~$ date
dg set 27 12:24:07 CEST 2015
armand@ubuntu:~$ date +"Són les %r hores del %d de %m de %y"
Són les 12:24:11 hores del 27 de 09 de 15
armand@ubuntu:~$
```

cal:

Ens mostra un calendari. La seva sintaxi és: `cal [mes] [any]`. Es pot fer servir de les següents maneres:

- **cal**: Ens mostra el calendari del mes i l'any actuals
- **cal any**: Ens mostra tot el calendari de l'any especificat.
- **cal mes any**: Ens mostra el calendari del mes i l'any especificat.

Exemples:

```
linux@ubuntu:~$ cal
Setembre 2015
dg dl dt dc dj dv ds
   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

linux@ubuntu:~$ cal 10 2016
Octubre 2016
dg dl dt dc dj dv ds
   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
linux@ubuntu:~$
```

uname:

Aquesta comanda es fa servir per a obtenir informació del sistema, tipus de màquina, sistema operatiu, tipus de processador, ... La seva sintaxi és `uname [-a][-m][-n][--v][-s][-r]`. Veiem per a què serveix cada modificador i exemples del seu ús:



- -a: Mostra tota la informació
- -m: Ens dóna el tipus de hardware i processador.
- -n: Ens dóna el nom del host.
- -v: Ens dóna la informació de quan va ser instal·lat el sistema.
- -s: Ens dóna el nom del sistema.
- -r: Ens dóna la versió del nucli del sistema.

Exemple:

```
armand@ubuntu:~$  
armand@ubuntu:~$ uname -a  
Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux  
armand@ubuntu:~$ uname -r  
3.13.0-32-generic  
armand@ubuntu:~$ uname -s  
Linux  
armand@ubuntu:~$ uname -v  
#57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014  
armand@ubuntu:~$ uname -m  
x86_64  
armand@ubuntu:~$ uname -n  
ubuntu  
armand@ubuntu:~$
```

passwd:

Ens permet modificar la nostra contrasenya. **ATENCIÓ:** En general no es pot fer servir qualsevol tipus de contrasenya. Depenent de la distribució podem tenir, per exemple, els següents requeriments: que la contrasenya sigui més llarga de 6 caràcters, diferent de l'usuari i de la darrera contrasenya.

clear:

Serveix per a netejar la pantalla.

script:

Aquesta comanda ens permet emmagatzemar en un arxiu tot el que l'usuari teclegi a partir del moment en que ho activem. És molt útil per a guardar una sessió de feina. Sintaxi: `script [-a] [nom_arxiu]`. Es fa servir de la següent manera:

- **script:** Emmagatzema tota la sessió en un arxiu anomenat `typescript`.
- **script nom_arxiu:** Emmagatzema tota la sessió en un arxiu anomenat `nom_arxiu`. Si aquest arxiu ja existeix l'esborra i crea un de nou.
- **script -a nom_arxiu:** Emmagatzema tota la sessió en un arxiu anomenat `nom_arxiu`. Si aquest arxiu ja existeix no l'esborra sinó que el completa.
- Per a finalitzar l'execució de `script` fem servir **exit**.



```
david@ubuntu:~$
david@ubuntu:~$
david@ubuntu:~$ script -a registre
S'ha iniciat l'execució de script, el fitxer és registre
david@ubuntu:~$ who
linux    tty1      2015-09-27 12:02
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
david    pts/3      2015-09-27 12:05 (192.168.0.101)
david@ubuntu:~$ whoami
david
david@ubuntu:~$ exit
exit
S'ha fet la seqüència, el fitxer és registre
david@ubuntu:~$ cat registre
S'ha iniciat l'execució de script a dg 27 set 2015 12:34:38 CEST
david@ubuntu:~$ who
linux    tty1      2015-09-27 12:02
armand   pts/1      2015-09-27 12:04 (192.168.0.101)
david    pts/3      2015-09-27 12:05 (192.168.0.101)
david@ubuntu:~$ whoami
david
david@ubuntu:~$ exit
exit
S'ha finalitzat l'execució de script a dg 27 set 2015 12:34:49 CEST
david@ubuntu:~$
```

man:

Totes les ordres del SO es troben descrites al Manual del Programador de Linux. Aquest manual ens dóna la informació sobre totes les ordres de Linux i està dividit en les següents seccions:


1. Ordres i programes d'aplicació.
2. Trucades al sistema.
3. Subrutines.
4. Dispositius.
5. Format d'arxis.
6. Jocs.
7. Miscel·lània.
8. Manteniment.

La sintaxi de man és: `man [secció] ordre`

Exemple:

