ASR

1. FHS: ~~File System H~~

filesystem Hierarchy Standart:

/: root, todo cuelga de aquí.

/boot/: ficheros relativos al arranque (tmb kernel).

/bin/: binaries, ejecutables esenciales (ls, cat, bash).

/sbin/: superuser binaries (fdisk, ifconfig...)

/lib/: librerías esenciales para los binaries esenciales.

/usr/: Unix System Resources, incluye el resto de las aplicaciones

/usr/bin/: app usuario.

/usr/sbin/: app superusuario.

/usr/lib/: librerías binaries.

/usr/share/: datos independientes de la arquitectura, fund. manuales

/usr/include/: ficheros .h estándar

/usr/src/: código fuente kernel y apps.

/usr/local/: apps instaladas por usuario, replica estructura de /usr/.

/opt/: apps que requieren de un subdirectorio independiente (comerciales).

/etc/: ficheros y scripts de configuración. (Gráfico /etc/X11/, usuarios /etc/passwd)

/var/: ficheros variables (logs, bases de datos...)

/var/log: logs del systema o apps

/var/spool/: ficheros temporales de impresión, e-mail...

/srv/: datos de servicios proporcionados por el sistema (ftp, http...).

/tmp/: ficheros temporales públicos que se borran durante el reinicio.

/home/: dir de usuarios
/root/: dir de superusuario

/dev/: dir de pseudoficheros de acceso a periféricos
/mnt/: punto de montaje para otros sistemas temporales
/proc/: info del sistema
/sys/: similar a proc (brillo, carga...)

2. PARTICIONAMIENTO

Es usual dividir el sistema en particiones dependiendo del número de usuarios:

- Un usuario:
  - swap: área de intercambio que sustituye a la RAM en casos extremos. Es usual reservar 2x MCRAM) de esta partición
  - /home/: cuentas usuario
  - /: resto del disco
- Multiusuario: se separan a demás
  - /usr/: que puede ponerse en read-only tras la instalación del sistema para dificultar la intr. de virus.
  - /var/ y /tmp/: a parte, para que ocupen un tamaño fijo y no coman el disco.

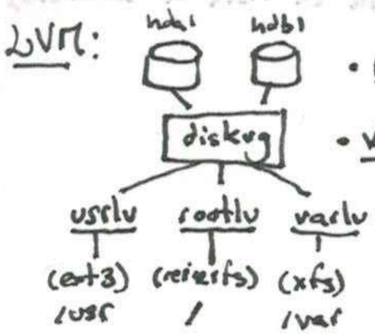
Sistemas de ficheros usados commmente:

- ext4: fourth EXTended filesystem
  - transaccional
  - reduce fragmentación
  - discos $\leq$ 1 EiB (2^{60} B) y ficheros $\leq$ 16 TiB
  - configurable con "tune2fs"
  - las versiones anteriores ext3 y ext2 siguen disponibles
- fat32, fat16: usado en MS-DOS y Windows 95/98/Me
- NTFS, ReFS: usado en Windows NT, no siempre disponible.

El gestor de arranque por defecto en Linux es el GRUB, el GRand Unified Bootloader. Este es cargado por la BIOS y pasa el control al programa de arranque eleccionado (localizado en /boot/).

Se suele instalar en el MBR o, preferiblemente, en el GPT.

LVM:



- PV: discos duros, particiones de discos...
- VG: agrupación de LVs
- LV: particiones lógicas sobre las que se montan los sistemas de ficheros

3. ARRANQUE DEL SISTEMA

1º BIOS: el procesador se sitúa en la posición de memoria de la BIOS, se ejecuta un programa que detecta los discos, carga el MBR o GPT y ejecuta el GRUB.

2º Bootloader: muestra un menú con la lista de S.O. disponibles y carga el kernel de Linux desde el disco y lo ejecuta.

3º Kernel: comienza a buscar y montar la partición que contiene el sistema de ficheros raíz, cargando el primer programa (init):

4º initramfs: en una primera fase se carga la memoria RAM con un fichero de imagen de un disco virtual que contiene una "partición raíz" y un programa init. Contiene lo mínimo para cargar el "verdadero" fs en el disco.

