Administración de Sistemas e Redes

Grao en Enxeñaría Informática Grao
Escola Técnica Superior de Enxeñaría
Universidade de Santiago de Compostela

Tomás Fernández Pena
tf.pena@usc.es

27 de agosto de 2025

Índice general

Capítulo 1

Introducción a los sistemas Linux/Unix

1.1. Instalación de Linux Debian

1.1.1. Tipos de instalación

A la hora de instalar un sistema, tenemos que tener en cuenta el tipo de funciones que va a desempeñar.
Podemos distinguir:

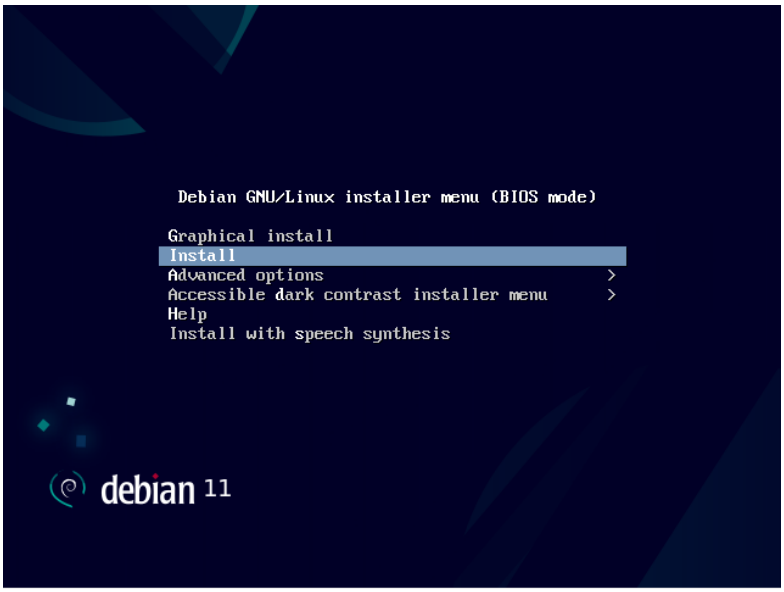
1. Sistema de escritorio: usado en tareas rutinarias (ofimática, acceso a Internet, etc.)
2. Estación de trabajo (*workstation*): sistema de alto rendimiento, generalmente orientado a una tarea específica
   - estación dedicada al cálculo (p.e. aplicaciones científicas)
   - estaciones gráficas (p.e. diseño 3D)
3. Servidores: ofrecen servicios a otras máquinas de la red
   - Servicios de disco: acceso a ficheros a través de FTP, servicio de disco transparente a través de NFS o Samba
   - Servicios de aplicaciones, por ejemplo, terminales, conexión remota (telnet, ssh), aplicaciones gráficas a través de X Window
   - Servicios de directorio, por ejemplo, LDAP (Protocolo Ligero de Acceso a Directorios), que almacena credenciales de usuarios, directorios, etc

- Servicios de base de datos
- Servicios de impresión
- Servicios de red, por ejemplo,
  - Servicios de configuración dinámica de máquinas, por ejemplo DHCP (*Dynamic Host Configuration Protocol*): permite configurar dinámicamente la red de los clientes
  - Servicios de acceso a Internet: NAT, proxy
  - Servicio de nombres (DNS)
  - Servicios de filtrado (firewall)
  - Correo electrónico
  - Servidor Web (p.e. Apache)

1.1.2. Instalación del sistema

Para detalles de instalación ver Guía de instalación de Debian

- Descargaremos la imagen de CD de instalación por red de la versión eleccionada (fichero `debian-`*versión*`-amd64-netinst.iso`)



- `Enter` para iniciar con opciones por defecto, `Advances options` para opciones de instalación avanzadas, `Help` para ayuda

Siguientes pasos en la instalación[1]

- Selección de idioma, localización y teclado
- Configuración de la red
  - Por defecto, intenta configurarla por DHCP
  - Si no lo consigue, pasa a configuración manual (indicar IP, máscara, pasarela y DNSs)
- Poner un nombre a la máquina e indicar el dominio (si alguno)
- Fijar el password del superusuario (root) y crear un usuario no privilegiado

Cuenta del superusuario

- El superusuario es un usuario especial que actúa como administrador del sistema
  - Tiene acceso a todos los ficheros y directorios del sistema
  - Tiene capacidad para crear nuevos usuarios o eliminar usuarios
  - Tiene capacidad de instalar y borrar software del sistema o aplicaciones
  - Puede detener cualquier proceso que se está ejecutando en el sistema
  - Tiene capacidad de detener y reiniciar el sistema
- El login del superusuario es `root` (aunque puede cambiarse)
- No es conveniente acceder al sistema directamente como root:
  - acceder como un usuario sin privilegios, y
  - obtener los permisos de root haciendo `su -` (necesitamos la contraseña de root)

[1] En cualquier momento de la instalación tenemos acceso a una consola pulsando `Alt-F2`; usar `Alt-F1` para volver a la instalación

Elección de contraseña

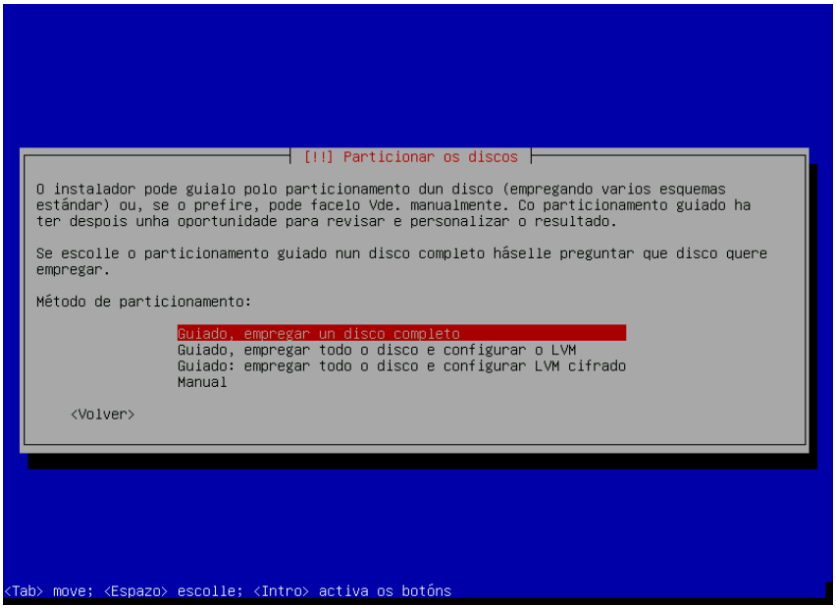
- Tener una contraseña de root adecuada es básico para la seguridad de un sistema
- Las contraseñas de usuario también deberían ser adecuadas
- Recomendaciones para elegir una contraseña:
  - No usar el nombre de usuario (login) ni variantes de este (p.e. login: pepe, passwd: pepe98), ni datos personales como nombre, apellidos, teléfono, DNI, etc.
  - No usar palabras contenidas en diccionarios, ni tampoco dos palabras de diccionario unidas
  - Usar contraseñas largas, por ejemplo, de 12 caracteres o más
  - Mezclar caracteres en mayúsculas y minúsculas, con caracteres no alfabéticos (números, signos de puntuación, etc.)
  - No usar la misma contraseña para diferentes tipo de accesos
- La contraseña se cambia con el comando `passwd`
  - `passwd`: cambia la contraseña (password) del usuario
  - Ejemplo: usuario pepe

```
# passwd
Changing password for pepe
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Continuación de la instalación

En una instalación por red los paquetes se traen de un repositorio remoto a través de http o ftp

- Seleccionar el huso horario
- Realizar el particionado del disco (modo guiado o manual)



Particionado del disco

Podemos optar por instalar todo el sistema en una sola partición, aunque no es nada recomendable

- preferible instalar diferentes directorios del sistema en diferentes particiones
- la estructura de directorios UNIX sigue el estándar FHS (*Filesystem Hierarchy Standard*)

Filesystem Hierarchy Standard

Localización estándar de los ficheros

- `/` (root o raíz) - directorio raíz del sistema, todo cuelga de aquí
- `/boot/` - ficheros usados para el arranque, incluyendo el kernel
- `/bin/` (***bin****aries*) - ejecutables esenciales (`ls`, `cat`, `bash`, etc.)
- `/sbin/` - (***su****peruser* ***bin****aries*) - ejecutables esenciales de configuración y administración para el superusuario (`fdisk`, `ifconfig`, etc.)
- `/lib/` - librerías esenciales para los ejecutables de `/bin/` y `/sbin/`

- `/usr/` (***U**nix **s**ystem **r**esources*) - resto de las aplicaciones usadas por los usuarios y el superusuario. Incluye ejecutables, librerías, cabeceras para programación en C, código fuente, manuales, etc, organizados en los siguientes subdirectorios:
  - `/usr/bin/` - la mayoría de las aplicaciones de usuario
  - `/usr/sbin/` - la mayoría de las aplicaciones para el superusuario
  - `/usr/lib/` - librerías que necesitan los ejecutables de `/usr/bin/` y `/usr/sbin/`
  - `/usr/share/` - datos independientes de la arquitectura, fundamentalmente manuales (`/usr/share/man`, `/usr/share/info`)
  - `/usr/include/` - ficheros de cabecera (`.h`) estándar
  - `/usr/src/` (opcional) - código fuente del kernel y de las aplicaciones
  - `/usr/local/` - aplicaciones que no forman parte de la distribución y que el superusuario ha instalado manualmente. Replica la estructura anterior, es decir, `/usr/local/bin/`, `/usr/local/lib/`, `/usr/local/share/`, etc.
- `/opt/` - aplicaciones que requieren un subdirectorio separado del resto (lo usan sobre todo los programas comerciales)
- `/etc/` - ficheros y scripts de configuración, tanto del sistema como de las aplicaciones. Por ejemplo:
  - `/etc/X11/` (opcional) - configuración de X Window
  - `/etc/skel/` (opcional) - plantillas para configurar usuarios
- `/var/` - ficheros **var**iables (logs, bases de datos, etc). Por ejemplo:
  - `/var/log/` - ficheros de log (información que el sistema recopila y guarda)
  - `/var/spool/` - ficheros temporales de impresión, e-mail y otros (los ficheros se borran cuando han sido procesados)
- `/tmp/` - ficheros temporales (cualquier usuario o proceso puede escribir en este directorio) que se borran durante el reinicio del sistema
- `/srv/` - datos de servicios proporcionados por el sistema (páginas web, ftp, cvs, etc.)

- `/home/` (opcional) - directorio de usuarios (directorio inicial o *home*)
- `/root/` (opcional) - directorio *home* del superusuario

Otros directorios del sistema:

- `/media/` - punto de montaje para medios removibles (USB, CDROM). El árbol de directorios de estas unidades cuelga de este punto
- `/mnt/` - punto de montaje para otros sistemas temporales (por ejemplo, el montaje de una unidad de red, de una partición de otro sistema operativo, etc)
- `/dev/` - directorio que contiene *seudo*ficheros de acceso a periféricos (operando sobre estos seudoficheros se dan órdenes a los dispositivos)
- `/proc/` - directorio que contiene información del sistema (CPU, memoria, buses, interrupciones, procesos en ejecución, etc)
- `/sys/` - similar al anterior, contiene información de dispositivos (por ejemplo, el brillo de la pantalla y la carga de la batería de los portátiles)

La mayoría de estos directorios están contenidos en discos, pero algunos están implementados en memoria RAM (de tipo `tmpfs`) o apuntan a otras localizaciones del sistema

Más información: http://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html

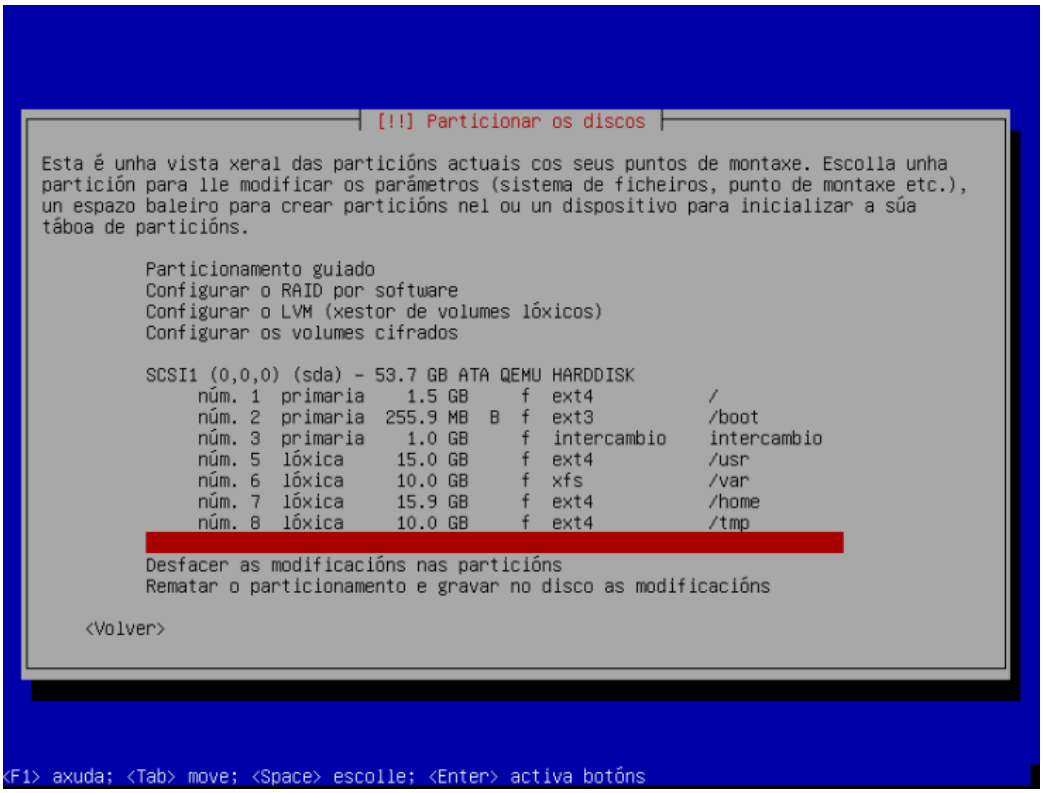
Esquemas de particionamiento

Dependiendo del tipo de sistema podemos escoger diferentes esquemas de particionamiento, algunos ejemplos:

- Máquina de escritorio (un sólo usuario trabajando a la vez), tres particiones que permiten actualizar el sistema sin tocar los datos de usuarios
  - `swap` - área de intercambio, es una zona del disco que en caso de emergencia puede utilizarse en sustitución de la memoria RAM (aunque localizada en disco y extremadamente lenta). Tamaño en función del tamaño de la RAM y del tipo de aplicaciones que se ejecuten (como orientación, tomar al menos el doble de la RAM)
  - `/home/` - cuentas de usuario, tamaño en función del número de usuarios
  - `/` - resto del disco (con el sistema operativo)

- Sistema multiusuario, además de las particiones anteriores crear particiones separadas para `/usr`, `/var` y `/tmp`
  - `/usr` podría montarse en modo sólo-lectura después de que todo el sistema esté instalado (dificulta la introducción de virus)
  - tener `/var` y `/tmp` en su partición evita que estos directorios crezcan hasta ocupar todo el disco (se llenan hasta que ocupen su partición, pero no pueden usar el resto de las particiones)
- Particiones adicionales:
  - `/boot` - para tener la función de arranque separada del resto. Permite incluso evitar las incompatibilidades de las BIOS con los discos duros grandes
  - `/chroot` - para aplicaciones en un entorno *enjaulado* que requieran seguridad y aislamiento (p.e. prueba de distribuciones)
  - `/var/lib` - partición para gestionar ficheros del servidor de bases de datos (DNS, Apache) o del proxy (MySQL, squid) (limita la posibilidad de un ataque por denegación de servicio)

Ejemplo de partición (disco de 50 G):



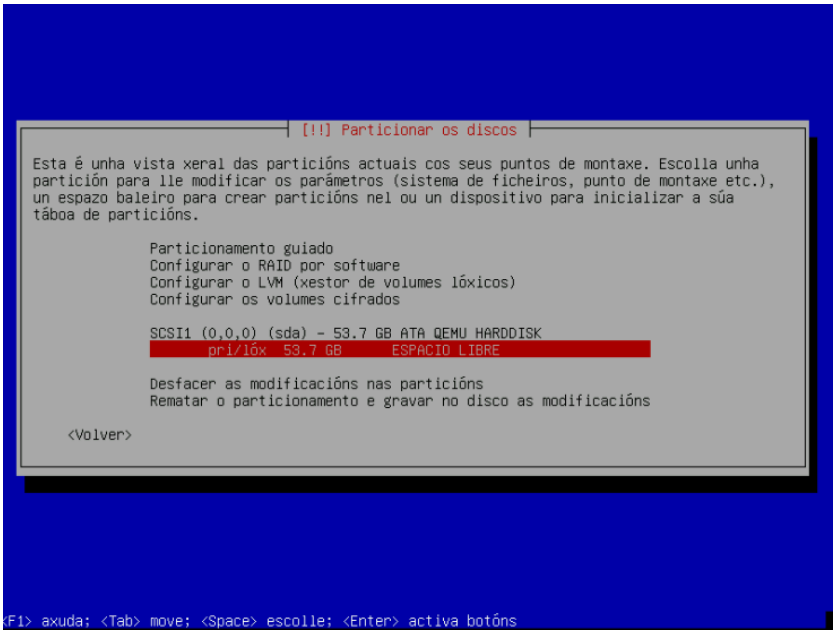
Particionamiento durante la instalación

Dos opciones:

- Particionamiento guiado (con o sin LVM)
  - Selecciona el tamaño de las particiones de manera automática
- Particionamiento manual
  - Particionamiento manual
    - control total del número y tamaño de las particiones

Particionamiento manual

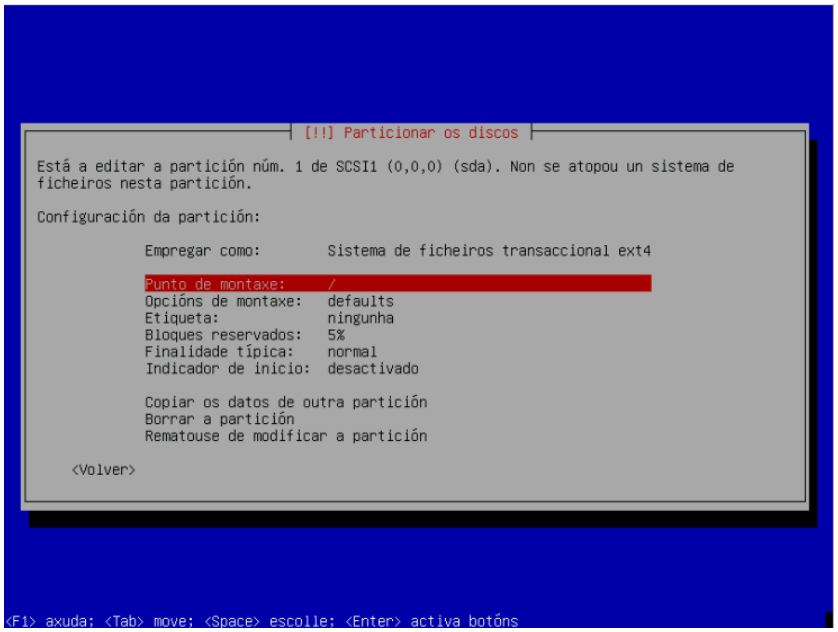
1. Seleccionamos el disco a particionar y crear nueva tabla de particiones:



2. Creamos una nueva partición indicándole el tamaño, el tipo (primaria o lógica) y la localización (comienzo o final)
   - Con la BIOS clásica puede haber 4 primarias o 3 primarias y una extendida, que a su vez se puede dividir en varias lógicas.
   - Con UEFI se pueden crear hasta 128 particiones primarias[2] por disco (de las cuales UEFI requiere para su propio uso la primera, con un tamaño de unos 256-512 MB).

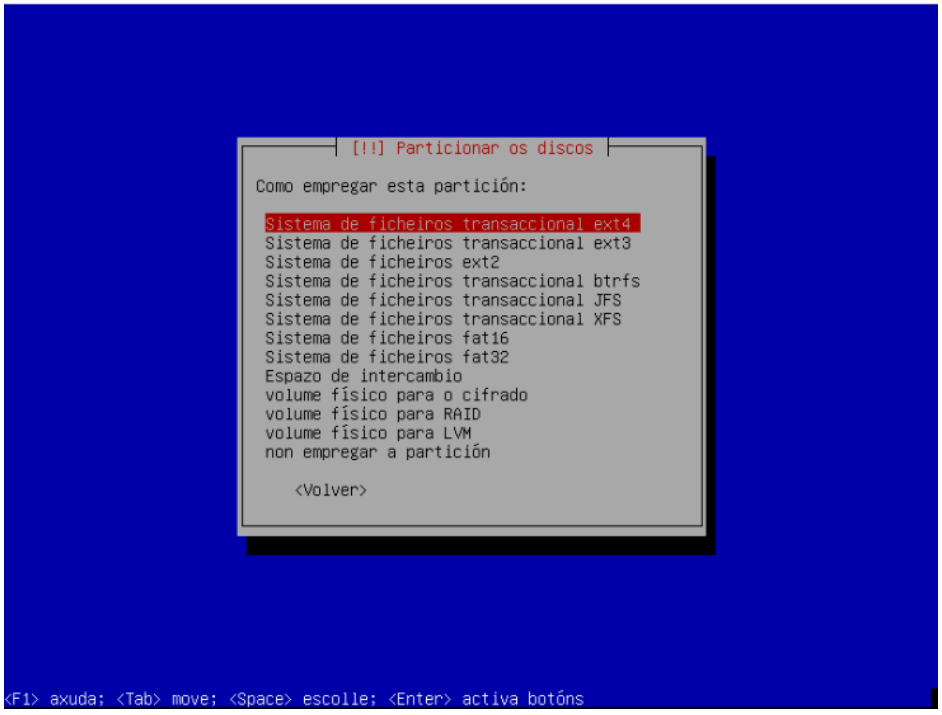
[2]Valor fijado por los SOs de Microsoft.

Ejemplo de partición para /



Sistemas de ficheros

Linux soporta múltiples sistemas de ficheros:

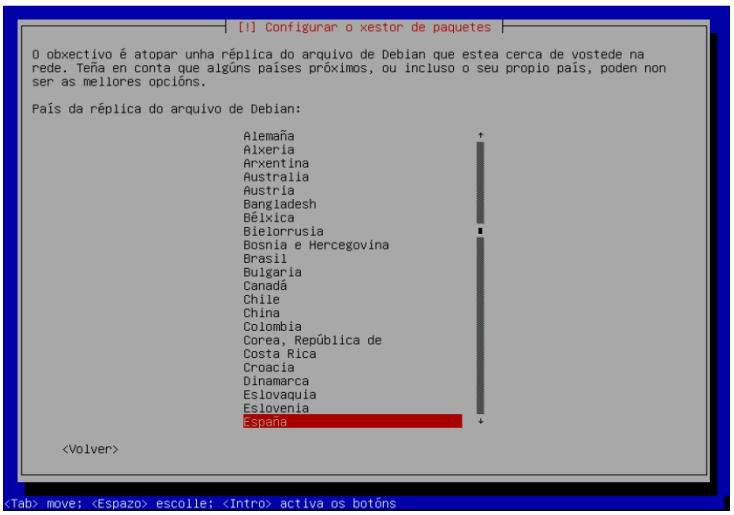


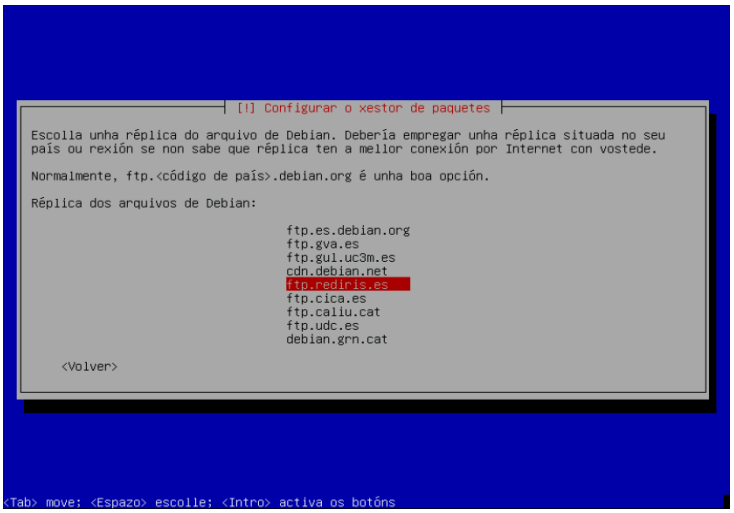
Para cada partición podemos seleccionar los siguientes:

- `ext4` - *Fourth EXTended filesystem* (cuarto sistema de ficheros extendido), es el sistema de ficheros estándar en Linux
  - Es un sistema de ficheros *transaccional* (*journaling*)
  - Tiene características que le permiten reducir la *fragmentación*
  - Puede trabajar con discos y ficheros de gran tamaño (volúmenes de hasta 1 EiB (exbibyte, 2^{60} bytes) y ficheros de tamaño de hasta 16 TiB).
  - Las opciones pueden configurarse con el comando `tune2fs`
  - Las anteriores versiones `ext3` y `ext2` están todavía disponibles
- `btrfs` (sistema de ficheros B-tree), posible sucesor de `ext4`, pues presenta características avanzadas que además de mejorar el rendimiento van dirigidas a la gestión y seguridad del almacenamiento. En concreto:
  - Gestiona de manera integrada el almacenamiento, pues incluye funciones que antes formaban parte del sistema de ficheros, del controlador RAID y del gestor lógico de volúmenes LVM.
  - Hace un uso extensivo de *copy-on-write* (copiar al escribir): si varios recursos son idénticos se devuelve un puntero a un único recurso; en el momento en que se modifica una "copia" del recurso, se crea una copia auténtica para prevenir que los cambios sean visibles a las demás copias
  - Permite *snapshots* de solo lectura o modificables
  - Incluye soporte nativo para sistemas de ficheros multidispositivo y soporta subvolúmenes
  - Protege la información (datos y metadatos) mediante checksums
  - Soporta compresión y empaquetado eficiente de ficheros pequeños
  - Optimizaciones para discos SSD
- `JFS`, `XFS` - otros tipos de sistemas transaccionales (*journaling*) portados de otros sistemas Unix (IBM y GSI Iris, respectivamente).
- `NTFS`, `exFAT` - usados por Windows en ordenadores domésticos y en unidades extraíbles. Soportados también en Linux.
- `ReFS` (*Resilent File System*) - usado por Windows para entornos empresariales. No disponible de forma nativa en Linux.

Últimos pasos en la instalación

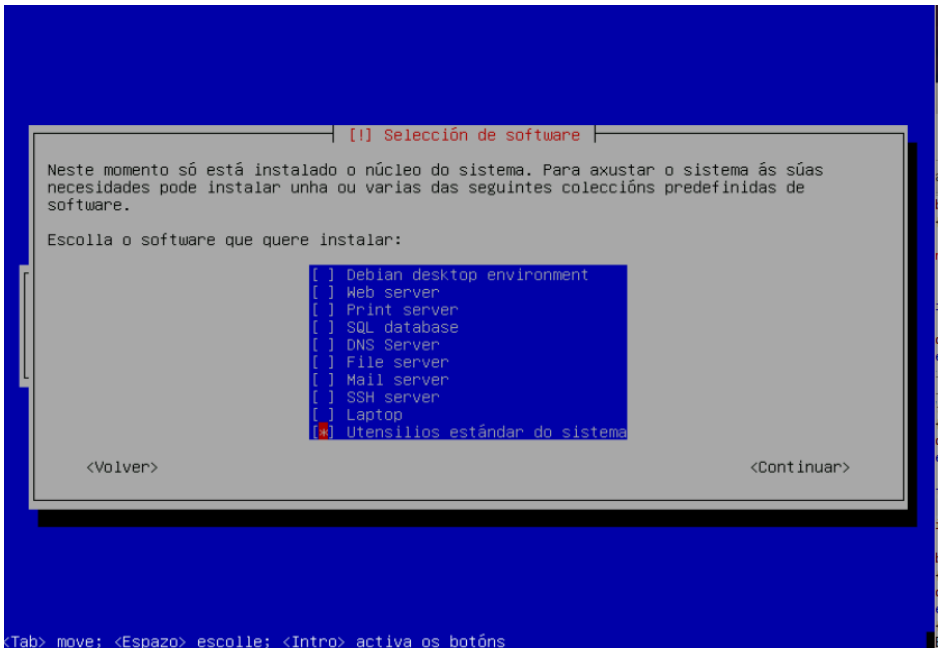
- Debemos seleccionar el *mirror* desde el que descargar el software
  - Existen varios repositorios de paquetes Debian ⟶ elegir el más cercano
  - Introducir la información del proxy, en el caso de trabajar en un PC de la red de cable de la USC (proxy2.usc.es y puerto 8080).





- Seleccionar los paquetes software a instalar
- Instalar del gestor de arranque

Selección de paquetes



- Elegir los paquetes a instalar:
  - elegir sólo "Utilidades estándar" (el resto lo instalaremos después)
  - desmarcar "Aplicaciones de escritorio", puesto que su instalación requiere un tiempo excesivo.
  - aunque optemos por no instalar nada, se instalarán todos los paquetes con prioridad "estándar", "importante" o "requerido" que aún no estén instalados en el sistema
- Podemos repetir este paso con el sistema instalado usando el comando `tasksel`

Instalación del gestor de arranque

Podemos tener diferentes distribuciones de Linux y Windows en el mismo ordenador, cada una con sus correspondientes particiones. El gestor de arranque (cargador o *bootloader*) nos permite seleccionar el SO a arrancar.

- Las distribuciones Linux usan el cargador del proyecto GNU denominado GRUB (*GRand Unified Bootloader*), actualmente la versión 2.

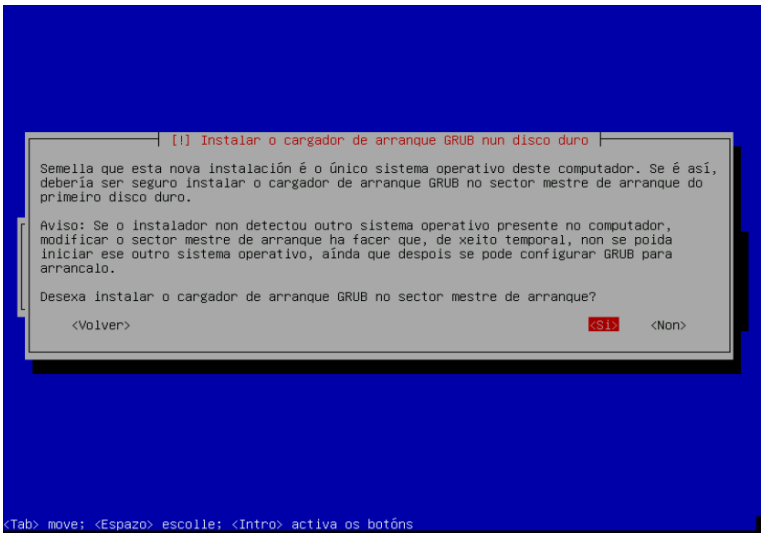
- Cuando el sistema se inicia, la BIOS carga el gestor de arranque que nos permite seleccionar el SO y a continuación transfiere el control al programa de inicio del correspondiente SO (localizado en `/boot`).

Para la instalación del gestor de arranque tenemos dos posibilidades:

1. Lo usual es instalarlo en el MBR o GPT del primer disco
   - El MBR (*Master Boot Record*, registro maestro de arranque) contiene información sobre las particiones del disco (*Master Partition Table*) y un pequeño código (*Master Boot Code*) del gestor de arranque. MBR se almacena solo en el primer sector del disco.
   - El GPT (*GUID Partition Table*) es el nuevo estándar que sustituye al anterior, más fiable y asociado a los sistemas UEFI. GPT crea múltiples copias redundantes a lo largo de todo el disco y su nombre hace referencia a que a cada partición se le asocia un identificador global único (GUID).
2. En caso de que tengamos otro cargador en el MBR o GTP, el gestor de arranque GRUB puede instalarse en el primer sector de la partición Linux que contenga `/boot`

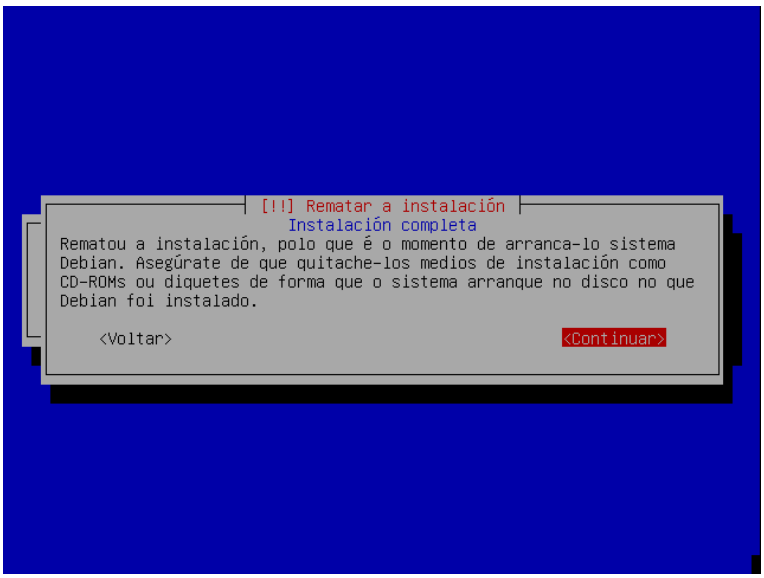
Instalación de GRUB en Debian

Debemos instalar al menos un gestor de arranque, en caso contrario no podríamos arrancar el sistema operativo. Si tenemos varios sistemas operativos en la misma máquina basta con instalar el gestor de arranque para uno de ellos (usualmente el del último sistema que instalemos).



Finalización de la instalación

Debian: la instalación termina aquí. Debemos reiniciar para continuar

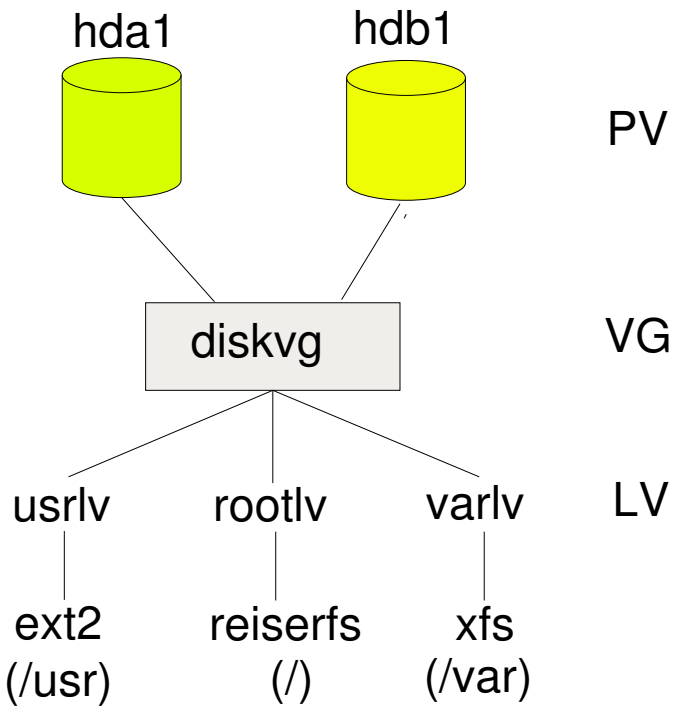


Logical Volume Management (LVM)

Proporciona una visión de alto nivel de los discos

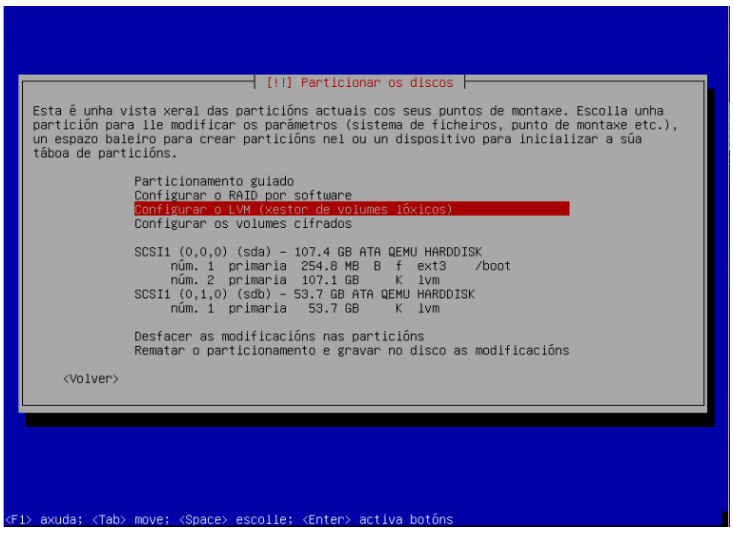
- permite ver varios discos como un único volumen lógico
- permite hacer cambios en las particiones sin necesidad de reiniciar el sistema
- permite gestionar los volúmenes en grupos definidos por el administrador
- **Volumen físico (PV)**: discos duros, particiones de los discos u otro dispositivo similar (p.e. RAID)
- **Grupo de volúmenes (VG)**: agrupación de LV, que forman una unidad administrativa
- **Volumen lógico (LV)**: *particiones* lógicas sobre las que se montan los sistemas de ficheros

Estructura de LVM:



Pasos para crear un sistema LVM
Suponemos un sistema con dos discos (`sda` y `sdb`)

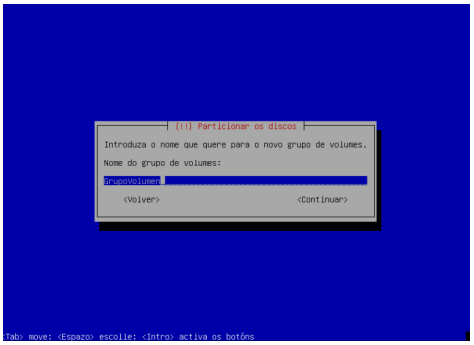
1. Crear los PV

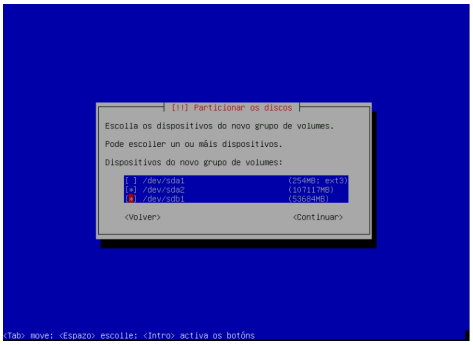


   - particionamos `sda` para reservar un espacio para `/boot` (dejamos `/boot` fuera de LVM para evitar problemas con el arranque, aunque extrictamente no es necesario)

- definimos 2 volúmenes físicos
  - el primero incluye todo `sda` menos `/boot` (`sda2`)
  - el segundo incluye todo `sdb` (`sdb1`)

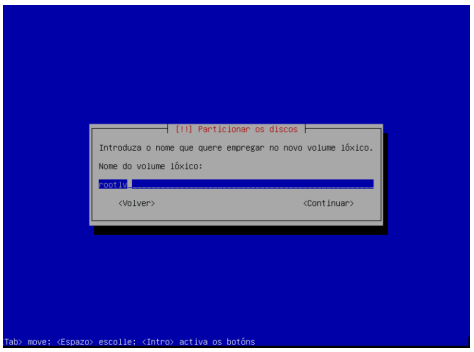
2. Crear un grupo de volumen que incluya los PVs

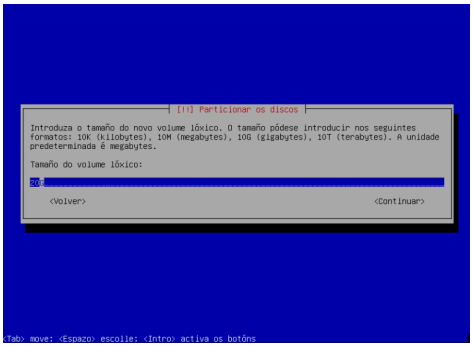




- podemos ponerle un nombre al grupo de volumen
- hacemos que incluya los dos volúmenes físicos que hemos definido en el punto anterior

3. Crear los volúmenes lógicos

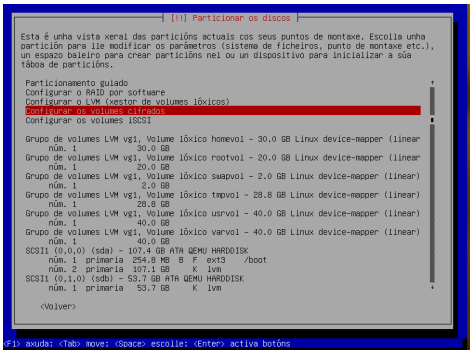


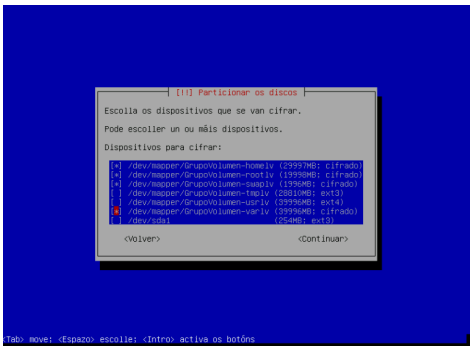


- creamos un volumen lógico por cada partición
- los LV pueden llevar un nombre identificativo

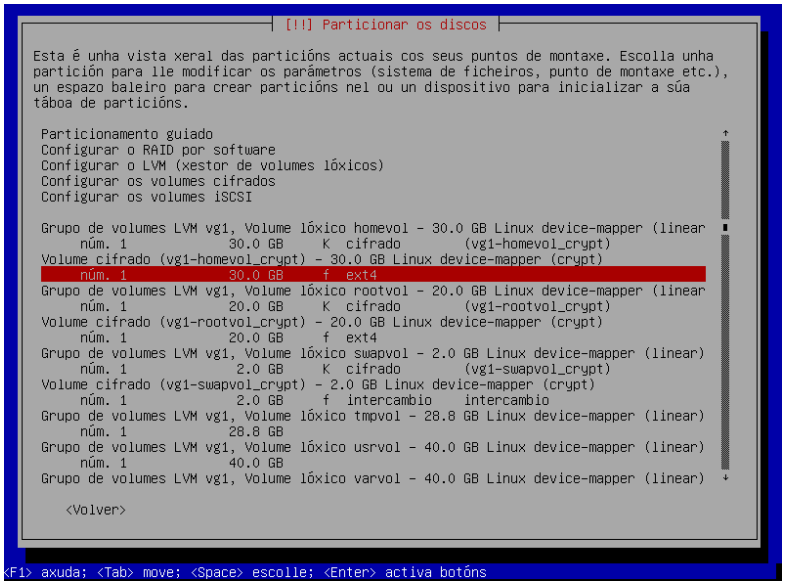
4. Cifrar sistemas de ficheros

- podemos usar algún LV como “volumen físico para cifrado”
- permite cifrar la información: contraseña para acceder a la misma





5. Asignar sistemas de ficheros a los volúmenes (cifrados o no)



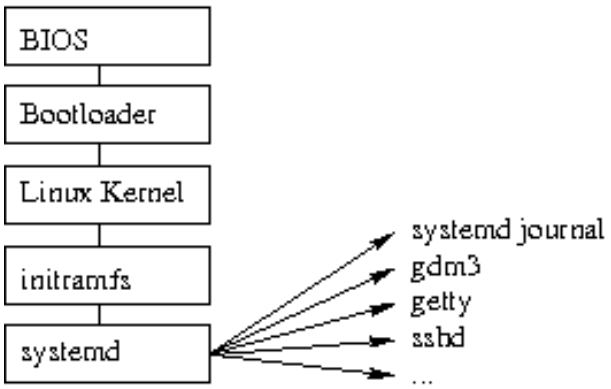
Configuración del gestor de arranque

Podemos configurar GRUB para evitar que sea modificado el menu de arranque. En concreto, podemos usar una contraseña para limitar:

- la modificación de los parámetros iniciales
- el acceso a determinadas imágenes
- el acceso a opciones avanzadas

1.1.3. Arranque del sistema

Al arrancar el ordenador se realizan las siguientes operaciones:



1. **BIOS**. En primer lugar, el procesador se dirige a una posición de memoria específica, donde se encuentra la BIOS. Se ejecuta un pequeño programa que detecta los discos, carga el registro maestro de arranque (MBR) (o su equivalente GPT), y ejecuta el gestor de arranque (usualmente GRUB).

   En el menú de la BIOS podemos fijar algunas opciones de arranque, por ejemplo, si tenemos varios discos, unidades de DVD o discos USB, podemos indicar de qué medio se leerá el MBR.

   NOTA: La BIOS puede ser configurada no para ejecutar el MBR, sino para buscar su equivalente en la red, por lo que es posible la utilización ordenadores sin disco duro de arranque por red.

2. **Gestor de arranque**. Ya en el disco, el gestor de arranque consiste en otro pequeño programa que usualmente muestra un menú con la lista de los S.O. disponibles, el usuario puede seleccionar el que le interese, y carga el `kernel` (núcleo) de Linux desde el disco y lo ejecuta.

3. **Kernel**. El kernel (localizado en `/boot/vmlinuz`) comienza a buscar y montar la partición que contiene el sistema de ficheros raíz, ejecutando el primer programa, `init`. Por su complejidad, este proceso se suele realizar en dos pasos:

4. **initramfs**. En una primera fase se carga en memoria RAM el fichero de imagen de un pequeño disco virtual que contiene una "partición raíz" virtual y un programa `init` (de ahí su nombre, `initramfs`, "disco RAM de inicialización").

`initramfs` contiene el mínimo requerido por el kernel para cargar el ‘ “verdadero” sistema de ficheros raíz contenido en el disco.

5. **init**. En una segunda fase, `initramfs` cede el control al “init real”, y la máquina inicia el proceso de arranque estándar.

   Este incluye módulos de controladores del disco duro y de otros dispositivos sin los cuales el sistema no puede arrancar, frecuentemente en la forma de scripts de inicialización y módulos para el montaje de RAIDs, inicio de particiones cifradas, activación de volúmenes LVM, etc.

systemd es el sistema de inicio actualmente utilizado en Debian. Sus características son las siguientes:

- `systemd` ejecuta varios procesos encargados de la creación del sistema: teclado, controladores, sistemas de ficheros, red, servicios, etc.
- Esto hace que ofrezca una visión global del sistema tanto software como hardware.
- Muchos de estos procesos ofrecen servicios, por ejemplo, `networking` (red), `cups` (servicio de impresión), `ssh`, {`xdm, gdm, ligthdm`} (diferentes gestores de display), `lvm` (gestor de volúmenes lógicos), etc.
- El arranque de los servicios se realiza en paralelo con el objetivo de acelerar el arranque.

Varias utilidades (por ejemplo, `systemctl` y `service`) permiten al administrador controlar los servicios que se inician y el estado del sistema.

- `systemctl` es un comando de gestión de servicios específico de `systemd`. Si se ejecuta sin argumentos muestra la lista de servicios activos, mientras que si inicamos un servicio, podemos realizar diferentes operaciones sobre el:

  `systemctl`
  `systemctl` *acción servicio*

  Donde la *acción* puede ser:

  `status` - muestra el estado del servicio
  `start` - inicia el servicio
  `stop` - detiene el servicio
  `restart` - detiene el servicio y a continuación lo reinicia
  `enable` - configura el servicio para ser iniciado en el arranque
  `disable` - el servicio no será iniciado durante el arranque

- Por ejemplo,