```
clear(1) clear(1)
NAME
    clear - clear the terminal screen
SYNOPSIS
    clear
DESCRIPTION
    clear clears your screen if this is possible. It looks in the environ-
    ment for the terminal type and then in the terminfo database to figure
    out how to clear the screen.
    clear ignores any command-line parameters that may be present.
SEE ALSO
    tput(1), terminfo(5)
    This describes ncurses version 5.6 (patch 20070716).
Manual page clear(1) line 1/25 (END) clear(1)
```

Si ens fixem a l'exemple veiem que hem executat `man clear` i ens ha donat tota la informació sobre la comanda `clear`. A més en diu `clear(1)` que vol dir que `clear` pertany a la secció 1. En general la informació sobre les comandes és molt extensa. Per moure'ns pel document farem servir `AvPag`, `RePag`, els cursors i per sortir farem `q`.

 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGS DAW DEPT. INF.
---	--	--

2. Ordres del sistema d'arxius

Veurem ara les comandes bàsiques per a tractar directoris i fitxers.

cd:

Com sabem a Linux es crea el directori arrel (/) que és on s'instal·la el sistema operatiu. La resta de particions i discs es consideren subdirectoris del directori arrel.

L'estructura de directoris es fa en format d'arbre (uns directoris dins d'uns altres). Els directoris d'usuaris es troben dins de la carpeta /home. Per a cada usuari del sistema es crea, dins de la carpeta anterior, un carpeta amb el nom de l'usuari, per exemple, si creem l'usuari alumne es crearà la carpeta /home/alumne.

La comanda cd es fa servir per a moure's pels directoris. Tots els directoris tenen dos subdirectoris especials, els subdirectoris . i el .. El primer es una referència al propi directori contenidor i el .. fa referència al directori pare. Per exemple, si ens troben a /home/alumne i fem cd .. passarem a estar al directori /home.

L'accés a un directori es pot realitzar de dues formes:

- Accés amb ruta absoluta: Per accedir a un directori posem el camí complet d'accés des del directori arrel. Per exemple si estem a /home/alumne i volem anar a /home/pere/Documents farem cd /home/pere/Documents. L'avantatge d'aquest tipus d'accés és que sempre és vàlid i la desavantatge és que per arribar a generar rutes molt llargues.
 - Accés amb ruta relativa: Podem accedir a un directori fent servir els subdirectoris especials .. Per exemple, si estem a /home/alumne i volem anar a /home/pere/Documents farem cd ../pere/Documents. L'avantatge d'aquest tipus d'accés és que ens estalviem escriure la ruta absoluta i la desavantatge és que aquestes rutes només són vàlides en funció de la posició que tenim al sistema de fitxers.

Si no especifiquem un directori concret ens envia al nostre HOME. Sintaxi: cd [directori]

Exemples:

- cd .. : Ens envia al directori pare.
- cd /etc : Anem al directori etc que es troba al directori arrel.
- cd ../etc : Accés al directori etc mitjançant una ruta relativa.

ls:

Llista els arxius continguts en un determinat directori. Sintaxi: ls [-lFad] [directori][arxius]

- ls: Dóna com a sortida tots els noms de fitxers i directoris.
- ls -F: Dóna com a sortida la mateixa llista anterior però afegint una símbol per a cada fitxer o directori que ens indica què és.

/	Directori
@	Enllaç simbòlic



*	Executable
=	Socket (arxiu especial emprat per comunicació en xarxa)
	Pipe (arxiu especial emprat per comunicar dos programes)