5º init: initramfs cede el control a init, y se realiza el proceso de arranque estándar. Esto incluye módulos de controladores sin los que el sistema no puede arrancar, inicio de particiones cifradas...

En Debian se usa systemd:

- Este distingue entre varios targets como:
  - rescue.target: arranca lo mínimo para intentar reparar un start dañado.
  - emergency.target: abre un único shell.
  - multi-user.target: multiusuario no gráfico
  - graphical.target
- Carga systemctl y service. systemctl muestra la lista de servicios activos, pero tiene los argumentos:
  - status
  - start
  - stop
  - restart
  - enable
  - disable

4. VERIFICACIÓN DE INSTALACIÓN DEL SISTEMA

Podemos ver la información del hardware:

- /proc/cpuinfo
- /proc/interrupts
- /proc/ioports
- /proc/dma
- lspci: lista disp PCI
- lsusb: lista disp USB

Podemos ver la información de las particiones:

- fdisk -l: muestra todas las particiones (solo su)
- df: muestra las particiones montadas (con -h se ven más legibles)

Podemos gestionar los módulos del kernel:

- lsmod: muestra los módulos cargados
- modprobe: instala un nuevo módulo
- rmmod: elimina un módulo

Tipos de discos duros gestionados:

1. Serial ATA (sata): más comunes (/sda, /sdb/...)
2. SCSI: usados en servidores de altas prestaciones, conectados por bus
3. IDE o Parallel ATA: poco usados actualmente (/hda, /hdb/...)

5. INSTALACIÓN DE PAQUETES

- **dpkg**: permite instalar paquetes previamente descargados (wget). Estos paquetes han de ser de tipo DEB:
  - Contienen bins de la app, metadatos y scripts.
  - Se nombran paquete_versión-build_arquitectura.deb.

    (Ej: ethereal_0.10.11-1_i386.deb)

dpkg cuenta con las siguientes opciones:

-i: instalar pkg	-s: estado pkg
-r: eliminar pkg (mantiene .conf)	-L: lista ficheros pkg
-P: purgar completamente pkg	-S: busca pkg de x fichero
-l: lista pkg instalados	dpkg-reconfigure

- **apt**: Advanced Packagin Tools, busca en las fuentes especificadas en "/etc/apt/sources.list" los paquetes a instalar, su formato es:

{deb (bin), deb-src (.h)} {uri} {stable, testing, unstable} {main, contrib, non-free}

Para configurar más apt ver /etc/apt/apt.conf y apt.conf.d/...

Opciones:

apt-get:
- update: actualiza lista pkg
- upgrade: actualiza pkg
- install: con -f corrige errores en la instalación
- remove, purge
- autoremove, clean: elimina pkg innecesarios
- source: descarga fichero fuente (usar --compile para .deb)
- build-dep: " dependencias de compilación

apt-cache:
- search: busca pkg para instalar
- show: muestra info pkg
- depends: lista dependencias de pkg
- policy: muestra fuentes y prioridades

- **A través del código fuente:**

1° Descargar el tar (.tar, .tar.gz, .tar.bz2)

2° Desempaquetado: con tar, de opciones:

-c: crear	-f: especificar tarball
-t: listar contenido	-v: verbose
-x: extraer	-z: para .gz
	-j: para .bz2

3° Leer fichero INSTALL, README o similar

4° Configurar: suele haber un archivo "configure" para esto. Ejecutar con ./configure (--prefix=dir para instalar en "dir" en vez de en /usr/local)

5° Compilar: si se hizo .configure, debe de haber un makefile. Hacer make o make all.

6° Instalar: si se compiló exitosamente hay que hacer un make install.

Un ejecutable puede tener las librerías linkeadas dinámicamente. Usando ldd podemos ver las librerías necesitadas:

- ldd /bin/ls

  librería => localización si la encontró

  lib => Not found

- Si no se encuentra puede que:
  - La librería se encuentra en la loc no estándar ($LD_LIBRARY_PATH)
  - La librería está desactualizada o no es compatible (usar ln o actualizar).