```
# configura un arranque sin entorno gráfico
systemctl disable xdm
# comprueba si hay servidor de ssh
systemctl status ssh
# reinicia la red
systemctl restart networking
```

- `service` es el comando clásico que tiene una sistaxis similar:

  `service --status-all`
  `service` *servicio acción*

  donde las acciones son parecidas a las del comando anterior (véase el manual).

- Por ejemplo:

```
# comprueba si hay servidor de ssh
service ssh status
# reinicia la red
service networking restart
```

`systemd` distingue varios *target* (que sustituyen a los clásicos *runlevels*). Entre ellos destacan:

- **Rescate** (`rescue.target`) arranca lo mínimo para intentar reparar un sistema dañado.
- **Emergencia** (`emergency.target`) abre un único shell (monousuario).
- **Multiusuario** (`multi-user.target`) es multiusuario no gráfico
- **Gráfico** (`graphical.target`) es multiusuario gráfico.

Disponemos de los siguientes comandos:

```
# Muestra el target actual
systemctl get-default
# Conmuta al target especificado
systemctl isolate <name of target>.target
# Fija el target que se ejecutará en el arranque
systemctl set-default <name of target>.target
```

1.1.4. Verificación de la instalación

- Las últimas distribuciones de Linux soportan la mayoría del hardware actual.
- Hay soporte Linux para múltiples arquitecturas: i386, AMD64, ARM, IBM Power, Intel Itanium, etc.
- En el proceso de instalación se configura automáticamente casi todo el hardware

Información de hardware

Disponemos de los siguientes comandos y ficheros:

`dmidecode`	vuelca la información DMI de la BIOS
`lshw`	infomación general de hardware
`lscpu`	información sobre el procesador
`nproc`	devuelve el número de núcleos disponible
`lspci`	muestra los dispositivos PCI
`lsusb`	muestra los dispositivos USB
`fdisk -l`	muestra todas las particiones
`df -h`	muestra las particiones montadas
`free -h`	memoria total, usada, disponible
`uname -a`	muestra versión de kernel
`lsb_release -a`	muestra distribución de Linux
`lsmod, rmmod, insmod`	operaciones con módulos del kernel
`/proc`, `/sys`	directorios con información de hardware
`/dev/sdxx, /dev/nvmexx`	dispositivos de discos y particiones

BIOS

- `dmidecode`: Este comando vuelca la información DMI (Desktop Management Interface) de la BIOS, incluyendo información de la placa base, configuraciones, opciones disponibles, módulos de RAM, discos y tarjetas insertadas, etc.

  DMI es un entorno estándar para gestión y seguimiento de componentes en un ordenador de sobremesa, portátil o servidor, creando una abstracción de esos componentes desde el software que los gestiona. Antes de la introducción de DMI, la información sobre los detalles de los componentes en un ordenador personal podían hallarse en fuentes de información no estandarizadas. Por ejemplo,

```
# dmidecode
BIOS Information
        Vendor: American Megatrends Inc.
        Version: F6
        ROM Size: 8 MB
        Characteristics:
                PCI is supported
                Boot from CD is supported
                ACPI is supported
                USB legacy is supported
                UEFI is supported
...
System Information
        Manufacturer: Gigabyte Technology Co., Ltd.
        Product Name: H81M-HD3
        Version: To be filled by O.E.M.
        Wake-up Type: Power Switch
...
Cache Information
        Socket Designation: CPU Internal L1
        Operational Mode: Write Back
        Installed Size: 128 kB
        Maximum Size: 128 kB
        Associativity: 8-way Set-associative
...
Processor Information
        Socket Designation: SOCKET 0
        Type: Central Processor
        Family: Core i7
        Manufacturer: Intel
        Signature: Type 0, Family 6, Model 60, Stepping 3
        Flags:
                FPU (Floating-point unit on-chip)
                VME (Virtual mode extension)
                DE (Debugging extension)
...
Memory Device
        Data Width: 64 bits
        Size: 4 GB
        Form Factor: DIMM
        Bank Locator: BANK 0
        Type: DDR3
        Type Detail: Synchronous
        Speed: 1333 MT/s
```

Si queremos información sobre un determinado componente hardware podemos ejecutar `dmidecode -t` n donde n puede ser (véase la página del manual):

n	Componente
0	BIOS
1	System
2	Baseboard
3	Chassis
4	Processor
5	Memory Controller
6	Memory Module
7	Cache
8	Port Connector
9	System Slots
10	On Board Devices
11	OEM Strings
12	System Configuration Options
13	BIOS Language
14	Group Associations
15	System Event Log
16	Physical Memory Array
17	Memory Device
18	32-bit Memory Error
19	Memory Array Mapped Address
20	Memory Device Mapped Address
21	Built-in Pointing Device
22	Portable Battery
23	System Reset
24	Hardware Security
25	System Power Controls
26	Voltage Probe
27	Cooling Device
28	Temperature Probe
29	Electrical Current Probe
30	Out-of-band Remote Access
31	Boot Integrity Services
32	System Boot
33	64-bit Memory Error
34	Management Device
35	Management Device Component
36	Management Device Threshold Data
37	Memory Channel
38	IPMI Device
39	Power Supply
40	Additional Information
41	Onboard Devices Extended Information
42	Management Controller Host Interface

Información general de hardware

- `lshw`: muestra información general del hardware. La información es similar a la del comando anterior, si bien en este caso el comando lee la información de los directorios `/proc` y `/sys`.

Información sobre la CPU

- `lscpu`: muestra información sobre el procesador, incluyendo núcleos, memoria caché, etc.
- `nproc`: devuelve el número de núcleos disponible.

Por ejemplo,

```
$ lscpu
```

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  2
  On-line CPU(s) list:   0,1
Vendor ID:               GenuineIntel
```

```
  Model name:            Intel(R) Pentium(R) CPU G3220 @ 3.00GHz
    Thread(s) per core:  1
    Core(s) per socket:  2
    Socket(s):           1
    Stepping:            3
    CPU max MHz:         3000,0000
    CPU min MHz:         800,0000
    BogoMIPS:            5986.29
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
...
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   64 KiB (2 instances)
  L1i:                   64 KiB (2 instances)
  L2:                    512 KiB (2 instances)
  L3:                    3 MiB (1 instance)
Vulnerabilities:
  Meltdown:              Mitigation; PTI
  Spectre v1:            Mitigation; usercopy/swapgs barriers and __user pointer
                          sanitization
  Spectre v2:            Mitigation; Retpolines, IBPB conditional, IBRS_FW, STIB
                         P disabled, RSB filling
...
```

```
$ nproc
```

```
2
```

Dispositivos PCI y USB

Comandos para verificar los dispositivos y periféricos conectados a los buses de nuestro sistema:

- `lspci`: lista dispositivos PCI, incluidos los PCI Express
- `lsusb`: lista dispositivos USB

1) Ejemplo: sistema con discos IDE, tarjeta VGA y dos tarjetas de red:

```
$ lspci
```

```
0000:00:00.0 Host bridge: Intel Corp. 440FX - 82441FX PMC [Natoma]
0000:00:01.0 ISA bridge: Intel Corp. 82371SB PIIX3 ISA [Natoma/Triton II]
0000:00:01.1 IDE interface: Intel C. 82371SB PIIX3 IDE [Natoma/Triton II]
0000:00:02.0 VGA compatible controller: Cirrus Logic GD 5446
0000:00:03.0 Ethernet controller: Realtek Semic. Co., Ltd. RTL-8029(AS)
0000:00:04.0 Ethernet controller: Realtek Semic. Co., Ltd. RTL-8029(AS)
...
```

2) Ejemplo: sistema con teclado, ratón, hub USB y dos pendrives:

```
$ lsusb
Bus 001 Device 005: ID 413c:1002 Dell Computer Corp. Keyboard Hub
Bus 001 Device 007: ID 413c:2002 Dell Computer Corp. Keyboard
Bus 002 Device 009: ID 413c:3010 Dell Computer Corp. Optical Wheel Mouse
Bus 005 Device 015: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 005 Device 016: ID 0ea0:2168 Transcend JetFlash 2.0 / Astone USB Drive
Bus 005 Device 019: ID 0c76:0005 JMTek, LLC. USBdisk
```

Discos duros

Los discos son dispositivos de bloques, esto es, que transfieren bloques de datos, en contraposición a los dispositivos de caracteres (como ratones, teclados, etc) que transfieren caracter a caracter. Por ejemplo, discos mecánicos, discos SSD o unidades Flash USB. En la arquitectura Intel y dependiendo del interface de comunicación, se distinguen:

1. SATA (Serial ATA)
   - Denominados así porque utilizan el bus SATA.
   - Identificados en Linux con `sd` seguido de una letra minúscula (`a`, `b`, `c`, ...) en función del orden del dispositivo descubierto: `/dev/sda` para el primer dispositivo descubierto, `/dev/sdb` para el segundo, `/dev/sdc` para el tercero, etc.
2. NVMe (NonVolatile Memory express)
   - Es el nuevo protocolo de acceso a las unidades de memoria no volátiles (discos SSD) basado en el bus PCI express. PCIe es utilizado para conectar todo tipo de tarjetas (GPUs, por ejemplo) y es mucho más rápido que el bus SATA. NVMe se suele usar combinado con un conector específico denominado M.2
   - Identificados en Linux con `nmve` seguido de un número que indica el controlador del bus, luego una `n` y un número que indica el dispositivo dentro del controlador, por ejemplo,
     `/dev/nvme0n1`$p2$ - controlador 0, dispositivo 1 (*partición 2*)
     `/dev/nvme2n5`$p3$ - controlador 2, dispositivo 5 (*partición 3*)

Particiones

En Linux las particiones en un disco se identifican con un número después del nombre del dispositivo. Para mostralas tenemos los siguientes comandos:

- `fdisk -l`: lista todas las particiones (sólo el superusuario).