```
linux@ubuntu:~$ ls
adeuu  adre  holaa  ip  material  treballs
linux@ubuntu:~$ ls -F
adeuu  adre@  holaa  ip*  material/  treballs/
linux@ubuntu:~$ ls -l
total 312
-rw-rw-r-- 1 linux adm      0 set 27 18:43 adeuu
lrwxrwxrwx 1 linux adm      2 set 27 18:51 adre -> ip
-rw-rw-r-- 1 linux linux    0 set 27 18:42 holaa
-rwxr-xr-x 1 linux adm 307328 set 27 18:50 ip
drwxrwxr-x 2 linux adm   4096 set 27 18:50 material
drwxrwxr-x 2 linux adm   4096 set 27 18:50 treballs
linux@ubuntu:~$
```

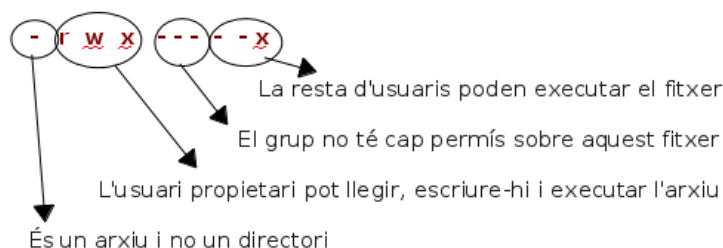
`ls -l`: Dóna com a sortida una informació el més extensa possible. El resultat d'una comanda com l'anterior pot ser el següent:

```
...
-rw-r--r-- 1 linux linux 144183 2007-04-09 12:10 Captura.png
drwxr-xr-x 3 root root 4096 2007-03-15 09:46 localhost
(1)      (2) (3)  (4)  (5)      (6)      (7)  (8)
```

(1). Permisos associats al fitxer o directori . Es desglossa de la següent forma:

d → directori
- → arxiu

Els 9 caràcters següents s'agrupen de tres en tres. El guió "-" indica absència de permís. 'r' lectura, 'w' escriptura i 'x' execució per fitxer o accés per directoris. El primer grup de tres pertanyen als permisos associats a l'usuari propietari del fitxer- El segon grup al grup i el tercer als altres usuaris. Un exemple:




Quan els permisos estan associats a un directori r,w i x tenen un significat diferent. 'r' implica poder llegir el directori (fer un ls), 'w' crear i esborrar fitxers en aquest directori i 'x' és l'accés (el que fem amb cd). Un exemple:

```
drwxr-xr-x 2 usuari users 48 Mar 25 18:44 dirprova
```

A dirprova hi poden accedir tots els usuaris. També el poden llistar tots els usuaris. Però només usuari hi pot crear o eliminar arxius

(2). Número d'enllaços durs que té el fitxer. Si és un directori indica el número de subdirectoris que conté (inclou les entrades . i .. que representen el directori actual i el directori pare.).

 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGS DAW DEPT. INF.
---	--	--

- (3). Usuari propietari del fitxer o directori
- (4). Grup al qual pertany l'arxiu o directori.
- (5). Grandaia en bytes del fitxer. En el cas de directoris fa referència a l'espai ocupat per l'entrada al sistema de fitxers (4096 bytes habitualment) i no l'espai del que conté.
- (6). Data de creació o modificació.
- (7). Hora de creació o modificació.
- (8). Nom del fitxer o directori.

- `ls -a`: Ens permet veure tots els arxius, fins i tot els arxius ocults (que a Linux són els arxius que comencen amb el caràcter `.`).
- `ls -ld directori`: Ens permet veure les propietats d'un directori i no el seu contingut.
- `ls -R directori`: Ens permet realitzar un llistat recursiu. És a dir que quan llistem un directori, ens mostri també el contingut dels seus subdirectoris. I dins del contingut d'aquests subdirectoris, si en troba més també mostra els seus continguts, ...

pwd:

Ens mostra el directori de feina actual en forma de camí absolut.

```
david@ubuntu:/etc/network$ pwd
/etc/network
david@ubuntu:/etc/network$ PS1="$> "
$> pwd
/etc/network
$> █
```

mkdir:

Aquesta comanda ens permet crear directoris. Es poden fer servir rutes absolutes o relatives. En general només els podrem crear al nostre HOME (l'únic lloc a on tenim permisos).

- `mkdir /home/dlopez/src` Crearia el directori `src` al nostre HOME.
- `mkdir src` Crearia el directori `src` al directori actiu.

rmdir:

Ens permet esborrar directoris (aquest ha d'estar buit). Es poden fer servir rutes absolutes o relatives. En general només podrem esborrar els directoris del nostre HOME.

- `rmdir src` Esborra el directori `src` que es troba al directori actiu.
- `rmdir /home/dlopez/src` Esborra el directori `src` que es troba al nostre HOME.

3. Editor de text: nano.

Per a crear i modificar fitxers podeu fer servir l'editor de texts nano. Per a obrir un fitxer només cal executar `nano nom_del_fitxer`.