6. SHELL

Variables de entorno predefinidas:

- HOME: dir base del usuario
- SHELL: shell por defecto
- USER: nombre del usuario
- PWD: dir actual
- PATH: path de los ejecutables
- LD_LIBRARY_PATH: path de las lib dinámicas
- PS1 / PS2: prompts primario y secundario

Expansión de comandos:

- ls ab* (cualquier fichero que empiece por "ab")
- ls `cat f.txt` (mira el contenido del comando)
- ls a{b,c} (mira "ab" y "ac")
- $(( )) o $[ ] evalua aritmética

  $[1+3 / 8]

  - let num = 1+3/8
- ignorar caracteres especiales: '' "" no ignora { $ \

Redirección entrada/salida:

- Estándar in → 0
- Estandar out → 1 } &
- Estandar err → 2 }

x < f: toma f de input

x > f: f de out

x >> f: append out

x > dev/tty1: al terminal

Comandos útiles con pipe:

- tee: muestra pantalla y guarda fichero
- xargs
- exec: puede usarse para redireccionar todos los comandos:

  exec 2> log: envia los errores a log

Cuidado con el orden de evaluación:

1° Redirección E/S

2° Sustit variables

3° Sustit nombres de ficheros

- Usar eval para sustituir 2 veces

SHELL-SCRIPT:

#!/bin/bash

- Parámetros:

  $0 → nombre script

  ${n} → entrada

  ${a:-b} → si no existe a, se usa b

  $! → PID último proceso

  $# → número de params

  $*, $@ → todos las params

  $? → éxito ultimo comando = 0

  shift n: desplaza el prompt n posiciones a la derecha izq

arrays:

```
num=( 1 2 )
echo ${num[0]}
for i in "${num[@]}"; do ...
```

- I/O:

  echo: out

  read: in

```
#!/bin/bash
IFS=: # /etc/passwd usa ":" para separar campos
cat /etc/passwd |
while read name pass uid gid fullname ignore
do
  ...
done
```

- Tests:

x && y → "y" solo se ejecuta si x acierta

x || y → "y" solo se ejecuta si x falla

if...then...else:

```
if x
then
  ...
elif y
then
  ...
fi
```

test: test ... o [[...]] o [...]

- strings:
  - s : s no es nulo
  - -z s : longitud s es cero
  - -n s : longitud s no es cero
  - s1 = s2 :
  - s1 != s2 :

- enteros:

e1 -eq e2	e1 -ge e2
e1 -ne e2	e1 -lt e2
e1 -gt e2	e1 -le e2

- ficheros:
  - -e f : f existe
  - -r f : f existe y es legible
  - -w f
  - -x f
  - -f f : f existe y es regular
  - -d f : f existe y es dir

- No hay && || !
- Los () necesitan "/"

case:

```
case valor in
  patron1)
    ...
    ...;;
  patron2)
    ...
    ...;;
esac
```

funciones:

```
funcion () {
  ...
}

funcion a b
```

wait: espera a que un proc en bg acabe

trap: reescribe comportamiento señales

```
trap "foo" SIGINT SIGQUIT
```

REG-EXP: (egrep '/.../', sed -r 's/.../reemplazo/g' file)

. → any

[] → cualquier char entre los corchetes, todo meta se anula

[^] → cualquier char que no esté entre los corchetes

^ → principio de linea

$ → final de linea

* → 0 o más

+ → 1 o más

? → 0 o 1

( ) → agrupa ER y son accesibles con \n

| → OR

{n} → n ocurrencias

{n, m} → entre n y m ocurrencias

\ → escapa

\\ o \\\ → \

PROCESAMIENTO TXTS:

- head/tail -n x: muestra x primeras/últimas líneas
- tac: invierte orden filas
- rev: invierte orden char en cada fila
- wc: cuenta palabras, líneas, bytes ... -l (líneas), -w (palabras)
- nl: añade números de línea
- sort: ordena líneas (-n para numéricamente)
- tr: cambia chars (tr 'a-z' 'A-Z')
- uniq: descarta líneas repes sucesivas
- cut: selecciona columnas o campos fichero
- paste: une líneas dos ficheros
- join: une ficheros ordenados por un campo
- split: divide en subficheros
- expand: cambia TAB por espacios
- fmt: formatea fichero para que las líneas sean del mismo tamaño.
- od: muestra en formato octal, hexadecimal... el fichero.
- awk:

```
#!/usr/bin/awk -f
BEGIN { print "Hola"; count=1 }
/patron/ { print $0 ; count=count+1 }
END {print count}
```

