Manual d'ordres GNU/Linux

Índex

Cicle formatiu: CFGM Sistemes Microinformàtics i Xarxes (SMX) / CFGS Administració de sistemes informàtics en xarxa (ASIX)

Mòdul: 0222 - Sistemes operatius monolloc / 0224 - Sistemes operatius en xarxa / 0369 - Implantació de sistemes operatius / 0374 - Administració de sistemes operatius

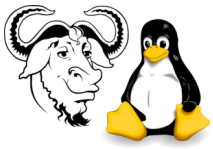

Figura 1: GNU/Linux logo

1. Gestió de fitxers i directoris

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

La gestió de fitxers i directoris és la base de qualsevol tasca d'administració en Linux. A diferència de Windows, on s'acostuma a treballar amb l'explorador gràfic, en entorns de servidor és habitual no disposar d'interfície gràfica, de manera que dominar aquestes ordres és imprescindible per a qualsevol tècnic de sistemes.

`ls` --- Llistar contingut d'un directori

`ls` (de l'anglès *list*) mostra el contingut d'un directori. És una de les ordres més usades i té moltes opcions per filtrar i formatar la sortida. Sense arguments, mostra els fitxers del directori on ens trobem. La informació que pot mostrar inclou el nom, la mida, els permisos, el propietari, el grup i la data de modificació de cada fitxer.

```
ls              # Llista fitxers del directori actual
ls -l           # Format llarg (permisos, mida, data)
ls -a           # Mostra fitxers ocults (els que comencen per .)
ls -lh          # Format llarg amb mides llegibles (KB, MB...)
ls -la          # Combinació de -l i -a
ls /etc         # Llista el contingut del directori /etc
```

Exemple pràctic:

```
ls -lh /var/log
# Mostra tots els fitxers de registre amb mida llegible
```

cd --- Canviar de directori

`cd` (de l'anglès *change directory*) permet moure's per l'estructura de directoris del sistema. És fonamental entendre la diferència entre **ruta absoluta** (comença per `/`, indica el camí complet des de l'arrel del sistema) i **ruta relativa** (indica el camí des del directori actual). El símbol ~ representa sempre el directori personal de l'usuari que ha iniciat sessió (`/home/nom_usuari` per a usuaris normals, `/root` per a l'administrador).

```
cd /home/usuari     # Va al directori indicat (ruta absoluta)
cd Documents        # Va al subdirectori Documents (ruta relativa)
cd ..               # Puja un nivell
cd ~                # Va al directori personal de l'usuari
cd -                # Torna al directori anterior
```

pwd --- Mostrar el directori actual

`pwd` (de l'anglès *print working directory*) mostra la ruta absoluta del directori on ens trobem en aquest moment. És molt útil quan treballem amb rutes relatives i volem saber el punt de partida exacte, o quan hem navegat per diverses carpetes i hem perdut la referència de la nostra ubicació.

```
pwd
# Exemple de sortida: /home/alumne/Documents
```

`mkdir` --- Crear directoris

`mkdir` (de l'anglès *make directory*) crea un o més directoris nous. L'opció `-p` és especialment útil perquè permet crear tota una jerarquia de directoris en un sol pas, sense que doni error si algun ja existeix. Sense `-p`, si el directori pare no existeix, l'ordre fallaria. L'opció `-m` permet especificar els permisos del directori en el moment de la creació, en notació octal.

```
mkdir proves            # Crea el directori "proves"
mkdir -p ruta/sub/dir   # Crea tota la ruta si no existeix
mkdir -m 755 projecte   # Crea el directori amb permisos específics
```

`rmdir` --- Eliminar directoris buits

`rmdir` (de l'anglès *remove directory*) elimina directoris, però amb una condició important: el directori ha d'estar completament buit. Si conté qualsevol fitxer o subdirectori, l'ordre falla i mostra un error. Aquesta restricció és una mesura de seguretat per evitar eliminar contingut per accident. Si volem eliminar un directori amb contingut, cal usar `rm -r`.

```
rmdir proves        # Elimina el directori "proves" (ha d'estar buit)
```

Si el directori no és buit, utilitza `rm -r`.

`rm` --- Eliminar fitxers i directoris

`rm` (de l'anglès *remove*) elimina fitxers i directoris de forma permanent. A Linux no hi ha cap paperera de reciclatge a la línia d'ordres: un cop eliminat, el fitxer no es pot recuperar fàcilment. L'opció `-r` (recursiu) permet eliminar directoris i tot el seu contingut. L'opció `-f` (force) suprimeix qualsevol missatge de confirmació i errors. La combinació `-rf` és una de les més perilloses del sistema, especialment executada com a root, ja que pot eliminar fitxers crítics del sistema operatiu sense cap advertència.

```
rm fitxer.txt           # Elimina un fitxer
rm -i fitxer.txt        # Demana confirmació abans d'eliminar
rm -r directori/        # Elimina el directori i tot el seu contingut
rm -rf directori/      # Elimina sense demanar confirmació (perillós!)
rm *.log                # Elimina tots els fitxers .log del directori
 ↪  actual
```

`rm -rf` és irreversible. Usar amb molta precaució.

`cp` --- Copiar fitxers i directoris

`cp` (de l'anglès *copy*) crea una còpia d'un fitxer o directori en una nova ubicació. Si el destí és un directori existent, el fitxer es copiarà dins d'aquest directori mantenint el nom original. Si el destí és un nom de fitxer, es crearà un nou fitxer amb aquest nom. L'opció `-r` és necessària per copiar directoris sencers, ja que sense ella `cp` no processarà el contingut del directori. L'opció `-p` preserva les metadades del fitxer original, incloent-hi permisos, propietari i data de modificació, cosa que és molt important en còpies de seguretat.

```
cp fitxer.txt còpia.txt             # Còpia un fitxer
cp fitxer.txt /home/usuari/         # Còpia a un altre directori
cp -r directori/ /backup/           # Còpia un directori sencer
cp -p fitxer.txt còpia.txt          # Preserva permisos i dates
cp -i fitxer.txt destí.txt          # Demana confirmació si el destí
 ↪  existeix
```

`mv` --- Moure o reanomenar fitxers

`mv` (de l'anglès *move*) té dues funcions en una: mou fitxers o directoris d'una ubicació a una altra, i també permet reanomenar-los. Si origen i destí estan al mateix sistema de fitxers, l'operació és instantània perquè simplement es modifica l'entrada al directori sense moure dades físicament. Si estan en sistemes de fitxers diferents, `mv` fa una còpia i elimina l'original. A diferència de `cp`, no cal l'opció `-r` per moure directoris.

```
mv fitxer.txt /tmp/                 # Mou el fitxer a /tmp
mv nom_vell.txt nom_nou.txt         # Reanomena el fitxer
mv directori/ /mnt/disc/            # Mou un directori sencer
```

`touch` --- Crear fitxers buits o actualitzar data

`touch` té dues utilitats principals. La primera és crear fitxers buits: si el fitxer especificat no existeix, `touch` el crea sense contingut. La segona és actualitzar la data i hora d'accés i modificació d'un fitxer existent a l'hora actual sense modificar-ne el contingut. Aquesta segona funció s'utilitza habitualment en scripts per indicar que una certa acció s'ha executat (fitxers de marca o *timestamp files*) o per forçar que certes aplicacions reprocessin un fitxer.

```
touch fitxer.txt        # Crea un fitxer buit (o actualitza la seva
 ↪  data)
touch fitxer1 fitxer2   # Crea diversos fitxers a la vegada
```

`find` --- Cercar fitxers i directoris

`find` és una de les ordres més potents i versàtils de Linux. Realitza cerques en temps real recorrent l'arbre de directoris (a diferència de `locate`, que usa una base de dades). Permet combinar múltiples criteris de cerca: nom, extensió, tipus (fitxer o directori), mida, data de modificació, propietari, permisos, i molts més. A més, pot executar accions sobre els fitxers trobats gràcies a l'opció `-exec`. La recerca per tot el sistema (`find /`) pot ser lenta i generar molts missatges d'error de permisos, que es poden suprimir redirigint els errors a `/dev/null`.

```
find / -name "fitxer.txt"           # Cerca a tot el sistema
find /home -name "*.pdf"            # Cerca PDFs a /home
find /var -mtime -7                 # Fitxers modificats els últims 7
 ↪  dies
find /tmp -size +10M                # Fitxers de més de 10 MB
find / -type d -name "log"         # Cerca directoris anomenats "log"
find /etc -name "*.conf" -user root # Fitxers .conf propietat de root
```

`locate` --- Cerca ràpida de fitxers (base de dades)

`locate` busca fitxers consultant una base de dades precompilada que conté la llista de tots els fitxers del sistema. Això el fa molt més ràpid que `find`, però té un inconvenient important: la base de dades s'actualitza periòdicament (normalment una vegada al dia amb una tasca cron), de manera que els fitxers creats o eliminats recentment pot ser que no apareguin o que segueixin apareixent als resultats. Quan necessitem resultats absolutament actuals, és millor usar `find`. Cal tenir en compte que `locate` no mostra fitxers als quals l'usuari no té permisos d'accés.

```
locate fitxer.txt           # Cerca el fitxer a la base de dades
locate -i Fitxer.txt        # Cerca sense distingir majúscules
sudo updatedb               # Actualitza la base de dades de locate
```

> `locate` és molt més ràpid que `find`, però la base de dades pot no estar actualitzada.

`cat` --- Mostrar contingut de fitxers

`cat` (de l'anglès *concatenate*) llegeix un o més fitxers i mostra el seu contingut a la sortida estàndard (la pantalla). Malgrat el seu nom, s'usa més freqüentment per mostrar el contingut d'un únic fitxer que per concatenar-ne diversos. Per a fitxers molt llargs no és la millor opció, ja que mostra tot el contingut de cop sense possibilitat de paginar. En aquells casos és preferible usar `less` o `more`. Sí que resulta molt útil per visualitzar fitxers de configuració curts o per combinar fitxers amb la redirecció (`cat fitxer1 fitxer2 > resultat`).

```
cat fitxer.txt              # Mostra tot el fitxer
cat fitxer1.txt fitxer2.txt # Concatena i mostra dos fitxers
cat -n fitxer.txt           # Mostra amb números de línia
```

`more` / `less` --- Navegar per fitxers llargs

Totes dues ordres permeten visualitzar fitxers grans de forma paginada, és a dir, mostrant-ne el contingut pantalla a pantalla sense sobrepassar els límits de la terminal. `more` és la versió original i més limitada: només permet avançar (no tornar enrere) i la navegació es fa amb la barra d'espai. `less` és una versió millorada que permet navegar en les dues direccions amb les tecles de cursor, fer cerques amb `/` (endavant) i `?` (enrere), i té moltes més opcions de navegació. El nom `less` és un joc de paraules: *less is more* (“menys és més”), fent referència al fet que és una millora de `more`.

```
more fitxer.txt     # Pagina el contingut (espai per avançar)
less fitxer.txt     # Navegació avançada (fletxes, / per cercar, q per
 ↪  sortir)
```

head / `tail` --- Primeres o últimes línies

`head` mostra les primeres línies d'un fitxer (10 per defecte) i `tail` mostra les últimes. Són molt útils per inspeccionar fitxers grans sense haver de carregar-los completament. `tail -f` és especialment valuós en l'administració de sistemes perquè segueix el fitxer en temps real, mostrant les noves línies a mesura que s'afegeixen. Aquesta opció s'usa habitualment per monitorar fitxers de registre (*logs*) mentre un servei s'executa, cosa que permet detectar errors en el moment en què es produeixen.

```
head fitxer.txt         # Mostra les primeres 10 línies
head -n 20 fitxer.txt   # Mostra les primeres 20 línies
tail fitxer.txt         # Mostra les últimes 10 línies
tail -n 50 fitxer.txt   # Mostra les últimes 50 línies
tail -f /var/log/syslog # Segueix el fitxer en temps real (molt útil
 ↪  per logs)
```

2. Permisos i propietaris

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

Linux és un sistema operatiu multiusuari, i el sistema de permisos és el mecanisme que controla qui pot fer què amb cada fitxer o directori. Cada fitxer té associat un propietari (un usuari) i un grup, i els permisos es defineixen per a tres nivells: el propietari, els membres del grup, i la resta d'usuaris del sistema. Entendre i gestionar correctament els permisos és essencial per a la seguretat del sistema.