```
GNU nano 2.2.6 Fitxer: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
```

^G Ajuda ^O Desa ^R Llegeix ^Y Pàg Ant ^K Retalla ^C Pos Act
^X Surt ^J Justifica ^W Cerca ^V Pàg Seg ^U Destalla el ^T Ortografia

4. Ús d'arxius i permisos

cat:

Serveix per a visualitzar el contingut d'arxius ASCII per pantalla. La seva sintaxi és: `cat [arxiu(s)]`. Si no incloem el nom de l'arxiu llavors llegeix des de la pantalla fins que l'usuari fa CTRL+D i després mostra el que s'ha escrit per pantalla.

more:

Imprimeix per pantalla un arxiu de text visualitzant-lo pantalla a pantalla.

La seva sintaxi és: `more [arxiu(s)]`. Per a moure's per a les diferents pantalles fem:

- Barra espaciadora: Saltar a la pròxima pantalla.
- Retorn de carro: Saltar una línia.
- q: Sortir.

head:

Ens permet visualitzar les primeres línies d'un fitxer de text. Sintaxi: `head [-N] arxiu`. Si posem el paràmetre -N ens mostra les N primeres línies, si no el posem ens mostra les 10 primeres línies.

tail:

Ens permet visualitzar les últimes línies d'un fitxer de text. Sintaxi: `tail [-N] arxiu`. Si posem el paràmetre -N ens mostra les N últimes línies, si no el posem ens mostra les 10 últimes línies.


cp:

Aquesta comanda ens permet copiar arxius entre els diferents directoris del sistema. La seva sintaxi és: `cp arxiu(s) destí`.

Es poden fer servir rutes absolutes o relatives. Si el nom del destí és un directori copiarà l'arxiu dins d'ell amb el mateix nom.

Veiem uns exemples:

- `cp serv.c /home/david/src/` – Copia l'arxiu `serv.c` a la carpeta `/home/david/src/` amb el mateix nom.

 Generalitat de Catalunya Departament d'Ensenyament Institut Joaquim Mir Vilanova i la Geltrú	M01. SISTEMES INFORMÀTICS UF1. INSTAL·LACIÓ, CONFIGURACIÓ I EXPLOTACIÓ DEL SISTEMA INFORMÀTIC	CFGS DAW DEPT. INF.
---	--	--

- `cp /etc/apache2.conf /home/david/src/hola.txt` – Copia l'arxiu `apache2.conf` que es troba a la carpeta `/etc`, a la carpeta `/home/david/src/` amb el nom `hola.txt`.

Algunes de les opcions que podem afegir al `cp` són:

- `-f`: Opció *force*. Força l'ordre a sobreescriure els fitxer que ja existeixen a la destinació sense preguntar. És la opció per defecte.
- `-i`: Opció *interactive*. Pregunta per a cada fitxer que s'ha de sobreescriure.
- `-p`: Opció *preserve*. Preserva els permisos i el propietari de l'arxiu origen.
- `-R`: Opció *recursive*. Realitzar una còpia recursiva. Es copia el directori sencer i tots els subdirectoris.
- `-u`: Opció *update*. Fa la còpia només si l'arxiu original és més nou que l'arxiu de destinació o no existeix la destinació.

mv:

La comanda `mv` és igual a la `cp` però en aquest cas l'arxiu no es copia sinó que es mou (l'arxiu original desapareix). Aquesta comanda també es fa servir per a renombrar arxius i directoris.

La seva sintaxi és: `mv arxiu(s) destí`. Veiem uns exemples:

- `mv Fotos/img.jpg tmp/` – Mou l'arxiu `img.jpg` de la carpeta `Fotos` a la carpeta `tmp` amb el mateix nom.
- `mv Fotos/img.jpg tmp/fotoGran.jpg` – Mou l'arxiu `img.jpg` de la carpeta `Fotos` a la carpeta `tmp` amb el nom `fotoGran.jpg`.
- `mv texto.txt texto2.txt` – Canvia el nom de l'arxiu `texto.txt` pel de `texto2.txt`.

rm:

Es fa servir per a esborrar arxius. La seva sintaxi és: `rm [-irf] arxiu(s)`. Les opcions són les següents:

- `-f` (force): Força l'esborrament de l'arxiu fins i tot si és de només lectura (l'usuari ha de ser el propietari de l'arxiu)
- `-i` (interactive): Pregunta abans d'esborrar l'arxiu. Útil si fas servir comodins per a identificar els arxius.
- `-r` (recursive): Esborra tots els arxius de subdirectoris dependents.

Exemples:

- `rm hola.txt` – Esborra l'arxiu `hola.txt`.
- `rm -f hola.txt` – Esborra l'arxiu `hola.txt` (fins i tot si és de només lectura)
- `rm -i /home/david/*` – Esborra tots els arxius de la carpeta `/home/david/` preguntant abans de fer-ho.
- `rm -r /home/david/*` – Esborra tots els arxius i directoris de la carpeta `/home/david/`.

file:

Linux no imposa cap format sobre els arxius. Per a saber de quin tipus és un arxiu concret fem servir `file`. La seva sintaxi és: `file arxiu(s)`.

Exemple:



```
linux@ubuntu:~$ file fitxer
fitxer: ASCII text
linux@ubuntu:~$ file teoria
teoria: directory
linux@ubuntu:~$ file ip2
ip2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (u
[sha1]=24f3c97aa8ce8d03471feb47285e9053d47f5118, stripped
linux@ubuntu:~$
```

touch:

Més endavant veurem més d'aquesta ordre, però de moment simplement la farem servir per crear arxius buits, per a fer les pràctiques o proves que convinguin

```
$ touch arxiu
```

Crea l'arxiu amb nom arxiu1.

dd:

dd és una ordre que emprarem més endavant per copiar sectors de disc a un arxiu. És una ordre que s'empra per fer còpies o imatges de disc. De moment, la podem emprar per crear arxius d'una mida determinada:

```
$ dd if=/dev/zero of=fitxer_creat bs=1MB count=100
$ dd if=/dev/random of=fitxer_creat bs=1MB count=100
```

El primer crea un arxiu de 100MB omplert amb "zeros" i el segon crea un arxiu amb contingut aleatori. Aquest darrer pot ser molt lent.

chmod:

Cada arxiu té associat un conjunt de permisos que determinen que pot fer un usuari amb ell.

Existeixen 3 tipus d'usuaris:

- Propietari de l'arxiu.
- Grup del propietari.
- Resta d'usuaris.

La determinació de permisos es realitza amb un sèrie de 9 caràcters. Cada caràcter pot ser:

- r: Permís de lectura.
- w: Permís d'escriptura.
- x: Permís d'execució.
- -: Sense permís.
-

Exemple: rw-r--r-- Aquest conjunt de permisos en diu que:

- L'usuari propietari pot llegir i escriure sobre el fitxer però no el pot executar.
- El grup al que pertany l'usuari propietari pot només llegir el document.

- La resta d'usuaris pot només llegir el document.
- Aquesta comanda ens permet modificar els permisos però hem de ser propietaris de l'arxiu o administrador. Sintaxi: `chmod mode arxiu(s)`. Hi ha dues formes de fer-ho servir:

- Triem els permisos, per exemple `rxr-xr--`. I determinem l'equivalència numèrica fent la següent taula.

Permisos	Propietari	Grup	Altres
Lletres	rwX	r-x	r--
Binari	111	101	100
Decimal	7	5	4

I llavors fem `chmod 754 arxiu`.

- Amb les lletres `u`, `g` i `o` determinarem a l'usuari propietari, el seu grup i la resta d'usuaris respectivament. Amb el `+` afegirem un permís i amb el `-` el traurem.

Exemples:

- `chmod u-w arxiu`
- `chmod o+r arxiu`

umask:

Quan creem un arxiu aquest té uns permisos per defecte. Si volem canviar aquests permisos haurem de fer servir la comanda `umask`. La seva sintaxi és `umask [màscara]` i funciona de la següent manera:

- Els permisos màxims que un arxiu pot rebre en crear-se són `rw-rw-rw-`, que traduït a decimal és 666.
- Si nosaltres volem el permís `rw-r-----` pels arxius creats, el primer que hem de fer és passar-ho a decimal, en el nostre cas el valor és 640 i després fer la resta $666 - 640 = 26$. Llavors farem `umask 26` per a aplicar els permisos.
- Si fem `umask` sense especificar la màscara ens informarà dels permisos actius en aquell moment.

whereis:

Els paràmetres que accepta aquesta comanda són ordres del SO Linux i ens retorna el directori a on resideix l'ordre i la pàgina del manual. La sintaxi és: `whereis [-b] [-m] [-s] ordre(s)`. Els paràmetres són els següents:

- `-b`: Només cerca l'arxiu binari.
- `-m`: Només cerca la pàgina del manual.
- `-s`: Cerca el codi font.