  - $1, $2 ... para campos
  - FS: file separator
  - NR: n línea
  - FNR: n línea fichero actual
  - NF: n palabra


```
#!/usr/bin/awk -f
NR == FNR {
  a[NR] = $0;
}
NR != FNR {
  if ( a[FNR] != $0 ) {
    print "Las lineas " FNR " difieren";
  }
}
```

- python:

```
#!/usr/bin/env python3
import argparse, re, os
parser = argparse.ArgumentParser( description='process files')
parser.add_argument( '-i', meta_var='input', help='no', required=True)
args = parser.parse_args()
print( args.i )
f = open ( 'file', 'r' )
f.write( 'pena' )
l = f.readLines() # ['hola mundo', 'que tal']
areas = findAreas( l ) #['hola', 'mundo', 'que', 'tal']
re.match ( r"___", l[0])
os.mkdir( "path" )
```

7. ADMINISTRACIÓN DE PROCESOS

Ver los procesos en ejecución:

- ps: lista los procesos con su PID, TTY, TIME y CMD. Tiene las opciones:
  - -e: muestra todo
  - -u: muestra los de un usuario
  - -o: permite definir el formato (user, pid, stat, ...)
  - --sort: ordena por campo

El estado puede ser R (running), S (sleep), D (uninterrumpible sleep), T (stopped) o Z (zombie).

- pstree
- top: muestra procesos en "tiempo real"
- strace: muestra las llamadas a sistemas realizadas por un proceso
- jobs: permite ver los procesos en bg

Señalización de procesos:

- kill: envía señales a un proceso, "kill -l" para verlas

  kill -9 ≡ kill -SIGKILL ≡ kill -KILL

  Señales:
  - SIGTERM: mata al proceso con cuidado
  - SIGKILL: mata inmediatamente al proceso (no ignorable)
  - SIGSTOP: detiene al proceso, si ctrl+z es SIGTSTP
  - SIGCONT: continúa proceso
  - SIGINT: ctrl+c, mata el proceso
  - SIGHUP: cuando terminal cierra, mata procesos y reinicia daemons.

- nohup: evita que se le envíe SIGHUP al proceso
- pgrep: busca el PID del proceso buscado
- pkill: envía señales a procesos por su nombre (pkill -KILL firefox)
- killall: similar a pkill pero sin patrones

Manejo de la prioridad: (-20 (más alta) ~ 19 (más baja))

- nice -n 10 top: lanza un comando con una prioridad
- renice 10 -p 5387: cambia la prioridad de un proceso

Control de los recursos de un proceso (con "ulimit")

Rendimiento del sistema:

- uptime: muestra la hora actual, el tiempo que lleva encendido, los usuarios conectados y la carga media del sistema para los últimos 1, 5 y 15 minutos.
- w: como uptime pero con info detallada de los usuarios
- free: muestra la memoria libre y usada en el sistema.

El directorio /proc:

- /cpuinfo: info de la cpu
- meminfo
- interrupts
- ioports
- XXXX: con el PID de cada proceso, tiene:
  - fd: descriptores de ficheros de enlace simbólico
  - maps: mapa de memoria

8. FICHEROS

Tipos de ficheros:

- **Regulares**: se crean con vi, cp, touch... y borran con rm.
- **Directorios**: se crean con mkdir y borran con rm -r.
- **Enlaces simbólicos**: creados con ln -s y se borran con rm.

Se usa "file ..." para determinar el tipo.