Interpretació dels permisos

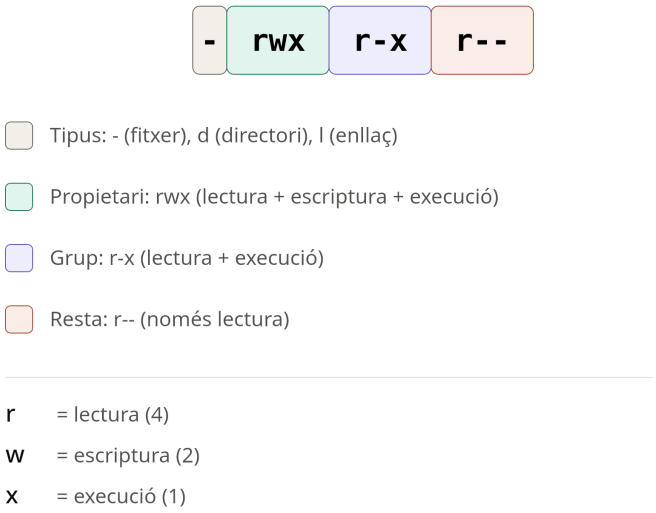

Figura 2: Permisos

Bits especials

A més dels 9 bits de protecció bàsics, en l'inode rauen uns altres 3 bits per al control dels permisos d'accés. Aquests tres bits (**setuid**, **setgid** i **sticky**) són especials i una mica complexos però essencials per a una correcta administració. Tots tres s'utilitzen principalment en programes executables i resulten potents i arriscats, ja que certs forats de seguretat es basen en la seva manipulació. Si s'activa el bit setuid d'un programa executable s'aconsegueix que els permisos d'accés als arxius que gestiona aquest executable es contrastin no amb el compte des del qual es vol executar el programa sinó amb el propietari del programa executable. Aquesta funció permet saltar de forma controlada els permisos de certs programes. L'exemple més conegut és el del programa passwd que permet canviar i actualitzar les contrasenyes que resideixen en l'arxiu `/etc/passwd`. En principi, qualsevol hauria de poder escriure en aquest arxiu perquè té dret a canviar la seva clau d'accés. Tanmateix, aquest fet, sense una restricció addicional, representaria un perill enorme, ja que un usuari podria canviar les claus de qualsevol altre compte mitjançant un editor de text. La solució a aquest conflicte és la utilització del bit setuid en el programa. L'usuari root és el propietari del programa i només el root té permís d'escriptura en l'arxiu `/etc/passwd`. Per tant, no es pot escriure en aquest arxiu directament, però quan s'executa l'ordre de canvi de clau, com aquest programa té activat el setuid. Els permisos que es verificaran seran els del seu propietari, que és root, i no els de l'usuari que vol canviar la clau. D'aquesta manera s'aconsegueix el permís d'escriptura de forma controlada. És a dir, gràcies al bit setuid s'aconsegueix escriure a l'arxiu, però només per mitjà d'aquest programa. El bit setgid és similar al setuid, però referit als permisos de grup en lloc dels de propietari. La funció del sticky bit és totalment diferent, encara que afecta també programes executables. L'objectiu d'aquest bit és fixar el programa permanentment en la memòria per evitar els temps de càrrega posteriors. També és un bit perillós, perquè els virus i derivats tenen precisament aquest objectiu, quedar-se residents en memòria. L'activació i

desactivació d'aquests bits és similar a la dels permisos. `u+s` activa el `setuid`, `g+s` activa el `setgid` i `o+t` l'`sticky`. Amb el signe menys es desactiven. Si desitgem modificar-los en manera numèrica hem d'usar un quart valor als tres ja coneguts, a l'esquerra dels anteriors. Per al càlcul del valor numèric setuid en val 4, setgid val 2 i sticky en val 1.

Els bits especials en les carpetes

L'activació del sticky en una carpeta serveix per aconseguir que qui tingui dret d'escriptura en una carpeta no pugui esborrar els arxius que hi hagi en ella i que no siguin de la seva propietat. És comú utilitzar aquesta possibilitat en carpetes que contenen arxius temporals com /tmp i carpetes que contenen arxius amb característiques diverses de protecció. Activant el bit setgid en una carpeta es pot aconseguir que els arxius que es vagin creant per sota d'aquesta carpeta heretin el grup propietari de la carpeta, que s'assignarà com a grup propietari de l'arxiu. Això és molt interessant quan treballaran en un projecte conjunt usuaris de diferents grups preassignats. En aquest cas es crearà un nou grup en el qual s'inclouran els comptes participants en el projecte. A continuació es crearà una carpeta arrel per al treball conjunt i, posteriorment, a aquesta carpeta se li assignarà com a grup propietari el nou grup i se li activarà el bit setgid. D'aquesta manera, quan participants del projecte creïn arxius i carpetes sota la carpeta principal el grup propietari que se li assignarà serà l'heretat, és a dir, el del projecte conjunt, i per tant els permisos de grup es comprovaran respecte al grup nou.

La representació d'aquests permisos en el resultat de l'ordre `ls --l` és:

Bit	Comportament
setuid	El caràcter `x` dels permisos del propietari es canvien per una `s` quan aquest permís està activat; una `S` significa que el permís en execució no està activat al mateix temps.
setgid	Igual que SUID però a escala dels permisos del grup.
sticky	Una `t` indica en el lloc de la `x` a escala dels permisos de l'entitat “others”. Si aquest permís és `T` significa que el permís `x` que està ocult no està actiu.

Resum

Permís	Arxiu	Carpeta
setuid	Executa l'arxiu amb la identitat del propietari de l'arxiu	
setgid	Executa l'arxiu amb la identitat del grup de l'arxiu	Els arxius creats en el directori hereten el grup del directori en lloc del grup principal de l'usuari
sticky	La imatge de l'executable segueix en memòria, la seva recàrrega és més ràpida	Només el propietari de l'arxiu o del directori poden esborrar els arxius

chmod --- Canviar permisos

`chmod` (de l'anglès *change mode*) modifica els permisos d'accés d'un fitxer o directori. Admet dues notacions: la **notació octal** (numèrica) i la **notació simbòlica**. En la notació octal, cada xifra representa un nivell (propietari, grup, resta) i és la suma dels valors dels permisos actius (r=4, w=2, x=1). Així, 7 significa rwx, 6 significa rw-, 5 significa r-x, i 4 significa r--. En la notació simbòlica s'usen les lletres u (user/propietari), g (group), o (others/resta) i a (all/tots), combinades amb + (afegir), - (treure) i = (establir exactament). L'opció `-R` aplica els canvis de forma recursiva a tots els fitxers i subdirectoris.

```
# Notació octal
chmod 755 fitxer.sh     # rwxr-xr-x (propietari tot, grup/resta
 ↪  lectura+execució)
chmod 644 document.txt  # rw-r--r-- (propietari lectura+escriptura,
 ↪  resta lectura)
chmod 600 clau.pem      # rw------- (només el propietari pot
 ↪  llegir/escriure)
chmod 777 script.sh     # rwxrwxrwx (tots els permisos per a tothom,
 ↪  evitar!)

# Notació simbòlica
chmod u+x script.sh     # Afegeix execució al propietari (user)
chmod g-w fitxer.txt    # Treu escriptura al grup
chmod o+r fitxer.txt    # Afegeix lectura a la resta (others)
chmod a+x script.sh     # Afegeix execució a tothom (all)
chmod -R 755 directori/ # Aplica permisos recursivament
```

Taula de valors octals comuns:

Octal	Permisos	Ús típic
777	rwxrwxrwx	Mai recomanat
755	rwxr-xr-x	Executables, directoris
644	rw-r--r--	Fitxers de text, configuració
600	rw-------	Claus privades, contrasenyes
700	rwx------	Directoris personals privats

chown --- Canviar propietari

`chown` (de l'anglès *change owner*) canvia el propietari o el grup associat a un fitxer o directori. Només l'usuari root pot canviar el propietari d'un fitxer; un usuari normal no pot transferir la propietat dels seus fitxers a un altre usuari. Sí que pot canviar el grup, però únicament a un grup del qual ell mateix sigui membre. La sintaxi `usuari:grup` permet canviar tots dos alhora. L'opció `-R` aplica el canvi recursivament, cosa que és molt habitual en l'administració de servidors web, on cal assegurar-se que el servidor web (normalment l'usuari `www-data`) sigui propietari dels fitxers del lloc web.

```
chown usuari fitxer.txt             # Canvia el propietari
chown usuari:grup fitxer.txt        # Canvia propietari i grup
```

```
chown -R www-data:www-data /var/www # Recursiu (típic per al servidor
 ↪  web)
```

`chgrp` --- Canviar grup

`chgrp` (de l'anglès *change group*) canvia únicament el grup propietari d'un fitxer o directori, sense modificar el propietari. Funciona de manera similar a `chown` però limitat als grups. Un usuari pot canviar el grup d'un fitxer seu a qualsevol grup del qual sigui membre. És útil en entorns col·laboratius on diversos usuaris pertanyents a un mateix grup han de compartir fitxers.

```
chgrp professors fitxer.txt         # Canvia el grup del fitxer
chgrp -R alumnes /home/compartit/   # Recursiu
```

`umask` --- Màscara de permisos per defecte

`umask` defineix els permisos que s'eliminaran automàticament cada vegada que es crea un fitxer o directori nou. Funciona com una màscara de substracció: els permisos base per a fitxers nous són 666 (rw-rw-rw-) i per a directoris 777 (rwxrwxrwx), i la màscara indica quins permisos cal treure. Amb la màscara per defecte 022, els fitxers nous tindran permisos 644 (666-022) i els directoris 755 (777-022). Canviar el `umask` en el fitxer `~/.bashrc` o `/etc/profile` afecta els permisos de tots els fitxers creats per l'usuari o el sistema respectivament.

```
umask           # Mostra la màscara actual (normalment 0022)
umask 027       # Estableix la màscara (fitxers nous: 640, dirs: 750)
```

`acl` - Llistes de control d'accés

La llista de control d'accés (ACL o access control list) és un concepte de seguretat informàtica utilitzat per fer complir la separació de privilegis. Això significa donar o no uns privilegis a un objecte determinat que està realitzant una consulta. Un sistema d'arxius ACL és una estructura de dades (normalment una taula) que conté entrades que especifiquen drets d'usuaris individuals o grups respecte a objectes específics del sistema específics com poden ser programes, processos o arxius. Els privilegis o permisos determinen drets d'accés específics: com per exemple si un usuari pot llegir, escriure o executar un objecte.

Instal·lació:

```
sudo apt-get install acl
```

Per poder fer ús de les ACL hem de muntar el suport d'ACL per a la partició que conté els arxius o les carpetes. Editem l'arxiu `/etc/fstab` i afegim acl a les opcions de muntatge de la partició afectada.

```
/dev/sda7    /var    ext4       defaults,acl    0    3
```

Ordres:

`setfacl` - per modificar la llista

`getfacl` - per obtenir informació de la llista

Exemple pràctic:

L'usuari Frederic és la persona que s'encarregarà de la pàgina web de l'empresa, que es troba en la carpeta /var/www. Aquest usuari no té accés via ftp a aquesta carpeta, per la qual cosa li suposarà un problema poder modificar els continguts d'aquest directori. L'usuari root pot solucionar el problema fent ús de les ACL i de l'ordre setfacl per assignar-li els permisos necessaris per tenir accés d'escriptura i execució a la carpeta:

```
sudo setfacl -m u:frederic:rwx /var/www
```

Per veure les propietats de la carpeta emprem l'ordre `getfacl`:

```
sudo getfacl /var/www
```

```
getfacl: Removing leading '/' from absolute path names
# file: /var/www
# owner: root
# group: root
user::rwx
user:frederic:rwx
group::r-x
mask::rwx
other::r-x
```

3. Gestió d'usuaris i grups

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

La gestió d'usuaris i grups és una tasca central en l'administració de sistemes Linux. Cada usuari té un identificador únic (UID), un directori personal, un shell per defecte i pertany a un o més grups. Els grups permeten organitzar usuaris amb necessitats d'accés similars i simplificar la gestió de permisos. Tota la informació d'usuaris s'emmagatzema principalment als fitxers `/etc/passwd`, `/etc/shadow` (contrasenyes xifrades) i `/etc/group`.

`useradd` --- Crear usuaris

`useradd` crea un compte d'usuari nou al sistema. Per si sola, l'ordre crea l'entrada a `/etc/passwd` però no necessàriament el directori personal ni un shell usable. Per això és habitual usar `-m` (crea el directori personal a `/home/nom_usuari`) i `-s` (especifica el shell). El directori personal es crea copiant el contingut de `/etc/skel`, que conté els fitxers de configuració inicials. Si no s'especifica un shell, el sistema pot assignar-ne un no interactiu com `/bin/sh` o `/usr/sbin/nologin`, que impedeix l'inici de sessió.

```
useradd alumne                          # Crea l'usuari "alumne"
useradd -m alumne                       # Crea l'usuari amb directori
 ↪  personal
useradd -m -s /bin/bash alumne          # Amb shell bash
useradd -m -g professors alumne         # Amb grup principal
 ↪  "professors"
useradd -m -G sudo,alumnes joan         # Afegeix a grups secundaris
useradd -m -c "Joan García" joan        # Amb comentari (nom complet)
```

`usermod` --- Modificar usuaris existents

`usermod` permet modificar qualsevol atribut d'un compte d'usuari existent sense necessitat d'eliminar-lo i tornar-lo a crear. Una de les opcions més usades és `-aG` per afegir l'usuari a grups secundaris addicionals. És crítica la diferència entre `-aG` (afegir als grups existents mantenint els altres) i `-G` sol (reemplaça completament la llista de grups secundaris, eliminant els anteriors). Les opcions `-L` i `-U` permeten bloquejar i desbloquejar comptes sense eliminar-los, cosa que és útil quan un empleat marxa temporalment o quan cal desactivar un compte per seguretat.

```
usermod -aG sudo alumne         # Afegeix l'usuari al grup sudo (sense
 ↪  treure'l d'altres)
usermod -s /bin/bash alumne     # Canvia el shell
usermod -l nou_nom vell_nom     # Canvia el nom d'usuari
usermod -L alumne               # Bloqueja el compte
usermod -U alumne               # Desbloqueja el compte
usermod -d /home/nou alumne     # Canvia el directori personal
```

Usar `-aG` i no `-G` sol, o es perdran els grups secundaris existents.