  La opción `-l` es necesaria porque en otro caso debemos indicarle un dispositivo de disco.

- `df -h`: lista las particiones montadas

  Algunas opciones:

  - `-h`: (humano) más fácil de leer pues da los tamaños en kilobytes, Megabytes y Gigabytes en lugar de en bloques de 1k
  - `-T`: (tipo) imprime el tipo de sistema de ficheros
  - `-l`: (local) sólo muestra sistemas de ficheros locales

Ejemplo: Disco duro Windows/Linux con UEFI y particiones físicas

```
# fdisk -l
Disk /dev/sda: 894.25 GiB, 960197124096 bytes, 1875385008 sectors
Disk model: KINGSTON SA400S3
Units: sectors of 1 * 512 = 512 bytes
Disklabel type: gpt
Disk identifier: 0A17951F-06A1-4A33-BB76-1A9280E6CBB6

Device          Start        End   Sectors   Size Type
/dev/sda1        2048     206847    204800   100M EFI System
/dev/sda2      206848     239615     32768    16M Microsoft reserved
/dev/sda3      239616  618097242 617857627 294.6G Microsoft basic data
/dev/sda4  1874237440 1875382271   1144832   559M Windows recovery environment
/dev/sda5   618098688 1457942527 839843840 400.5G Linux filesystem
/dev/sda6  1457942528 1582944255 125001728  59.6G Linux swap
/dev/sda7  1582944256 1680601087  97656832  46.6G Linux filesystem
...
```

Ejemplo: Un disco duro (*/dev/nvme0n1*) y un pendrive montado (*/dev/sda*)

```
$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
tmpfs          tmpfs     1.6G  2.4M  1.6G   1% /run
/dev/nvme0n1p8 ext4       32G   23G  7.3G  76% /
tmpfs          tmpfs     7.7G     0  7.7G   0% /dev/shm
tmpfs          tmpfs     5.0M   16K  5.0M   1% /run/lock
efivarfs       efivarfs  192K  100K   88K  54% /sys/firmware/efi/efivars
/dev/nvme0n1p5 ext4      146G   93G   46G  68% /home
/dev/nvme0n1p1 vfat       96M   33M   64M  35% /boot/efi
tmpfs          tmpfs     1.6G  124K  1.6G   1% /run/user/1000
/dev/sda1      exfat     117G  2.8G  114G   3% /mnt/usb
```

Las particiones de tipo `tmpfs` son sistemas de ficheros virtuales en memoria RAM usados por el SO.

Información de memoria

- `free -h`: muestra la memoria total, disponible y libre tanto de RAM como de *swap*.

  La opción `-h` da los tamaños en Kibibyte (KiB=2^{10}), Mebibyte (MiB=2^{20}) y Gibibyte (GiB=2^{30}) y Tebibyte (TiB=2^{40}) en lugar de en bloques de 1k.

Por ejemplo,

```
$ free -h
```

```
        total    used    free   shared  buff/cache   available
Mem:    7,6Gi    1,0Gi   5,7Gi    19Mi       928Mi       6,4Gi
Swap:    52Gi    182Mi    51Gi
```

Versión de Linux

- `uname -a`: muestra la versión del kernel de Linux (`-a` da toda la información).
- `lsb_release -a`: muestra la distribución de Linux.

Por ejemplo,

```
$ uname -a
```

```
Linux H610M-S2H-DDR4 6.8.0-45-generic #45-Ubuntu SMP x86_64 GNU/Linux
```

```
$ lsb_release -a
```

```
Distributor ID:        Ubuntu
Description:    Ubuntu 24.04.1 LTS
Release:        24.04
Codename:       noble
```

Módulos del kernel

Los módulos del kernel son fragmentos de código que pueden ser cargados y eliminados del núcleo bajo demanda. Muchos de ellos funcionan como controladores de dispositivos (tarjetas de sonido, tarjetas gráficas, etc). Se encuentran en ficheros con la extensión `*.ko`. Los comandos para trabajar con módulos son:

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

Por ejemplo:

```
$ lsmod
snd_hda_intel     40960  0
snd_hda_codec    135168  4 snd_hda_codec_realtek,snd_hda_codec_hdmi,snd_hda
kvm_intel        172032  0
kvm              540672  1 kvm_intel
nvidia_uvm       745472  0
nvidia_drm        45056  1
nvidia_modeset   765952  4 nvidia_drm
nvidia         11489280  59 nvidia_modeset,nvidia_uvm
```

```
# rmmod kvm_intel
# modprobe soundcore
```

Directorios `/proc` y `/sys`

Los ficheros contenidos en estos directorio contienen mucha información sobre los recursos usados por el hardware. De hecho, la mayoría de los comandos anteriores se limitan a leer estos ficheros y a presentar la información al usuario.

Ejemplo: procesador con dos núcleos:

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 60
model name      : Intel(R) Pentium(R) CPU G3220 @ 3.00GHz
stepping        : 3
microcode       : 0x1e
cpu MHz         : 1252.031
cache size      : 3072 KB
flags           : fpu vme de pse tsc msr pae mce cx8 apic mmx fxsr sse sse2  ...
...
processor       : 1
vendor_id       : GenuineIntel
...
```

1.2. Instalación de software

1.2.1. Formas de instalación

Tenemos, básicamente dos formas de instalar programas en Linux:

- Instalación desde paquetes precompilados
  - Menos optimización
  - Más sencilla
- Compilación e instalación desde las fuentes
  - Optimización para nuestro sistema
  - Más compleja

Gestores de paquetes

En la mayoría de distribuciones Linux, es posible obtener los programas precompilados en formato de paquetes

- Ventajas:
  - Fáciles de instalar y desinstalar
  - Fáciles de actualizar
  - Fácil control de los programas instalados
- Inconvenientes
  - Binarios menos optimizados
  - Problemas de dependencias de paquetes
  - Problemas si la base de datos de paquetes se corrompe

Formatos de paquetes más populares

- Paquetes `DEB` (distribuciones Debian, Ubuntu, etc)
- Paquetes `RPM` (**R**edHat **P**ackage **M**anager, distribuciones Fedora, RedHat, Mandriva, etc.)

Gestión de paquetes en Debian

La distribución Debian incluye un gran número de paquetes (unos 50.000). Existen varias herramientas para el manejo de esos paquetes.

- `dpkg` - herramienta de bajo nivel, para gestionar directamente los paquetes DEB
- `apt` - herramientas APT, permiten gestionar los paquetes, descargándolos de varias fuentes (CDs, http)
- Muchas otras herramientas con interface gráfico o semigráfico, a veces con formato *store* o tienda.
- `alien` - permite convertir e instalar en Debian paquetes de otro tipo, p.e. RPMs

Otras herramientas para instalar paquetes:

- `snap` - estos paquetes son autocontenidos por lo que funcionan en una amplia gama de distribuciones de Linux. Incluyen dentro del paquete todas las componentes que necesitan, por ejemplo las librerías. Esto permite que funcionen independientemente de las versiones de librerías que tenemos en Linux pero también introducen mucha redundancia.

  Las aplicaciones *snap* se ejecutan en contenedores con acceso limitado al sistema host. Sin embargo, los usuarios pueden otorgarle el acceso a funciones adicionales del host, por ejemplo, la grabación de audio, el acceso a dispositivos USB, etc (los permisos son similares a los de las aplicaciones android). Este enfoque aumenta la seguridad pero tiene la desventaja que las aplicaciones tardan más tiempo en arrancar.

  snap fue desarrollado por Ubuntu. En otras distribuciones existen proyectos smilares, como *Flatpak* propuesto por Fedora y RedHat.

- `pip` - es el instalador de paquetes de Python (pip: package installer for Python). Como su nombre indica, solo se utiliza para instalar componentes y librerías de Python. Dependiendo que estemos usando la versión 2.7 o la 3.x de python el comando es `pip` o `pip3`.

  Las operaciones que se pueden realizar son las usuales, por ejemplo,

```
apt-get install python3-pip
pip3 install numpy
pip3 remove pytorch
```

1.2.2. dpkg - Debian Package

Permite instalar, actualizar, desinstalar o listar paquetes DEB. Los paquetes tienen que haber sido previamente descargados.

Contenido de los paquetes DEB. Cada paquete DEB contiene:

- Los binarios de la aplicación (ejecutables, librerías, documentación)
- Metadatos, con información sobre el paquete, *scripts* para su configuración, lista de dependencias, etc.

Nombre de los paquetes

- *paquete_versión-build_arquitectura*.deb, donde
  - *paquete* - nombre de la aplicación
  - *versión* - número de versión de la aplicación
  - *build* - número de "compilación" (subversión)
  - *arquitectura* - plataforma para la que está compilado
- Ejemplos:

  `ethereal_0.10.11-1_i386.deb`
  `libgtk-3-0_3.18.9-1_amd64.deb`

Opciones de dpkg

Las operaciones con *dpkg* son las usuales de los gestores de paquetes (instalar, eliminar, etc):

`-i`	`--install`	instalación del paquete
`-r`	`--remove`	eliminación del paquete
`-P`	`--purge`	eliminación completa del paquete
`-l`	`--list`	lista los paquetes
`-s`	`--status`	imprime el estado del paquete
`-L`	`--listfiles`	lista el contenido del paquete
`-S`	`--search`	busca el paquete al que pertenece un fichero
`dpkg-reconfigure`		reconfigura el paquete
`/var/lib/dpkg/lock`		fichero de bloqueo (cerrojo)

NOTA: siguiendo un convenio bastante extendido en Linux las opciones de una letra vienen precedidas por un guión, mientras que las opciones en formato palabra están precedidas por dos guiones.

NOTA: la opción `-l` (minúscula) da la lista de los paquetes instalados, mientras que `-L` (mayúscula) muestra el contenido de un paquete (lista de ficheros que lo componen). Lo mismo para `-s` (minúscula) que da el estado del paquete y `-S` (mayúscula) usado para hacer búsquedas de ficheros. La opción `-P` (mayúscula) se utiliza para purgar un paquete puesto que la misma letra en minúscula tiene una cierta función de impresión.

Instalación y eliminación de paquetes con dpkg:

- Instalación de paquetes

  `dpkg -i` *nombre_completo_paquete*`.deb`
  `dpkg --install` *nombre_completo_paquete*`.deb`

  - NOTA: Esta es la única operación que veremos de *dpkg* que necesitamos escribir el nombre del paquete con todos sus campos, *paquete_versión-build_arquitectura*.deb. En las demás operaciones solo necesitamos escribir el primer campo del nombre.
  - la instalación chequea la existencia de dependencias, paquetes en conflicto, sobreescritura de ficheros existentes, etc.
  - si el comando falla, se puede forzar la instalación usando la opción `--force-all`.

- Eliminación de paquetes, manteniendo los ficheros de configuración

  `dpkg -r` *paquete*
  `dpkg --remove` *paquete*

- Eliminación total de paquetes, eliminando los ficheros de configuración

  `dpkg -P` *paquete*
  `dpkg --purge` *paquete*

- Reconfiguración de las opciones de un paquete ya instalado

  `dpkg-reconfigure` *paquete*

Información sobre los paquetes

- Listar paquetes (si no se pone *patrón* muestra todos los paquetes instalados):

  `dpkg -l` [*patrón*]
  `dpkg --list` [*patrón*]

```
$ dpkg -list "telnet*"

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Nome                  Versión               Descripción
+++-=====================-=====================-=====================================
ii  telnet                0.17-29               The telnet client
un  telnet-client         <ningunha>            (non hai ningunha descripción dispoñible)
un  telnet-hurd           <ningunha>            (non hai ningunha descripción dispoñible)
un  telnet-server         <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnet-ssl            <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnetd               <ningunha>            (non hai ningunha descripción dispoñible)
un  telnetd-hurd          <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnetd-ssl           <ningunha>            (non hai ningunha descripción dispoñible)
```

Las dos primeras letras representan:

- **Estado de selección**: indica el estado del paquete para las acciones que se llevarán a cabo posteriormente
  - u, *Unknown* - estado no conocido
  - i, *Install* - paquete seleccionado para instalar (se instala con `dselect install`)
  - r, *Remove* - paquete seleccionado para eliminar (se elimina con `dselect install`)
  - p, *Purge* - paquete seleccionado para purgar (se elimina con `dselect install`)
  - h, *Hold* - paquete retenido (no puede actualizarse)
- **Estado actual**: indica el estado actual del paquete
  - n, *Not Installed* - paquete no instalado
  - i, *Installed* - paquete instalado en el sistema
  - c, *Config-files* - paquete no instalado, pero ficheros de configuración presentes (p.e. después de un remove)
  - u, *Unpacked* - paquete desempaquetado y listo para instalación
  - f, *Failed-config* - durante la instalación falló la configuración del paquete
  - h, *Half-installed* - paquete a medio instalar debido a algún problema

Para que un programa esté correctamente instalado tiene que aparecer la combinación `ii`.

- Estado del paquete

  ```
  dpkg -s paquete
  dpkg --status paquete
  ```

  Ejemplo:

```
$ dpkg --status wget
Package: wget
Status: install ok installed
Priority: important
Section: web
Installed-Size: 1428
Maintainer: Noel Kothe <noel@debian.org>
Architecture: i386
Version: 1.10-2
Depends: libc6 (>= 2.3.2.ds1-21), libssl0.9.7
Conflicts: wget-ssl
Description: retrieves files from the web
 Wget is a network utility to retrieve files from the Web
```

- Ficheros que forman parte de un paquete

  ```
  dpkg -L paquete
  dpkg --listfiles paquete
  ```

  Ejemplo:

```
$ dpkg --listfiles wget
/.
/etc
/etc/wgetrc
/usr
/usr/bin
/usr/bin/wget
/usr/share
/usr/share/doc
/usr/share/doc/wget
```

- Identificar el paquete al que pertenece un fichero:

  ```
  dpkg -S fichero
  dpkg --search fichero
  ```

  Ejemplo:

```
$ dpkg --search /usr/bin/wget
wget: /usr/bin/wget
```

1.2.3. APT - Advanced Packaging Tools

APT es un sistema de gestión de paquetes creado por el proyecto Debian que simplifica en gran medida la instalación y eliminación de programas. Permite descargar e instalar paquetes desde una fuente local y/o remota, resolviendo automáticamente todas las dependencias, actualizaciones, etc.

Fichero de fuentes de apt

Es el fichero de configuración que contiene los distintos servidores desde los cuales se obtienen los paquetes.

- `/etc/apt/sources.list`

Ejemplo:

```
# See sources.list(5) for more information
deb ftp://ftp.rediris.es/debian/ stable main contrib non-free
deb http://security.debian.org/ stable/updates main contrib non-free
# Para descargar fuentes, a través de apt-get source
deb-src ftp://ftp.rediris.es/debian/ stable main
```

- formato de `sources.list`

  # Para binarios
  `deb` *uri distribución componentes*
  # Para ficheros fuente
  `deb-src` *uri distribución componentes*

- *distribución* puede ser:
  - `stable` (o el nombre que tenga la versión estable, actualmente, *trixie*), es la versión recomendada.
  - `testing` contiene versiones más recientes de paquetes, aunque todavía no probados exhaustivamente.
  - `unstable` (también denominada *sid*), es la versión en desarrollo, probablemente con errores sin corregir.
- *componentes* pueden ser (se admiten varios en la misma línea):
  - `main` - conjunto principal de paquetes
  - `contrib` - paquetes adicionales
  - `non-free` - paquetes que no son libres

La primer línea `deb` suele apuntar al servidor oficial o a un *mirror*, mientras que *security.debian.org* es el servidor de actualizaciones de seguridad. Estas últimas se tratan separadamente puesto que su importancia es máxima (muchas actualizaciones corrigen fallos o añaden nuevas características, por lo que no instalarlas no suele suponer mayores problemas, pero las actualizaciones de seguridad son cruciales).

Otras opciones de configuración de `APT`

Además del fichero `sources.list`, APT admite los ficheros de configuración siguientes:

1. Fichero `/etc/apt/apt.conf`

   Este fichero puede contener todo tipo de opciones de configuración. Este el formato clásico de los ficheros de configuración de Linux, en donde un solo fichero se utiliza para configurar todas las opciones o parámetros de un servicio o aplicación.

2. Ficheros en el directorio `/etc/apt/apt.conf.d`

   El directorio contiene diferentes ficheros, cada uno de los cuales configura una opción diferente. Este estilo de configuración se utiliza en servicios o aplicaciones complejos con muchas opciones o que son utilizados para diferentes tareas. En este caso es recomendable que cada conjunto de opciones se almacene en un fichero diferente. El formato del nombre de los ficheros consta de un número, de una etiqueta identificativa. Los ficheros se procesan en el orden indicado por los números que aparecen en el nombre del fichero.

   Por su parte, el nombre del directorio se suele construir a partir del nombre del fichero de configuración clásico, añadiéndole el sufijo `_d` o la extensión `.d`.

3. Fichero `/etc/apt/preferences` es específico para configurar las prioridades de los paquetes.

NOTA: Para la incorporación de repositorios externos puede ser necesario instalar previamente una clave.

Existen también comandos para instalar automáticamente repositorios. El comando `add-apt-repository`, típico de Ubuntu, puede utilizarse para añadir repositorios a *sources.list*. El formato es: `add-apt-repository ppa:user/ppa_name`, donde `ppa:user/ppa_name` es el PPA (personal package archive) que se quiere añadir.

Por ejemplo, para utilizar el repositorio *deb-multimedia.org*:

```
# nos traemos la clave manualmente (no partir la URL):
wget http://www.deb-multimedia.org/pool/main/d/deb-multimedia-keyring/
    deb-multimedia-keyring_2016.8.1_all.deb
dpkg --install deb-multimedia-keyring_2016.8.1_all.deb
# o bien nos traemos la clave en modo no seguro (solo por esta vez):
apt-get update -oAcquire::AllowInsecureRepositories=true
apt-get install deb-multimedia-keyring
```

Comandos y opciones de APT

Existen dos comandos básicos de APT y un tercero que combina las funcionalidades de los dos primeros:

- `apt-get` - se utiliza para hacer modificaciones (instala paquetes, reconstruye la base de datos de paquetes, etc).[3]
- `apt-cache` - da únicamente información (busca en una caché).
- `apt` - combina las funcionalidades de los dos primeros (admite la mayoría de las opciones de ambos) y presenta algunas modificaciones en la presentación de la información en pantalla.

La lista de opciones de los dos primeros comandos (que también se pueden usar con `apt`):

`apt-get`	`update`	actualiza la lista de paquetes
	`upgrade`, `dist-upgrade`	actualiza los paquetes
	`install`	instala el paquete
	`remove`, `purge`	desinstala el paquete
	`autoremove`, `clean`	limpia paquetes innecesarios
	`source`	descarga un fichero fuente
	`build-dep`	descarga dependencias de compilación
`apt-cache`	`search`	busca paquetes para instalar
	`show`	muestra información del paquete
	`depends`	lista las dependencias del paquete
	`policy`	muestra fuentes y prioridades
`apt-get -f install`		corrige los errores de dependencias
`/var/lib/dpkg/lock`		fichero de bloqueo (cerrojo)

[3] Apt-get tiene un huevo de pascua, también conocido como easter egg, muy famoso. Se trata de escribir `apt-get moo` en una línea de comandos.

Veamos estos comandos y opciones:

Comando `apt-get`

1. Actualizar la lista de paquetes

   ```
   apt-get update
   ```

2. Actualizar los paquetes

   ```
   apt-get upgrade
   apt-get dist-upgrade
   ```

   - debe hacerse un `apt-get update` antes de actualizar los paquetes
   - `upgrade` se limita a reemplazar un paquete por otro
   - `dist-upgrade` borra o instala nuevos paquetes si es necesario

3. Instalar un paquete

   `apt-get install` *nombre_paquete*

4. Eliminar paquetes

   `apt-get remove` *nombre_paquete* # borra solo el paquete
   `apt-get purge` *nombre_paquete* # borra el paquete y toda su configuración
   `apt-get autoremove` # elimina los paquetes que ya no se necesitan

5. Eliminar las copias de ficheros descargados.

   - Cuando se instala un paquete a través de `apt-get` se guarda una copia en `/var/cache/apt/archives/`.

   ```
   apt-get clean  # Elimina todos los ficheros descargados
   ```

6. Descargar ficheros fuente

   `apt-get source` *nombre_paquete*

   - la opción `--compile` compila el paquete después de descargarlo (y genera el `.deb`)

7. Descargar las dependencias para compilar un paquete

`apt-get build-dep` *nombre_paquete*

8. También puede usarse el comando `apt-build` para descargar, compilar e instalar un paquete a partir de las fuentes

Comando apt-cache

APT proporciona información de paquetes no instalados ni descargados, pues trae la lista (caché) completa de paquetes del servidor. El comando `apt-cache` proporciona las siguientes opciones:

1. `search`: busca cadenas en nombre de paquetes (el argumento puede ser una expresión regular). Ejemplo:

```
$ apt-cache search firefox
bookmarkbridge - tool to synchronize bookmarks between browsers
gtkcookie - Editor for cookie files
latex-xft-fonts - Xft-compatible versions of some LaTeX fonts
libflash-mozplugin - GPL Flash (SWF) Library - Mozilla plugin
iceweasel web browser based on Firefox - Trasitional package
...
```

2. `show`: muestra información del paquete
3. `depends`: muestras las dependencias del paquete
4. `policy`: muestra fuentes y prioridades (general o de un paquete)

Dependencias entre paquetes

Durante la instalación de paquetes con APT pueden presentarse las siguientes situaciones:

- El paquete A depende (*Depends*) del paquete B si B es absolutamente necesario para usar A
- El paquete A recomienda (*Recommends*) el paquete B si se considera que la mayoría de los usuarios no querrían A sin las funcionalidades que proporciona B
- El paquete A sugiere (*Suggests*) el paquete B si B está relacionado y mejora las funcionalidades de A
- El paquete A está en conflicto (*Conflicts*) con B en el caso de que A no funciona correctamente si B está instalado

Corrección de problemas

Dependencias

La opción `apt-get -f install` (o `--fix-broken`) intenta corregir los errores de dependencias. Por ejemplo:

```
The following packages have unmet dependencies:
arduino-core : Depends: gcc-avr but it is not going to be installed
avr-libc : Depends: gcc-avr (>= 1:4.3.4) but it is not going to
be installed
...
E: Unmet dependencies. Try 'apt-get -f install' with no packages
(or specify a solution).
```

Bloqueo

Por su parte, el fichero `/var/lib/dpkg/lock` funciona a mode cerrojo, de la siguiente forma:

- Solo se permite ejecutar una instancia de `apt-get` de cada vez.
- Al inicio de la ejecución del comando se creará el fichero.
- No se permitirá volver a ejecutar el comando mientras exista el fichero.
- Al acabar la ejecución, `apt-get` borrará el fichero.
- Además, a los usuarios no privilegiados que no puedan crear este fichero, no se les permitirá ejecutar la aplicación.

Puede por tanto, producirse el siguiente problema:

- Si durante la ejecución de `apt-get` el programa termina inesperadamente el fichero de bloqueo no se borrará.
- Cuando volvamos a intentar volver a ejecutar el comando, este no nos permitirá continuar. En estos casos el fichero de bloqueo puede borrarse manualmente.

NOTA: En algunas configuraciones o distribuciones como Ubuntu, la descarga e instalación de las actualizaciones de seguridad se realiza de forma automática (servicio `unattended-upgrades`), por lo que en ocasiones podemos encontrarnos que el uso de `apt-get` está bloqueado por el mecanismo indicado.

1.2.4. Instalación desde el código fuente

Los pasos para la instalar manualmente una aplicación desde el código fuente son los siguientes:

1. Descarga, normalmente se distribuyen en forma de *tarballs*:
   - `.tar` (empaquetado pero sin comprimir)
   - `.tar.gz` (empaquetado y comprimido *gzip*, abreviado `.tgz`)
   - `.tar.bz2` (empaquetado y comprimido *bzip2*, abreviado `.tbz`)
2. Desempaquetado: comando `tar` (*Tape ARchive format*)
   - `tar` - crea y extrae ficheros de un archivo
   - Opciones principales:
     - `-c` (Create). Crea un archivo `tar`
     - `-t` (lisT). Lista el contenido de un archivo
     - `-x` (eXtract). Extrae los ficheros de un archivo
   - Otras opciones
     - `-f` *tarball*. Necesaria para operar con el archivo *tarball*, pues si no, usaría la entrada/salida estándar (denotada "-")
     - `-v` (Verbose). Lista los ficheros según se van procesando
     - `-z` (gZip). Comprime/descomprime ficheros `gzip`
     - `-j` (bzip2). Comprime/descomprime ficheros `bzip2`
   - Ejemplos
     - Crea un tar.gz con los ficheros del directorio dir
       `$ tar czvf archivo.tar.gz dir`
     - Muestra el contenido de un tar.gz
       `$ tar tzvf archivo.tar.gz`
     - Extrae un fichero tar.bz2
       `$ tar xjvf archivo.tar.bz2`
3. Leer el fichero INSTALL, INSTALAR o similar
4. Configuración
   - El código fuente desarrollado con ayuda de las herramientas GNU (*autoconf*) contienen un script `configure`, que se encarga de:
     - chequear el entorno de compilación

- chequear las librerías necesarias
- generar los `Makefiles` que nos permitirán compilar el código

- Ejecución
  - `./configure` <*opciones*>
- Opciones:
  - `./configure --prefix=dir`
    instala el programa en `dir/bin`, `dir/lib`, etc, en vez de en el directorio por defecto (normalmente `/usr/local`)
  - Para ver opciones: `./configure --help`
- Ejemplo:
  - `./configure --prefix=/opt`
    instala la aplicación en `/opt/`

5. Compilación
   - El proceso de configuración genera ficheros `makefile` o `Makefile` en los directorios del código fuente
     - indican reglas (*rules*) que especifican como ejecutar ciertas tareas (*targets*) sobre el código: compilar, enlazar, crear páginas de manual, instalar
   - Funcionamiento:
     - `make` (ejecuta el *target* por defecto, normalmente todo, menos instalar)
     - `make all` (si no existe el *target* por defecto)
     - `make clean` (borra ficheros objetos, ejecutables, etc)
6. Instalación
   - Si la compilación terminó con éxito, simplemente
     - `make install` (instala el programa ejecutable, librerías, páginas de manual)
   - Todo el proceso de compilación la puede realizar cualquier usuario y la instalación también en el caso de que el usuario tenga permiso de escritura en el directorio destino (que se puede indicar al configurar con la opción *--prefix*)

Tipos de ejecutables:

1. Enlazados estáticamente (*statically linked*): son "completos"
2. Enlazados dinámicamente (*dynamically linked*): para ejecutarse necesitan librerías instaladas en el sistema
   - ocupan menos que los estáticos
   - librerías compartidas por varios programas

Enlazamiento dinámico

- El comando `ldd` nos permite ver las librerías que un ejecutable necesita. El formato de la salida es: `librería requerida =>librería encontrada`. Por ejemplo:

```
# ldd /bin/ls
     libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
     libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3
     ...
```

  Si no se encuentra una librería:

```
# ldd /bin/ls
     libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
     libpcre.so.3 => Not found
     ...
```

  En el caso de no encontrar una librería, puede deberse a dos problemas:

  1. La librería se encuentra en una localización no estándar. Esto puede resolverse indicando la ubicación de la librería.
  2. La librería que necesita el programa es de una versión diferente a la que tenemos en el ordenador. El problema puede intentar arreglarse haciendo un enlace simbólico (comando `ln`) a la librería existente con el nombre de la librería requerida. Si las versiones de las librerías son muy diferentes, esta solución probablemente no funcionará y será necesario instalar la nueva versión de la librería (pueden convivir sin problemas ambas versiones de librería en el mismo ordenador). Por ejemplo,

     `ln -s libgdal.so.30.0.1 libgdal.so.30.0.2`

- El cargador dinámico (ld - dynamic linker) se encarga de enlazar y cargar las librerías que necesitan los ejecutables.
  - El fichero de configuración `/etc/ld.so.conf` (y los ficheros a los cuales nos redirige) lista los directorios que contienen librerías:

```
# cat /etc/ld.so.conf
    include /etc/ld.so.conf.d/*.conf
# ls /etc/ld.so.conf.d/*.conf
    /etc/ld.so.conf.d/libc.conf
    /etc/ld.so.conf.d/x86_64-linux-gnu.conf
    /etc/ld.so.conf.d/libc.conf
    ...
# cat /etc/ld.so.conf.d/libc.conf
    # libc default configuration
    /usr/local/lib
```

  - En estos ficheros podemos incluir rutas a librerías que añadamos al sistema en localizaciones no estándar. A continuación debemos ejecutar el comando `ldconfig` para activar los cambios.

- En el espacio de usuario también podemos indicar nuevos directorios con librerías usando la variable de entorno `LD_LIBRARY_PATH`
  - Para mostrar en pantalla el contenido de esta variable:

```
echo $LD_LIBRARY_PATH
⟶ /usr/lib:/usr/local/lib:/usr/X11R6/lib
```

  - Podemos modificarla y exportar (hacer global) la modificación:

```
export LD_LIBRARY_PATH="dir1:dir2:$LD_LIBRARY_PATH"
echo $LD_LIBRARY_PATH
⟶ dir1:dir2:/usr/lib:/usr/local/lib:/usr/X11R6/lib
```

1.3. Automatización de tareas

En esta sección veremos la utilización de comandos que permiten automatizar tareas repetitivas

1. Tareas que se deben ejecutar en momentos concretos o de forma periódica:
   - `at`, `batch` permiten ejecutar trabajos a una hora específica o bajo determinadas condiciones
   - `cron` permite correr trabajos a intervalos regulares
2. Herramientas para automatizar la configuración de servidores

1.3.1. Tareas periódicas

Comando `at`

Permite indicar el momento en que se quiere ejecutar un trabajo

- Sintaxis:

  `at [`*`opciones`*`]` *TIME*

- Al ejecutar `at` pasamos a un nuevo prompt, que nos permite introducir comandos que se ejecutarán a la hora indicada
  - para salvar el trabajo y salir `CTRL-D`
  - al terminar, la salida estándar se envía como un mail al usuario (si está instalado un agente de correo)
  - el trabajo se ejecuta mientras el sistema esté encendido a la hora indicada
- Ejemplo:

```
$ at 11:45
warning: commands will be executed using /bin/sh
at> ls /tmp > lista
at> env DISPLAY=:0 zenity -info -text="Hola"
at> <EOT>
job 4 at Wed Nov 16 11:45:00 2005
```

- *TIME* puede especificarse de varias formas:

- `HH:MM` por ejemplo `12:54`
- `HH:MMAM/PM`, por ejemplo `1:35PM`
- `HH:MM MMDDYY`, por ejemplo `1:35PM 122505`
- `now + numero unidades`, donde `unidades` puede ser `minutes`, `hours`, `days`, o `weeks`

```
$ at now+2hours
```

- `today`, `tomorrow`, por ejemplo `12:44tomorrow`
- `midnight` (00:00), `noon` (12:00), `teatime` (16:00)

Comandos relacionados

- `atq` lista los trabajos pendientes del usuario
  - si es el superusuario, lista los trabajos de todos los usuarios
- `atrm` borra trabajos identificados por su número de trabajo
- `batch` ejecuta trabajos cuando la carga del sistema es baja
  - el trabajo empieza en cuanto la carga caiga por debajo de 1.5
  - la carga se obtiene del fichero `/proc/loadavg`

Procesos periódicos, `cron`

Para crear trabajos que se ejecuten periódicamente se utilizan el demonio `cron` y el comando `crontab`

- Sintaxis que permite editar, modificar o borrar un trabajo:

```
crontab [-u usuario] {-e|-l|-r}
```

- Opciones:
  - `-e` edita o crea nuevos trabajos
  - `-l` muestra los trabajos
  - `-r` borra los trabajos
  - `-u usuario` para operar como otro usuario (solo root)
- Sintaxis utilizando un fichero previamente escrito:

```
crontab [-u usuario] fichero
```

Los trabajos se especifican en un fichero que puede tener tres tipos de líneas:

- Comentarios, que empiezan por `#`
- Definición de variables (si es necesario), de tipo *`nombre = valor`*

```
# shell usada para ejecutar los comandos
SHELL=/bin/bash
# Usuario al que se envía (por mail) la salida del comando
# (por defecto, se envían al propietario del fichero)
MAILTO=pepe
```

- Especificación del trabajo y hora de ejecución, de la siguiente forma:

  `minuto hora día mes día_semana comando`

  - el día de la semana de 0 a 7 (0 ó 7 domingo)
  - `*` indica cualquier valor
  - se pueden indicar rangos, listas o repeticiones:
    - `1-5` para indicar de lunes a viernes
    - `0,15,30,45` para indicar cada 15 minutos
    - `0-23/2` en el campo hora indica realizar cada dos horas en todo el rango de horas (0, 2, 4, etc.)

- Ejemplos:

  - Borra el `/tmp` todos los días laborables a las 4:30 am

    ```
    30 4 * * 1-5 rm -rf /tmp/*
    ```

  - Escribe la hora, cada 15 minutos, durante la noche:

```
0,15,30,45 0-8,20-23 * * * echo Hora: $(date) >> /tmp/horas
```

Además, el administrador puede crear scripts que se ejecuten con periodicidad horaria, diaria, semanal y mensual

- sólo tiene que colocar esos scripts en los directorios

```
/etc/cron.hourly
/etc/cron.daily
/etc/cron.weekly
/etc/cron.monthly
```

- estos ficheros suelen ser de mantenimiento del sistema
- la ejecución de estos scripts se controla en el fichero `/etc/crontab` (entre otras cosas se indica la fecha y hora concreta, que es común)
- estos scripts también pueden ejecutarse con `anacron` si está instalado (véase a continuación)

Cron está pensado para sistemas funcionando 24/7

- Si el sistema está apagado a la hora de una acción cron, esa iteración no se realiza
- Problema en sistemas de sobremesa y/o domésticos

Solución complementaria: **Anacron**

Anacron

Ejecuta asíncronamente tareas periódicas programadas

- Al iniciarse el sistema comprueba si hay tareas periodicas pendientes (que no se realizaron por estar el sistema apagado)
- De ser así, espera un cierto retardo y las ejecuta

Fichero de configuración: `/etc/anacrontab`

`período retardo id_del_trabajo comando`

- El período se especifica en días (o como `@monthly`) y el retardo en minutos:

```
1        5   cron.daily   run-parts /etc/cron.daily
7        10  cron.weekly  run-parts /etc/cron.weekly
@monthly 15  cron.monthly run-parts /etc/cron.monthly
```

- En este ejemplo, cada día, los scripts en `cron.daily` se ejecutan 5 minutos después de iniciar el anacron, cada 7 días (10 minutos de espera), se ejecutan los scripts en `cron.weekly` y cada mes (15 minutos) los de `cron.monthly`

Limitaciones de anacron

- Solo para uso del administrador
- Solo permite períodos de días (no vale para tareas que se deban ejecutar varias veces al día)

1.3.2. Automatización de la configuración

Comandos de sincronización de ficheros y clonado:

`rsync` Comando de sincronización clásico, rápido y versátil

- sólo copia los ficheros modificados, preservando el propietario, grupo, modo y fechas de modificación
- funciona sobre `ssh` y puede utilizarlo cualquier usuario
- ejemplo:

```
# rsync -av /home/tomas maquina1:/tmp
```

`pdsh` Envía comandos a un grupo de hosts en paralelo

- permite copias en paralelo con los comandos `pdcp` (copia de uno a muchos) y `rpdcp` (copia de muchos a uno)
- ejemplo, copia /etc/hosts a los hosts foo0, foo4 y foo5:

```
# pdcp -w foo[0-5] -x foo[1-3] /etc/hosts /etc
```

`unison` Aplicación para sincronizar ficheros y directorios entre sistemas

- puede sincronizar entre sistemas Windows y UNIX
- no requiere permisos de root
- permite sincronización en los dos sentidos

Imágenes del sistema

Herramientas que nos permiten obtener imágenes completas del sistema para sincronización de ficheros o réplicas (clones)

partimage salva particiones completas a un fichero de imagen

- permite recuperar la partición completa en caso de errores
- permite realizar clones de un PC

clonezilla aplicación opensource para hacer clones masivos

- permite hacer clones de múltiples PCs (40 o más) simultáneamente
- puede usar multicast para distribuir las imágenes

systemimager herramienta de instalación y distribución de software

- permite automatizar la instalación de Linux y la distribución de software en una red de PCs (usado en clusters, granjas de servidores o redes en general)

1.4. Copias de seguridad

Realizar copias de seguridad es una de las tareas más importantes del administrador del sistema

- Es casi inevitable que se produzcan pérdidas de información, debido a, entre otras causas:
  - deterioro o borrado accidental por parte de un usuario autorizado
  - ataque intencionado por parte de personas no autorizadas
  - fallo software o hardware
  - incendio, robos, desastres naturales, etc.
- es imprescindible poder recuperar la información perdida

En esta sección veremos los comandos básicos para realizar copias de seguridad en UNIX/Linux.

Componentes de las copias de seguridad

Hay básicamente tres componentes que intervienen en una copia de seguridad:

- Los medios de almacenamiento: cintas, discos, etc.
- El programa de copia: los comandos que mueven los datos de los discos a los medios
- El planificador: decide que información se copia y cuando

Medios de almacenamiento: Dispositivos donde se guarda la información; los más populares son:

- Cintas, por ejemplo cartuchos LTO (Linear Tape Open) de varios TB (p.e. 18 TB sin comprimir en LTO-9)
- Silos robotizados (Tape library) para grandes infraestructuras (varios exabytes = millones de terabytes)
- Discos duros externos, pocos TB por unidad
- Discos ópticos (DVDs, Blue-Ray), hasta 128 GB por disco
- Backup en la nube: capacidad “ilimitada”, problemas de tiempo de acceso y problemas legales

Programa de copia: Se encarga de copiar los ficheros seleccionados en el medio de almacenamiento; dos mecanismos básicos

- Basado en imagen: accede al disco a bajo nivel
  - normalmente copias más rápidas, pero mas lento restaurar ficheros individuales
  - programas específicos para diferentes filesystems
  - comandos de este tipo son `dump` y `dd`
- Fichero a fichero
  - acceden a los ficheros a través de llamadas al SO
  - copias más lentas, pero restauración de ficheros individuales más simple
  - un comando de este tipo es `tar`

El planificador: Decide cuándo realizar el backup (mediante cron o similar) y cuánta información copiar

Tipos de backup:

Completo se salva toda la información. Se le asigna el nivel 0

Diferencial se salvan los ficheros modificados desde el último backup completo

- los backups son más grandes que en el caso incremental
- para restaurar sólo necesitamos el backup completo y el último diferencial
- se les asignan niveles con números mayores que 0, por ejemplo, una secuencia de backups podría ser 0, 5, 5, 5

Incremental sólo se salvan los ficheros modificados desde el último backup completo o incremental

- la copia de seguridad necesita menos tiempo y espacio
- para restaurar los datos necesitaremos el último backup completo y todos los incrementales
- por ejemplo, una secuencia de backups podría ser 0, 2, 4, 6, 8

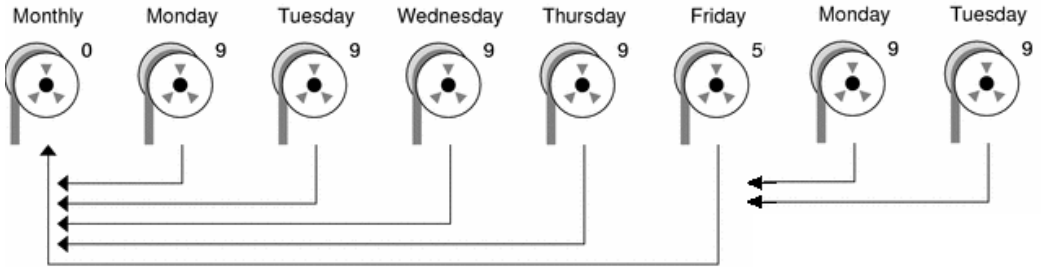
En general, en una secuencia el nivel j contiene solo modificaciones con respecto al previo más cercano i que sea menor ($i < j$). Por ejemplo, en la secuencia 0, 2, 4, 3, el backup 3 contiene modificaciones con respecto al 2.

Planificación de los backups

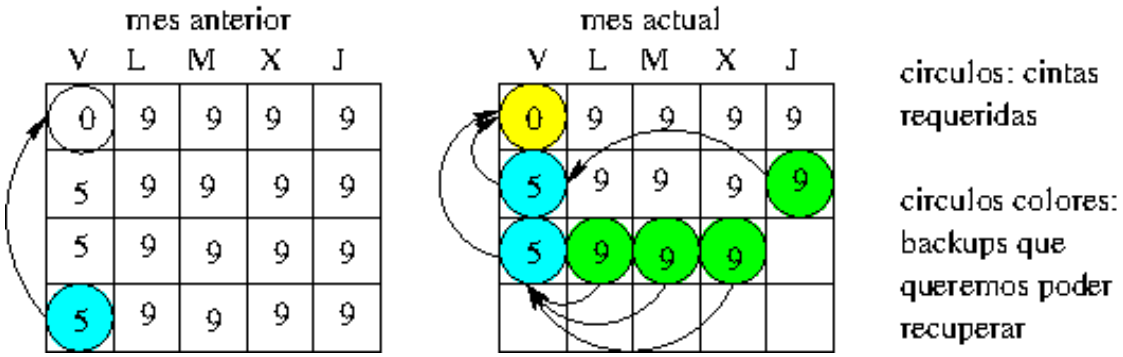
Podemos seguir diferentes estrategias a la hora de planificar los backups. A modo de ejemplo, supongamos que:

- Los backups se hacen de lunes a viernes y supondremos meses de 4 semanas (28 días por mes).
- Queremos ser capaces de recuperar cualquier versión diaria de la última semana (de los últimos 5 días laborables) y cualquier versión semanal del último mes.

Ejemplo 1: backup mensual de nivel 0, semanal de nivel 5 y diario de nivel 9, siendo los dos últimos diferenciales (los diarios diferenciales con respecto al último semanal y los semanales diferenciales con respecto al mensual).



- En este ejemplo se podrían haber usado otros números en el rango 1-9 para producir los mismos resultados. La clave es usar el mismo número cuatro días a la semana, con cualquier número menor el día restante (para el backup semanal). Por ejemplo, se podría haber especificado los niveles {4, 4, 4, 4, 2} o {7, 7, 7, 7, 5}. Además, el día de la semana en el cual se hace el backup semanal es irrelevante.
- Este esquema necesita de nueve cintas: dos para el nivel 0, tres para el nivel 5 y cuatro para el nivel 9. Un caso que requiere todas las cintas se muestra a continuación:

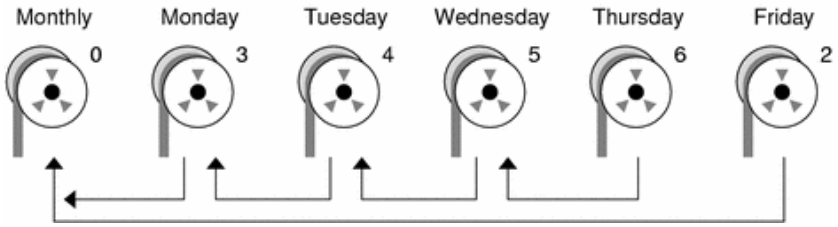


Se necesitan dos cintas de nivel 0, puesto que si solo tuviéramos una, cada vez que hagamos un backup completo perderíamos toda la información de los backups anteriores.

Como las cintas de niveles 5 y 9 contienen información diferencial con respecto al backup de nivel previo (0 y 5, respectivamente), no dependen de ninguna otra cinta de su mismo nivel y por tanto no es necesario duplicarlas.

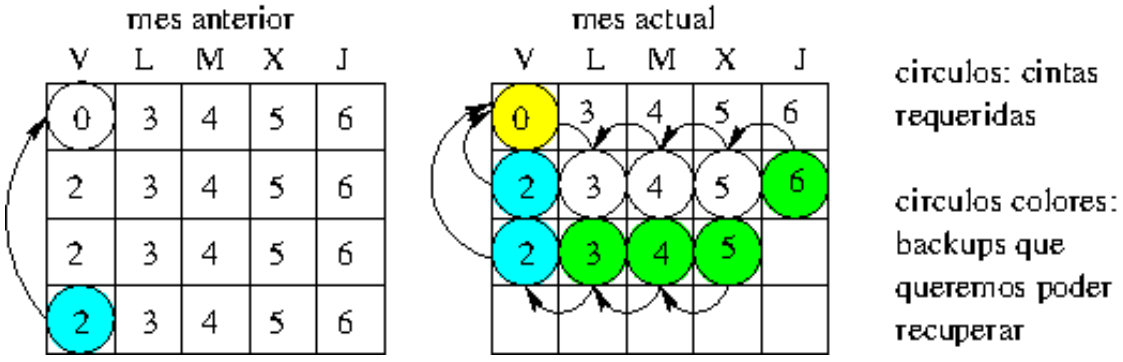
- Para restaurar necesitamos las últimas cintas de nivel 0, 5 y 9.

Ejemplo 2: backup mensual de nivel 0, semanal diferencial de nivel 2 e incrementales diarios de niveles 3, 4, 5 y 6.



- En este ejemplo se podría haber usado la secuencia {6, 7, 8, 9 seguida de 2}, o {5, 6, 7, 8 seguido de 3}. Los números no tienen un significado propio, si no que lo obtienen al ordenarlos en una secuencia especificada, como se describe en los ejemplos.

- Necesita al menos 12 cintas, dos de nivel 0, tres de nivel 2 y siete para los niveles diarios. Un caso que requiere de todas las cintas se muestra a continuación:



- para restaurar necesitamos en orden: las últimas cintas de niveles 0 y 2 y las diarias desde el viernes anterior de forma consecutiva

1.4.1. Comandos básicos

Veremos los comandos básicos para hacer backups en Linux/UNIX

Comando `dump`: Hace copias de un sistema de ficheros (no de un directorio), con las siguientes características:

- Puede salvar todo tipo de ficheros (incluidos ficheros de dispositivo)
- Los permisos, propietarios y fechas de modificación son preservados
- Puede realizar copias incrementales
- Pueden ser copias multivolumen (de varias cintas)

El formato en general es:

`dump [-nivel] [`*`opciones`*`] [`*`ficheros_a_salvar`*`]`

- Nivel de backup, un entero entre 0-9:
  - 0 implica backup completo
  - mayor que 0 implica copiar sólo los ficheros nuevos o modificados desde el último backup de nivel inferior
  - la información sobre los backups realizados se guarda en el fichero `/var/lib/dumpdates` si se especifica con la opción `-u`, lo cual es casi siempre necesario para restaurarlos
- Algunas opciones:
  - `-f` especifica el dispositivo o fichero donde salvar la copia
  - `-u` actualiza el fichero `/var/lib/dumpdates` después de una copia correcta
  - `-z`, `-j` usa compresión con `gzip` o `bzip2`, respectivamente
- Ejemplo: backup de nivel 0 en la cinta */dev/st0* de la partición `/home`

```
# dump -0 -u -f /dev/st0 /home
```

- Ejemplo: backup en una máquina remota usando `ssh` como transporte

```
# export RSH=ssh
# dump -0u -f sistema_remoto:/dev/st0 /home
```

Comando `restore`: Restaura ficheros salvados por `dump`

- Formato:

  `restore` *acción* `[`*opciones*`] [ficheros_a_recuperar]`

- Acciones principales:
  - `r` restaura la copia completa
  - `t` muestra los contenidos de la copia
  - `x` extrae sólo los ficheros indicados
  - `i` modo interactivo
    - permite ver los ficheros de la copia
    - con `add` indicamos los ficheros a extraer y con `extract` los extraemos
    - usar `?` para ayuda
- Algunas opciones:
  - `-f` especifica el dispositivo o fichero de la copia
  - `-a` no pregunta de que volumen extraer los ficheros (lee todos los volúmenes empezando en 1)
- Ejemplo: restaurar el backup de `/dev/st0`

  ```
  # restore -rf /dev/st0
  ```

- Ejemplo: restaurar el backup desde un sistema remoto

  ```
  # export RSH=ssh
  # restore -rf sistema_remoto:/dev/st0
  ```

- Ejemplo: restaurar sólo un fichero

  ```
  # restore -xaf /dev/st0 fichero
  ```

Fichero `restoresymtable`: Se crea cuando se restaura un filesystem completo, en el directorio donde se restaura

- Contiene información sobre el sistema restaurado
- Puede eliminarse una vez finalizada la restauración

Comando `tar` (Tape ARchiver)

Permite empaquetar varios ficheros en uno sólo, manteniendo la estructura de directorios (ya lo hemos tratado en un apartado anterior)

Comando `dd`

Comando de copia y conversión basado en imagen

- Sintaxis.

  `dd [if=`*`fichero_entrada`*`] [of=`*`fichero_salida`*`] [opciones]`

- Ejemplo de copia de una partición de disco a fichero

```
dd if=/dev/sda3 of=/tmp/fichero.imagen
```

- Ejemplo de copia de un cierto número de bytes

```
# bs indica el tamaño del bloque y count el número de bloques
# (en total se copian bs*count bytes)
$ dd if=/dev/urandom of=fichero.random bs=1024 count=4
```

- Ejemplo ignorando un error de lectura (opción `noerror`)