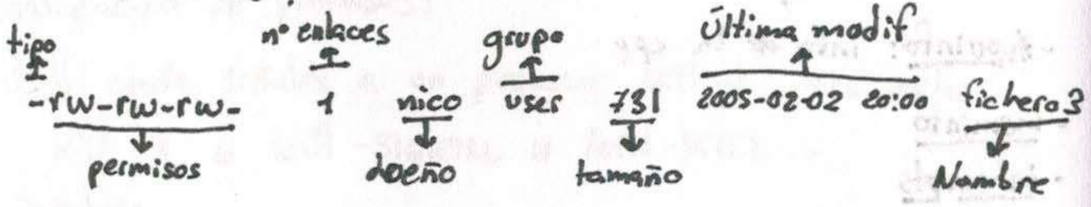
Creación de enlaces: Hay dos tipos:

- **Duros**: crean otra entrada en el dir que apunta al mismo nodo-i que el original. No se puede enlazar a un fichero de otra partición. Hay que borrar enlaces para borrar fichero.
- **Blandos**: si el fichero se borra el enlace no apunta a nada. No hay problema con las particiones.

Para crear:

- **ln**: ln [-s] [opciones] destino [enlace]. Crea enlaces duros por defecto, pero con -s se hacen blandos.

Atributos de "ls":



El tipo puede ser regular (-), directorio (d), enlace (l)...

Linux guarda 3 tipos de fecha:

- **mtime**: última modif. ls -l.
- **atime**: último acceso. ls -l --time=atime.
- **ctime**: último cambio de estado. ls -l --time=ctime.

Permisos de ficheros:

r) Lectura: permite leer y abrir. En dirs permite listar contenido.
w) Escritura: permite modificar. En dirs permite crear, borrar... contenido.
x) Ejecutar: permite ejecutar. En dirs permite entrar.

- **chmod**: para cambiarlos, formatos:
  - chmod u+x ... → añade ejecución a usuarios
  - chmod 400... → añade lectura solo a usuarios

+**setuid** y **setgid**: cambian usuario y grupo. Hay un UID y GID efectivos y reales. Si un programa pide sudo podría tener RUID de nico y EUID de root. (chmod {g+s / u+s})

+**sticy bit**: solo en directorios, permite solo borrar archivos al dueño del archivo, del dir y al root. (chmod +t dir)

- **chown** y **chgrp**

Búsqueda de ficheros:

- **find**: muestra los ficheros desde el dir actual de forma recursiva.
  - **-name "x"**: busca ficheros que coincidan con el patrón x.
  - **-iname "x"**: no distingue entre mayúsculas y minúsculas.
  - **-regex x**: igual pero con regexes.
  - **-user, group "x"**: igual pero buscando usuario o grupo.
  - **-size x**: tamaño { igual / mayor (+) / menor (-) } a n { bytes (c) / KB (k) / MB (M) / GB (G) }
  - **-perm**: permisos
    - -perm 770 → exactos
    - -perm -770 → por lo menos
  - **-atime, mtime, ctime [+,-]n**: días.

- print : imprime el nombre de los ficheros por pantalla.
- ls : " en formato largo.
- exec comando {} \; : ejecuta comando
- ok comando {} \; : igual pero pregunta antes.
- Aritméticas:

expr1 expr2 → AND	!expr1 → NOT
expr1 -o expr2 → OR	\(expr1 \)

- wich: muestra la loc. de comandos.
- whereis: muestra la loc. de la fuente, binario y man.
- locate: find rápido.

9. PARTICIONES

- fdisk: crea o lista particiones. (fdisk /dev/hda)
- parted: programa de GNU de más alto nivel.
- blkid: muestra el UUID, el cual es más fiable que "/dev/sd*".
- mkfs.tipo: crea fs del tipo "tipo" (ext3, ext4, xfs, vfat...)
- mkswap: crea un swap
- swapon: activa un swap
- fsck.tipo: testea y repara fs
- tune2fs: ajusta parámetros de ext2, 3 y 4.
- dump2fs: muestra info de ext2, ext3 y ext4.
- e2label: fija la etiqueta de ext2/3/4.
- mount: asocia un dir a un fs. (mount /dev/sdb1 /home)
- umount: (umount /home).
- du: muestra el espacio ocupado por ficheros y subdirectorios de dir.
  - -h: humano
  - -s: solo ocupación total
- df: muestra el espacio de disco usado por los fs
- /etc/fstab: dicta como se montan los fs listados
- /etc/mtab: lista los fs montados

- Sistemas con LVM:

-Son más flexibles, permite total libertad de control para los volúmenes que gestiona. Hasta se pueden reescalar en caliente.