`userdel` --- Eliminar usuaris

`userdel` elimina un compte d'usuari del sistema. Sense opcions addicionals, elimina l'entrada de l'usuari als fitxers del sistema (`/etc/passwd`, `/etc/shadow`, `/etc/group`) però deixa intacte el directori personal i tots els seus fitxers. Amb l'opció `-r`, elimina també el directori personal i el correu de l'usuari. Cal tenir en compte que els fitxers propietat de l'usuari eliminat que estiguin fora del seu directori personal quedaran sense propietari reconegut (es mostrarà el UID numèric en lloc d'un nom), i caldrà reassignar-los manualment.

```
userdel alumne          # Elimina l'usuari (manté el directori
 ↪  personal)
userdel -r alumne       # Elimina l'usuari i el directori personal
```

`groupadd` / `groupmod` / `groupdel` --- Gestió de grups

Aquestes tres ordres gestionen el cicle de vida dels grups al sistema. `groupadd` crea un grup nou amb un GID (identificador de grup) assignat automàticament o especificat manualment. `groupmod` permet modificar atributs d'un grup existent, com ara el nom o el GID. `groupdel` elimina un grup, però cal assegurar-se prèviament que cap usuari tingui aquest grup com a grup principal, o l'ordre fallarà. Els grups existents es poden consultar al fitxer `/etc/group`.

```
groupadd alumnes            # Crea el grup "alumnes"
groupmod -n nous alumnes    # Reanomena el grup
groupdel alumnes            # Elimina el grup
```

`passwd` --- Gestionar contrasenyes

`passwd` gestiona les contrasenyes dels comptes d'usuari. Les contrasenyes mai s'emmagatzemen en text pla: Linux les xifra amb un algorisme criptogràfic (habitualment SHA-512) i guarda el resultat al fitxer `/etc/shadow`, que només és llegible per root. Quan un usuari normal executa `passwd` sense arguments, se li demana la contrasenya actual com a verificació abans de poder canviar-la. L'administrador (root) pot canviar la contrasenya de qualsevol usuari sense conèixer-ne l'actual. L'opció `-e` (expire) força l'usuari a canviar la contrasenya en el proper inici de sessió, cosa útil quan es crea un compte nou.

```
passwd                  # Canvia la contrasenya de l'usuari actual
passwd alumne          # Canvia la contrasenya d'un usuari (com a root)
passwd -l alumne        # Bloqueja el compte
passwd -u alumne        # Desbloqueja el compte
passwd -e alumne        # Força canvi de contrasenya en el proper
 ↪  inici de sessió
```

Si vols saber més respecte a la gestió de les contrasenyes en Linux ves a: Gestió de les contrasenyes en GNU/Linux

su --- Canviar d'usuari

`su` (de l'anglès *switch user* o *substitute user*) permet canviar d'usuari dins d'una sessió oberta. Quan s'usa `su` sense l'opció `-`, es canvia l'usuari, però es mantenen les variables d'entorn de la sessió original (directori de treball, variables d'entorn, etc.). Amb l'opció `-` o `-l`, s'inicialitza un entorn complet com si l'usuari hagués iniciat sessió des de zero, incloent-hi el directori personal i les variables d'entorn pròpies. En sistemes Ubuntu, root no té contrasenya configurada per defecte, i per accedir-hi s'usa `sudo -i` o `sudo su -`.

```
su alumne           # Canvia a l'usuari "alumne" (manté l'entorn
 ↪  actual)
su - alumne         # Canvia a l'usuari amb el seu entorn complet
su -                # Canvia a root amb el seu entorn
```

sudo --- Executar ordres com a root

`sudo` (de l'anglès *superuser do*) permet a usuaris autoritzats executar ordres amb privilegis d'administrador (root) o d'un altre usuari, sense necessitat de conèixer la contrasenya de root. Els usuaris autoritzats es configuren al fitxer `/etc/sudoers` (editat amb `visudo`). En distribucions com Ubuntu, el primer usuari creat durant la instal·lació pertany al grup `sudo` i pot usar aquesta ordre. `sudo` registra totes les ordres executades al fitxer de registre del sistema, cosa que proporciona un registre d'auditoria molt valuós per a la seguretat. Demana la contrasenya de l'usuari (no la de root) i la recorda durant uns minuts.

```
sudo apt update             # Executa l'ordre com a root
sudo -i                     # Obre un shell de root
sudo -u alumne ordre        # Executa una ordre com un altre usuari
sudo !!                     # Repeteix l'última ordre amb sudo
```

`id`, `who`, `whoami`, `w` --- Informació d'usuaris

Aquestes ordres proporcionen informació sobre els usuaris que usen el sistema. `id` mostra el UID (User ID), el GID (Group ID) principal i tots els grups als quals pertany l'usuari. Aquesta informació és molt útil per diagnosticar problemes de permisos: si un usuari no pot accedir a un fitxer, `id` ens permet verificar si pertany al grup correcte. `who` i `w` mostren els usuaris connectats al sistema, incloent-hi des d'on s'han connectat (terminal local o IP remota) i fa quant de temps. `w` proporciona informació addicional sobre l'activitat de cada usuari.

```
id                      # Mostra UID, GID i grups de l'usuari actual
id alumne               # Mostra informació d'un usuari concret
whoami                  # Mostra el nom de l'usuari actual
who                     # Mostra els usuaris connectats
w                       # Mostra els usuaris connectats i la seva
 ↪  activitat
```

4. Processos i serveis

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

En Linux, un procés és una instància en execució d'un programa. El sistema operatiu assigna a cada procés un identificador únic anomenat PID (Process ID). Els processos s'organitzen en una jerarquia: cada procés té un procés pare (PPID), i el primer procés del sistema (init o systemd) és l'ancestre de tots els altres. La gestió correcta dels processos i dels serveis del sistema és fonamental per mantenir un servidor estable i segur.

`ps` --- Llistar processos

`ps` (de l'anglès *process status*) mostra una instantània dels processos en execució en el moment de l'execució. A diferència de `top`, no s'actualitza en temps real. La combinació d'opcions `aux` és la més habitual: `a` mostra els processos de tots els usuaris, `u` mostra el format orientat a l'usuari (amb nom d'usuari, ús de CPU i memòria), i `x` inclou els processos que no estan associats a cap terminal (com els serveis o dimonis del sistema). La sortida es pot filtrar amb `grep` per trobar processos específics.

```
ps                      # Processos de la sessió actual
ps aux                  # Tots els processos del sistema (format
 ↪  complet)
ps aux | grep nginx     # Cerca el procés nginx
ps -ef                  # Format alternatiu amb PPID
```

`top` / `htop` --- Monitor de processos en temps real

`top` és un monitor de processos interactiu que s'actualitza periòdicament (per defecte cada 3 segons) i mostra informació sobre l'ús de CPU, memòria i swap del sistema, així com els processos ordenats per consum de recursos. A la capçalera es pot veure el temps que porta engegat el sistema (*uptime*), el nombre d'usuaris connectats, la càrrega mitjana del sistema i un resum de l'ús de memòria. `htop` és una versió millorada amb interfície en colors, navegació amb cursor i possibilitat de gestionar processos de forma visual. Cal instal·lar-lo manualment (`apt install htop`) perquè no sempre ve inclòs per defecte.

```
top                     # Monitor bàsic (q per sortir, k per matar un
 ↪  procés)
htop                    # Monitor avançat i visual (cal instal·lar-lo)
```

Tecles útils dins de `top`:

- `k` → Matar un procés (introduir PID)
- `r` → Canviar la prioritat (renice)
- `M` → Ordenar per ús de memòria
- `P` → Ordenar per ús de CPU
- `q` → Sortir

`kill` / `killall` --- Finalitzar processos

`kill` envia un senyal a un procés identificat pel seu PID. Malgrat el nom, no tots els senyals finalitzen el procés: el senyal per defecte és SIGTERM (15), que demana al procés que s'aturi de forma ordenada, alliberant recursos i tancant fitxers. El procés pot ignorar aquest senyal o demorar la resposta. En canvi, SIGKILL (9) és enviat directament pel nucli del sistema i el procés no pot ignorar-lo ni interceptar-lo, de manera que l'atura immediatament. SIGKILL s'ha d'usar com a últim recurs perquè el procés no té l'oportunitat d'alliberar recursos correctament. `killall` fa el mateix, però admet el nom del procés en lloc del PID, cosa que simplifica la feina quan no sabem el PID exacte.

```
kill 1234               # Envia el senyal SIGTERM al procés 1234
 ↪  (tancament suau)
kill -9 1234            # Envia SIGKILL (força el tancament)
kill -l                 # Llista tots els senyals disponibles
killall firefox         # Mata tots els processos de Firefox
killall -9 nginx        # Força el tancament de tots els processos
 ↪  nginx
```

Senyals més comuns:

Senyal	Número	Descripció
SIGTERM	15	Tancament suau (per defecte)
SIGKILL	9	Tancament forçat (no es pot ignorar)
SIGHUP	1	Recarrega la configuració

`nice` / `renice` --- Prioritat de processos

`nice` i `renice` controlen la prioritat dels processos, és a dir, quanta atenció del processador reben en comparació amb els altres. La prioritat s'expressa amb un valor de “niceness” que va de -20 (màxima prioritat, el procés acapara recursos) a 19 (mínima prioritat, cedeix recursos a tots els altres). El valor per defecte és 0. Valors positius de niceness signifiquen que el procés és “amable” amb els altres i cedeix el processador. Valors negatius indiquen que el procés té preferència, però per assignar-ne, cal ser root. `nice` s'usa en arrencar un procés nou, mentre que `renice` modifica la prioritat d'un procés que ja s'està executant.

```
nice -n 10 programa         # Executa un programa amb prioritat baixa
 ↪  (10)
nice -n -5 programa        # Prioritat alta (negatiu = més prioritat,
 ↪  cal root)
renice -n 5 -p 1234         # Canvia la prioritat del procés 1234
```

El rang de prioritat va de -20 (màxima) a 19 (mínima). Per defecte és 0.

`systemctl` --- Gestió de serveis (systemd)

`systemctl` és l'ordre principal per interactuar amb `systemd`, el sistema d'inicialització i gestió de serveis que utilitzen la majoria de distribucions Linux modernes (Ubuntu, Debian, Fedora, CentOS, etc.). Permet iniciar, parar, reiniciar i comprovar l'estat de qualsevol servei del sistema. La distinció entre `restart` i `reload` és important: `restart` para el servei completament i el torna a engegar (interrompent les connexions actives), mentre que `reload` demana al servei que torni a llegir la seva configuració sense interrompre les connexions existents. `enable` i `disable` controlen si el servei s'engega automàticament quan el sistema arrenca, però no l'inicien ni l'aturen immediatament.

```
systemctl start nginx                # Inicia el servei
systemctl stop nginx                 # Para el servei
systemctl restart nginx              # Reinicia el servei
systemctl reload nginx              # Recarrega la configuració sense
 ↪  parar
systemctl status nginx               # Mostra l'estat del servei
systemctl enable nginx               # Activa l'inici automàtic en
 ↪  arrencar
systemctl disable nginx              # Desactiva l'inici automàtic
systemctl is-active nginx            # Comprova si està actiu
systemctl list-units --type=service  # Llista tots els serveis
```

`journalctl` --- Registres del sistema (systemd)

`journalctl` permet consultar el registre centralitzat del sistema gestionat per `systemd-journald`. A diferència dels sistemes antics on els registres eren fitxers de text dispersos per `/var/log`, `systemd` recull tots els missatges de serveis, del nucli i del sistema en un únic registre binari. `journalctl` permet filtrar per servei, per data, per nivell de gravetat i per molts altres criteris. L'opció `-f` (follow) és equivalent al `tail -f` dels fitxers de log tradicionals i permet monitorar els registres en temps real. L'opció `-p err` filtra per nivell de prioritat, mostrant només errors i missatges més greus.

```
journalctl                      # Tots els registres
journalctl -u nginx             # Registres d'un servei concret
journalctl -f                   # Segueix els registres en temps real
journalctl --since "1 hour ago" # Registres de l'última hora
journalctl -b                   # Registres del darrer arrencament
journalctl -p err               # Només errors
```

5. Emmagatzematge i particions

Mòdul relacionat: **0221 Muntatge i manteniment d'equips · 0222 Sistemes operatius monolloc**

En Linux, tots els dispositius d'emmagatzematge (discos durs, SSD, memòries USB) apareixen com a fitxers especials al directori `/dev`. Els discos IDE/SATA/NVMe s'anomenen `/dev/sda`, `/dev/sdb`, etc., i les seves particions `/dev/sda1`, `/dev/sda2`, etc. Per accedir al contingut d'una partició cal muntar-la, és a dir, fer-la accessible a través d'un directori de l'arbre de fitxers del sistema (punt de muntatge). Tota la informació sobre les particions que s'han de muntar automàticament en l'arrencada es troba al fitxer `/etc/fstab`.