```
# Crea una copia de imagen de una partición
$ dd if=/dev/sda5 of=disco.img
# Monta la imagen en un directorio para acceder a los datos
$ mount -o loop disco.img /mnt/disco
# Extrae los datos de una cinta con error
$ dd conv=noerror if=/dev/st0 of=/tmp/bad.tape.img
```

Comando `mt`

Permite la manipulación directa de la unidad de cinta

- Sintaxis.

  `mt [-f` *`unidad_de_cinta`*`]` *`operación`* `[`*`número`*`]`

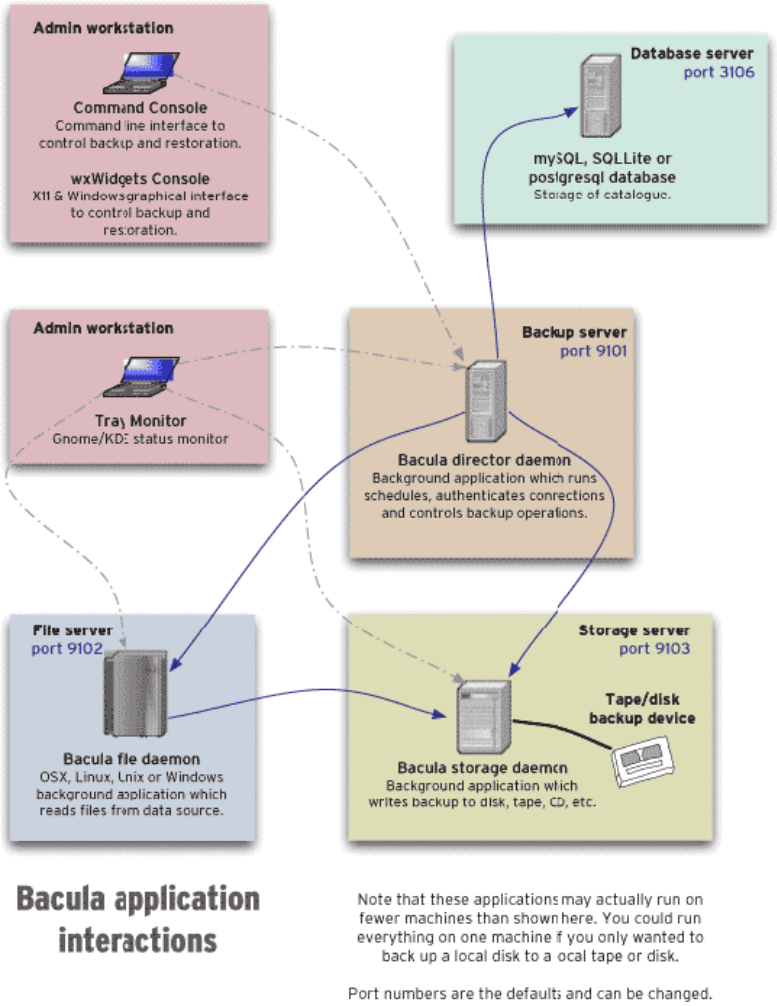
- Incluye comandos para rebobinar la cinta, alisar y dar tensión a la cinta, borrarla completamente, avanzar/retroceder un cierto número de ficheros, saltar al final de la parte grabada, etc.

1.4.2. Otras aplicaciones de backups, sincronización y clonado

Existen otras aplicaciones y/o comandos que permiten hacer backups:

Bacula

- Sofisticado sistema de backup en red con diseño modular
- Permite hacer copias de seguridad de todas las máquinas de una LAN a diferentes medios de backups (cinta, disco,...)
- Soporta MySQL, PostgreSQL o SQLlite para el catálogo
- Hace backups de sistemas UNIX, Linux y Windows
- Para más información, ver www.bacula.org/documentation o la sección 10.8 del libro *UNIX and Linux System Administration Handbook*, Evi Nemeth et al, 5ª ed.



Amanda

`Amanda`: *Advanced Maryland Automatic Network Disk Archiver*

- Sofisticado sistema de backup en red
- Permite hacer copias de seguridad de todas las máquinas de una LAN a una unidad de cinta en un servidor
- Está disponible en la mayoría de los UNIX y soporta muchos tipos de medios de backup
- Puede hacer uso de SAMBA para copias de sistemas Windows NT
- Se basa en `dump` y `tar`
- Para más información, ver wiki.zmanda.com

Flexbackup

Flexbackup Herramienta de backup flexible para instalaciones de pequeño y medio tamaño

- Más simple de configurar y utilizar que `Amanda` para sitios con un número no muy alto de sistemas
- Usa distintos formatos de archivo: `dump`, `afio`, `GNU tar`, `cpio`, `zip`, etc.
- Permite backups completos e incrementales, como `dump`
- Permite backups remotos a través de `rsh` o `ssh`
- Para más información, ver flexbackup.sourceforge.net

rdiff-backup

rdiff-backup copia un directorio en otro, permitiendo copias remotas

- Hace una copia exacta de los directorios (*mirror*), guardando las propiedades de los ficheros (propietario, permisos, etc.)
- Guarda las diferencias entre copias de los ficheros para poder recuperar un fichero antiguo (*incremental*)
- Sólo transmite las diferencias de los ficheros (similar a `rsync`)
- Para más información ver www.nongnu.org/rdiff-backup

DAR

DAR *Disk ARchiver* comando para hacer backups de árboles de directorios y ficheros

- Permite copiar un filesystem entero a un archivo
- Permite hacer backups completos y diferenciales
- Permite hacer copias multivolumen:
  - divide en archivo en varios ficheros (*slices*) parando antes de crear cada nuevo *slice*
  - interesante para hacer copias CD o DVD
- Más información en: dar.linux.free.fr

BackupPC

BackupPC solución de alto rendimiento para backups de sistemas GNU/Linux, WinXX y MacOSX PCs a un servidor o NAS

- No necesita software en el cliente
- Obtiene los backups mediante SAMBA, tar sobre ssh/rsh/nfs o rsync
- Compresión opcional
- Interfaz web para el administrador
- Más información en: backuppc.sourceforge.net/info.html

UrBackup

UrBackup sistema cliente/servidor para GNU/Linux y/o Windows

- Backups completos o incrementales
- Salva particiones completas o directorios
- Configurable desde el servidor o los clientes
- Interfaz web para el administrador
- Más información en: http://www.urbackup.org/documentation.html

Capítulo 2

Programación de scripts

2.1. Línea de comandos

Veremos conceptos básicos para usar nuestro sistema desde la línea de comandos

2.1.1. El interprete de comandos (shell)

El shell se inicia cuando accedemos a nuestra cuenta y proporciona:

- un intérprete de comandos
- un entorno de programación

Para salir del shell o volver al shell anterior: `exit` o `Ctrl-D`

Comandos externos o internos al shell

- Comandos externos
  - son programas ajenos al shell
  - cuando se lanzan inician un nuevo proceso
  - se buscan en los directorios indicados en la variable PATH
  - por ejemplo: `ls`, `cat`, `mkdir`, etc
- Comandos internos (*builtin commands*)
  - se ejecutan en el mismo proceso del shell, sin lanzar un nuevo proceso
  - por ejemplo: `alias`, `cd`, `pwd`, `eval`, `exec`, `bg`, etc.

  - ver el manual del shell para más información, en el caso del shell bash: `man bash-builtins`

- Para saber si un comando es externo o interno en bash usar el comando interno `type`:

```
$ type cd
cd is a shell builtin
$ type cat
cat is /bin/cat
```

Comandos y parámetros

- Un comando consta del nombre del comando y de cero, uno o más parémtros o argumentos.
- Para que interprete los espacios como parte de un parámetro usar comillas simples o dobles:

```
$ echo 'hola            amigo'
```

  - Comando $\longrightarrow$ `echo`
  - Argumento 1 $\longrightarrow$ `hola            amigo`

Gestión de comandos

- `su`, `sudo` - permiten ejecutar comandos con la identidad de otro usuario o como administrador
- `alias` - Permiten crear alias de comandos complejos (para eliminarlos `unalias`)

```
$ alias l='ls -la'
```

- `history` - muestra una lista con los últimos comandos ejecutados y permite reejecutarlos

Manejo del historial de comandos

Nos permiten recuperar un comando tecleado anteriormente, realizar búsquedas, modificaciones, etc.

Comando	Descripción
`<up-arrow>/<down-arrow>`	Comando anterior/posterior
`!!`	Último comando ejecutado
`!`*n*	*n*-ésimo comando del historial
`!-`*n*	*n* comandos hacia atrás
`!`*cadena*	Último comando ejecutado que empieza por cadena
`!?`*cadena*	Último comando ejecutado que contiene cadena
`^`*cadena1*`^`*cadena2*	Ejecuta el último comando cambiando *cadena1* por *cadena2*
`Ctrl-r`	Busca hacia atrás en el historial
`fc`	Permite ver, editar y reejecutar comandos del historial

2.1.2. Variables de shell

Uso de variables:

- control del entorno (*environment control*)
- programación shell

Dos tipos

- variables locales: visibles sólo desde el shell actual
- variables globales, de entorno o exportadas: visibles en otros shells

Para ver las variables definidas:

- Para ver todas las variables definidas en nuestra shell usar `set`
- Para ver las variables de entorno definidas usar `env` o `printenv`

El nombre de las variables debe:

- empezar por una letra o _
- seguida por cero o mas letras, números o _ (sin espacios en blanco)

Uso de las variables

- Asignar un valor: *nombre_variable=valor*

```
$ un_numero=15
$ nombre="Pepe Pota"
```

- Acceder a las variables: *$nombre_variable* o *${nombre_variable}*

```
$ echo $nombre
Pepe Pota
```

- Número de caracteres de una variable

```
$ echo ${#un_numero}
2
```

- Eliminar una variable: *unset nombre_variable*

```
$ unset nombre
$ echo ${nombre}mo
mo
```

- Variables de solo lectura: *readonly nombre_variable*

```
$ readonly nombre
$ unset nombre
bash: unset: nombre: cannot unset: readonly variable
```

Variables globales, de entorno o exportadas

- Cada shell se ejecuta en un *entorno* (*environment*)
- El entorno de ejecución especifica aspectos del funcionamiento del shell a través de la definición de variables de entorno (=globales=exportadas)
- Algunas variables de entorno predefinidas son:

Nombre	Propósito
HOME	directorio base del usuario
SHELL	shell por defecto
USER	el nombre de usuario
PS1/PS2	*prompts* primario y secundario
PWD	el directorio actual
PATH	el *path* para los ejecutables
LD_LIBRARY_PATH	el *path* para las librerías dinámicas
MANPATH	el *path* para las páginas de manual
LANG	aspectos de localización geográfica e idioma
LC_*	aspectos particulares de loc. geográfica e idioma

- Para definir una nueva variable de entorno: `export`

```
$ a=1 ; b=2 ; c=3 ; d=4    # Define cuatro variables de shell
$ export d                 # Exporta la variable d
$ cat script.sh            # Ejemplo de script
  #!/bin/bash              # Dos variables serán parametros,
  echo "$1, $2, $c, $d"    # c será local y d será global
$ ./script.sh $a $b        # Ejecutamos el script
   --> 1, 2, , 4           # no tiene acceso a la variable local
```

- La variable exportada será visible en los shell hijos (y para los scripts y procesos) que se creen a continuación
  - el shell hijo crea una copia local de la variable y la usa
  - las modificaciones de una copia no afectan a los demás shell

2.1.3. Expansiones del shell

Expansión de parámetros

Es la sustitución de las variables por su valor

```
$ A=Pepe
$ echo $A
Pepe
```

Expansión de nombres de ficheros

Es la sustitución de los *comodines* (*wildcards*), que nos permiten especificar múltiples ficheros al mismo tiempo. También se conoce como *glob*.

Lista de comodines:

Carácter	Corresponde a
`*`	0 o más caracteres
`?`	1 carácter
`[ ]`	uno de los caracteres entre corchetes
`[! ]` o `[^ ]`	cualquier carácter que no esté entre corchetes

`$ ls -l *html` # Lista los ficheros del directorio actual con terminación html

`$ ls -l [abd].html` # Implica `a.html`, `b.html` y `d.html`.[1]

- podemos ver como se hace la expansión
  - `set -x` muestra los detalles
  - `set +x` para no ver detalles
- podemos desactivar la expansión con `set -f`

Los ficheros "ocultos" (que empiezan por `.`) no se expanden

- debemos poner el `.` de forma explícita

Para referirnos a mayúsculas o minúsculas podemos usar los patrones:

- `[[:lower:]]`: corresponde a un carácter en minúsculas
- `[[:upper:]]`: corresponde a un carácter en minúsculas
- `[[:alpha:]]`: corresponde a un carácter alfabético
- `[[:digit:]]`: corresponde a un número

Para más detalles: `man 7 glob`

[1] Nota importante: en **bash** el comportamiento de los rangos depende de la configuración de nuestro sistema, en particular, de la definición de la variable `LC_COLLATE`. Si `LC_COLLATE=C`, `[L-N]` implica `LMN` y `[l-n]` implica `lmn`. En otro caso (p.e. si `LC_COLLATE="es_ES.UTF-8"` o `"gl_ES@euro"`) entonces `[L-N]` implica `LmMnN` y `[l-n]` implica `lLmMn`.

Expansión de comandos

Permite que la salida de un comando reemplace el propio comando
Formato:

`$(comando)` o `` `comando` ``

Ejemplos:

```
$ echo date
date
$ echo `date`
Xov Xul 21 13:09:39 CEST 2005
$ echo líneas en fichero=$(wc -l fichero)
```

`wc -l` cuenta el número de líneas en el fichero; el comando se ejecuta y su salida se pasa al `echo`

Expansión de llaves

Permite generar strings arbitrarios

- no tiene para nada en cuenta los ficheros existentes en el directorio actual

```
$ echo a{d,c,b}e
ade ace abe
```

Expansión de la tilde

Expande la tilde como directorio HOME del usuario indicado

- si no se indica usuario, usa el usuario actual

```
cd ~              # Accedemos al nuestro HOME
cd ~root          # Accedemos al HOME de root
ls ~pepe/cosas/   # Vemos el contenido del directorio
                  cosas de pepe
```

Expansión aritmética

Permite evaluar expresiones aritméticas enteras

- se usa `$(( expresión ))` o `$[ expresión ]`
- `expresión` tiene una sintaxis similar a la del lenguaje C

  - permite operadores como `++`, `+=`, `&&`,...

- También se puede usar `let` (usar comillas dobles si hay espacios)

```
$ let numero=(numero+1)/2
```

- Ejemplos:

```
$ echo $(((4+11)/3))                          --> 5
$ numero=15 ; echo $((numero+3))              --> 18
$ echo $numero                                --> 15
$ echo $((numero+=4))                         --> 19
$ echo $numero                                --> 19
$ numero=$(((numero+1)/2)) ; echo $numero     --> 10
```

Eliminación del significado especial

- bash permite eliminar el significado de los caracteres especiales
- para ello se usan las comillas simples, dobles o \
- El caracter o caracteres que vengan a continuación simplemente se escriben

Carácter	Acción
'	el shell ignora todos los caracteres especiales contenidos entre un par de comillas simples
"	el shell ignora todos los caracteres especiales entre comillas dobles excepto $, ` y \
\	el shell ignora el carácter especial que sigue a \

Ejemplos:

`ls "/usr/bin/a*"` (el nombre del fichero contiene un asterisco)
`echo '$PATH'` (escribe literalmente $PATH)
`echo "$PATH"` (escribe el contenido de la variable)
`echo I\'m Pepe` (escribe literalmente la comilla)

2.1.4. Redirección de la entrada/salida

Es posible cambiar la fuente de la entrada o el destino de la salida de los comandos

- toda la E/S se hace a través de ficheros

- cada proceso tiene asociados 3 ficheros para la E/S

Nombre	Descriptor de fichero	Destino por defecto
entrada estándar (*stdin*)	0	teclado
salida estándar (*stdout*)	1	pantalla
error estándar (*stderr*)	2	pantalla

  - por defecto, un proceso toma su entrada de la entrada estándar, envía su salida a la salida estándar y los mensajes de error a la salida de error estándar

Ejemplo

```
$ ls /bin/bash /kaka
ls: /kaka: Non hai tal ficheiro ou directorio # Error
/bin/bash           # Salida estándar
$
```

Para cambiar la entrada/salida se usan los siguientes caracteres:

Carácter	**Resultado**
`comando < fichero`	Toma la entrada de `fichero`
`comando > fichero`	Envía la salida de `comando` a `fichero`; sobreescribe cualquier cosa de `fichero`
`comando >> fichero`	Añade la salida de `comando` al final de `fichero`
`comando << etiqueta`	Toma la entrada para `comando` de las siguientes lineas, hasta una línea que tiene sólo `etiqueta`
`comando 2> fichero`	Envía la salida de error de `comando` a `fichero` (el 2 puede ser reemplazado por otro descriptor de fichero)
`comando 2>&1`	Envía la salida de error a la salida estándar (el 1 y el 2 pueden ser reemplazado por otro descriptor de fichero, p.e. `1>&2`)
`comando &> fichero`	Envía la salida estándar y de error a `fichero`
`comando1 | comando2`	pasa la salida de `comando1` a la entrada de `comando2` (*pipe*)

Ejemplos:

- `ls -l > lista.ficheros`
  Crea el fichero `lista.ficheros` conteniendo la salida de `ls -l`
- `ls -l /etc >> lista.ficheros`
  Añade a `lista.ficheros` el contenido del directorio `/etc`

- `cat < lista.ficheros | more`
  Muestra el contenido de `lista.ficheros` página a página (equivale a `more lista.ficheros`)
- `ls /kaka 2> /dev/null`
  Envía los mensajes de error al dispositivo nulo (a la *basura*)
- `> kk`
  Crea el fichero `kk` vacío
- `cat > entrada`
  Lee información del teclado, hasta que se teclea `Ctrl-D`; copia todo al fichero entrada
- `cat << END > entrada`
  Lee información del teclado, hasta que se introduce una línea con `END`; copia todo al fichero `entrada`
- `ls -l /tmp /kaka > salida 2> error`
  Redirige la salida estándar al fichero `salida` y la salida de error al fichero `error`
- `ls -l /tmp /kaka > salida.y.error 2>&1`
  Redirige la salida estándar y de error al fichero `salida.y.error`; el orden es importante:

  ```
  ls -l /tmp /kaka 2>&1 > salida.y.error
  ```

  no funciona, ¿por qué?
- `ls -l /tmp /kaka &> salida.y.error`
  Igual que el anterior
- `cat /etc/passwd > /dev/tty2`
  Muestra el contenido de `/etc/passwd` en el terminal `tty2`
  - usar el comando `tty` para ver el nombre del terminal en el que estamos

Comandos útiles con pipes y redirecciones

1. `tee`
   - copia la entrada estándar a la salida estándar y también al fichero indicado como argumento

- `ls -l | tee lista.ficheros | less`

  hace dos cosas: almacena la salida de `ls -l` en `lista.ficheros` y la muestra en pantalla página a página

- La opción `-a` permite añadir la salida a un fichero existente.

2. **`xargs`**

   - permite pasar un elevado número de argumentos a otros comandos
   - lee la entrada estándar, y ejecuta el comando uno o más veces, tomando como argumentos la entrada estándar (ignorando líneas en blanco). Ejemplos:
   - `$ locate README | xargs cat`

     El comando `locate` busca los ficheros `README` en el ordenador y mediante `xargs` los ficheros se envían a `cat` que muestra su contenido. Equivale a:
     `cat /var/lib/aspell/README /usr/share/tool/README.txt ...`
   - `$ locate README | xargs -I{} cp {} /tmp/`

     Copia los `README` en el directorio `/tmp`; la opción `-I` permite que {} sea reemplazado por los nombres de los ficheros. Equivale a:
     `cp /var/lib/aspell/README /usr/share/tool/README.txt ... /tmp/`

3. **`exec`**

   - ejecuta un programa reemplazando el shell actual con el programa (es decir, al programa se le asigna el PID del shell, dejando el shell de existir)

     ```
     $ echo $$ # donde $$ indica el PID del shell actual
     4946
     $ exec sleep 20
     ```
     En otro terminal, ejecutamos
     ```
     $ ps a | grep 4946
     4946 pts/13   Ss+   0:00 sleep 20
     ```
   - si no se especifica el programa, exec puede usarse para redireccionar las entradas y salidas de todos los comandos:

     `$ exec > /tmp/salida` : envía toda la salida estándar al fichero

     `$ exec < /tmp/entrada` : toma el fichero como entrada estándar

2.1.5. Orden de evaluación

Desde que introducimos un comando hasta que se ejecuta, el shell ejecuta los siguientes pasos, y en el siguiente orden:

1. Redirección E/S
2. Sustitución (expansión) de variables: reemplaza cada variable por su valor
3. Sustitución (expansión) de nombres de ficheros: sustituye los comodines por los nombres de ficheros

Si no se tiene en cuenta ese orden, pueden aparecer problemas:

```
$ star=\* ; pipe=\|
$ ls -d $star
cuatro  dos  tres  uno
$ cat uno $pipe more
cat: |: Non hai tal ficheiro ou directorio
cat: more: Non hai tal ficheiro ou directorio
```

En el segundo caso, primero se mira si hay redirección E/S y no hay; se sustituye `$pipe` por su valor y finalmente se miran los comodines (que no hay) así `$pipe` se toma por el carácter "`|`", no por la redirección

Comando `eval`

Evalúa la línea de comandos 2 veces:

- la primera hace todas las substituciones
- la segunda ejecuta el comando

Ejemplo:

```
$ pipe=\|
$ eval cat uno $pipe more
Este es el fichero uno
...
$
```

- En la primera pasada reemplaza `$pipe` por "`|`"
- En la segunda ejecuta el comando `cat uno | more`

2.1.6. Ficheros de inicialización de bash

Se distinguen tres formas de iniciar bash: dependiendo de quién lo invoque, si el usuario (*shell interactivo*) o el sistema (*shell no-interactivo*) y además en el primer caso se distingue entre el primer acceso que necesita clave (*login shell*) de los posteriores (*non-login shell*). Cuando se inicia bash se leen automáticamente distintos ficheros de inicialización.

- En estos ficheros el usuario define variables de entorno, alias, el prompt, el path, etc.
- Los ficheros que se leen dependen de la forma de invocar bash

Formas de invocar bash:

1. Invocado como un *login shell interactivo*
   - cuando entramos en el sistema con login y password, usamos `su -`, o iniciamos bash con la opción `--login`
   - cuando se inicia, se leen los siguientes ficheros:
     a) `/etc/profile`
     b) el primero que exista de : `~/.bash_profile`, `~/.bash_login` o `~/.profile`
   - al dejar el shell se lee `~/.bash_logout`
2. Invocado como un *non-login shell interactivo*
   - cuando abrimos una nueva ventana de comandos (entramos sin login ni password), iniciamos bash sin opciones o usamos `su`
   - se leen los ficheros:
     a) `/etc/bash.bashrc`
     b) `~/.bashrc`[2]
   - al salir no se ejecuta nada
3. Invocado como un *shell no interactivo*
   - por ejemplo, cuando se lanza un script
   - en un shell no interactivo, la variable `$PS1` no está disponible
   - se lee el fichero definido en la variable `BASH_ENV`

[2]Usualmente, desde `.bash_profile` se invoca al `bashrc` de la siguiente forma:
`if [ -f ~/.bashrc ]; then . ~/.bashrc; fi`

2.2. Scripts de administración

Un administrador de sistemas debe crear scripts para realizar tareas complejas

- La mayoría de los ficheros de configuración de Unix son ficheros ASCII
- Disponemos de potentes herramientas para manejar estos ficheros

Veremos

- Programación de scripts con bash
- Herramientas de manejo de ficheros de texto usando expresiones regulares
- Programación en Python

2.2.1. Programación Shell-Script

Bash (y otros *shells*) permiten programar *scripts*:

Script o programa *shell*: fichero de texto conteniendo comandos externos e internos, que se ejecutan línea por línea

- El programa puede contener, además de comandos
  1. variables
  2. constructores lógicos (`if...then`, `AND`, `OR`, etc.) y lazos (`while`, `for`, etc.)
  3. funciones
  4. comentarios

Para saber más:

- *Advanced Bash-Scripting Guide*, Mendel Cooper, Última revisión 10 de marzo de 2014, www.tldp.org/guides.html
- *The Deep, Dark Secrets of Bash*, Ben Okopnik, Linux Gazette, okopnik.freeshell.org/articles/Shell_Scripting-4.html
- *Introduction to Shell Scripting*, Ben Okopnik, okopnik.freeshell.org/writings.html

Ejecución de un script

Los scripts deben empezar por el *número mágico* `#!` seguido del programa a usar para interpretar el script:

- `#!/bin/bash` - script de bash
- `#!/bin/sh` - script de shell
- `#!/usr/bin/env python3` - script de python3
- `#!/usr/bin/awk -f` - script de awk

Las formas usuales de ejecutar un script son:

- darle permiso de ejecución al fichero y ejecutarlo como un comando:

```
$ chmod +x helloworld
./helloworld
```

- ejecutar una shell poniendo como argumento el nombre del script (sólo necesita permiso de lectura)

```
$ bash helloworld
```

- ejecutarlo en la shell actual (dos posibles formas):

```
$ . helloworld
$ source helloworld
```

Paso de parámetros

Es posible pasar parámetros a un scripts: los parámetros se recogen en las variables `$1` a `$9`

Variable	Uso
`$0`	el nombre del script
`$1` a `$9`	parámetros del 1 al 9
`${10}`, `${11}`,...	parámetros por encima del 10
`${a:-b}`	si no existe `$a` se usa el valor `b`
`$#`	número de parámetros
`$*`, `$@`	todos los parámetros

Ejemplo:

```
$ cat parms1.sh
#!/bin/bash
VAL=$(($1:-0} + ${2:-0} + ${3:-0}))
echo $VAL
$ bash parms1.sh 2 3 5
10
$ bash parms1.sh 2 3
5
```

El comando `shift` desplaza los parámetros hacia la izquierda el número de posiciones indicado:

```
$ cat parms2.sh
#!/bin/bash
echo $#
echo $*
echo "$1 $2 $3 $4 $5 $6 $7 $8 $9 ${10} ${11}"
shift 9
echo $1 $2 $3
echo $#
echo $*
$ bash parms2.sh a b c d e f g h i j k l
12
a b c d e f g h i j k l
a b c d e f g h i j k
j k l
3
j k l
```

Uso de arrays

```
#!/bin/bash
name=( "cero" "uno" "dos" "tres" )
num=( 1 2 3 4 )
echo ${name[0]}
echo ${num[0]}
for i in "${name[@]}" ; do
   echo $i
done
for i in "${num[@]}" ; do
   echo $i
done
echo "total= ${#name[@]}"
```

Salida

El comando `exit` permite salir del script en cualquier punto

- se le puede dar un argumento numérico que toma como estado de salida, p.e. `exit 0` si el script acaba bien y `exit 1` en caso contrario
- si no se usa `exit`, el estado de salida del script es el del último comando ejecutado
- `$?` recoge el código se salida del script o del último comando ejecutado.

2.2.2. Entrada/salida

Es posible leer desde la entrada estándar o desde fichero usando `read` y redirecciones:

```
#!/bin/bash
echo -n "Introduce algo: "
read x
echo "Has escrito $x"
echo -n "Escribe 2 palabras: "
read x y
echo "Primera palabra $x; Segunda palabra $y"
```

Si queremos leer o escribir a un fichero utilizamos redirecciones:

```
echo $X > fichero
read X < fichero
```

Este último caso lee la primera línea de `fichero` y la guarda en la variable `X`

- Si queremos leer un fichero línea a línea podemos usar `while`:

```
#!/bin/bash
# FILE: linelist
# Usar: linelist filein fileout
# Lee el fichero pasado en filein y
# lo salva en fileout con las lineas numeradas
count=0
while read BUFFER
do
   count=$((++count))
   echo "$count $BUFFER" » $2
done < $1
```

  - el fichero de entrada se va leyendo línea a línea y almacenando en BUFFER
  - `count` cuenta las líneas que se van leyendo

- El uso de lazos para leer ficheros es bastante ineficiente
  - deberían evitarse (por ejemplo, usar `cat fichero`)

Ejemplo de lectura de fichero

```
#!/bin/bash
# Usa $IFS para dividir la línea que se está leyendo
# por defecto, la separación es "espacio"
echo "Lista de todos los usuarios:"
OIFS=$IFS # Salva el valor de IFS
IFS=: # /etc/passwd usa ":" para separar los campos
cat /etc/passwd |
while read name passwd uid gid fullname ignore
do
  echo "$name ($fullname)"
done
IFS=$OIFS # Recupera el $IFS original
```

- El fichero `/etc/passwd` se lee línea a línea
  - para cada línea, sus campos se almacenan en las variables que siguen a `read`
  - la separación entre campos la determina la variable `$IFS` (por defecto, espacio en blanco)

Redirecciones

Las redirecciones y pipes pueden usarse en otras estructuras de control

Ejemplo: lee las 2 primeras líneas de un fichero

```
if true
then
  read x
  read y
fi < fichero1
```

Ejemplo: lee líneas de teclado y guardalas en un fichero temporal convirtiendo minúsculas en mayúsculas

```
#/bin/bash
read buf
while [[ "$buf" ]]
do
  echo $buf
  read buf
done | tr 'a-z' 'A-Z' > tmp.$$
```

2.2.3. Tests

Los comandos que se ejecutan en un shell tienen un *código* de salida, que se almacena en la variable `$?`

- si `$?` es 0 el comando terminó bien
- si `$?` es > 0 el comando terminó mal

Ejemplo:

```
$ ls /bin/ls
/bin/ls
$ echo $?
0
$ ls /bin/ll
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
```

Podemos chequear la salida de dos comandos mediante los operadores `&&` (AND) y `||` (OR)

- estos operadores actúan en cortocircuito:

  `comando1 && comando2`
  `comando2` sólo se ejecuta si *`comando1`* acaba bien
  `comando1 || comando2`
  `comando2` sólo se ejecuta si *`comando1`* falla

- comandos `true` y `false`: devuelven 0 y 1, respectivamente

Ejemplo con `&&`:

```
$ ls /bin/ls && ls /bin/ll
/bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
$ ls /bin/ll && ls /bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
```

Ejemplo con `||`:

```
$ ls /bin/ls || ls /bin/ll
/bin/ls
$ echo $?
0
$ ls /bin/ll || ls /bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
/bin/ls
$ echo $?
0
```

2.2.4. Estructura `if...then...else`

Podemos usar el estado de salida de uno o varios comandos para tomar decisiones:

```
if comando1
then
  ejecuta otros comandos
elif comando2
then
  ejecuta otros comandos
else
  ejecuta otros comandos
fi
```

- debe respetarse la colocación de los `then`, `else` y `fi`
  - también puede escribirse `if comando1 ; then`
- el `elif` y el `else` son opcionales, no así el `fi`

Ejemplo:

```
$ cat if.sh
#!/bin/bash
if (ls /bin/ls && ls /bin/ll) >/dev/null 2>&1
then
  echo "Encontrados ls y ll"
else
  echo "Falta uno de los ficheros"
fi
$ bash if.sh
Falta uno de los ficheros
```

Comando `test`

Notar que `if` sólo chequea el código de salida de un comando, no puede usarse para comparar valores: para eso se usa el comando `test`
El comando test permite:

- chequear la longitud de un string
- comparar dos strings o dos números
- chequear el tipo de un fichero
- chequear los permisos de un fichero
- combinar condiciones juntas

`test` puede usarse de tres formas:

`test` *`expresión`*

`[` *`expresión`* `]`[3]

`[[` *`expresión`* `]]` (comando test extendido)

Si la expresión es correcta `test` devuelve un código de salida 0, si es falsa, devuelve 1:

- este código puede usarse para tomar decisiones:

[3]Notar los espacios en blanco entre los [ ] y *expresión*

```
if [[ "$1" = "hola" ]]
then
  echo "Hola a ti también"
else
  echo "No te digo hola"
fi
if [[ $2 ]]
then
  echo "El segundo parámetro es $2"
else
  echo "No hay segundo parámetro"
fi
```

- en el segundo `if` la expresión es correcta si `$2` tiene algún valor; falsa si la variable no está definida o contiene null (`""`)

2.2.5. Expresiones

Existen expresiones para chequear strings, números o ficheros

Chequeo de strings

Expresión	Verdadero si
string	el string es no nulo (`""`)
`-z` *string*	la longitud del string es 0
`-n` *string*	la longitud del string no es 0
string1 `=` *string2*	los strings son iguales
string1 `!=` *string2*	los strings son distintos

Chequeo de enteros

Expresión	Verdadero si
int1 `-eq` *int2*	los enteros son iguales
int1 `-ne` *int2*	los enteros son distintos
int1 `-gt` *int2*	*int1* mayor que *int2*
int1 `-ge` *int2*	*int1* mayor o igual que *int2*
int1 `-lt` *int2*	*int1* menor que *int2*
int1 `-le` *int2*	*int1* menor o igual que *int2*

Chequeo de ficheros

Expresión	Verdadero si
`-e` *file*	*file* existe
`-r` *file*	*file* existe y es legible
`-w` *file*	*file* existe y se puede escribir
`-x` *file*	*file* existe y es ejecutable
`-f` *file*	*file* existe y es de tipo regular
`-d` *file*	*file* existe y es un directorio
`-c` *file*	*file* existe y es un dispositivo de caracteres
`-b` *file*	*file* existe y es un dispositivo de bloques
`-p` *file*	*file* existe y es un pipe
`-S` *file*	*file* existe y es un socket
`-L` *file*	*file* existe y es un enlace simbólico
`-u` *file*	*file* existe y es *setuid*
`-g` *file*	*file* existe y es *setgid*
`-k` *file*	*file* existe y tiene activo el *sticky bit*
`-s` *file*	*file* existe y tiene tamaño mayor que 0

Operadores lógicos de `test`

- Los operadores lógicos clásicos se utilizan con el comando `test` y con `[ ]`.

Expresión	Propósito
`!`	niega el resultado de una expresión
`-a`	operador AND
`-o`	operador OR
`\(` *expr* `\)`	agrupación de expresiones; los paréntesis tienen un significado especial para el shell, por lo que hay que *escaparlos*

Ejemplos:

```
$ test -f /bin/ls -a -f /bin/ll ; echo $?  -->  1
$ [ ! -w /etc/passwd ] ; echo $?  -->  0
$ [ $$ -gt 0 -a \( $$ -lt 5000 -o -w file \) ]
```

Operadores lógicos de `test` extendido

- A partir de la versión 2.02 de Bash se introduce el *extended test command*: `[[ expr ]]`, que permite realizar comparaciones similares a los lenguajes estándar:
- permite usar los operadores && y || para unir expresiones
- no necesita *escapar* los paréntesis

Expresión	Propósito
`!`	niega el resultado de una expresión
`&&`	operador AND
`||`	operador OR
`( expr )`	agrupación de expresiones

Ejemplos:

```
$ [[ -f /bin/ls && -f /bin/ll ]] ; echo $?  -->  1
$ [[ ! -w /etc/passwd ]] ; echo $?  -->  0
$ [[ $$ -gt 0 && ($$ -lt 5000 || -w file) ]]
```

2.2.6. Control de flujo

Además del `if` bash permite otras estructuras de control de flujo: `case`, `for`, `while` y `until`

Estructura `case`

```
case valor in
  patrón_1)
    comandos si value = patrón_1
    comandos si value = patrón_1 ;;
  patrón_2)
    comandos si value = patrón_2 ;;
  *)
    comandos por defecto ;;
esac
```

- si *`valor`* no coincide con ningún patrón se ejecutan los comandos del apartado después del `*)`. Este apartado es opcional
- *`patrón`* puede incluir comodines y usar el símbolo | como operador OR

Ejemplo:

```
#!/bin/bash
echo -n "Respuesta:" read RESPUESTA
case $RESPUESTA in
  S* | s*)
    RESPUESTA="SI";;
  N* | n*)
    RESPUESTA="NO ";;
  *)
    RESPUESTA="PUEDE";;
esac
echo $RESPUESTA
```

Lazos `for`

```
for var in lista
do
  comandos
done
```

- *var* toma los valores de la lista
  - puede usarse *globbing* para recorrer los ficheros

Ejemplo: recorrer una lista

```
LISTA="10 9 8 7 6 5 4 3 2 1"
for var in $LISTA
do
  echo $var
done
```

Ejemplo: recorrer los ficheros `*.bak` de un directorio

```
dir="/var/tmp"
for file in $dir/*.bak
do
  rm -f $file
done
```

Sintaxis alternativa, similar a la de C

```
LIMIT=10
for ((a=1, b=LIMIT; a <= LIMIT; a++, b--))
do
  echo "$a-$b"
done
```

Bucle `while`

```
while comando
do
  comandos
done
```

- se ejecuta mientras que el código de salida de `comando` sea cierto

Ejemplo:

```
while [[ $1 ]]
do
  echo $1
  shift
done
```

Bucle `until`

```
until comando
do
  comandos
done
```

- se ejecuta hasta que el código de salida de `comando` sea hace cierto

Ejemplo:

```
until [[ "$1" = "" ]]
do
  echo $1
  shift
done
```

`break` y `continue`

- `break` permite salir de un lazo.
- Con `break` *n* especificamos el número de lazos que queremos salir
- `continue` permite saltar a la siguiente iteración

Ejemplo con `break`:

```
# Imprime el contenido de los ficheros hasta que
# encuentra una línea en blanco
for file in $*
do
  while read buf
  do
    if [[ -z "$buf" ]]
    then
      break 2
    fi
    echo $buf
  done <$file
done
```

Ejemplo con `continue`:

```
# Muestra un fichero pero no las líneas de más
# de 80 caracteres
while read buf
do
  cuenta=`echo $buf | wc -c`
  if [[ $cuenta -gt 80 ]]
  then
    continue
  fi
  echo $buf
done < $1
```

2.2.7. Funciones

Podemos definir funciones en un script de shell:

```
funcion() {
  comandos
}
```

y para llamarla:

```
funcion p1 p2 p3
```

Siempre tenemos que definir la función antes de llamarla:

```
#!/bin/bash
# Definición de funciones
funcion1() {
  comandos
}
funcion2() {
  comandos
}
# Programa principal
funcion1 p1 p2 p3
```

Paso de parámetros La función referencia los parámetros pasados por posición, es decir, `$1`, `$2`, ..., y `$*` para la lista completa:

```
$ cat funcion1.sh
#!/bin/bash
funcion1()
{
  echo "Parámetros pasados a la función: $*"
  echo "Parámetro 1: $1"
  echo "Parámetro 2: $2"
}
# Programa principal
funcion1 "hola" "que tal estás" adios
$
$ bash funcion1.sh
Parámetros pasados a la función: hola que tal estás adios
Parámetro 1: hola
Parámetro 2: que tal estás
```

return Una función de bash puede devolver dos posibles valores: 0 y 1.

```
#!/bin/bash
funcion2() {
  if [[ -f /bin/ls && -f /bin/ln ]]; then
    return 0
  else
    return 1
  fi
}
# Programa principal
if funcion2; then
  echo "Los dos ficheros existen"
else
  echo "Falta uno de los ficheros - adiós"
  exit 1
fi
```

2.2.8. Otros comandos

wait Permite esperar a que un proceso lanzado en *background* termine

```
sort $largefile > $newfile &
ejecuta comandos
wait
usa $newfile
```

Si lanzamos varios procesos en background podemos usar `$!`

- `$!` devuelve el `PID` del último proceso lanzado

```
sort $largefile1 > $newfile1 &
SortPID1=$!
sort $largefile2 > $newfile2 &
SortPID2=$!
ejecuta comandos
wait $SortPID1
usa $newfile1
wait $SortPID2
usa $newfile2
```

trap Permite *atrapar* las señales del sistema operativo

- permite hacer que el programa termine limpiamente (p.e. borrando ficheros temporales, etc.) aún en el evento de un error

```
$ cat trap.sh
#!/bin/bash
capturado() {
  echo "Me has matado!!!"
  kill -SIGTERM $$
}
trap "capturado" SIGINT SIGQUIT
while true; do
  true
done
$ bash trap.sh
(Ctrl-C)
Me has matado!!!
Terminado
```

Las señales más comunes para usar con **trap** son:

Señal	Significado
0	salida del shell (por cualquier razón, incluido fin de fichero)
SIGHUP	cuelgue (hang-up) del terminal o muerte del proceso controlador
SIGINT	interrupción de teclado (Ctrl-C)
SIGTERM	terminate (mata el proceso permitiéndole terminar)
SIGKILL	kill (no puede ser parada ni ignorada)
SIGQUIT	salida de teclado

Referencias indirectas

Permiten definir variables cuyo contenido es el nombre de otra variable:

```
a=letra
letra=z
# Referencia directa
echo "a = $a"  # a = letra
# Referencia indirecta
eval a=\$$a
echo "Ahora a = $a"  # Ahora a = z
```

Las versiones de bash a partir de la 2 permiten una forma más simple para las referencias indirectas:

```
a=letra
letra=z
# Referencia directa
echo "a = $a"  # a = letra
# Referencia indirecta
echo "Ahora a = ${!a}"  # Ahora a = z
```

Otro ejemplo con `eval`

```
$ cat dni.sh
#!/bin/bash
dniPepe=23456789
dniPaco=98765431
echo -n "Nombre: "; read nombre
eval echo "DNI = \$dni${nombre}"
$ bash dni.sh
Nombre: Pepe
DNI = 23456789
```

2.2.9. Optimización de scripts

El shell no es especialmente eficiente a la hora de ejecutar trabajos pesados

- Ejemplo: script que cuenta las líneas de un fichero:

```
$ cat cuentalineas1.sh
#!/bin/bash
count=0
while read line
do
  count=$(expr $count + 1)
done < $1
echo "El fichero $1 tiene $count líneas"
```

- si medimos el tiempo que tarda

```
$ time bash cuentalineas1.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m59.757s
user 0m17.868s
sys  0m41.462s
```

- Podemos mejorarlo si usamos aritmética de shell en vez de el comando `expr`

```
$ cat cuentalineas2.sh
#!/bin/bash
count=0
while read line
do
  count=$(($count+1))
done < $1
echo "El fichero $1 tiene $count líneas"
```

  - el tiempo ahora

```
$ time bash cuentalineas2.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m1.014s
user 0m0.887s
sys  0m0.108s
```

- Y todavía mejor:

```
$ cat cuentalineas3.sh
#!/bin/bash
count=$(wc -l $1 | cut -d " " -f 1)
echo "El fichero $1 tiene $count líneas"
$
$ time bash cuentalineas3.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m0.096s
user 0m0.005s
sys  0m0.009s
```

- Conclusiones
  - Intenta reducir el número de procesos creados al ejecutar el script, por ejemplo, usando las funciones aritméticas del shell
  - Siempre que sea posible, intenta usar comandos del shell (`wc`, `tr`, `grep`, `sed`, etc.) en vez de lazos

Depuración

Para depurar un script de shell podemos usar la opción `-x` o `-o xtrace` de bash:

- muestra en la salida estándar trazas de cada comando y sus argumentos, después de que el comando se haya expandido pero antes de que se sea ejecutado

```
$ bash -x cuentalineas3.sh Quijote.txt
++ wc -l Quijote.txt
++ cut -d ' ' -f 1
+ count=36855
+ echo 'El fichero Quijote.txt tiene 36855 líneas'
El fichero Quijote.txt tiene 36855 líneas
```

Es posible depurar sólo parte de un script:

- poner `set -x` o `set -xv` al inicio del trozo a depurar
- `set +x` o `set +xv` para cancelar

```
$ cat cuentalineas3.sh
#!/bin/bash
set -x
count=$(wc -l $1 | cut -d " "-f 1)
set +x
echo "El fichero $1 tiene $count líneas"
$
$ bash cuentalineas3.sh Quijote.txt
++ wc -l Quijote.txt
++ cut -d ' '-f 1
+ count=36855
+ set +x
El fichero Quijote.txt tiene 36855 líneas
```

2.3. Expresiones regulares

Las expresiones regulares permiten reconocer en texto una serie de cadenas de caracteres que obedecen a cierto patrón.

- Los ficheros de configuración y logs de Unix son, normalmente, ficheros de texto.
- Muchos comandos de procesamiento y búsqueda de texto como `grep`, `egrep`, `sed`, `awk` o `vi` usan expresiones regulares.
- No hay que confundir las expresiones regulares con los comodines, que son utilizados por el shell para referenciar ficheros.
- Se distinguen las expresiones regulares básicas y las extendidas, más completas.
- se pueden usar con ficheros o con cualquier otro de tipo de entrada, como la entrada estándar, pipes, etc.
- Para evitar confusiones conviene escribir las expresiones regulares entre comillas, simples o dobles.

Comandos egrep y sed -r

- `egrep`. Busca patrones en ficheros de texto. Si el patrón se encuentra en una línea devuelve la línea indicando la coincidencia. En caso contrario no devuelve nada. `egrep` (o `grep -E`) utiliza expresiones regulares extendidas, mientras que `grep` usa solo expresiones regulares básicas. El formato básico de uso es:

  `egrep '`*patrón*`'`

- `sed -r`. Sustituye patrones por la cadena de texto que se le indica (la opción `-r` indica que se utilicen expresiones regulares extendidas). El formato básico de utilización es:

  `sed -r 's/`*patrón*`/`*sustitución*`/g'`

- Estos comandos se pueden usar con ficheros de texto o con cualquier otro de tipo de entrada, como la entrada estándar, tuberías, etc.
- Pueden usar comillas dobles o simples. Las comillas dobles permiten sustituir variables y ejecutar comandos, mientras que las comillas simples no.

Ejemplos:

- `egrep 'unix' tmp.txt`
  busca en el fichero `tmp.txt` las líneas que contienen la palabra `unix`
- `egrep '[Uu]nix' tmp.txt`
  busca las líneas que contienen `unix` o `Unix`
  (los corchetes indican que hay varias posibilidades de coincidencia)
- `egrep 'hel.' tmp.txt`
  busca las líneas que contienen `hel` seguido de cualquier carácter
  (el punto indica cualquier caracter)
- `egrep 'ab*c' tmp.txt`
  localiza las cadenas que empiecen por `a`, que continúen con 0 o más `b`, y que sigan con una `c`, por ejemplo: `abbbc` o `aaacb`, pero no `axc` o `cba`
  (* indica repetición de cero, una, dos o más veces)
- `egrep 't[^aeiouAEIOU][a-zA-Z]*' tmp.txt`
  localiza las cadenas que empiecen por `t`, seguido de algún carácter no vocálico y de 0 o más apariciones de letras
- `sed -r "s/inicio/fin/g" tmp.txt`
  sustituye la cadena `inicio` por `fin`
- `sed -r "s/[Ww]indows/Linux/g" tmp.txt`
  sustituye las cadenas `Windows` y `windows` por Linux

Importante: no debemos confundir las expresiones regulares con los comodines para la sustitución de nombres de ficheros (*glob*)

- Por ejemplo, el asterisco (*) como comodín indica cualquier cadena en nombres de fichero, mientras que como expresión regular es repetición.

Si no especificamos fichero, `egrep` y `sed` usan la entrada estándar:

- Podemos usarlos para probar las expresiones regulares escribiendo en el terminal después del comando (se finaliza con CNTL-D):

```
$ egrep --color '[Uu]nix'
unix → unix (coincidencia)
Unix → Unix (coincidencia)
Linux → (sin coincidencia)

$ sed -r 's/x/HOLA/g'
x → HOLA
```

Las comillas dobles permiten utilizar variables y comandos dentro de las expresiones regulares.

Ejemplo, suponiendo que el contenido de `fichero.txt` es `x`:

```
$ a=5
$ sed -r "s/x/$a/g" fichero.txt → 5
$ sed -r 's/x/$a/g' fichero.txt → $a
$ sed -r "s/x/`date`/g" fichero.txt → mié ago 3 09:15:30
$ sed -r 's/x/`date`/g' fichero.txt → `date`
```

2.3.1. Expresiones regulares básicas

ER de un sólo carácter

ER	concuerda con
`.`	cualquier carácter
`[ ]`	cualquiera de los caracteres entre corchetes, p.e. `[abc]` concuerda con `a`, `b` o `c`; `[a-z]` concuerda con cualquier letra minúscula
`[^ ]`	cualquier carácter que no esté entre corchetes
`^`	principio de línea
`$`	final de línea
`*`	0 o más ocurrencias de la expresión regular anterior
`\( \)`	permite agrupar ER
`\`	*escapa* un metacarácter

Repetición Podemos repetir una expresión regular usando `\{ \}`

Constructor	Propósito
`\{`n`\}`	concuerda con exactamente n ocurrencias de la ER previa
`\{`n`,\}`	concuerda con al menos n ocurrencias de la ER previa
`\{`n`, `m`\}`	concuerda con entre n y m ocurrencias de la ER previa

Nótese que en las expresiones regulares básicas `\( \)` y `\{ \}` requieren caracteres de escape, cosa que no ocurre con las expresiones regulares extendidas.

2.3.2. Expresiones regulares extendidas

Los sistemas UNIX actuales admiten extensiones a las expresiones regulares básicas.

Concordancia:

- La mayoría de los caracteres son tratados como literales, esto es, concuerdan (*match*) consigo mismos:
  - Por ejemplo: `egrep "a"`
    `a` concuerda con `a`, `ax`, `xa`, `xay`, etc.
- la excepción son los metacaracteres:

  `. [ ] ^ $ * ( ) \ + ? |{ }`

ER	concuerda con
`.`	cualquier carácter
`[ ]`	cualquiera de los caracteres entre corchetes, p.e. `[abc]` concuerda con `a`, `b` o `c`; `[a-z]` concuerda con cualquier letra minúscula
`[^ ]`	cualquier carácter que no esté entre corchetes
`^`	principio de línea
`$`	final de línea
`*`	0 o más ocurrencias de la expresión regular anterior
`+`	una o más ocurrencias de la ER anterior
`?`	cero o una ocurrencia de la ER anterior
`( )`	permite agrupar ER
`|`	OR
`{`n`}`	concuerda con exactamente n ocurrencias de la ER previa
`{`n`,}`	concuerda con al menos n ocurrencias de la ER previa
`{`n`, `m`}`	concuerda con entre n y m ocurrencias de la ER previa
`\`	*escapa* un metacarácter
`\\` o `\\\`	`\`

- Dentro de `[ ]` los metacaracteres pierden su significado especial: p.e. `[a.]c` concuerda con `ac` y `.c`
- Para incluir un carácter `]` en una lista colocarlo al principio; para incluir un `^` en cualquier lugar menos al principio; para incluir un `-` al final: p.e. `[a^]c` concuerda con `ac` y `^c`

Ejemplos:

ER	concuerda con
`a..c`	cadena que contenga el caracter `a`, seguido por dos caracteres cualquiera y luego `c`, por ejemplo, `zaxyct`, ...
`[abc]`	cadenas que contengan un carácter `a`, `b` o `c`
`[^abc]`	cadenas que contengan un carácter distinto de `a`, `b` y `c`
`[a-z]`	cadenas que con tengan una letra minúscula
`^abc`	líneas que empiecen por `abc`
`abc$`	líneas que terminen por `abc`
`ab*c`	cadenas que contengan el caracter `a`, que continúen con 0 o más `b`, y luego una `c`: `abc`, `ac`, `abbc`, `aaccab`,... pero no `cba` o `aaab`
`b[cq]*e`	cadenas que contengan `b`, que continúen con 0 o más `c` o `q`, y luego una `e`: `be`, `bcce`, `bccqqee` o `bqqqce`
`.*`	cualquier cadena
`abc.*`	cualquier cadena que contenga `abc`
`x(abc)*x`	cadenas que tengan una `x` seguida de 0 o más ocurrencias de `abc`, y seguida de otra `x`: `xabcx`, `xx`, `xabcabcx`,..., pero no `xacx` o `xcbax`
`(a|b)c`	cadena que contenga `ac` o `bc`
`ab+c`	cadena que contenga `abc`, `abbc`, pero no `ac`
`ab?c`	cadena que contenga `ac`, `abc`, pero no `abbc`
`a{5}`	5 ocurrencias del carácter `a`
`.{5,}`	al menos 5 ocurrencias de cualquier carácter
`^#.*\.$`	línea que empiece por `#` y termine por `.` (notar que el segundo `.` está *escapado* por la `\`; la ER `.*` implica 0 o más caracteres cualquiera)

Etiquetado Las ER que se ponen entre `( )` quedan etiquetadas, y podemos hacer referencia a ellas mediante `\`n, con n el número de la etiqueta

- Ejemplos:
  - `(.)oo\1` concuerda con `moom`, `noon`, pero no con `moon`
    (en el primer caso `(.)` copia la primera `m` de `moom` y la pega después de `oo` con `\1`, obteniendo la coincidencia).
  - `(.)oo\1-(.)aa\1\2` concuerda con `moom-paamp`
    (aquí el primer paréntesis copia la primera `m` y el segundo paréntesis copia la primera `p` que luego se pegan con `\1` y `\2`, respectivamente, obteniendo la coincidencia)
- Con el comando `sed` permiten realizar la operación de copia-pega:

  ```
  sed -r "s/(.)(.)/\2\1/g"
  ```

  Así, sustituye la cadena `ab` por `ba`, la cadena `xy` por `yx`, etc.
  (en el primer caso, el primer paréntesis copia la `a` y el segundo paréntesis copia la `b`, luego `\2` pega la `b` y `\1` pega la `a`)

- También podemos usar el caracter & para hacer referencia a la secuencia reconocida. Por ejemplo:

  `sed -r "s/a.*a/[reconocido: &]/g"`

  (por ejemplo, sustituye `axyza` por `[reconocido: axyza]`)

Las expresiones regulares intentan reconocer la concordancia más larga.

- Ejemplo:

  `egrep --color "a(.*)a"`

  Para la secuencia `yyyaxaxayyy` detecta `axaxa`

Otros caracteres Además de los ya vistos, pueden usarse otros metacaracteres:

ER	concuerda con
`\n`, `\r`, `\t`	LF, CR y tab (no siempre funcionan)
`[:space:]`	caracteres en blanco ([\t\n\r\f\v])
`[:blank:]`	espacio y tabulado
`[:alnum:]` o `\w`	caracteres alfánuméricos (letras y números)
`[:digit:]`	dígitos
`[:alpha:]`	alfabéticos
`[:upper:]`	mayúsculas
`[:lower:]`	minúsculas
`[:xdigit:]`	dígitos hexadecimales
`[:punct:]`	signos de puntuación
`[:cntrl:]`	caracteres de control
`[:graph:]`	caracteres imprimibles (sin espacio)
`[:print:]`	caracteres imprimibles (con espacio)
`\<`, `\>`	inicio/fin de palabra
`\b`	posición entre palabras
`\B`	posición en medio de una palabra

- `[[:upper:]]bc` concuerda con `Abc`, pero no `abc`
- `\babc\b` concuerda con `ab abc df`, pero no con `abcdef`
- `\Babc\B` concuerda con `ababcdf`, pero no con `ab abc df`

2.3.3. Más ejemplos

Ejemplos de uso con `sed`:

```
$ echo "abc1234def"| sed -r "s/[0-9]+/NUMERO/"
abcNUMEROdef
$ echo "abc1234def"| sed -r 's/[0-9]+/<&>/'
abc<1234>def
# En el siguiente ejemplo, notar que las ER intentan siempre
reconocer la secuencia más larga posible
$ echo "000x111x222x333"| sed 's/x.*x/<&>/'
000<x111x222x>333
# Eliminar blancos a principio y al final de línea y sustituir
más de un blanco seguido por uno solo
$ sed -r "s/^␣+// ; s/␣+$// ; s/␣␣+/␣/g" fich
# Pon los 4 primeros caracteres de cada línea al final
de la misma
$ sed -r 's/^(.{4,4})(.*)/\2\1/' fich
# Cambia de minúsculas a mayúsculas la primera letra de
cada palabra
$ sed -r 's/\<./\u&/g'
# Convierte retornos de DOS (CR/LF) a formato Unix (LF)
$ sed 's/^M$//' [4]
# también funcionaría
$ sed 's/\r//'
```

Ejemplos más complejos:

1. `\w+@\w+\.\w+((\.\w+)*)?` concuerda con direcciones de e-mail
2. `(0[1-9]|[12][0-9]|3[01])-(0[1-9]|1[012])-(19|20)[0-9]{2}` concuerda con fechas en el formato *dd-mm-yyyy* (años entre el 1900 y 2099)
3. `[-+]?([0-9]*\.)?[0-9]+([eE][-+]?[0-9]+)?` concuerda con números en punto flotante (con o sin exponente)

2.3.4. Más opciones de los comandos grep y sed

`grep` y `egrep` buscan en ficheros por un patrón determinado

[4] Para introducir un carácter de control, como `^M`, tenemos que pulsar primero `Ctrl-V` y luego el carácter, en este caso `Enter`