-Se pueden añadir nuevos discos al LVM y extender las "particiones" para que los ocupen.

-Estructura:

1. Por el lado físico teremos a los volúmenes físicos (PV). Siendo estos los discos y sus particiones.
2. A partir de los PV se construyen agrupaciones lógicas llamadas Grupos de Volúmenes (VG). Han de etiquetarse.
3. Los VG se dividen en LVs etiquetados.
   - Estos pueden cifrarse.
4. Para usarlos necesitan ser formateados.
4. Luego se asigna el LV a un dir.

Comandos ver info:

- pvdisplay: muestra info PV (/dev/sda)
- vgdisplay: muestra info VG (groupVol)
- lvdisplay: muestra info LV (/dev/groupVol/homelv)

Manejo PV y VG:

- Crear PV: pvcreate /dev/sda1
- Crear VG: vgcreate x /dev/sda1 /dev/sda2
- Activar VG: vgchange -a y x
- Borrar VG: vgchange -a n x
  vgremove x
- Añadir PV a VG: vgextend x /dev/sda3
- Quitar PV a VG: vgreduce x /dev/sda2

Manejar LV:

- Crear LV: lvcreate -L4G -n homelv x
- Destruir LV: # umount /dev/x/homelv
  lvremove /dev/x/homelv
- Agrandar LV: lvextend -L+10G /dev/x/homelv

- Los LV se encuentran en:
  ~~/dev/x/~~ /dev/VG/LV
  /dev/mapper/VG-LV
- Si hemos agrandado el LV, hay que agrandar el fs:
  fsadm resize /dev/mapper/x-homelv

- Manejo de discos cifrados:

El fichero /etc/crypttab indica cómo descifrar discos.

- LUKS: cifrado estandar de Linux.
- Cifrado sin LUKS:

cryptsetup create sda7_crypt /dev/sda7 --key-file /dev/urandom
vim /etc/crypttab
> sda7_crypt /dev/sda7 /dev/urandom ..., ..., ..., swap
vim /etc/fstab
> /dev/sda7_crypt none swap rw 0 0

- Cifrado con LUKS:

cryptsetup luksAddKey /dev/sdb1

(cryptsetup luksDump /dev/sdb1) → para ver los slots
luksFormat → primera clave
luksUUID → obtener UUID

10. USUARIOS

En UNIX se recogen las siguientes características:

- nombre e identificador numérico UID
- contraseña
- grupos a los que pertenece con sus GID
- un directorio en home
- un login shell
- un conjunto de ficheros de inicio

Hay varios tipos:

- cuentas normales
- root
- cuentas para los servicios (nobody, lp, bin...).

La información sobre estos se guarda en los ficheros:

- /etc/passwd:

  name : password: UID : GID : data : /home/name : shell /bin/bash

  - Si en password ha hay una "x", entonces significa que está guardada en shadow

- /etc/shadow: contiene contraseñas encriptadas, acceso restringido.
- /etc/group: info sobre grupos.

  name : password : GID : users

- /etc/gshadow: contiene las contraseñas de los grupos y una copia de su lista de usuarios. root puede cambiar/establecer las contraseñas con "gpasswd".

Un usuario puede cambiar de grupo con "newgrp":

- Si es miembro, puede.
- Si no, pero hay contraseña, puede.
- Si no, y no hay contraseña, no puede.
- Si hay entrada en gshadow, esta tiene prioridad.

- /etc/skel: plantilla para ficheros de inicio nuevo usuario.

Creación manual de una cuenta:

1. Editar con "vipw" /etc/passwd y añadir una nueva línea.
2. Editar /etc/shadow con "vipw -s" usando una contraseña cifrada generada por "passwd".
3. Editar /etc/group.
4. Si es necesario, también /etc/shadow.
5. Crear el directorio del usuario y copiar el contenido de /etc/skel ahí.
6. Usar chown, chgrp y chmod para fijar al directorio los valores correctos.
7. Fijar la contraseña con passwd. (preferiblemente con -e)

Comandos:

- passwd: permite cambiar/establecer la contraseña de un usuario.

  passwd [options] [user]

  -e: obliga a cambiar la contraseña en el siguiente login.