`fdisk` --- Gestionar particions (MBR)

`fdisk` és l'eina clàssica de Linux per crear, modificar i eliminar particions en discos amb esquema de particions MBR (Master Boot Record), que suporta fins a 4 particions primàries i discos de fins a 2 TB. Per a discos més grans o quan es vol usar l'esquema GPT (GUID Partition Table), és preferible usar `parted` o `gdisk`. `fdisk -l` pot executar-se sense privilegis sobre alguns sistemes, però per modificar particions cal ser root. Tots els canvis que es fan a `fdisk` es guarden en memòria fins que s'escriuen al disc amb l'opció `w`, de manera que si ens equivoquem podem sortir sense guardar amb `q`.

```
fdisk -l                    # Llista tots els discos i particions
fdisk /dev/sdb              # Obre l'editor de particions del disc sdb
```

Opcions dins de `fdisk`:

- `n` → Crea una nova partició
- `d` → Elimina partició
- `p` → Mostra la taula de particions
- `w` → Desa i surt
- `q` → Surt sense desar

`parted` --- Gestionar particions (MBR i GPT)

`parted` és una eina de gestió de particions més moderna que `fdisk` i suporta tant MBR com GPT. A diferència de `fdisk`, `parted` aplica els canvis immediatament sense necessitat de confirmar, de manera que cal tenir molta cura. Admet tant un mode interactiu (sense arguments o amb el nom del disc) com un mode no interactiu passant totes les opcions a la línia d'ordres, cosa que el fa ideal per a scripts d'automatització. Pot redimensionar particions existents (amb algunes limitacions) i gestionar discos de qualsevol mida.

```
parted /dev/sdb print                      # Mostra les particions del
 ↪  disc
parted /dev/sdb mklabel gpt                 # Crea una taula de
 ↪  particions GPT
parted /dev/sdb mkpart primary ext4 0% 50%  # Crea una partició
```

`mkfs` --- Formatar particions

`mkfs` (de l'anglès *make filesystem*) crea un sistema de fitxers en una partició o dispositiu. En realitat és una família d'ordres: `mkfs.ext4`, `mkfs.ntfs`, `mkfs.vfat`, etc., on cada una crida l'eina específica per a cada tipus de sistema de fitxers. **Formatant una partició s'eliminen totes les dades que conté de forma irrecuperable**, de manera que cal verificar sempre el dispositiu destí abans d'executar l'ordre. `ext4` és el sistema de fitxers estàndard de Linux, `NTFS` és compatible amb Windows, i `FAT32` és universal però limitat a fitxers de màxim 4 GB.

```
mkfs.ext4 /dev/sdb1         # Formata amb ext4
mkfs.ntfs /dev/sdb2         # Formata amb NTFS
mkfs.vfat /dev/sdb3         # Formata amb FAT32
mkfs.xfs /dev/sdb1          # Formata amb XFS
```

`mount` / `umount` --- Muntar i desmuntar unitats

`mount` connecta un sistema de fitxers d'una partició o dispositiu a un punt de l'arbre de directoris del sistema, fent-ne el contingut accessible. El punt de muntatge ha d'existir prèviament com a directori buit. Sense arguments, `mount` mostra tots els sistemes de fitxers muntats actualment. `umount` desmunta un sistema de fitxers; cal assegurar-se que cap procés ni usuari estigui accedint al sistema de fitxers en aquell moment, o l'ordre fallarà amb un error de “dispositiu ocupat”. Per montar dispositius automàticament en l'arrencada del sistema, cal afegir-los al fitxer `/etc/fstab`.

```
mount /dev/sdb1 /mnt/disc       # Munta la partició a /mnt/disc
mount -t ext4 /dev/sdb1 /mnt/   # Especifica el sistema de fitxers
mount -o ro /dev/sdb1 /mnt/     # Munta en mode només lectura
mount                           # Mostra totes les unitats muntades
umount /mnt/disc                # Desmunta la unitat
umount -l /mnt/disc             # Desmunta de forma diferida (lazy)
```

`df` --- Espai lliure en disc

`df` (de l'anglès *disk free*) mostra l'espai total, l'espai usat i l'espai lliure de tots els sistemes de fitxers muntats. L'opció `-h` (human-readable) mostra les mides en unitats llegibles (KB, MB, GB) en lloc de blocs. Un aspecte important de `df` és que mostra l'espai del sistema de fitxers complet on es troba el directori indicat, no l'espai del directori en si. Cal tenir en compte que en sistemes de fitxers `ext4`, per defecte es reserva un 5% de l'espai per a l'usuari root, de manera que un disc que `df` mostra al 95% d'ús pot estar completament ple per als usuaris normals.

```
df                          # Espai de tots els sistemes de fitxers
df -h                       # Format llegible (GB, MB)
df -hT                      # Inclou el tipus de sistema de fitxers
df /home                    # Espai del sistema de fitxers que conté
 ↪  /home
```

du --- Espai ocupat per fitxers i directoris

`du` (de l'anglès *disk usage*) mesura l'espai que ocupa en disc un fitxer o el contingut d'un directori de forma recursiva. A diferència de `df`, que informa sobre el sistema de fitxers complet, `du` informa sobre el contingut específic d'un directori o fitxer. L'opció `-s` (summary) mostra només el total del directori sense llistar cada subdirectori. La combinació `du -sh * | sort -h` és molt útil per identificar ràpidament quins directoris o fitxers ocupen més espai dins d'una carpeta, cosa que ajuda a alliberar espai en discos plens.

```
du -sh /home/alumne         # Mida total del directori (llegible)
du -sh /var/*               # Mida de cada element dins /var
du -sh * | sort -h          # Ordena per mida
```

`lsblk` --- Llistar dispositius de bloc

`lsblk` (de l'anglès *list block devices*) mostra tots els dispositius de bloc del sistema (discos, particions, dispositius de loopback, etc.) en forma d'arbre que reflecteix la seva relació jeràrquica. A diferència de `fdisk -l`, `lsblk` no requereix privilegis de root per mostrar la informació bàsica. L'opció `-f` afegeix informació sobre el sistema de fitxers, l'UUID (identificador únic universal de la partició) i el punt de muntatge, cosa que és molt útil per configurar el fitxer `/etc/fstab`.

```
lsblk                   # Arbre de discos i particions
lsblk -f                # Inclou el sistema de fitxers i UUID
```

`blkid` --- Informació de particions

`blkid` mostra l'UUID, el tipus de sistema de fitxers i altres atributs de les particions i dispositius de bloc. L'UUID és un identificador únic que no canvia fins i tot si el dispositiu es connecta a un port diferent o si es canvia l'ordre dels discos, al contrari que els noms de dispositiu com `/dev/sda1` que poden variar. Per aquest motiu, al fitxer `/etc/fstab` és recomanable usar l'UUID en lloc del nom del dispositiu per garantir que el sistema arrenca correctament independentment de l'ordre en què es detecten els discos.

```
blkid                    # Mostra UUID i tipus de totes les particions
blkid /dev/sdb1            # Informació d'una partició concreta
```

6. Còpies de seguretat

Mòdul relacionat: **0226 Seguretat informàtica**

Les còpies de seguretat (backups) són una de les tasques més crítiques de l'administració de sistemes. Una bona política de còpies de seguretat ha de definir la freqüència (diària, setmanal, mensual), el tipus (completa, incremental, diferencial), la ubicació (local, remota, núvol) i el procediment de restauració. A Linux, les eines de còpia de seguretat integrades al sistema operatiu permeten implementar solucions robustes sense necessitat de programari de tercers.

`tar` --- Crear i gestionar arxius comprimits

`tar` (de l'anglès *tape archive*) és l'eina estàndard de Linux per empaquetar múltiples fitxers i directoris en un únic arxiu i, opcionalment, comprimir-lo. El nom ve del seu ús original amb cintes magnètiques. Un arxiu `tar` sense compressió s'anomena *tarball* i té l'extensió `.tar`; amb compressió gzip s'usa `.tar.gz` o `.tgz`, i amb bzip2 s'usa `.tar.bz2`. La compressió bzip2 ofereix millor ràtio de compressió, però és més lenta que gzip. L'ordre de les opcions és important: `f` ha d'anar sempre just abans del nom del fitxer. `tar` és l'ordre fonamental per a còpies de seguretat manuals o en scripts.

```
# Crear
tar -czvf backup.tar.gz /home/alumne/    # Crea un arxiu comprimit
 ↪  (gzip)
tar -cjvf backup.tar.bz2 /home/alumne/   # Compressió bzip2 (millor
 ↪  ràtio)
tar -cvf backup.tar /home/alumne/        # Sense compressió

# Extreure
tar -xzvf backup.tar.gz                  # Extreu al directori actual
tar -xzvf backup.tar.gz -C /tmp/         # Extreu a /tmp

# Llistar contingut
tar -tzvf backup.tar.gz                  # Llista sense extreure
```

Opcions clau:

- `c` → Crear arxiu
- `x` → Extreure
- `t` → Llistar
- `z` → Comprimir amb gzip
- `j` → Comprimir amb bzip2
- `v` → Mode verbose (mostra el progrés)
- `f` → Especifica el fitxer

`gzip` / `gunzip` --- Comprimir fitxers individuals

`gzip` comprimeix un fitxer únic usant l'algorisme Deflate (similar al format ZIP). Per defecte, substitueix el fitxer original per la versió comprimida afegint l'extensió `.gz`. A diferència de `tar`, no permet empaquetar múltiples fitxers en un sol arxiu; per fer-ho cal combinar-lo amb `tar`. `gunzip` és equivalent a `gzip -d` i descomprimeix fitxers `.gz`. La relació entre `tar` i `gzip` és complementària: `tar` empaqueta i `gzip` comprimeix, i és per això que s'usen conjuntament amb les opcions `z` de `tar`.

```
gzip fitxer.txt            # Comprimeix (crea fitxer.txt.gz i elimina
 ↪  l'original)
gzip -k fitxer.txt         # Comprimeix però manté l'original
gzip -d fitxer.txt.gz      # Descomprimeix
gunzip fitxer.txt.gz       # Descomprimeix (equivalent a gzip -d)
```

`rsync` --- Sincronització de fitxers

`rsync` (de l'anglès *remote sync*) és una eina de sincronització de fitxers molt més intel·ligent que una simple còpia: compara origen i destí i transfereix únicament les diferències, cosa que fa les sincronitzacions molt més eficients quan els fitxers canvien poc entre execucions. Pot operar localment o a través de la xarxa (usant SSH com a transport segur amb l'opció `-z` per comprimir les dades en trànsit). L'opció `--delete` fa que el destí sigui un mirall exacte de l'origen, eliminant al destí els fitxers que ja no existeixen a l'origen. Això permet implementar còpies de seguretat incrementals eficients. L'opció `-n` (dry-run) simula l'operació sense fer cap canvi, per verificar el que es faria.

```
rsync -av /origen/ /destí/                  # Sincronitza localment
rsync -avz /home/alumne/ usuari@servidor:/backup/   # Cap a un
 ↪  servidor remot
rsync -avz --delete /origen/ /destí/        # Elimina al destí el que
 ↪  no existeix a l'origen
rsync -avzn /origen/ /destí/               # Simulació (dry-run, no fa
 ↪  canvis)
```

`dd` --- Còpia bit a bit (imatges de disc)

`dd` (de l'anglès *data duplicator* o *disk dump*) copia dades bloc a bloc, bit a bit, independentment del sistema de fitxers. Això el fa ideal per crear imatges exactes de discos o particions, clonar discos, crear imatges d'arrencada o esborrar discos de forma segura. A diferència de les còpies de fitxers convencionals, `dd` ho copia tot: dades, espai buit, metadades del sistema de fitxers i fins i tot les particions eliminades. `if` significa *input file* (origen) i `of` significa *output file* (destí). `bs` especifica la mida del bloc de lectura/escriptura; valors més grans (1M) milloren el rendiment. `dd` no dona cap avís si el destí és incorrecte, de manera que una equivocació pot destruir tot el contingut d'un disc.

```
dd if=/dev/sda of=/backup/disc.img          # Còpia del disc a fitxer
```

```
dd if=/backup/disc.img of=/dev/sdb          # Restaura la imatge al
 ↪  disc
dd if=/dev/sda of=/dev/sdb                  # Clona un disc sencer
dd if=/dev/zero of=/dev/sdb bs=1M          # Esborra el disc amb zeros
```

`dd` és molt potent i perillós. Verificar sempre origen (`if`) i destí (`of`) abans d'executar.

7. Diagnòstic i monitoratge de xarxa

Mòdul relacionat: **0225 Xarxes locals** · **0227 Serveis de xarxa**

El diagnòstic de xarxa és una de les habilitats més importants d'un tècnic de sistemes. Quan un servei no és accessible o una connexió falla, cal seguir un procés sistemàtic de resolució de problemes: primer comprovar la connectivitat bàsica (ping), després la ruta (traceroute), la resolució de noms (dig/nslookup), els ports oberts (netstat/ss) i finalment la configuració de les interfícies (ip/ifconfig).