```
grep [opciones] patrón [fichero...]
egrep [opciones] patrón [fichero...]
```

Opciones:

- `-E` o `egrep`: usa expresiones regulares extendidas
- `-R` o `rgrep`: lee todos los ficheros bajo cada directorio, recursivamente
- `-i` o `--ignore-case`: busca ignorando diferencias entre mayúsculas y minúsculas
- `-n` o `--line-number`: muestra el número de línea dentro del fichero

`sed` (*stream editor*, editor de flujo) permite realizar transformaciones básicas de un flujo de entrada (un fichero o una entrada desde una tubería)

Formato:

- Substitución de cadenas (`s`):

```
sed -r [opciones] 's/ER/reemplazo/flag' [fichero]
```

- Borrado de la línea completa (`d`):

```
sed -r [opciones] '/ER/d' fichero]
```

- Reemplazo de la línea completa (c\):

```
sed -r [opciones] '/ER/c\reemplazo' [fichero]
```

- Insertar/añadir una línea antes o después de la afectada (i\, a\):

```
sed -r [opciones] '/ER/i\reemplazo' [fichero]
```

```
sed -r [opciones] '/ER/a\reemplazo' [fichero]
```

Algunas opciones:

- `-i` modifica el fichero de entrada (edición *in-place*)

Algunos flags:

- `g`: aplica los cambios globalmente (por defecto, sólo se cambia la primera aparición en cada línea)

- *NUMERO*: reemplaza solo la aparición número *NUMERO* dentro de cada línea
- `w` *fichero*: escribe las líneas con sustituciones al fichero indicado

Ejemplos:

- Reemplaza, en cada línea de `fichero`, la quinta ocurrencia de `stop` por `STOP`

```
$ sed 's/stop/STOP/5' fichero
```

- Sustituye `stop` por `STOP` y guarda cada línea reemplazada en el fichero `fich2`

```
$ sed -r 's/stop/STOP/w fich2' fichero
```

- Borra las líneas que contengan `jaime`

```
$ sed -r '/jaime/d' amigos
```

- Cambia las líneas que contengan `jaime` por CAMBIADO

```
$ sed -r '/jaime/c\CAMBIADO' amigos
```

- Inserta una línea, con la palabra 'APARICION', antes de las líneas que contengan `jaime`

```
$ sed -r '/jaime/i\APARICION' amigos
```

Comandos desde fichero: la opción -f permite leer comandos de `sed` agrupados en un fichero

Ejemplo: reemplazo desde la línea 3 hasta la 10 de *Linux/linux* por *GNU/Linux* y de *samba* por *Samba*

```
$ cat file.sed
3,10 {
  s/[Ll]inux/GNU\/Linux/g
  s/samba/Samba/g
}
$ sed -f file.sed fichero
```

2.4. Procesamiento de textos

Comandos simples Existe una serie de comandos simples para realizar operaciones concretas sobre ficheros de texto

- también se conocen como *filtros*: obtienen su entrada de la entrada estándar (o un fichero) y envían la salida a la salida estándar:

```
sort < fichero.txt | head -3 > otro_fichero.txt
```

`head`, `tail`	muestran el principio/final de un fichero
`tac`, `rev`	muestran el fichero al revés por líneas/caracteres de la línea
`wc`	cuenta el número de líneas, palabras y bytes de un fichero
`sort`	ordena las líneas alfabéticamente
`tr`	borra o reemplaza caracteres
`uniq`	elimina líneas sucesivas repetidas
`cut`	selecciona campos o columnas de un fichero
`paste`	combina texto de varios ficheros por líneas
`join`	combina varios ficheros por campos
`split`	divide un fichero en ficheros más pequeños
`nl`	añade números de línea
`expand`	convierte `TABs` en espacios
`fmt`	formatea párrafos
`od`	muestra un fichero en diferentes formatos

1) head, tail

- `head` muestra el principio de un fichero
- `tail` muestra el final de un fichero

Formato:

`head / tail` [opciones] *fichero*

Algunas opciones de ambos comandos:

- `-n` N ó `-`N muestra las primeras/últimas N líneas
- `-c` N muestra los primeros/últimos n bytes
- `-v` le añade una línea de cabecera, con el nombre del fichero

Algunas opciones adicionales de `tail`:

- `-f` hace que `tail` corra en un lazo, añadiendo líneas a medida que el fichero crece (útil para cuando queremos ver como se modifica un fichero)
- `--retry` útil con `-f`; aunque el fichero no exista o sea inaccesible continua intentando hasta que puede abrirlo

Ejemplos:

```
$ head -n 2 -v quijote.txt
==>quijote.txt <==
En un lugar de la Mancha, de cuyo nombre
no quiero acordarme, no ha mucho tiempo

$ tail -n 2 -v quijote.txt
==>quijote.txt <==
astillero, adarga antigua, rocín flaco y
galgo corredor.
```

2) tac, rev

- `tac` imprime el fichero de la última a la primera línea (opuesto a `cat`, de ahí el nombre)
- `rev` invierte las líneas del fichero

Ejemplos:

```
$ tac quijote.txt
galgo corredor.
astillero, adarga antigua, rocín flaco y
que vivía un hidalgo de los de lanza en
no quiero acordarme, no ha mucho tiempo
En un lugar de la Mancha, de cuyo nombre

$ rev quijote.txt
erbmon oyuc ed ,ahcnaM al ed ragul nu nE
opmeit ohcum ah on ,emradroca oreiuq on
ne aznal ed sol ed ogladih nu aíviv euq
y ocalf nícor ,augitna agrada ,orellitsa
.roderroc oglag
```

3) wc

Muestra el número de líneas, palabras y bytes de un fichero

Formato:

`wc` [opciones] *fichero*

Algunas opciones:

- `-l` muestra sólo el número de líneas
- `-w` muestra sólo el número de palabras
- `-c` muestra sólo el número de bytes
- `-L` muestra la longitud de la línea más larga

Ejemplo:

```
$ wc quijote.txt
  5 33 178 quijote.txt

$ wc -l quijote.txt
5 quijote.txt

$ wc -w quijote.txt
33 quijote.txt

$ wc -c quijote.txt
178 quijote.txt
```

4) sort

ordena alfabéticamente líneas de texto y las muestra en la salida estándar

Formato:

`sort` [opciones] *fichero*

Algunas opciones:

- `-b` ignora blancos al principio de línea
- `-f` no distingue mayúsculas/minúsculas
- `-r` orden inverso
- `-m` mezcla ficheros previamente ordenados

- `-n` ordena numéricamente
- `-k` *POS1[, POS2]* ordena según los campos desde *POS1* a *POS2*, o el final si no está *POS2* (el primer campo es 1)

Ejemplos:

```
$ cat nombres.txt   $ sort nombres.txt   $ sort -f nombres.txt
María Pérez         Adriana Gómez        Adriana Gómez
luis Andión         María Pérez          jorge pena
Adriana Gómez       jorge pena           luis Andión
jorge pena          luis Andión          María Pérez

$ sort -f -k 2,2 nombres.txt
luis Andión
Adriana Gómez
jorge pena
María Pérez
```

5) tr

Borra caracteres o reemplaza unos por otros

Formato:

`tr` [opciones] *set1 set2*

Algunas opciones:

- `-d` borra los caracteres especificados en *set1*
- `-s` reemplaza caracteres repetidos por un único carácter

Ejemplos:

```
$ tr 'a-z' 'A-Z' < quijote.txt
EN UN LUGAR DE LA MANCHA, DE CUYO NOMBER...

$ tr -d ' ' < quijote.txt
EnunlugardelaMancha,decuyonombre...

$ tr au pk < quijote.txt
En kn lkgpr de lp Mpnchp, de ckyo nombre...

$ tr lcu o < quijote.txt | tr -s o
En on ogar de oa Manoha, de oyo nombre
```

6) uniq

Descarta todas (menos una) las líneas idénticas sucesivas en el fichero

Formato:

`uniq` [opciones] *fichero*

Algunas opciones:

- `-d` muestra las líneas duplicadas (sin borrar)
- `-u` muestra sólo las líneas sin duplicación
- `-i` ignora mayúsculas/minúsculas al comparar
- `-c` muestra el número de ocurrencias de cada línea
- `-s` n no compara los n primeros caracteres
- `-f` n no compara los n primeros campos

Ejemplo:

```
$ cat nombres.txt
Julio Lorenzo
Pedro Andión
Celia Fernández
Celia Fernández
Juan Fernández
Enrique Pena
```

```
$ uniq nombres.txt
Julio Lorenzo
Pedro Andión
Celia Fernández
Juan Fernández
Enrique Pena
```

```
$ uniq -f 1 -c nombres.txt
  1 Julio Lorenzo
  1 Pedro Andión
  3 Celia Fernández
  1 Enrique Pena
```

7) cut

Escribe partes seleccionadas de un fichero a la salida estándar; puede usarse para seleccionar columnas o campos de un fichero específico

Formato:

`cut` [opciones] *fichero*

Algunas opciones:

- `-b`, `-c`, `-f` corta por bytes, caracteres o campos, respectivamente
- `-d` fija el carácter delimitador entre campos (por defecto, `TAB`)

Ejemplos:

```
$ cat nombres-ord.txt
Luis Andión
Adriana Gómez
Jorge Pena
María Pérez
```

```
$ cut -c 1-7 nombres-ord.txt
Luis An
Adriana
Jorge P
María P
```

```
$ cut -c 1-5,9-10 nombres-ord.txt
Luis ió
AdriaGó
Jorgena
Maríare
```

```
$ cut -d ' ' -f 1 nombres-ord.txt
Luis
Adriana
Jorge
María
```

8) paste

Permite unir texto de varios ficheros, uniendo las líneas de cada uno de los ficheros

Formato:

`paste` [opciones] *fichero1 [fichero2] ...*

Algunas opciones:

- `-s` pega los ficheros secuencialmente, en vez de intercalarlos
- `-d` especifica los carácteres delimitadores en la salida (por defecto, `TAB`)

Ejemplos:

```
$ cat nombres.txt
Luis
Adriana
Jorge
María
```

```
$ cat apellidos.txt
Andión
Gómez
Pena
Pérez
```

```
$ paste nombres.txt apellidos.txt
 Luis     Andión
 Adriana  Gómez
 Jorge    Pena
 María    Pérez
```

```
$ paste -d ' ' nombres.txt apellidos.txt
Luis Andión
Adriana Gómez
Jorge Pena
María Pérez

$ paste -s -d '\t\n' nombres.txt
 Luis   Adriana
 Jorge  María
```

9) join

Permite combinar dos ficheros usando campos: busca en los ficheros por entradas comunes en el campo y une las líneas; los ficheros deben estar ordenados por el campo de unión

Formato:

`join` [opciones] *fichero1 fichero2*

Algunas opciones:

- `-i` ignora mayúsculas/minúsculas
- `-1` *FIELD* une en el campo *FIELD* (entero positivo) de *fichero1*
- `-2` *FIELD* une en el campo *FIELD* de *fichero2*
- `-j` *FIELD* equivalente a `-1` *FIELD* `-2` *FIELD*
- `-t` *CHAR* usa el carácter *CHAR* como separador de campos
- `-o` *FMT* formatea la salida (*M.N* fichero *M* campo *N*, 0 campo de unión)
- `-v` *N* en vez de la salida normal, muestra las líneas que no se unen del fichero *N*
- `-a` *N* además la salida normal, muestra las líneas que no se unen del fichero *N*

Ejemplo:

```
$ cat nombres1.txt    $ cat nombres2.txt
Luis Andión           Pedro Andión
Adriana Gómez         Celia Fernández
Jorge Pena            Julio Lorenzo
María Pérez           Enrique Pena

$ join -j 2 nombres1.txt nombres2.txt
Andión Luis Pedro
Pena Jorge Enrique

$ join -j 2 -o 1.1 2.1 0 nombres1.txt nombres2.txt
Luis Pedro Andión
Jorge Enrique Pena
```

10) split

Divide un fichero en ficheros más pequeños; los ficheros más pequeños se nombran a partir del *prefijo* especificado (*prefijo*aa, *prefijo*ab,...)

Formato:

`split` [opciones] *fichero prefijo*

Si no se pone *fichero*, o se pone `-` se lee la entrada estándar

Algunas opciones:

- `-l` *n* pone *n* líneas en cada fichero de salida (por defecto 1000)
- `-b` *n* pone *n* bytes en cada fichero de salida
- `-C` *n* pone en cada fichero de salida tantas líneas completas como sea posible sin sobrepasar *n* bytes
- `-d` usa números en vez de letras para el nombre de los ficheros de salida

Ejemplo:

```
$ split -l 2 quijote.txt quij
$ ls quij*
quijaa quijab quijac quijote.txt
$ cat quijaa
En un lugar de la Mancha, de cuyo nombre
no quiero acordarme, no ha mucho tiempo
```

```
$ cat quijac
galgo corredor.
$ split -l 2 -d quijote.txt quij
$ ls quij*
quij00 quij01 quij02 ...
```

11) nl

Añade números de línea; `nl` considera los ficheros separados en *páginas lógicas*, cada una de ellas con una cabecera, cuerpo y pie, cada una de estas secciones se numera de forma independiente, y la numeración se reinicia para cada página; los comienzos de cabecera, cuerpo y pie de cada página se marcan, respectivamente, con `\:\:\:`, `\:\:` y `\:`

Formato:

`nl` [opciones] *fichero*

Algunas opciones:

- `-b`, `-h` o `-f` *ESTILO* indica el estilo de numeración para cuerpo, cabecera o pie, que puede ser:
  - `a`: numera todas las líneas
  - `t`: numerar sólo las líneas no vacías (por defecto para el cuerpo)
  - `p` *ER*: numera sólo las líneas que concuerdan con *ER*
  - `n`: no numera ninguna línea (por defecto para cabecera y pie)
- `-v` *n* inicia la numeración en *n* (por defecto, 1)
- `-i` *n* incrementa los números por *n* (por defecto, 1)
- `-p` no reinicia la numeración al principio de cada página
- `-s` *STRING* una STRING para separar los números de línea del texto (por defecto ' ')

Ejemplo:

```
$ nl -s 'q ' quijote.txt
  1q En un lugar de la Mancha, de cuyo nombre
  2q no quiero acordarme, no ha mucho tiempo
  3q que vivía un hidalgo de los de lanza en
  4q astillero, adarga antigua, rocín flaco y
  5q galgo corredor.
```

12) expand

Convierte `TABs` en espacios; útil debido a que la representación del `TAB` puede ser diferente en distintos sistemas

Formato:

`expand` [opciones] *fichero* ...

Algunas opciones:

- `-t` n reemplaza cada `TAB` por n espacios (por defecto, 8)
- `-i` solo reemplaza los `TABs` de principio de línea

Ejemplos:

```
$ cat hola.c
main() {
        for(i=0; i<10;i++)
                printf("Hola mundo!\n");
}
```

```
$ expand -t 2 hola.c
main() {
  for(i=0; i<10;i++)
    printf("Hola mundo!\n");
}
```

El comando `unexpand` hace la operación contraria

13) fmt

Formatea cada párrafo, uniendo o separando líneas para que todas tengan el mismo tamaño

Algunas opciones:

- `-`n o `-w` n pone la anchura de las líneas a n (por defecto, 75)
- `-c` conserva la indentación a principio de línea y alinea a la izquierda la segunda línea

- `-s` las líneas pueden dividirse, no unirse
- `-u` uniformiza el espaciado entre palabras

Ejemplo:

```
$ cat quijote.txt
En un lugar de la Mancha, de    cuyo nombre no
quiero acordarme, no ha mucho tiempo
que vivía un
hidalgo    de los de lanza en astillero, adarga
antigua, rocín flaco y galgo corredor.

$ fmt -w 45 -u quijote.txt
En un lugar de la Mancha, de cuyo nombre
no quiero acordarme, no ha mucho tiempo
que vivía un hidalgo de los de lanza en
astillero, adarga antigua, rocín flaco y
galgo corredor.
```

14) od

Muestra un fichero en octal, hexadecimal o otros formatos; en cada línea muestra (en la primera columna) el *offset*

Formato:

`od` [opciones] *fichero*

Algunas opciones:

- `-t` *TIPO* especifica el formato de la salida (por defecto octal): `o` para octal, `x` para hexadecimal, `d` para decimal, `c` para caracteres ASCII, `a` para caracteres con *nombre...*
- `-A` *TIPO* especifica el formato del offset (por defecto octal): `o`, `x`, `d` como antes, `n` para que no aparezca
- `-w` *BYTES* número de bytes por linea (por defecto 16)

Ejemplo:

```
$ od -t x -A x quijote.txt
000000 75206e45 756c206e 20726167 6c206564
000010 614d2061 6168636e 6564202c 79756320
000020 6f6e206f 6572626d 206f6e0a 65697571
...
```

2.5. awk

Lenguaje diseñado para procesar datos basados en texto; el nombre AWK deriva de los apellidos de los autores.

- los administradores de sistemas utilizan `awk` para procesar los ficheros de configuración y logs de los sistemas
- estos ficheros, normalmente, se organizan en forma de *tabla* (líneas compuestas por campos). `awk` es ideal para tratar esos ficheros
- sólo veremos algunos de los aspectos más importantes del uso de `awk` para el manejo de ficheros de texto

2.5.1. Funcionamiento básico

`awk` lee el fichero que se le pase como entrada (o la entrada estándar) línea a línea, y sobre cada línea ejecuta una serie de operaciones

El programa más sencillo de `awk` consiste en imprimir un mensaje de texto por cada línea que recibe de entrada:

```
'{ print "Hola mundo!" }'
```

Nótese las comillas simples y las llaves al principio y al final, que son obligatorias.

Formas de ejecutar `awk`

- Usando la entrada estándar: si no especificamos fichero, `awk` usa la entrada estándar como es usual:

  `awk` *PROGRAMA*

  Ejemplo:

  ```
  $ awk '{ print "Hola mundo!" }'
  $ (introducimos enter)
  Hola mundo!
  $ (introducimos enter)
  Hola mundo!
  ```

- Tomando la entrada de un fichero:

```
awk PROGRAMA fichero_entrada
```

Ejemplo:

```
# creamos un fichero con dos líneas vacías (dos enter):
$ echo -e "\n" > fichero

# ejecutamos awk sobre este fichero:
$ awk '{ print "Hola mundo!" }' fichero
Hola mundo!
Hola mundo!
```

- Escribiendo el programa en un fichero:

```
awk -f FICHERO_PROGRAMA entrada/fichero_entrada
```

Ejemplo:

```
# Partimos del programa awk en un fichero:
$ cat hola.awk
    { print "Hola mundo!" }
$ chmod +x hola.awk

# ejecutamos el programa sobre una entrada de dos líneas:
$ echo -e "\n" | awk -f hola.awk
Hola mundo!
Hola mundo!
```

- Ejecutando el *FICHERO_PROGRAMA* como un script:

poner `#!/usr/bin/awk -f`
al principio de *FICHERO_PROGRAMA*

Ejemplo:

```
$ cat hola.awk
     #!/usr/bin/awk -f
     { print "Hola mundo!" }
$ chmod +x hola.awk

# Ejecutamos el script sobre una entrada de dos líneas:
$ echo -e "\n" | ./hola.awk
Hola mundo!
Hola mundo!
```

Estructura de un programa `awk` Un programa `awk` tiene tres secciones:

1. Parte inicial, que se ejecuta sólo una vez, antes de empezar a procesar la entrada:

   `BEGIN { operaciones }`

2. Parte central, con instrucciones que se ejecutan para cada una de las líneas de la entrada; tienen en siguiente formato:

   `/PATRÓN/ { operaciones }`

   las *operaciones* se realizan sólo sobre las líneas que verifiquen la ER indicada en *PATRÓN*

   - si no ponemos ningún *PATRÓN* las *operaciones* se realizan sobre todas las líneas
   - si ponemos `!/PATRÓN/` las operaciones se ejecutan en las líneas que no concuerden con el patrón

   La parte central es la que toma por defecto el intérprete `awk`

3. Parte final, se efectúa sólo una vez, después de procesar la entrada:

   `END { operaciones }`

Ejemplo de script:

```
#!/usr/bin/awk -f
BEGIN{ print "Inicio" }
{ print "Hola mundo!" }
END{ print "Final" }
```

La ejecución de este script dará como resultado:

```
Inicio
Hola mundo!
Hola mundo!
Final
```

2.5.2. Manejo de ficheros de texto

`awk` divide las líneas de la entrada en campos:

- la separación entre campos la determina la variable `FS` (por defecto, uno o más blancos y `TABs`)
- las variables $1, $2, . . . , $N contienen los valores de los distintos campos
  - $0 contiene la línea completa

Ejemplos:

1. Ejemplo simple:

```
$ awk '{ print $1, $3 }'
$ uno dos tres cuatro → uno tres
$ cinco seis siete ocho → cinco siete
```

2. Lista los ficheros (comando `ls`) y selecciona las campos 9 y 5:

```
$ ls -ldh * | awk '{print "Fichero ", $9, "ocupa ", $5, "bytes"}'
Fichero fich1 ocupa 36 bytes
Fichero fich2 ocupa 9,1K bytes
Fichero fich3 ocupa 3,7M bytes
```

3. Lista las particiones (comando `df`) e imprime su ocupación:

```
$ df -h | sort -rnk 5,5 |  \
> awk 'BEGIN { print "Nivel de ocupación" }  \
> /^\/dev\/sd/ { print "Partición ",$6,": ",$5 }  \
> END { print "Terminado" }'
Nivel de ocupación
Partición /home : 87% ocupación
Partición /var : 51% ocupación
Partición / : 38% ocupación
Terminado

$ # Lo mismo usando un fichero
$ cat ocupacion.awk
BEGIN { print "Nivel de ocupación" }
/^\/dev\/sd/ { print "Partición ",$6,": ", $5 }
END { print "Terminado" }
$ df -h | sort -rnk 5,5 | awk -f ocupacion.awk
```

Variables predefinidas: `awk` tiene un conjunto de variables predefinidas, por ejemplo, nos permite especificar el separador de campos

Esas variables son:

Nombre	Significado
`FS`	Carácter separador entre campos de entrada (por defecto, blanco o tabulado)
`NR`	Número de registros de entrada (líneas)
`NF`	Número de campos en el registro de entrada
`RS`	Carácter separador entre registros de entrada (por defecto, nueva línea)
`OFS`	Carácter separador entre campos en la salida (por defecto, un espacio en blanco)
`ORS`	Carácter separador entre registros de salida (por defecto, nueva línea)
`FILENAME`	Nombre del fichero abierto

Ejemplo:

```
$ cat usuarios.awk
BEGIN { FS = ":"; OFS = "-->"; ORS = "\n=========\n"; }
{ print NR, $1, $5 }
$ awk -f usuarios.awk /etc/passwd
...
37 -->tomas -->Tomás Fernández Pena„,
==========
38 -->caba -->José Carlos Cabaleiro Domínguez„,
==========
...
```

2.5.3. Otras características

`awk` es un lenguaje completo:

- permite definir variables de usuario
- permite realizar operaciones aritméticas sobre las variables
- permite utilizar condiciones, lazos, etc.
- permite definir funciones

La sintaxis de `awk` es prácticamente idéntica a la del lenguaje C

- podemos usar `printf` en lugar de `print` (con la sintaxis de C)
- también podemos usar arrays

Ejemplos:

1. Realiza acumulaciones (se finaliza con CNTL-D)

```
$ cat acumula.awk
BEGIN{ sum=0; print "Introduce números" }
{ sum=sum+$1; }
END{ print "La suma acumulada es: ", sum}
```

2. Lista el tamaño de los ficheros y el tamaño total

```
$ cat lista-ficheros.awk
BEGIN{ total = 0; }
{   total += $5;
   printf("Fichero%s ocupa%d bytes\n", $8,$5); }
END{ printf("Ocupación total =%d bytes\n",total); }

$ ls -ld * | awk -f lista-ficheros.awk
Fichero ancestros.awk ocupa 370 bytes
Fichero hola.c ocupa 66 bytes
Fichero lista-ficheros.awk ocupa 143 bytes
Ocupación total = 579 bytes
```

3. Muestra una advertencia si el nivel de ocupación de una partición supera un límite

```
$ cat ocupacion2.awk
BEGIN { limite = 85; }
/^\/dev\/sd/ { if($5 >limite)
printf("PELIGRO: el nivel de ocupación de%s es%s\n%", $6, $5);}

$ df -ah | tr -d '%' | awk -f ocupacion2.awk
PELIGRO: el nivel de ocupación de /home es 87%
```

Paso de parámetros: es posible pasar parámetros en la llamada a `awk`
Ejemplo: Indicando el `PID` de un proceso obtiene el `PID` de todos sus ancestros (padres, abuelos, . . .)

```
$ cat ancestros.awk
BEGIN { ind=0; }
function padre(p) {
   for(i=0; i <ind; i++)
     if(pid[i] == p) return(ppid[i]);
}
!/PID/ { pid[ind]=$3; ppid[ind]=$4; ind++; }
END {
   do {
     printf("%d --> ", proc); proc = padre(proc);
   } while(proc >= 1);
   printf("\n\n");
}

$ ps axl | awk -f ancestros.awk proc=4258
4258 --> 3326 --> 1 -->
```

Arrays asociativos: awk permite el uso de arrays asociativos, es decir, que pueden tener como índice una cadena de caracteres
Ejemplo

```
$ cat usuarios2.awk
BEGIN { FS = ":"}
{ nombre[$1] = $5; }
END {
  for(;;){
    printf("Nombre de usuario: ");
    getline user < ";
    if( user ==  ) break;
    printf("<%s>: %s\n", user, nombre[user]);
  }
}

$ awk -f usuarios2.awk /etc/passwd
Nombre de usuario: tomas
<tomas>: Tomás Fernández Pena„,
Nombre de usuario:
```

2.6. Python

Además de Bash existen otros lenguajes adecuados para la creación de scripts de administración. En la actualidad el lenguaje de script que domina a todos los demás es Python.

Principales características:

- Énfasis en la legibilidad
- Uso de identación para delimitar bloques de código
- Soporte de diversos paradigmas: imperativo, orientado a objetos y funcional
- Sistema de tipos dinámico y gestión automática de memoria
- Gran librería con módulos para múltiples tareas

Hay dos versiones actuales de Python, la 2.7 y la 3.x, que son incompatibles entre sí. Por ejemplo, la sentencia `print` de Python 2.7 ha sido remplazada por la función `print()` (que requiere paréntesis) en Python 3.x. En este curso utilizaremos Python 3.x.

Ejemplo sencillo que abre un fichero e imprime cada línea:

```
#!/usr/bin/env python3
# Abre el fichero sólo lectura
try:
  f = open('/etc/passwd','r')
except IOError:
  print('No puedo abrir /etc/passwd')
else:
  # Lee las líneas en una lista
  lista = f.readlines()
  # Recorre e imprime la lista
  for l in lista:
    print(l, end='')    # end='' elimina el retorno de línea
  f.close()
```

Nótense los : y los sangrados del texto (mediante espacios o tabulados) que definen los bloques de código. La codificación por defecto de los programas en Python3.x es utf-8.

2.6.1. Tipos de datos en Python

Python usa un tipado dinámico en donde son los valores, no las variables, las que definen el tipo de dato (*int, float, complex, bool, str, ...*). Los enteros tienen una precisión ilimitada, mientras que los float suelen tener la precisión de *double* en C. Las cadenas pueden delimitarse por comillas simples o dobles. En general, no es necesario indicar el tipo de dato. Por ejemplo,

```
a=5      a=7.8      a=5+3j      a=True      a='a'      a='cadena'
```

Otro ejemplo, la función `input` por defecto obtiene una cadena, mientras que `print` puede imprimir cualquier tipo de datos.