- chage: permite cambiar la expiración de la contraseña.

  -l: muestra info de expiración.

- gpasswd: añade/borra usuarios a grupos por root.
- newgrp: cambio de grupo por el usuario.

- useradd, userdel, usermod: trabajan sobre passwd y shadow.
- groupadd, groupdel, groupmod: trabajan sobre group y gshadow.
- adduser, deluser: alto nivel.
- addgroup, delgroup: alto nivel.

- newusers: toma un fichero como /etc/passwd con las contraseñas sin encriptar y hace todo. A usuarios repes actualiza.

- chpasswd: igual pero solo actualiza password, formato user:pass.

PAM: Pluggable Autentification Method es una biblioteca de autentificación genérica para la validación de usuarios utilizando varios esquemas (ficheros locales, OTPW, DNI-e...). Para esto usa módulos de varios tipos:

- De autentificación (auth): para la identificación del usuario
- De cuentas (account): controlan las condiciones para permitir auth.
- De contraseña (password): condiciones y proced para cambio pass.
- De sesión (session): configuran y administran sesiones de usuarios. (montaje dirs, manejo logs...).

Cada servicio tiene su .conf.

Cuotas de disco: evitan que los usuarios monopolicen el disco, pero son molestas. Estas pueden tener límite:

- Duro: se deniega el intento de escribir datos superado este límite.
- Débil: se le da al usuario un "período de gracia" para liberar la ocupación. Si no se resuelve el problema, se bloquea la cuenta.

Las cuotas pueden establecerse a usuarios o a grupos, con los comandos:

- quotacheck: construye el índice y chequea integridad.
- quotaon/off: activa/desactiva quotas.
- quota: informa al usuario del uso de su cuota.
- edquota
- repquota: genera informes de uso.

En /etc/fstab se indican los fs que estarán sujetos a cuotas

II. REDES DE ÁREA LOCAL

Linux soporta múltiples protocolos (TCP/IP, TCP/IP v6, IPX/SPX, PPP...) y hardware para redes ethernet, wifi, ATM...

Diferentes NICs (Network Interface Cards) corresponden a diferentes dispositivos de comunicación.

- En los esquemas clásicos se denominan ethx, wlanx, pppx...
- En las nuevas versiones es más concreto (enp3s0)

Ficheros de configuración de red:

- /etc/network/interfaces: interfaces que usa el servicio "networking"
- /etc/resolv.conf: configuración DNS
- /etc/hosts: asocia nombres de hosts a direcciones IP
  - Así se puede resolver una IP sin acceder a un DNS
  - El nombre/dominio del host puede verse con "hostname" y "dnsdomainname".
  - Tmb en /etc/hostname.

Configuración de ~~dhcp~~: DHCP: Domain Host Configuration Protocol.

- Permite la configuración automática la red de los sistemas a través de un servidor. Esto es la conf. IP, DNS...
- /etc/dhcp/dhcpd.conf: configuración del servidor DHCP
- dhclient: se pide la configuración. (dhclient eth0). En el cliente en /etc/network/interfaces debe aparecer:

```
auto eth0
iface eth0 inet dhcp
```

- /proc/sys/net/ipv4/ip_forward: permite (1) o prohibe (0) el routing entre interfaces (0 por defecto)

Comandos de configuración de red: (no permanentes)

- ifconfig: configuración de la interfaz de red. También sirve para visualizar las interfaces activas y no activas (esto último con la opción -a). Su salida se divide en:

eth0 Direcciones hardware
Direccion IPv4
Direccion IPv6
Flags (UP, RUNNING, BROADCAST, MULTICAST...), MTU y métrica.
N paquetes enviados y errores en ello
N paquetes recividos y fallos en ello
colisiones y longitud de la cola de transmisión
N bytes enviados y recibidos

Se usa para subir interfaces así:

ifconfig eth0 193.144.84.77 netmask 255.255.255.0 broadcast/[illegible]
193.144.84.77 up

- ifup, ifdown: activa/desactiva interfaz.
- ip: muestra y modifica dispositivos y rutas. Más completo que ifconfig, route y nestat juntos.
- route: muestra rutas y su estado. Dice si está accesible (U), y si es host (H) o gateway (G) o nada ().