`ping` --- Comprovar connectivitat

`ping` envia paquets ICMP Echo Request a un host i espera les respostes (ICMP Echo Reply). Permet comprovar si un host és accessible a la xarxa i mesurar la latència (temps de resposta) i la pèrdua de paquets. En Linux, `ping` s'executa indefinidament fins que s'atura amb Ctrl+C, a menys que s'especifiqui el nombre de paquets amb `-c`. Si el ping a una IP funciona, però el ping a un nom de domini no funciona, el problema és de resolució DNS, no de connectivitat de xarxa. Si el ping a la porta d'enllaç funciona, però no a Internet, el problema és de routing o del proveïdor.

```
ping 8.8.8.8                    # Ping continu a Google DNS
ping -c 4 8.8.8.8               # Envia 4 paquets i para
ping -c 4 www.google.com        # Ping a un nom de domini (prova DNS)
ping -i 2 servidor              # Un ping cada 2 segons
```

`traceroute` --- Ruta dels paquets

`traceroute` (o `tracepath` en algunes distribucions) mostra el camí que segueixen els paquets des del nostre equip fins a la destinació, passant per tots els routers intermedis (*salts*). Per a cada salt mostra la seva adreça IP, el nom DNS si en té, i els temps de resposta. És molt útil per diagnosticar on es produeix un problema de xarxa: si els paquets arriben fins a cert punt i després es perden o el temps de resposta s'incrementa molt, indica on pot ser el problema. Alguns routers de la ruta ignoren els paquets de `traceroute` per seguretat i apareixen com `* * *` a la sortida.

```
traceroute www.google.com       # Mostra el camí fins a la destinació
traceroute -n www.google.com    # Sense resolució de noms (més ràpid)
```

`netstat` --- Estadístiques de xarxa (clàssic)

`netstat` mostra estadístiques i informació sobre les connexions de xarxa, les taules de rutes, les estadístiques d'interfícies i molt més. Ha estat substituïda per `ss` en les distribucions modernes, però encara és àmpliament usada. Les opcions `tuln` mostren les connexions TCP (`t`) i UDP (u) en mode escolta (`l`), és a dir, els ports que el sistema té oberts esperant connexions entrants, en format numèric (n, sense resolució de noms per ser més ràpid). L'addició de p requereix privilegis de root i mostra el procés que controla cada port, cosa molt útil per identificar quin servei usa cada port.

```
netstat -tuln          # Ports oberts (TCP/UDP, mode escolta, numèric)
netstat -an             # Totes les connexions
netstat -r              # Taula de rutes
netstat -i              # Estadístiques d'interfícies
netstat -tulnp         # Ports oberts amb el procés que els usa (root)
```

`ss` --- Estadístiques de xarxa (modern, substitueix netstat)

`ss` (de l'anglès *socket statistics*) és la substituta moderna de `netstat`, amb millor rendiment i més funcionalitats. Accedeix directament al nucli del sistema per obtenir la informació, de manera que és més ràpida i precisa. Usa les mateixes opcions principals que `netstat` (`t`, u, `l`, n, p), cosa que facilita la transició. A més, pot mostrar informació detallada sobre l'estat intern de les connexions TCP, molt útil per a diagnòstics avançats.

```
ss -tuln                # Ports oberts
ss -tulnp               # Ports oberts amb el procés (root)
ss -s                   # Resum d'estadístiques
ss -an                  # Totes les connexions
```

`nmap` --- Escàner de xarxa i ports

`nmap` (de l'anglès *network mapper*) és l'eina d'exploració de xarxa i auditoria de seguretat més usada. Permet descobrir hosts a la xarxa, identificar quins ports estan oberts, detectar els serveis que s'executen i fins i tot el sistema operatiu dels equips. En un context de seguretat informàtica, s'usa tant per a auditories legítimes (comprovant l'exposició dels nostres propis sistemes) com per a reconeixement (descobrint sistemes a la xarxa). Cal tenir present que escanejar sistemes aliens sense autorització és il·legal en la majoria de jurisdiccions. En entorns de pràctiques, `nmap` s'usa per explorar la xarxa local i verificar la configuració dels servidors.

```
nmap 192.168.1.1                  # Escaneig bàsic d'un host
nmap -p 80,443 192.168.1.1        # Escaneig de ports concrets
nmap -p 1-1000 192.168.1.1        # Escaneig d'un rang de ports
nmap 192.168.1.0/24               # Escaneig de tota la xarxa local
nmap -sV 192.168.1.1              # Detecta versions dels serveis
nmap -O 192.168.1.1              # Detecta el sistema operatiu (root)
```

`ifconfig` --- Configuració d'interfícies (clàssic)

`ifconfig` (de l'anglès *interface configuration*) és l'ordre clàssica per mostrar i configurar les interfícies de xarxa. Ha estat deprecada a favor de l'ordre `ip` en les distribucions modernes, però segueix estant disponible i és molt usada per la seva simplicitat. En molts sistemes cal instal·lar el paquet `net-tools` per disposar d'ella. Mostra informació com l'adreça IP, la màscara de xarxa, l'adreça MAC, l'MTU i les estadístiques de paquets enviats i rebuts.

```
ifconfig                    # Mostra totes les interfícies actives
ifconfig -a                 # Inclou les interfícies desactivades
ifconfig eth0               # Informació d'una interfície concreta
```

`ip` --- Gestió de xarxa (modern, substitueix ifconfig)

l'ordre `ip` és l'eina moderna per a la gestió de xarxa a Linux, part del paquet `iproute2`. Substitueix `ifconfig`, `route`, `arp` i altres ordres clàssiques en una sola eina unificada amb una sintaxi consistent. Permet gestionar adreces IP, rutes, interfícies, regles de routing i molt més. A diferència de `ifconfig`, els canvis fets amb `ip` en la línia d'ordres no sobreviuen a un reinici del sistema; per fer-los permanents cal modificar els fitxers de configuració de xarxa o usar `nmcli`.

```
ip addr                         # Mostra adreces IP de totes les
 ↪  interfícies
ip addr show eth0               # Informació de la interfície eth0
ip link                         # Estat de les interfícies
 ↪  (activa/inactiva)
ip link set eth0 up             # Activa la interfície
ip link set eth0 down           # Desactiva la interfície
ip route                        # Taula de rutes
ip route show                   # Mostra les rutes
```

`arp` --- Taula ARP

ARP (de l'anglès *Address Resolution Protocol*) és el protocol que resol adreces IP a adreces MAC (físiques) dins d'una xarxa local. Quan un equip vol comunicar-se amb una IP de la seva xarxa, primer consulta la seva taula ARP per trobar la MAC corresponent; si no la té, envia una petició ARP broadcast a tota la xarxa. L'ordre `arp` permet consultar i manipular aquesta taula. La taula ARP es va omplint automàticament a mesura que el sistema es comunica amb altres equips i les entrades caduquen passats uns minuts si no s'han usat.

```
arp -a                          # Mostra la taula ARP (IP ↔ MAC)
arp -d 192.168.1.1              # Elimina una entrada de la taula
```

`nslookup` / `dig` / `host` --- Resolució DNS

Aquestes tres ordres permeten fer consultes al sistema DNS (Domain Name System) per resoldre noms de domini a adreces IP i viceversa. DNS és un sistema jeràrquic i distribuït que actua com la "guia telefònica" d'Internet. `nslookup` és l'ordre més antiga i simple, disponible en tots els sistemes (inclòs Windows). `dig` (de l'anglès *Domain Information Groper*) és la més completa i detallada, preferida pels administradors de sistemes per al diagnòstic. `host` és la més senzilla i mostra la informació de forma concisa. Totes permeten especificar un servidor DNS concret per a la consulta, cosa molt útil per verificar si un servidor DNS resol correctament un domini.

```
# nslookup
nslookup www.google.com             # Consulta DNS bàsica
nslookup www.google.com 8.8.8.8     # Usa un servidor DNS específic

# dig (més detallat)
dig www.google.com                  # Consulta DNS completa
dig www.google.com A                # Registre de tipus A (IPv4)
dig www.google.com MX               # Registres de correu
dig @8.8.8.8 www.google.com         # Usa el servidor 8.8.8.8

# host (més senzill)
host www.google.com                 # Resolució ràpida
host 8.8.8.8                        # Resolució inversa (IP → nom)
```

8. Configuració de xarxa

Mòdul relacionat: **0225 Xarxes locals** · **0224 Sistemes operatius en xarxa**

La configuració de xarxa a Linux es pot fer de forma temporal (canvis que es perden en reiniciar) o permanent (modificant fitxers de configuració). En distribucions modernes, `NetworkManager` és el servei responsable de gestionar les connexions de xarxa, i `nmcli` és la seva interfície de línia d'ordres. En servidors, sovint es prefereix configurar la xarxa directament en fitxers com `/etc/network/interfaces` (Debian/Ubuntu) o fitxers a `/etc/sysconfig/network-scripts/` (Red Hat/CentOS).

`ip addr` --- Assignar adreces IP

La subordre `ip addr` gestiona les adreces IP assignades a les interfícies de xarxa. Permet afegir múltiples adreces IP a una sola interfície (ús típic en servidors web virtuals), eliminar-ne i mostrar les actuals. La notació CIDR (per exemple, `/24`) indica la màscara de subxarxa: `/24` equival a `255.255.255.0`. Cal recordar que aquests canvis són temporals; per fer-los permanents cal modificar la configuració de NetworkManager o els fitxers de configuració de xarxa del sistema.

```
ip addr add 192.168.1.100/24 dev eth0      # Afegeix una IP a
 ↪  l'interfície
ip addr del 192.168.1.100/24 dev eth0      # Elimina la IP
```

```
ip addr flush dev eth0                    # Elimina totes les IPs de
 ↪  l'interfície
```

`ip route` --- Gestió de rutes

La subordre `ip route` gestiona la taula de rutes del sistema, que determina per on s'envien els paquets en funció de la seva destinació. La ruta per defecte (*default gateway*) s'usa per a tots els paquets que no coincideixen amb cap altra ruta, normalment per sortir a Internet. Les rutes estàtiques s'usen per especificar camins concrets a xarxes específiques, cosa habitual en xarxes corporatives amb múltiples segments. La taula de rutes es consulta per cada paquet que el sistema ha d'enviar, seguint el principi de la ruta més específica (la que té el prefix de xarxa més llarg coincident).

```
ip route add default via 192.168.1.1      # Afegeix la porta d'enllaç
 ↪  per defecte
ip route add 10.0.0.0/8 via 192.168.1.254  # Afegeix una ruta estàtica
ip route del default                     # Elimina la ruta per defecte
```

`nmcli` --- Gestió de xarxa amb NetworkManager

`nmcli` (de l'anglès *NetworkManager command-line interface*) és la interfície de línia d'ordres de NetworkManager, el dimoni que gestiona les connexions de xarxa en la majoria de distribucions Linux d'escriptori i molts servidors. Les connexions configurades a través de NetworkManager es guarden de forma permanent i es restauren automàticament en reiniciar. Resulta especialment útil per gestionar connexions Wi-Fi des de la línia d'ordres en servidors sense entorn gràfic. `nmtui` és la variant amb interfície de text (TUI) que proporciona un menú interactiu més amigable per a configuracions puntuals.

```
nmcli device status                        # Estat de les interfícies
nmcli connection show                      # Llista les connexions
 ↪  guardades
nmcli connection up "Nom-Connexio"         # Activa una connexió
nmcli connection down "Nom-Connexio"       # Desactiva una connexió
nmcli device wifi list                     # Llista xarxes Wi-Fi
 ↪  disponibles
nmcli device wifi connect "SSID" password "contrasenya"  # Connecta a
 ↪  Wi-Fi
```

`ifup` / `ifdown` --- Activar/desactivar interfícies

`ifup` i `ifdown` activen i desactiven interfícies de xarxa llegint la seva configuració del fitxer `/etc/network/interfaces` (en sistemes Debian/Ubuntu que no usen NetworkManager). Apliquen la configuració definida en aquest fitxer, incloent-hi l'adreça IP, la màscara, la porta d'enllaç i el DNS. Són ordres simples però menys flexibles que `nmcli`. En sistemes moderns amb NetworkManager, pot ser que no funcionin correctament si les interfícies són gestionades per NetworkManager.

```
ifup eth0           # Activa la interfície eth0
ifdown eth0         # Desactiva la interfície eth0
```

9. Transferència de fitxers i accés remot

Mòdul relacionat: **0227 Serveis de xarxa**

L'accés remot segur és una de les habilitats fonamentals en l'administració de sistemes. En entorns de producció, tots els equips de servidors es gestionen de forma remota des d'una estació de treball, sovint a través de la xarxa corporativa o fins i tot a través d'Internet. El protocol SSH (Secure Shell) és l'estàndard actual per a accés remot segur, ja que xifra tot el tràfic. Els protocols antics com Telnet o FTP han d'evitar-se perquè transmeten les dades (incloent-hi les contrasenyes) en text pla i qualsevol persona a la xarxa podria interceptar-les.