```
linux@ubuntu:~$
linux@ubuntu:~$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
linux@ubuntu:~$
```

id:

Retorna l'identificador d'usuari i de grup, que no és res més que un número que fa servir Linux per a saber a qui ha d'aplicar els permisos. La sintaxi és: `id [-ug] [usuari]`.

Els paràmetres són:



- Sense paràmetres: mostra el nostre id.
- -u: Mostra únicament l'identificador d'usuari.
- -g: Mostra únicament l'identificador de grup.

```
armand@ubuntu:/home/linux$  
armand@ubuntu:/home/linux$ id david  
uid=1001(david) gid=1001(david) grups=1001(david)  
armand@ubuntu:/home/linux$
```

```
linux@ubuntu:~$ id  
uid=1000(linux) gid=1000(linux) grups=1000(linux),4(adm),24(cdrom),27(sudo),30(dip),  
,46(plugdev),108(lpadmin),110(sambashare)
```

su:

Ens permet canviar els nostre identificador d'usuari i deixar-nos actuar com un altre usuari (evidentment ens demana la password).

La sintaxi és:

`su [-] [usuari].`

El funcionament és el següent:

- Si no especifiquem usuari és com si intentem identificar-nos com a root (administrador).
- -: Serveix per a prendre els paràmetres d'inici de l'usuari al que ens convertim.
- Per sortir de la sessió del nou usuari fem exit.

```
linux@ubuntu:~$  
linux@ubuntu:~$ su armand  
Contrasenya:  
armand@ubuntu:/home/linux$ pwd  
/home/linux  
armand@ubuntu:/home/linux$ exit  
exit  
linux@ubuntu:~$ su - armand  
Contrasenya:  
armand@ubuntu:~$ pwd  
/home/armand  
armand@ubuntu:~$ exit  
logout  
linux@ubuntu:~$ su -  
Contrasenya:  
root@ubuntu:~# pwd  
/root  
root@ubuntu:~#
```

sudo:

Aquesta ordre ens permet executar una comanda com si fóssim l'usuari administrador (evidentment ens demana la password).

La sintaxi és:

`sudo ordre.`

Pot ocórrer que des d'algun usuari no ens permeti fer-ho dient-nos que el nostre usuari no es troba al fitxer sudoers. Cal que el teu usuari es trobi en el grup "admin" o "sudo" del teu sistema. Més endavant ja veurem com fer-ho.



```
armand@ubuntu:/home/linux$  
armand@ubuntu:/home/linux$ sudo rm *  
[sudo] password for armand:  
armand is not in the sudoers file. This incident will be reported.  
armand@ubuntu:/home/linux$ █
```

newgrp:

Aquesta comanda ens permet canviar d'identificador de grup (un usuari pot pertànyer a molts grups).
La sintaxi és: `newgrp grup`.

```
linux@ubuntu:~$  
linux@ubuntu:~$ id  
uid=1000(linux) gid=1000(linux) groups=1000(linux),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),110(sambashare)  
linux@ubuntu:~$ touch holaa  
linux@ubuntu:~$ ls -l  
total 0  
-rw-rw-r-- 1 linux linux 0 set 27 18:42 holaa  
linux@ubuntu:~$ newgrp adm  
linux@ubuntu:~$ touch adeuu  
linux@ubuntu:~$ id  
uid=1000(linux) gid=4(adm) groups=1000(linux),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),110(sambashare)  
linux@ubuntu:~$ ls -l  
total 0  
-rw-rw-r-- 1 linux adm 0 set 27 18:43 adeuu  
-rw-rw-r-- 1 linux linux 0 set 27 18:42 holaa  
linux@ubuntu:~$ █
```