Para añadir ruta a 1.1.0.0 a través de 2.2.2.2:

route add -net 1.1.0.0 netmask 255.255.0.0 gw 2.2.2.2

- netstat: muestra estado red. Sin parámetros muestra los sockets abiertos, y con "-s" muestra estadísticas clasificadas por protocolo.

Otros comandos de red:

- ping: muestra disponibilidad y velocidad de una conexión
- traceroute: muestra la ruta que hace un paquete hasta su destino.
  - envía poco a poco paquetes con mayor tiempo de vida para ir recibiendo errores de los routers que atraviesa la solicitud.
- host/dig: permiten obtener la IP de un host y viceversa.
- arp: muestra y permite modificar la tabla cache de ARP.

12. AUTOMATIZACIÓN DE TAREAS

Tareas periódicas:

- at: permite indicar el momento en el que se va a ejecutar un trabajo.
  - Una vez ejecutado nos pone un prompt donde introducimos los comandos.
  - La salida estándar se envía como un mail al usuario.
  - Si el sistema está encendido a la hora indicada, se ejecuta.

atq: muestra los trabajos pendientes

- atrm: elimina un trabajo pendiente
- batch: ejecuta trabajos cuando la carga caiga de 1.5 en el sistema.

- cron y crontab: especifica trabajos periódicos en un fichero:

Si quiere periodicidad cambiar usar /etc/cron.hourly .daily .weekly .monthly

```
#shell usado
SHELL = /bin/bash
#usuario del mail
MAILTO = rossianniken
#comandos
minuto hora día mes día_semana comando
```

* → cualquier valor

0, 15, 30, 45 → cada 15 min

1-5 → de lunes a viernes

0-23/2 → cada 2 h entre las 0h y las 23h

13. COPIAS DE SEGURIDAD

Una buena estrategia para copias de seguridad debe tener las siguientes características:

- Fácil de usar o automáticas
- Eficiencia y rapidez
- Facilidad de restauración
- Capacidad de verificar las copias
- Tolerancia a fallos en los medios de almacenamiento
- Portabilidad

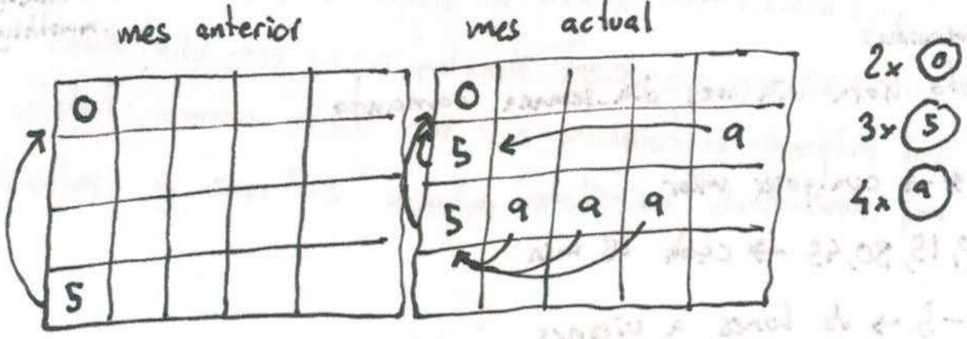
Suelen estar formados por:

- Medio de almacenamiento (cinta, disco duro, nube, DVD...)
- El programa de copia (puede ser basado en imagen o fichero a fichero)
- El planificador (decide cuándo copiar y cuánto)

Hay cuatro tipos de backup:

- Completo
- Parcial: solo se salva la información más importante
- Diferencial: se salvan las modif con respecto al último completo.
- Incremental: se salva el diff con respecto al último completo o inc.
  - Se usan niveles, el 0 es el completo y cada nivel depende del inferior.

Ejemplo: queremos recuperar cualquier versión diaria de la última semana y cualquier semanal del último mes.



Comandos usados:

- dump: hace un backup de imagen. (dump -0u -8 -f /dev/st0 /home)
- restore: restaura un dump. (restore -rf /dev/st0)
- tar: sirve para backups fichero a fichero.
- dd: copia y conversión. (dd if=/home of=/tmp/bk.img)