`ssh` --- Connexió remota segura

`ssh` (de l'anglès *Secure Shell*) estableix una connexió xifrada amb un servidor remot, que permet executar ordres en ell com si estiguéssim asseguts davant del teclat. El xifratge és bidireccional: tant les dades enviades com les rebudes estan protegides. L'autenticació pot ser per contrasenya o per parell de claus criptogràfiques (clau pública/privada), sent aquesta última molt més segura i convenient per a l'automatització. `ssh-keygen` genera el parell de claus, i `ssh-copy-id` copia la clau pública al servidor per permetre l'autenticació sense contrasenya. La configuració del client SSH es pot personalitzar al fitxer `~/.ssh/config`.

```
ssh usuari@servidor                          # Connexió bàsica
ssh usuari@192.168.1.10                      # Connexió per IP
ssh -p 2222 usuari@servidor                  # Port alternatiu
ssh -i clau.pem usuari@servidor              # Autenticació per clau
ssh-keygen -t rsa -b 4096                   # Genera un parell de claus
 ↪  RSA
ssh-copy-id usuari@servidor                  # Copia la clau pública al
 ↪  servidor
```

`scp` --- Còpia segura de fitxers

`scp` (de l'anglès *Secure Copy Protocol*) permet copiar fitxers entre equips locals i remots usant el protocol SSH per al transport, de manera que les dades viatgen sempre xifrades. La sintaxi per a l'especificació de fitxers remots és `usuari@host:/ruta/fitxer`. Si la ruta del destí és un directori, el fitxer es copia dins d'aquest mantenint el nom original. L'opció `-r` permet copiar directoris sencers de forma recursiva. Una limitació de `scp` és que no permet reprendre transferències interrompudes; per a fitxers grans o connexions poc fiables és millor usar `rsync`.

```
scp fitxer.txt usuari@servidor:/destí/          # Còpia local → remot
scp usuari@servidor:/ruta/fitxer.txt /local/    # Còpia remot → local
scp -r directori/ usuari@servidor:/destí/       # Còpia recursiva
scp -P 2222 fitxer.txt usuari@servidor:/destí/  # Port alternatiu
```

`sftp` --- Transferència de fitxers segura (interactiu)

`sftp` (de l'anglès *SSH File Transfer Protocol*) proporciona una interfície interactiva per a la transferència de fitxers a través de SSH, similar a la del client FTP clàssic però amb el xifratge de SSH. Una vegada establerta la connexió, s'obre una sessió interactiva on es poden executar ordres per navegar pels directoris remots i locals, llistar fitxers i transferir-los en les dues direccions. Les ordres que comencen per `l` (local) operen sobre el sistema local (per exemple, `lls` per llistar fitxers locals, `lcd` per canviar el directori local).

```
sftp usuari@servidor            # Obre la sessió SFTP

# ordres dins de sftp:
# get fitxer.txt                # Baixa un fitxer
# put fitxer.txt                # Puja un fitxer
# ls                            # Llista fitxers remots
# lls                           # Llista fitxers locals
# cd directori                  # Canvia directori remot
# lcd directori                 # Canvia directori local
# quit                          # Surt
```

`ftp` --- Transferència de fitxers (sense xifrar)

`ftp` (de l'anglès *File Transfer Protocol*) és el protocol clàssic de transferència de fitxers. La seva limitació principal és que totes les comunicacions, incloses les credencials d'autenticació, viatgen en text pla sense cap xifratge, cosa que el fa inadequat per a entorns on la seguretat importa. Malgrat això, segueix usant-se en entorns controlats, xarxes internes aïllades, i és important conèixer-lo per als continguts del mòdul de serveis de xarxa (0227). El mode passiu (`-p`) s'usa quan el client es troba darrere d'un tallafoc o NAT i no pot acceptar connexions entrants del servidor.

```
ftp servidor                    # Connexió FTP
ftp -p servidor                 # Mode passiu
```

```
# ordres dins de ftp:
# user nom contrasenya          # Autenticació
# get fitxer.txt                # Baixa un fitxer
# put fitxer.txt                # Puja un fitxer
# ls                            # Llista fitxers
# binary                       # Mode binari (per fitxers no de text)
# ascii                         # Mode text
# bye                           # Surt
```

`wget` --- Baixar fitxers d'Internet

`wget` (de l'anglès *web get*) és una eina no interactiva per descarregar fitxers des de la web usant els protocols HTTP, HTTPS i FTP. “No interactiva” significa que pot funcionar en segon pla i no requereix que l'usuari estigui present durant la descàrrega; fins i tot pot reprendre descàrregues interrompudes amb l'opció `-c`. És molt útil en scripts d'instal·lació automatitzada per baixar programari, i en servidors sense entorn gràfic on no hi ha navegador disponible. Pot descarregar llocs web sencers recursivament per consultar-los sense connexió.

```
wget https://exemple.com/fitxer.zip          # Baixa un fitxer
wget -c https://exemple.com/fitxer.zip       # Continua una baixada
 ↪  interrompuda
wget -O nom.zip https://exemple.com/f.zip   # Canvia el nom del fitxer
 ↪  baixat
wget -r https://exemple.com/                 # Baixa un lloc web
 ↪  recursivament
```

`curl` --- Transferència de dades amb URL

`curl` (de l'anglès *client URL*) és una eina molt versàtil per transferir dades amb diversos protocols (HTTP, HTTPS, FTP, SFTP, SMTP, i molts més). A diferència de `wget`, que se centra en la descàrrega de fitxers, `curl` està orientat a la interacció amb URL i API web: permet especificar capçaleres HTTP personalitzades, enviar dades en peticions POST, gestionar galetes, autenticar-se i molt més. Per defecte mostra la sortida a la pantalla en lloc de guardar-la en un fitxer. És l'eina preferida per provar API REST i serveis web des de la línia d'ordres.

```
curl https://exemple.com                  # Mostra el contingut d'una
 ↪  URL
curl -O https://exemple.com/fitxer.zip       # Baixa un fitxer
curl -o nom.zip https://exemple.com/f.zip    # Baixa amb nom
 ↪  personalitzat
curl -X POST -d "dades" https://api.com/    # Petició POST (per a APIs)
curl -I https://exemple.com                # Mostra les capçaleres HTTP
```

10. Tallafoc i seguretat

Mòdul relacionat: **0226 Seguretat informàtica**

El tallafoc (*firewall*) és la primera línia de defensa d'un sistema Linux en xarxa. Controla el tràfic de xarxa entrant i sortint basant-se en regles predefinides. A Linux, el tallafoc s'implementa en l'àmbit del nucli del sistema operatiu a través de `netfilter`, i les eines `iptables` i `nftables` proporcionen la interfície per configurar les seves regles. `ufw` és una capa d'abstracció sobre `iptables` que simplifica la gestió del tallafoc per a casos d'ús habituals.

`iptables` --- Tallafoc per regles (baix nivell)

`iptables` és la interfície de línia d'ordres per configurar el tallafoc `netfilter` del nucli Linux. Organitza les regles en tres cadenes principals: `INPUT` (tràfic entrant al sistema), `OUTPUT` (tràfic sortint del sistema) i `FORWARD` (tràfic que passa pel sistema cap a un altre equip, usada en encaminadors). Per a cada paquet, el nucli recorre les regles de la cadena corresponent en ordre i aplica la primera que coincideixi. Si cap regla coincideix, s'aplica la política per defecte de la cadena. `iptables` és molt potent, però té una sintaxi complexa; en distribucions modernes, `nftables` és el seu successor, tot i que `iptables` segueix estant àmpliament documentat i usat.

```
# Llistar regles
iptables -L                             # Llista totes les regles
iptables -L -v -n                       # Format detallat i numèric

# Regles bàsiques
iptables -A INPUT -p tcp --dport 22 -j ACCEPT      # Permet SSH
iptables -A INPUT -p tcp --dport 80 -j ACCEPT      # Permet HTTP
iptables -A INPUT -p tcp --dport 443 -j ACCEPT     # Permet HTTPS
iptables -A INPUT -j DROP                          # Bloqueja la resta

# Eliminar regles
iptables -D INPUT -p tcp --dport 80 -j ACCEPT      # Elimina una regla
iptables -F                             # Elimina totes les regles
 ↪  (flush)

# Guardar regles
iptables-save > /etc/iptables/rules.v4
iptables-restore < /etc/iptables/rules.v4
```

`ufw` --- Tallafoc simplificat (Ubuntu/Debian)

`ufw` (de l'anglès *Uncomplicated Firewall*) és una interfície simplificada per gestionar `iptables`, dissenyada per facilitar la configuració del tallafoc sense necessitat de conèixer la sintaxi complexa d'`iptables`. Permet definir regles basant-se en ports, protocols, serveis pel seu nom (llegint el fitxer `/etc/services`) o adreces IP. Les regles d'`ufw` es guarden i es restauren automàticament en reiniciar. Per defecte, `ufw` usa una política de “denegar tot el tràfic entrant i permetre tot el sortint”, que és una configuració segura per a la majoria de servidors.

```
ufw status                          # Estat del tallafoc
ufw enable                          # Activa el tallafoc
ufw disable                         # Desactiva el tallafoc
ufw allow 22                        # Permet el port 22 (SSH)
ufw allow ssh                       # Equivalent a l'anterior
ufw allow 80/tcp                    # Permet HTTP
ufw allow from 192.168.1.0/24       # Permet tota una xarxa
ufw deny 23                         # Bloqueja Telnet
ufw delete allow 80                 # Elimina una regla
ufw reset                           # Restableix la configuració per
 ↪  defecte
```

`gpg` --- Xifratge i signatures

`gpg` (de l'anglès *GNU Privacy Guard*) implementa l'estàndard OpenPGP per al xifratge de fitxers i comunicacions i per a la creació de signatures digitals. Usa criptografia asimètrica: cada usuari té un parell de claus, una pública (que es pot compartir lliurement) i una privada (que mai s'ha de compartir). Per xifrar un fitxer per a algú, s'usa la seva clau pública; el destinatari el pot desxifrar amb la seva clau privada. Per a les signatures digitals, s'usa la clau privada per signar i la pública per verificar. `gpg` s'usa àmpliament per verificar la integritat i autenticitat de paquets de programari.

```
gpg --gen-key                                 # Genera un parell de
 ↪  claus
gpg --encrypt -r usuari@exemple.com fitxer.txt  # Xifra un fitxer
gpg --decrypt fitxer.txt.gpg                  # Desxifra un fitxer
gpg --sign fitxer.txt                         # Signa un fitxer
gpg --verify fitxer.txt.gpg                  # Verifica la signatura
```

`openssl` --- Certificats i xifratge

`openssl` és una implementació de codi obert dels protocols SSL/TLS i d'una àmplia biblioteca de funcions criptogràfiques. S'usa per generar claus RSA, crear certificats X.509 (usats en HTTPS), verificar certificats, xifrar fitxers i provar connexions SSL/TLS. En entorns de pràctiques i en servidors interns, s'usen certificats autosignats (creats amb `openssl`) que proporcionen el xifratge però no la verificació d'identitat per una autoritat certificadora externa. Per al HTTPS públic, cal usar certificats d'una CA reconeguda (com Let's Encrypt).

```
# Generar clau privada i certificat autosignat
openssl genrsa -out clau.pem 2048
openssl req -new -x509 -key clau.pem -out cert.pem -days 365

# Verificar un certificat
openssl x509 -in cert.pem -text -noout

# Comprovar un servidor HTTPS
openssl s_client -connect exemple.com:443
```

`md5sum` / `sha256sum` --- Verificació d'integritat

Aquestes ordres calculen el hash criptogràfic d'un fitxer, que és una “empremta digital” única del seu contingut. Si el fitxer canvia ni que sigui un sol bit, el hash resultant serà completament diferent. Això permet verificar que un fitxer baixat d'Internet no s'ha corromput durant la transmissió ni ha estat modificat maliciosament (sempre que el hash de referència s'obtingui d'una font fiable). MD5 ja no es considera segur per a aplicacions criptogràfiques (és possible generar col·lisions), però segueix sent útil per a la verificació d'integritat simple. SHA-256 és el recomanat per a usos actuals. Molts llocs de descàrrega de distribucions Linux publiquen els hashes SHA-256 dels seus ISO perquè els usuaris puguin verificar les descàrregues.

```
md5sum fitxer.iso               # Calcula el hash MD5
sha256sum fitxer.iso            # Calcula el hash SHA-256 (recomanat)
sha256sum -c checksums.txt      # Verifica fitxers contra una llista
 ↪  de hashes
```

11. Gestió de programari

Mòdul relacionat: **0222 Sistemes operatius monolloc**

Linux usa sistemes de gestió de paquets per instal·lar, actualitzar i eliminar programari de forma centralitzada i segura. Un paquet és un arxiu comprimit que conté els binaris del programa, fitxers de configuració, documentació i metadades (com la llista de dependències). El gestor de paquets s'encarrega de descarregar el paquet des dels dipòsits oficials, verificar la seva autenticitat (signatura GPG), instal·lar les dependències necessàries i registrar la instal·lació al sistema. Existeixen dues famílies principals: Debian/Ubuntu (`apt`, `dpkg`) i Red Hat/Fedora/CentOS (`dnf`, `yum`, `rpm`).

Sistemes basats en Debian/Ubuntu (`apt`)

`apt` (de l'anglès *Advanced Package Tool*) és el gestor de paquets d'alt nivell de les distribucions basades en Debian. Gestiona automàticament les dependències: si un programa necessita una biblioteca que no està instal·lada, `apt` la descarrega i instal·la automàticament. La llista de paquets disponibles es troba als dipòsits, les adreces dels quals es configuren a `/etc/apt/sources.list` i als fitxers de `/etc/apt/sources.list.d/`. `apt update` actualitza la llista local de paquets disponibles (no instal·la res), i `apt upgrade` instal·la les versions noves dels paquets ja instal·lats.

```
apt update                          # Actualitza la llista de paquets
apt upgrade                         # Actualitza tots els paquets
 ↪  instal·lats
apt full-upgrade                  # Actualitza i gestiona dependències
apt install nginx                   # Instal·la el paquet nginx
apt install -y nginx                # Instal·la sense demanar
 ↪  confirmació
apt remove nginx                    # Desinstal·la (manté la
 ↪  configuració)
apt purge nginx                     # Desinstal·la i elimina la
 ↪  configuració
apt autoremove                      # Elimina paquets que ja no cal
apt search paraula                  # Cerca paquets pel nom
apt show nginx                     # Informació detallada d'un paquet
apt list --installed                # Llista els paquets instal·lats
```

`dpkg` --- Gestió de paquets .deb (baix nivell)

`dpkg` (de l'anglès *Debian Package*) és l'eina de baix nivell que gestiona els paquets `.deb` directament, sense connexió a cap dipòsit. A diferència d'`apt`, no resol dependències automàticament; si un paquet requereix una biblioteca que no està instal·lada, `dpkg` fallarà i indicarà quines dependències manquen. S'usa principalment per instal·lar paquets `.deb` baixats manualment (per exemple, Google Chrome) o per consultar informació sobre els paquets instal·lats. `apt` usa internament `dpkg` per a la instal·lació física dels paquets.

```
dpkg -i paquet.deb                  # Instal·la un paquet .deb local
dpkg -r nginx                       # Elimina un paquet
dpkg -l                           # Llista tots els paquets instal·lats
dpkg -l | grep nginx                # Cerca un paquet concret
dpkg -L nginx                      # Fitxers instal·lats per un paquet
```

Sistemes basats en Red Hat/CentOS (`dnf` / `yum`)

`dnf` (de l'anglès *Dandified YUM*) és el gestor de paquets modern de les distribucions basades en Red Hat Enterprise Linux (RHEL), Fedora i CentOS. Substitueix `yum` (que segueix disponible per compatibilitat en moltes distribucions) amb millor rendiment i gestió de dependències. Funciona de manera similar a `apt` però amb una sintaxi lleugerament diferent. Els paquets es distribueixen en format `.rpm` i els dipòsits es configuren a `/etc/yum.repos.d/`.

```
dnf update                          # Actualitza tots els paquets
dnf install httpd                   # Instal·la Apache
dnf remove httpd                    # Desinstal·la
dnf search paraula                  # Cerca paquets
dnf info httpd                      # Informació d'un paquet
dnf list installed                  # Paquets instal·lats
```

`rpm` --- Gestió de paquets .rpm (baix nivell)

`rpm` (de l'anglès *RPM Package Manager*, anteriorment *Red Hat Package Manager*) és l'eina de baix nivell per a la gestió de paquets `.rpm`, equivalent a `dpkg` en els sistemes Debian. Com `dpkg`, no gestiona dependències automàticament, cosa que fa que la instal·lació de paquets individuals amb dependències complexes pugui ser complicada (el fenomen conegut com a “infern de les dependències”, que va motivar la creació de `yum`/`dnf`). `rpm` és molt útil per consultar informació sobre paquets instal·lats, com ara veure tots els fitxers d'un paquet o comprovar si un fitxer prové d'algun paquet.

```
rpm -ivh paquet.rpm                    # Instal·la un paquet .rpm local
rpm -e httpd                           # Desinstal·la
rpm -qa                              # Llista tots els paquets instal·lats
rpm -qi httpd                          # Informació d'un paquet
```

12. Utilitats generals

Les utilitats generals de processament de text i de la línia d'ordres són eines fonamentals en l'administració de sistemes Linux. La filosofia Unix es basa en el principi que cada programa fa una cosa i la fa bé, i que les eines es combinen a través de canonades (pipes) per dur a terme tasques complexes. Dominar aquestes utilitats permet processar i analitzar fitxers de text de manera molt eficient, cosa que és especialment útil per treballar amb fitxers de configuració i de registre.

`grep` --- Cercar text en fitxers

`grep` (de l'anglès *global regular expression print*) és una de les ordres més usades a Linux. Busca línies que coincideixen amb un patró (que pot ser text simple o una expressió regular) dins d'un o més fitxers, i les mostra a la sortida. La seva versatilitat rau en la possibilitat d'usar expressions regulars, que permeten definir patrons de cerca molt complexos. L'opció `-r` el fa recórrer tots els fitxers d'un directori recursivament, cosa molt útil per cercar un text de configuració a tot `/etc`. La combinació de `grep` amb altres ordres a través de pipes és omnipresent a la línia d'ordres Linux.

```
grep "error" fitxer.log             # Cerca la paraula "error"
grep -i "error" fitxer.log          # Sense distingir majúscules
grep -r "error" /var/log/           # Cerca recursiva en un directori
grep -n "error" fitxer.log          # Mostra el número de línia
grep -v "info" fitxer.log           # Mostra les línies que NO
 ↪  contenen "info"
grep -c "error" fitxer.log          # Compta les ocurrències
grep "^root" /etc/passwd            # Línies que comencen per "root"
grep "bash$" /etc/passwd            # Línies que acaben per "bash"
```

`awk` --- Processament de text per columnes

`awk` és un potent llenguatge de processament de text especialitzat en el tractament de fitxers estructurats en columnes. El seu nom ve dels seus creadors: Aho, Weinberger i Kernighan. Processa el fitxer línia per línia i per a cada línia separa els camps (columnes) basant-se en un delimitador (espai en blanc per defecte, o el caràcter especificat amb `-F`). Les columnes s'accedeixen amb `$1`, `$2`, etc., i `$0` representa la línia sencera. A més de seleccionar columnes, `awk` pot calcular sumes, comptar coincidències, aplicar condicions i generar informes. El fitxer `/etc/passwd` és un exemple clàssic d'ús d'`awk` per extreure informació.

```
awk '{print $1}' fitxer.txt         # Mostra la primera columna
awk '{print $1, $3}' fitxer.txt     # Mostra la primera i tercera
 ↪  columna
awk -F: '{print $1}' /etc/passwd    # Usa ":" com a separador (llista
 ↪  usuaris)
awk '{sum += $1} END {print sum}' nums.txt  # Suma la primera columna
```

sed --- Editor de flux (substitucions)

`sed` (de l'anglès *stream editor*) és un editor de text no interactiu que processa fitxers línia per línia i aplica transformacions especificades per l'usuari. La seva operació més habitual és la substitució de text amb la sintaxi `s/patró/substitució/opcions`, on el patró pot ser text simple o una expressió regular. L'opció `g` (global) al final de la substitució fa que s'apliqui a totes les ocurrències de la línia, no només a la primera. L'opció `-i` modifica el fitxer directament (in-place) en lloc de mostrar el resultat per pantalla. `sed` és molt usat en scripts d'automatització per modificar fitxers de configuració de forma programàtica.

```
sed 's/vell/nou/' fitxer.txt         # Substitueix la primera
 ↪  ocurrència per línia
sed 's/vell/nou/g' fitxer.txt        # Substitueix totes les
 ↪  ocurrències
sed -i 's/vell/nou/g' fitxer.txt     # Modifica el fitxer directament
sed '/patró/d' fitxer.txt           # Elimina les línies amb el patró
sed -n '5,10p' fitxer.txt            # Mostra les línies 5 a 10
```

Més informació de sed.

`cut` --- Extreure columnes de text

`cut` extreu seccions (columnes o caràcters) de cada línia d'un fitxer. És més simple que `awk` però suficient per a moltes tasques de selecció de camps. L'opció `-d` especifica el delimitador de camps i `-f` indica quins camps extreure. L'opció `-c` extreu caràcters per posició. `cut` no pot reordenar els camps (els mostra en l'ordre original), a diferència d'`awk`. És especialment útil per processar fitxers CSV o fitxers de text amb camps separats per un caràcter fix com els punts i comes o els dos punts.

```
cut -d: -f1 /etc/passwd            # Primera columna amb separador ":"
cut -d, -f2,4 fitxer.csv            # Columnes 2 i 4 d'un CSV
cut -c1-10 fitxer.txt               # Caràcters de l'1 al 10 de cada
 ↪  línia
```

`sort` --- Ordenar línies

`sort` ordena les línies d'un fitxer de text. Per defecte, l'ordenació és lexicogràfica (alfabètica, caràcter per caràcter), cosa que pot donar resultats inesperats amb números (per exemple, 10 apareixeria abans que 9 en un ordre lexicogràfic). L'opció `-n` força l'ordenació numèrica correcta. L'opció `-k` permet ordenar per una columna específica, i és molt útil combinada amb el delimitador `-t`. `sort` és molt usat en combinació amb `uniq` per eliminar duplicats d'un fitxer, ja que `uniq` només elimina duplicats consecutius i, per tant, necessita que el fitxer estigui ordenat prèviament.

```
sort fitxer.txt                    # Ordena alfabèticament
sort -r fitxer.txt                 # Ordre invers
sort -n nums.txt                   # Ordre numèric
```

```
sort -k2 fitxer.txt                # Ordena per la segona columna
sort -u fitxer.txt                 # Elimina duplicats
```

`uniq` --- Eliminar o comptar duplicats

`uniq` elimina o compta les línies duplicades consecutives d'un fitxer. El punt clau és que `uniq` només detecta duplicats si les línies iguals són adjacents, de manera que normalment cal ordenar el fitxer primer amb `sort`. L'opció `-c` afegeix al principi de cada línia el nombre de vegades que apareix, cosa molt útil per analitzar freqüències en fitxers de registre (per exemple, per comptar quants cops ha aparegut cada missatge d'error). L'opció `-d` mostra únicament les línies que apareixen més d'una vegada, útil per trobar entrades repetides.

```
uniq fitxer.txt                    # Elimina línies consecutives
 ↪  duplicades
uniq -c fitxer.txt                 # Compta les repeticions
uniq -d fitxer.txt                # Mostra només les línies duplicades
sort fitxer.txt | uniq             # Elimina tots els duplicats
```

`wc` --- Comptar paraules, línies i caràcters

wc (de l'anglès *word count*) compta el nombre de línies, paraules i caràcters (o bytes) d'un fitxer o de l'entrada estàndard. Sense opcions, mostra les tres estadístiques. És molt útil per conèixer ràpidament les dimensions d'un fitxer o per comptar resultats filtrats: per exemple, `ls /etc | wc -l` compta el nombre de fitxers al directori `/etc`, o `grep "ERROR" fitxer.log | wc -l` compta el nombre de línies d'error en un fitxer de registre. La distinció entre bytes (`-c`) i caràcters (`-m`) és rellevant en fitxers que usen codificació UTF-8, on alguns caràcters ocupen més d'un byte.

```
wc fitxer.txt              # Línies, paraules i caràcters
wc -l fitxer.txt           # Només línies
wc -w fitxer.txt           # Només paraules
wc -c fitxer.txt           # Només bytes
ls /etc | wc -l            # Compta els fitxers d'un directori
```

`echo` / `printf` --- Mostrar text

`echo` és l'ordre bàsica per mostrar text a la sortida estàndard. S'usa àmpliament en scripts per mostrar missatges informatius, per generar contingut de fitxers amb redirecció o per mostrar el valor de variables. La seva limitació és que el format de sortida és bàsic. `printf` és una alternativa més potent inspirada en la funció `printf` del llenguatge C, que permet especificar el format de sortida amb especificadors de format (`%s` per a cadenes, `%d` per a enters, `%f` per a decimals, `\n` per a salt de línia, etc.). `printf` no afegeix un salt de línia automàticament al final, a diferència d'`echo`, de manera que cal afegir-hi explícitament `\n`.

```
echo "Hola, món"                          # Mostra text
echo $HOME                               # Mostra el valor d'una variable
echo -n "Sense salt de línia"            # Sense salt de línia al final
printf "Nom: %s, Edat: %d\n" "Joan" 25   # Format com printf de C
```

`man` --- Manual d'ordres

`man` (de l'anglès *manual*) mostra les pàgines del manual del sistema per a qualsevol ordre, fitxer de configuració, funció de sistema o concepte. Les pàgines del manual estan organitzades en seccions numerades: 1 (ordres d'usuari), 5 (formats de fitxer), 8 (ordres d'administrador del sistema), entre d'altres. Quan el nom és ambigu i existeix en diverses seccions (com `passwd`, que existeix com a ordre i com a fitxer de configuració), cal especificar la secció. Dins del manual, es pot cercar amb `/`, navegar amb les tecles de cursor i sortir amb `q`. L'opció `-k` cerca en els resums de totes les pàgines del manual, actuant com un cercador d'ordres quan no sabem el nom exacte.

```
man ls              # Manual de l'ordre ls
man -k paraula      # Cerca en tots els manuals
man 5 passwd        # Secció 5 (formats de fitxer) del manual de passwd
```

Redirecció i canonades (pipes)

La redirecció i les canonades (pipes) són mecanismes fonamentals de la shell que permeten encadenar ordres i gestionar els fluxos de dades. En Linux, cada procés té tres fluxos estàndard: entrada estàndard (`stdin`, descriptor 0), sortida estàndard (`stdout`, descriptor 1) i sortida d'error estàndard (`stderr`, descriptor 2). La redirecció (>, >>, <, 2>) connecta aquests fluxos a fitxers. La canonada (`|`) connecta la sortida estàndard d'una ordre amb l'entrada estàndard de la següent, que permet construir *pipelines* on la sortida d'una ordre és processada per la següent sense necessitat de fitxers intermedis. Aquesta composabilitat és una de les característiques més poderoses de Linux.

```
# Redirecció de sortida
ls > llista.txt             # Escriu la sortida a un fitxer
 ↪  (sobreescriu)
ls >> llista.txt            # Afegeix al final del fitxer

# Redirecció d'errors
ordre 2> errors.txt       # Redirigeix els errors a un fitxer
ordre > sortida.txt 2>&1  # Redirigeix sortida i errors al mateix
 ↪  fitxer

# Canonades (pipe)
ls -la | grep ".txt"        # Filtra la sortida de ls
cat /etc/passwd | grep bash | cut -d: -f1   # Encadena diverses ordres
ps aux | grep nginx         # Cerca processos nginx
```

Més informació de redireccions.

`alias` --- Dreceres d'ordres

`alias` permet crear dreceres personalitzades per a ordres llargues o complexes que s'usen freqüentment. Quan s'executa un àlies, la shell el substitueix per l'ordre real que representa. Els àlies definits a la línia d'ordres existeixen únicament durant la sessió actual; per fer-los permanents cal afegir-los al fitxer d'inicialització de la shell (`~/.bashrc` per a bash). Moltes distribucions ja inclouen àlies predefinits (com `ll='ls -alF'` o `la='ls -A'`). `unalias` elimina un àlies i `alias` sense arguments mostra tots els àlies actius.

```
alias ll='ls -lh'                    # Crea un àlies
alias update='sudo apt update && sudo apt upgrade'
alias ..='cd ..'
unalias ll                           # Elimina un àlies
alias                                # Llista tots els àlies actius
```

Per fer-los permanents, afegir-los al fitxer `~/.bashrc`.

`history` --- Historial d'ordres

La shell bash emmagatzema l'historial d'ordres executades al fitxer `~/.bash_history`. L'ordre `history` mostra aquest historial numerat, i diverses dreceres permeten reutilitzar ordres anteriors sense tornar-les a escriure. `!!` és un àlies per a l'última ordre executada, molt útil quan hem oblidat posar `sudo` davant d'una ordre que ho requeria. `!n` executa l'ordre número n de l'historial. La drecera Ctrl+R permet cercar a l'historial de forma interactiva mentre s'escriu. El nombre d'ordres guardades es controla amb la variable d'entorn `HISTSIZE`.

```
history                      # Mostra l'historial complet
history 20                   # Mostra les últimes 20 ordres
!!                           # Repeteix l'última ordre
!50                          # Executa l'ordre número 50 de l'historial
!ssh                         # Executa l'última ordre que comença per
 ↪  "ssh"
history | grep apt           # Cerca a l'historial
```

13. Automatització de tasques

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

L'automatització de tasques repetitives és un dels principals avantatges de l'administració de sistemes Linux. `cron` és el planificador de tasques del sistema que permet executar scripts o ordres automàticament a intervals regulars, sense necessitat d'intervenció manual. És essencial per a tasques com còpies de seguretat automatitzades, actualitzacions periòdiques, enviament d'informes, neteja de fitxers temporals o qualsevol altra tasca que calgui fer regularment.

`cron` / `crontab` --- Programació de tasques

`cron` és el dimoni (servei en segon pla) que executa les tasques planificades. `crontab` és l'eina per editar i gestionar les taules de tasques. Cada usuari pot tenir la seva pròpia taula de tasques cron, i root pot gestionar la taula de qualsevol usuari. A més, existeixen directoris del sistema (`/etc/cron.daily/`, `/etc/cron.weekly/`, `/etc/cron.monthly/`) on es poden deixar scripts que s'executaran automàticament amb la freqüència que indica el nom del directori. Quan es programa una tasca cron, cal tenir en compte que l'entorn d'execució és molt limitat (poc PATH, sense variables d'entorn de l'usuari), de manera que en els scripts cal usar rutes absolutes per a les ordres.

```
crontab -e              # Edita les tasques de l'usuari actual
crontab -l              # Llista les tasques programades
crontab -r              # Elimina totes les tasques
sudo crontab -e         # Tasques de root
```

Format del crontab:

```
# min  hora  dia-mes  mes  dia-setmana  ordre
  *    *     *        *    *            ordre     # Cada minut
  0    2     *        *    *          backup.sh   # Cada dia a les 2:00
  0    0     1        *    *           informe.sh  # L'1 de cada mes a
 ↪  mitjanit
  30   8     *        *    1-5         tasca.sh    # Dilluns-divendres
 ↪  a les 8:30
  0    */4   *        *    *           tasca.sh    # Cada 4 hores
```

Caràcters especials: - `*` → Qualsevol valor - `*/n` → Cada n unitats - `n,m` → Els valors n i m - `n-m` → De n a m

Més informació a Planificació de tasques: at i cron

14. Bash bàsic (scripts)

Mòdul relacionat: **0222 Sistemes operatius monolloc** · **0224 Sistemes operatius en xarxa**

Un script de shell és un fitxer de text que conté una seqüència d'ordres que la shell executa en ordre. Permeten automatitzar tasques repetitives, encadenar operacions complexes i prendre decisions basades en condicions. Bash (*Bourne Again Shell*) és la shell per defecte de la majoria de distribucions Linux i proporciona un llenguatge de scripts complet amb variables, condicionals, bucles, funcions i molt més. Els scripts de bash que s'han d'executar han de tenir el permís d'execució (`chmod +x script.sh`).

Estructura bàsica d'un script

La primera línia del script, anomenada *shebang* o *hashbang*, indica al sistema operatiu quin intèrpret ha d'usar per executar el fitxer. La forma `#!/bin/bash` especifica explícitament bash. Sense el shebang, el sistema intentaria executar el fitxer com a script de la shell actual, cosa que pot donar resultats inesperats si la shell de l'usuari no és bash. A continuació del shebang es recomana afegir comentaris explicatius (línies que comencen per #) i inicialitzar les variables que s'usaran al llarg del script.

```
#!/bin/bash
# Descripció del script

# Variables
NOM="alumne"
EDAT=18

# Mostrar valors
echo "Nom: $NOM"
echo "Edat: $EDAT"
```

La primera línia `#!/bin/bash` s'anomena **shebang** i indica l'intèrpret a usar.

Variables

Les variables de bash emmagatzemen valors de text (bash no distingeix entre enters i cadenes de caràcters de forma nativa). L'assignació no pot tenir espais al voltant del signe `=`: `VAR=valor` és correcte, però `VAR = valor` donaria un error. Per accedir al valor d'una variable s'usa el prefix `$`. Les claus `${}` s'usen per delimitar el nom de la variable quan va seguit directament d'altres caràcters. A més de les variables definides per l'usuari, bash proporciona diverses variables especials automàtiques molt útils en els scripts: `$0` conté el nom del script, `$1`-`$9` els arguments passats al script, `$?` el codi de retorn de l'última ordre (0 significa èxit, qualsevol altre valor indica error) i `$$` el PID del procés actual.

```
VARIABLE="valor"           # Assignació (sense espais al voltant de =)
echo $VARIABLE              # Accés al valor
echo "${VARIABLE}_sufix"    # Amb claus per evitar ambigüitat

# Variables especials
echo $0                     # Nom del script
echo $1, $2                 # Primers arguments
echo $@                     # Tots els arguments
echo $#                     # Nombre d'arguments
echo $?                   # Codi de retorn de l'última ordre (0 = èxit)
echo $$                     # PID del procés actual
```

Condicionals

Les estructures condicionals permeten executar blocs de codi de forma selectiva basant-se en condicions. En bash, les condicions s'avaluen entre claudàtors `[ ]` (o `[[ ]]` en bash modern, que és més robust). Cal deixar espais entre els claudàtors i l'expressió. Per a la comparació de nombres s'usen operadors específics (`-eq`, `-ne`, `-gt`, etc.) en lloc dels símbols matemàtics (que s'interpreten com a redirecció). Per a la comparació de cadenes s'usa `=` i `!=`. Les proves de fitxers (`-f`, `-d`, `-r`, etc.) comproven si un fitxer existeix i quins permisos té, cosa molt útil en scripts d'administració.

```
# if / elif / else
if [ $EDAT -ge 18 ]; then
    echo "Major d'edat"
elif [ $EDAT -ge 16 ]; then
    echo "Quasi major"
else
    echo "Menor d'edat"
fi

# Comparació de nombres
# -eq  igual
# -ne  diferent
# -gt  major que
# -ge  major o igual
# -lt  menor que
# -le  menor o igual

# Comparació de cadenes
if [ "$NOM" = "alumne" ]; then
    echo "Ets l'alumne"
fi

# Comprovació de fitxers
if [ -f /etc/passwd ]; then echo "Existeix el fitxer"; fi
if [ -d /home ]; then echo "Existeix el directori"; fi
if [ -r fitxer ]; then echo "El fitxer és llegible"; fi
```

Bucles

Els bucles permeten repetir un bloc de codi múltiples vegades. El bucle `for` és ideal quan es coneix el nombre d'iteracions o quan es vol iterar sobre una llista de valors. El bucle `while` s'usa quan cal repetir fins que es compleixi una condició. La notació `{1..10}` genera una seqüència de números de l'1 al 10. Per a operacions aritmètiques en bash s'usa la sintaxi `$((expressió))`. Els bucles `for` sobre fitxers (amb comodins com `*.txt`) són molt útils per processar grups de fitxers en scripts d'administració.

```
# Bucle for
for i in 1 2 3 4 5; do
    echo "Iteració $i"
done
```

```
# Bucle for amb rang
for i in {1..10}; do
    echo "Número $i"
done

# Bucle for sobre fitxers
for fitxer in *.txt; do
    echo "Processant $fitxer"
done

# Bucle while
COMPTADOR=0
while [ $COMPTADOR -lt 5 ]; do
    echo "Comptador: $COMPTADOR"
    COMPTADOR=$((COMPTADOR + 1))
done
```

Exemple complet: script de còpia de seguretat

Aquest script integra la majoria dels conceptes vistos: variables, substitució d'ordres (`$(ordre)`), condicionals per comprovar directoris i el resultat de les operacions, i l'ordre `tar` per crear la còpia. La substitució d'ordres `$(date +%Y%m%d)` executa l'ordre `date` i substitueix el resultat al lloc on s'usa, que permet generar noms de fitxer dinàmics amb la data actual. La comprovació de $? al final determina si `tar` ha tingut èxit (codi 0) o ha fallat (qualsevol altre codi).

```
#!/bin/bash
# Script de còpia de seguretat

DATA=$(date +%Y%m%d)
ORIGEN="/home/alumne"
DESTÍ="/backup"
FITXER="backup_$DATA.tar.gz"

echo "Iniciant còpia de seguretat..."

# Comprovar si existeix el directori de destí
if [ ! -d "$DESTÍ" ]; then
    mkdir -p "$DESTÍ"
    echo "Directori $DESTÍ creat"
fi

# Crear la còpia
tar -czvf "$DESTÍ/$FITXER" "$ORIGEN"

# Comprovar si ha anat bé
if [ $? -eq 0 ]; then
    echo "Còpia de seguretat creada: $DESTÍ/$FITXER"
else
```

```
    echo "Error en crear la còpia de seguretat!"
    exit 1
fi
```

15. Referència ràpida

Acció	ordre
Llistar fitxers	`ls -lh`
Copiar fitxers	`cp origen destí`
Mou/reanomena	`mv origen destí`
Cerca fitxers	`find / -name "nom"`
Canviar permisos	`chmod 755 fitxer`
Canviar propietari	`chown usuari:grup fitxer`
Crear usuari	`useradd -m usuari`
Canviar contrasenya	`passwd usuari`
Llistar processos	`ps aux`
Matar un procés	`kill -9 PID`
Gestionar serveis	`systemctl start/stop/status`
Espai lliure	`df -h`
Espai ocupat	`du -sh directori/`
Comprimir	`tar -czvf fitxer.tar.gz dir/`
Descomprimir	`tar -xzvf fitxer.tar.gz`
Comprovar xarxa	`ping -c 4 8.8.8.8`
Ports oberts	`ss -tuln`
Connexió remota	`ssh usuari@servidor`
Copia remota	`scp fitxer usuari@host:/destí/`
Instal·lar paquet	`apt install paquet`
Cercar text	`grep "paraula" fitxer`
Ajuda d'ordre	`man ordre`

Versions d'aquest document

- HTML - basics.html
- PDF - basics.pdf
- ODT - basics.odt
- MD - basics.md