```
cadena=input('introduce una cadena: ')
a=int( input('introduce un entero: ') )
b=float( input('introduce un float: ') )
print( 'datos: ', cadena, a, b )
```

- Podemos convertir de un tipo de dato a otro, por ejemplo, `int(x)`, `float(x)` y `str(x)` convierten a entero, float y a cadena, respectivamente.
- Entre las operaciones aritméticas se incluyen +, −, *, /, // (división entera), % (resto), $x**y$ (potencia)
- Las operaciones lógicas bit a bit entre enteros incluyen $x|y$ (or), $x\hat{}y$ (exor), $x\&y$ (and), $x << n$ (desplazamiento a la izquierda), $x >> n$ (desplamiento a la derecha) y $\sim x$ (negación).
- Por último, las operaciones lógicas incluyen `and`, `or` y `not`.

Colecciones

Python proporciona diferentes tipos de colecciones (arrays):

1. Listas: mutables (se pueden modificar), que pueden contener tipos mezclados (string, enteros, etc.) Se definen mediante corchetes o usando el procedimiento `list()`.

   ```
   frutas=['naranjas', 'uvas', 123, 'limones', 'uvas']
   frutas.append('peras')
   frutas.remove(123)
   frutas.remove('uvas') # [naranjas,limones,uvas,peras]
   frutas[2:2] = ['fresas', 'pomelos'] # inserta en pos 2
   ```

```
print(frutas)       # naranjas,limones,fresas,pomelos,uvas,peras
print(len(frutas))  # 6
print(frutas[0:3])  # naranjas, limones, fresas
print(frutas[-3])   # pomelos
print(frutas[1:-3]) # limones, fresas
frutas.pop()        # Elimina el último elemento
del frutas[2:4]     # Elimina los elementos 2 y 3
frutas.sort()       # Ordena
print(frutas)       # [limones,naranjas,uvas]
a=list('hola')      # a=['h','o','l','a']
'o' in a            # True
```

Nótese que el último elemento puede referenciarse como -1, que el rango [0:3] incluye los elementos 0,1,2 y que la inserción en [2:2] sustituye los elementos comprendidos entre 2 y 2-1=1 (por tanto no se borra ninguno de los existentes). Además, podemos desglosar fácilmente los caracteres de una cadena.

Las listas pueden enlazarse

```
a = [ [0,1,3,4], [5,6,7,8,8], [9,10] ]
a.append([13,14,15,16])
print( a[2][0] )  # 9
del a[1]
print( a )        # [[0,1,3,4], [9,10], [13,14,15,16]]
```

2. Tuplas: listas inmutables (no se pueden modificar). Se definen usando paréntesis (o sin ellos) o mediante el procedimiento `tuple()`.

```
meses=('enero','febrero','marzo','abril', 'mayo', 'junio',\
'julio','agosto','septiembre','octubre','noviembre',\
'diciembre')        # Los paréntesis son opcionales
print(meses[3])     # abril
'sabado' in meses   # False
'enero' in meses    # True
```

3. Conjuntos (Sets): sin elementos duplicados. Se generan con `set()`.

```
cesta=['naranjas', 'uvas', 'limones', 'uvas']  # una lista
frutas=set(cesta)                  # generamos un conjunto
print(frutas)                      # naranjas,uvas,limones
a = set('abracadabra')
b = set('alacazam')
```

```
print( a )     # 'a', 'r', 'b', 'c', 'd'
print( a-b )   # 'r', 'b', 'd'
print( a | b ) # 'a', 'c', 'b', 'd', 'm', 'l', 'r', 'z'
print( a & b ) # 'a', 'c'
print( a ^ b ) # 'b', 'd', 'm', 'l', 'r', 'z'
```

4. Diccionarios: constan de clave (`key`) y valor (`value`). Se definen usando llaves o con el procedimiento `dict`.

```
d=dict(a=1, b=2, c=3)
d=dict([('a',1),('b',2),('c',3)])
edad_de = {'Eva':23, 'Ana':19, 'Oscar':41}
print edad_de['Ana']  # Imprime 19
edad_de['Eva'] = 18   # Cambia un valor
edad_de['Juan'] = 26  # Añade un elemento
del edad_de['Oscar']  # Borra un elemento
edad_de.keys()        # ['Eva', 'Juan', 'Ana']
edad_de.values()      # [18, 26, 19]
for key,value in edad_de.items():
   print(key,'->',value)
```

Referencias

La copia de una variable referencia a un objeto diferente, mientras que la copia de una colección referencia al mismo objeto.

```
a = 1      # a apunta al objeto 1
b = a      # b apunta al objeto 1
a += 5     # ahora a apunta al objeto 6
print( b ) # b no se ve modificado, a=6, b=1 objetos diferentes
a = [1, 2] # a apunta al objeto lista
b = a      # a y b referencian al mismo objeto, la lista
a[0] += 5  # se modifica el objeto lista
print( b ) # b=[6, 2] tambien refleja la modificación
```

Para que las listas referencien a diferentes objetos:

```
a = [1, 2] # nuevo objeto lista
b = a[:]   # a y b referencian a objetos diferentes
a[0] += 5  # se modifica el objeto a
print( b ) # [1, 2], b no se ha modificado
c=list(a)  # otra forma
```

Generación de listas

```
l = range(5)         # l = [0, 1, 2, 3, 4]
l = range(2, 5)      # l = [2, 3, 4]
l = range(2, 10, 3)  # l = [2, 5, 8]
l = range(5, -5, -2) # l = [5, 3, 1, -1, -3]
s = sum(range(1,4))  # s = 6
# iniciar una lista
a = [0]*3            # a = [0, 0, 0]
# iniciar una lista enlazada
for i in range(3):
   a[i]=[0]*2        # a = [ [0, 0], [0, 0], [0, 0] ]
```

Matrices

Python no incluye ningún tipo para matrices. Sin embargo, podemos tratar una lista enlazada como una matriz. Por ejemplo:

```
A = [[1, 4, 5],
     [-5, 8, 9]]
```

Podemos referenciar a sus elementos, por ejemplo, `A[0][0]=1`, `A[0][1]=4`, etc. También podemos referenciar a las filas, por ejemplo `A[0]=[1,4,5]`.

Además, los módulos y librerías de Python pueden extender o introducir sus propios tipos de datos. Por ejemplo, el módulo `NumPy` introduce el tipo de dato `array` con numerosas operaciones predefinidas sobre arrays y matrices.

2.6.2. Control de flujo

Lazos

```
frutas=['naranjas', 'uvas']
for f in frutas:
  print( f, len(f) ) # naranjas, 8; uvas, 4

for i in range(len(frutas)):
  print( i, frutas[i] ) # 0, naranjas; 1, uvas

nf = input('Añade otra fruta: ')
while nf:              # Si la entrada no está vacía
  frutas.append(nf)    # añádela a la lista
  nf = input('Añade otra fruta: ')
```

Condicionales

Hay 8 tipos de comparaciones en Python, que tienen todas la misma prioridad: < (menor), <= (menor o igual), > (mayor), >= (mayor o igual), == (igual), ! = (distinto), `is` (identidad de objeto) y `is not` (identidad de objeto negativa). Por ejemplo:

```
x = int(input('Introduce un entero: '))
if x < 0:
  x = 0
  print( 'Negativo cambiado a 0' )
elif x == 0:
  print( 'Cero' )
else:
  print( 'Positivo' )
```

Funciones

- Ejemplo 1:

```
def opera(a,b):
  c=a+b
  d=a-b
  return(c,d)

suma,resta=opera(8,7.5)
```

- Ejemplo 2 (nótese el valor por defecto de `nf`):

```
def compra(fr, nf='manzanas'):
  fr.append(nf)

frutas=[]     # También frutas=list()
compra(frutas, 'peras')
compra(frutas)
compra(nf='limones', fr=frutas)
print( frutas )  # peras, manzanas, limones
```

Funciones con un numero arbitrario de argumentos

Si ningún argumento es un diccionario:

```
def fun(*args):
  for arg in args:
     print( arg )

fun('peras', [1,2,3], 6)  ->  peras
                          ->  [1,2,3]
                          ->  6
```

En general, se distinguen dos tipos de argumentos: diccionarios (con dos asteriscos) y el resto (con un asterisco).

```
def fun(*args, **kwargs):
  for arg in args:
     print( arg )
  for kw in kwargs.keys():
      print( kw, kwargs[kw] )

fun('peras', 1, manzanas=2, limones=3)  ->  peras
                                        ->  1
                                        ->  limones 3
                                        ->  manzanas 2
```

Funciones en ficheros separados

Se utiliza la sentencia `import`

```
$ cat myutils.py
#!/usr/bin/env python3
def sqrt(x): return x ** 0.5
def double(x): return x + x
def main(): print( sqrt(5), double(5) )
if __name__ == "__main__":
   main()

$ cat main.py
#!/usr/bin/env python3
import myutils
print( myutils.sqrt(42) )
```

Cuando el intérprete de Python lee un fichero ejecuta todo el código que encuentra en el. Si el fichero es importado desde otro programa Python, puede interesarnos que el programa principal del archivo importado no se

ejecute, sino que solo se incluyan las funciones. Es por ello, que se ha escrito el condicional `if __name__ == "__main__"`. De esta forma:

1. Si el fichero se invoca directamente se ejecuta la función `main()`.
2. Si se invoca desde otro fichero no se ejecuta la función `main()`.

2.6.3. Orientación a objetos

A continuación se muestra la definición de una clase:

```
class fruteria:
   '''Ejemplo simple de clase'''
   def __init__(self, f):
       self.stock = list()
       self.stock.append(f)
   def compra(self, f):
       self.stock.append(f)
   def vende(self, f):
       if f in self.stock:
           self.stock.remove(f)
       else:
           print( f, 'no disponible' )
```

El método `__init__` representa el constructor en Python. Cuando se llama a la clase, Python crea un objeto y le pasa como primer parámetro la variable `self`, que representa la instancia del objeto en sí. A continuación se le pasa el resto de los parámetros.

En cuanto al programa principal:

```
mi_fruteria = fruteria('pera')
mi_fruteria.compra('manzana')
print( mi_fruteria.stock )    # ['pera', 'manzana']
mi_fruteria.vende('pera')
mi_fruteria.vende('platano')  # platano no disponible
print( mi_fruteria.stock )    # ['manzana']
mi_fruteria.vende('pera')     # pera no disponible
```

Métodos y atributos privados

Los métodos o atributos privados se definen con dos guiones bajos antes del nombre (y no pueden terminar en dos guiones bajos)

```
class Ejemplo:
    def publico(self):
        print( 'Uno' )
        self.__privado()

    def __privado(self):
        print( 'Dos' )

ej = Ejemplo()
ej.publico()   # Imprime Uno Dos
ej.__privado() # Da un error
```

Herencia múltiple

Se permite la herencia múltiple:

```
class fruteria:
    def que_vendo(self):
        print( 'Vendo frutas' )

class carniceria:
    def que_vendo(self):
        print(  'Vendo carne' )

# Herencia múltiple
class tienda(carniceria, fruteria):
    pass  # operación nula

# La clase carniceria está primera
# en la definición de tienda
tienda().que_vendo() # Vendo carne
```

Nota: La sentencia `pass` es una operación nula, es decir, no hace nada cuando se ejecuta, pero ayuda a mantener el formato del lenguaje.

2.6.4. Módulos y librerías

Existe una colección innumerable de módulos y librerías para Python. Se utilizan mediante la sentencia `import`. Dos ejemplos:

```
import math
print( math.cos(5) )
```

```
from math import cos, sin
print( cos(5), sin(5) )
```

1) Procesamiento de textos

Python incorpora numerosos métodos para procesar cadenas de texto:

```
# Elimina caracteres y separa por espacios
l = 'Hola que tal!'.strip('!').split()
# l=['Hola', 'que', 'tal']

# Une utilizando un caracter
s = ','.join(l)  # s='Hola,que,tal'

# Cuenta el número de ocurrencias de un caracter
c = s.count(',') # c=2

# Reemplaza un caracter por otro
ss = s.replace(',', '\t')  # ss='Hola    que    tal'

# Separa por otro tipo de caracter, e invierte la lista
l=ss.split('\t')
l.reverse()     # l=['tal', 'que', 'Hola']

# Localiza una subcadena en el string
c=ss.find('tal')  # c=9
c=ss.find('tall') # c=-1 (no encuentra la subcadena)

# Separa por líneas
ml = '''Esto es
un texto con
varias lineas'''
l = ml.splitlines()
# l=['Esto es', 'un texto con', 'varias lineas']
```

2) Expresiones regulares

El módulo `re` permite trabajar con expresiones regulares.

- Comprueba palabras

```
import sys, re
s=input('Introduce una palabra: ')
if re.match('^[A-Z]+$', s):
   print( 'Todas las letras mayusculas' )
```

- Busca URLs en un fichero de texto

```
import sys, re
try:
  f = open('fich.txt','r')
except IOError:
  print( 'No puedo abrir' )
  sys.exit(1)
for l in f:
  # Busca todas las URLs en la línea actual y guárdalas (sin http)
  # en la lista h. [^\s]+ es cualquier caracter distinto de SP, TAB y CR
  h = re.findall('http://([^\s]+)', l)
  if h:          # Si la lista no está vacía
    for w in h: # recorrela e imprime las URLs
        print( w )

# Separa un string en una lista
s = 'Uno:Dos.Tres-Cuatro'
l = re.split('[:.-]', s)
```

3) Parámetros y funciones dependientes del sistema

El módulo sys incluye parametros y funciones dependientes del sistema

- `sys.argv` Lista de argumentos en línea de comandos (`sys.argv[0]` es el nombre del script)
- `sys.exit(`*`[code]`*`)` Termina el script con código de salida *`code`*

4. Subprocesos

El módulo `subprocess` permite lanzar subprocesos, por ejemplo, comandos del SO o scripts de bash

1. Para ejecutar un comando redireccionando la salida estándar y de error a /dev/null (también podría enviarse a un fichero)

```
#!/usr/bin/env python3
import subprocess
code = subprocess.call(['mkdir', '/tmp/dir'],
    stdout=open('/dev/null','w'), stderr=open('/dev/null','w') )
if code==0:
    print('Ejecucion correcta')
else:
    print('Error en la ejecucion')
```

2. Para ejecutar un comando del sistema operativo y leer su salida:

```
#!/usr/bin/env python3
import subprocess
try:
    output = subprocess.check_output(['echo', 'Hello World!'],
       encoding='utf8')
except:
    print( 'No puedo ejecutar' )
else:
    print(output, end='')
```

3. De forma similar, para ejecutar un script de bash y leer su salida:

```
#!/usr/bin/env python3
import subprocess
try:
    output = subprocess.check_output(['./script.sh', 'parametro'],
       encoding='utf8')
except:
    print('No puedo ejecutar')
else:
    print(output, end='')
```

4. Ejecuta un comando y la salida estándar y de error va al objeto p

```
#!/usr/bin/env python3
import subprocess
p = subprocess.Popen(['df', '-h'], stdout=subprocess.PIPE,
    stderr=subprocess.PIPE, encoding='utf8')
# Lee e imprime las lineas de la salida y de error de df -h
out = p.stdout.readlines()
for line in out:
```

```
        print(line, end='')
    err = p.stderr.readlines()
    for line in err:
        print(line, end='')
```

5) Parseo de argumentos

El módulo `argparse` parsea las opciones en línea de comandos

6) Operaciones dependientes del sistema operativo

os Uso de funcionalidades dependientes del SO

- `os.getlogin()` nombre de login del usuario
- `os.getloadavg()` carga media del sistema
- `os.getcwd()` obtiene el directorio actual
- `os.chdir(`*`path`*`)` cambia el directorio actual a *`path`*
- `os.listdir(`*`path`*`)` lista de todas las entradas del directorio *`path`*

os.path Manipulación de ficheros y/o directorios

- `os.path.isfile(`*`path`*`)` True si *`path`* es un fichero regular
- `os.path.split(`*`path`*`)` Divide *`path`* en directorio+fichero
- `os.path.splitext(`*`path`*`)` Divide *`path`* en nombre_fichero+ extensión
- `os.path.getsize(`*`path`*`)` Devuelve el tamaño de *`path`*

glob Expansión de nombres de ficheros estilo UNIX (*globbing*)

- `glob.glob(`*`expr`*`)` Lista de ficheros indicados por *`expr`* (puede contener comodines)

shutil Operaciones de alto nivel con ficheros

- `shutil.copy(`*`src`*`, `*`dst`*`)` Copia el fichero *`src`* al fichero o directorio *`dst`*
- `shutil.move(`*`src`*`, `*`dst`*`)` Mueve recursivamente un fichero o directorio

tempfile Genera ficheros y directorios temporales

- `tempfile.NamedTemporaryFile()` Crea un fichero temporal con nombre

Ejemplo: Renombrar en un directorio los ficheros *.xml a *.html

Utilizamos `argparse` para recoger los argumentos:

```
#!/usr/bin/env python3
import os.path, glob, shutil, sys, argparse
def main():
    # definimos un parseador
    parsear = argparse.ArgumentParser(description='Renombra XML a HTML')
    # le añadimos los argumentos que queramos
    parsear.add_argument('directory', help='nombre de directorio')
    # recogemos todos los valores introducidos
    args = parsear.parse_args()
    # seleccionamos uno de los argumentos
    dir = args.directory

    # Chequea que sea un directorio
    if not os.path.isdir(dir):
        print( dir + ' no es un directorio' )
        sys.exit(1)
    try:
        os.chdir(dir) # Cambia al directorio
        # Recorre los ficheros .xml
        for f in glob.glob('*.xml'):
            # Construye el nuevo nombre y renombra los ficheros
            new = os.path.splitext(f)[0] + '.html'
            shutil.move(f, new)
    except:
        print( 'Hubo un problema ejecutando el programa.' )

if __name__ == "__main__":
    main()
```

7) Operaciones con ficheros comprimidos

Los módulos `gzip, bz2, zipfile` y `tarfile` permiten el manejo de ficheros comprimidos

Ejemplo: acciones sobre un tar, seleccionándolas de un menú

```
#!/usr/bin/env python3
import tarfile, sys
```

```
try:
  f = True
  while f:
    # Abre el fichero tar (especificado como argumento)
    tar = tarfile.open(sys.argv[1], 'r')

    # Presenta el menú y obtiene la selección
    selection = input('''
    Selecciona
       1 para extraer un fichero
       2 para mostrar información sobre un fichero en '''
         + sys.argv[1] + '''
       3 para listar los ficheros de ''' + sys.argv[1] + '''
       4 para terminar''' + '\n')

    # Realiza la acción en función de la selección
    if selection == '1':
      filename = input('Indica el fichero a extraer:  ')
      tar.extract(filename)
    elif selection == '2':
      filename = input('Indica el fichero a inspeccionar:  ')
      for tarinfo in tar:
        if tarinfo.name == filename:
          print( '\nNombre:\t', tarinfo.name,
          '\nTamaño:\t', tarinfo.size, 'bytes\n' )
    elif selection == '3':
      print( tar.list(verbose=True) )
    elif selection == '4':
      f = False
    else:
        print( 'Selección incorrecta' )
except:
    print( 'Hubo un problema ejecutando el programa.' )
```

Referencias

- Python Official Website: página principal de Python
- The Python Tutorial: un buen sitio para empezar
- The Python Standard Library: la librería estándar
- Python Para Todos: libro en PDF descargable y gratuito

Capítulo 3

Actividades administrativas básicas

3.1. Gestión de procesos

Un proceso es una instancia de un programa en ejecución. En cada momento se están ejecutando un gran número de procesos:

- procesos de sistema (kernel, servicios o *daemons*)
- procesos de usuarios

Una CPU puede constar de varios núcleos. El planificador de procesos se encarga de asignar los procesos a los núcleos en función su prioridad. En la planificación con asignación dinámica los procesos pueden cambiar de núcleo al pasar a estado activo si el último núcleo en el que se ejecutó está ocupado y existe otro núcleo ocioso. No obstante, la migración de un proceso de un núcleo a otro es costosa. Por tanto, siempre que sea posible se intentará evitar para no incurrir en la penalización de los fallos de caché que resultan de la migración de procesos.

Un proceso se pueden componer de varios subprocesos (hilos). Tanto los procesos como los hilos son secuencias independientes de ejecución. La diferencia es que los hilos (del mismo proceso) comparten su espacio de direcciones virtuales y recursos del sistema, mientras que los procesos se ejecutan en espacios de memoria separados.

En esta sección trataremos la gestión de los procesos que se ejecutan:

- listar procesos en ejecución
- detener y matar procesos
- controlar la prioridad de ejecución

En concreto, veremos los siguientes programas y ficheros:

Comandos	Función
`ps`, `pstree`, `top`	listan los procesos
`strace`	muestra las llamadas al sistema de un proceso
`fg` / `bg`	ejecución en primer o segundo plano
`jobs`	comandos lanzados desde el shell actual
`kill`	envía señales a procesos por PID
`CNTL-C`, `CNTL-Z`	envían señales de teclado
`pgrep`	busca procesos por nombre
`pkill`, `killall`	envían señales a procesos por nombre
`nohup`	lanza procesos ignorando la señal SIGHUP
`exec`	remplaza un proceso por otro
`nice`, `renice`	fija/cambia la prioridad de un proceso
`ulimit`	controla los recursos del shell actual
`uptime`	información sobre el estado y carga del sistema
`w`	información sobre los usuarios y sus procesos
`free`	información sobre la memoria total, usada y libre
`/proc`, `/sys`	recopilan información del sistema y procesos

3.1.1. Ver los procesos en ejecución

Existen varias herramientas para ver los procesos en ejecución, la más importante es el comando `ps`

`ps` (*process status*)

lista los procesos con su PID (identificador del proceso), TTY (número de terminal), TIME (tiempo de CPU usado) y CMD (línea de comando usada)

```
$ ps
  PID TTY          TIME CMD
 1836 pts/0    00:00:00 bash
 7661 pts/0    00:00:00 pdflatex
```

Sin opciones, `ps` sólo muestra los procesos lanzados desde el terminal actual y con el mismo `EUID` que el usuario que lo lanzó

Opciones de ps. Existe de un gran número de opciones, que se pueden especificar de 3 maneras:

1. opciones UNIX: pueden agruparse y se preceden por un guión: `ps -e`

2. opciones BSD: pueden agruparse y van sin guión: `ps ax`
3. opciones largas GNU: precedidas de dos guiones: `ps --user tomas`

Algunas opciones:

- `-e`: muestra todos los procesos
- `-u` *usuario*: muestra los procesos de un usuario
- `-o` *formato*: permite definir el formato de salida, por ejemplo,

Ejemplos:

- Para ver con formato los procesos lanzados desde el terminal por el usuario:

```
$ ps -o user,pid,state,comm
```

```
USER       PID S COMMAND
amo       1357 S bash
amo       1992 S gedit
amo       2279 R ps
```

- Para ver con formato todos los procesos ordenados por usuario:

```
$ ps -eo user,pid,state,comm --sort user
```

Para más opciones ver la página de manual de `ps`

En cuanto al estado (código `state`), son posibles entre otros:

Código	**significado**
R	*Running* (ejecutándose o en cola de ejecución)
S	*interruptible Sleep* (dormido y esperando un evento, p.e, la entrada de un dato por teclado)
D	*uninterruptible sleep* (usualmente esperando IO, p.e., la copia de un fichero requiere que los datos lleguen desde el disco)
T	*sTopped* (detenido por una señal de control: kill, Ctrl-Z)
Z	*Zombie* (proceso terminado, pero que todavía está en la tabla de procesos porque su padre no lo liberó)

La opción de formato (`o`) nos permite seleccionar un gran número de elementos a visualizar (se pueden ver muchas más en el manual):

Código	Etiqueta	Significado
pid	PID	identificador del proceso
ppid	PPID	identificador del proceso padre
state	S	estado del proceso
ruser	RUSER	usuario
euser	EUSER	usuario efectivo
rgroup	RGROUP	grupo
egroup	EGROUP	grupo efectivo
ruid	RUID	identificador del usuario
euid	EUID	identificador del usuario efectivo
rgid	RGID	identificador del grupo
egid	EGID	identificador del grupo efectivo
psr	PSR	procesador (núcleo) en el que se ejecuta
pcpu	%CPU	porcentaje de CPU usado
pmem	%MEM	porcentaje de memoria usada
rss	RSS	memoria residente (RAM) en kB
vsz	VSZ	memoria virtual en kB
comm	COMMAND	nombre del comando
cmd	CMD	comando con todos sus parámetros
start	START	hora de inicio del proceso
etime	ELAPSED	tiempo transcurrido desde el inicio
time	TIME	tiempo de CPU consumido
pri	PRI	prioridad del proceso (actualmente)
nice	NI	nice (prioridad asignada inicialmente)
tty	TT	terminal de control

pstree muestra el árbol de procesos:

```
systemd-+-ModemManager-+-{gdbus}
        |              `-{gmain}
        +-NetworkManager-+-dhclient
        |                +-dnsmasq
        |                +-{gdbus}
        |                `-{gmain}
        +-accounts-daemon-+-{gdbus}
        |                 `-{gmain}
        +-lightdm-+-Xorg
        |         +-lightdm-+-fvwm2-+-FvwmAnimate
        |         |         |       +-FvwmButtons
        |         |         |       +-xfce4-terminal-+-bash-+-acroread-
        |         |         |       |                |      +-gedit-+-
        ...
```

top

- `top` nos da una lista de procesos actualizada periódicamente

```
top - 17:34:08 up  7:50,  6 users,  load average: 0.12, 0.31, 0.27
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.2% us,  2.0% sy,  0.0% ni, 91.0% id,  0.0% wa,  0.8% hi,  0.0% si
Mem:   1026564k total,   656504k used,   370060k free,    65748k buffers
Swap:  2048248k total,        0k used,  2048248k free,   336608k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6130 root      15   0 63692  48m 9704 S  8.7  4.9   8:03.34 XFree86
 6341 tomas     15   0 14692 8852 6968 S  4.3  0.9   1:55.13 metacity
 6349 tomas     16   0 32792  14m 9232 S  1.3  1.5   0:41.60 gnome-terminal
 6019 tomas     15   0  7084 3184 1896 D  0.3  0.3   0:23.22 famd
 6401 tomas     15   0 16756 8280 6856 S  0.3  0.8   0:02.49 geyes_applet2
 6427 tomas     15   0 18288  10m 8112 S  0.3  1.0   0:09.04 wnck-applet
 7115 tomas     15   0 26312  13m  11m S  0.3  1.4   0:00.61 kio_uiserver
 7390 tomas     15   0 45016  30m  18m S  0.3  3.0   0:38.69 kile
    1 root      16   0  1516  536  472 S  0.0  0.1   0:00.61 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    ......................
```

- en la cabecera nos muestra un resumen del estado del sistema
  - hora actual, tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos
  - número total de tareas y resumen por estado
  - estado de ocupación de la CPU y la memoria. En un sistema de n núcleos el máximo de uso de CPU es $n \times 100\,\%$
- por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU (los más costosos arriba)
- pulsando `h` mientras se ejecuta `top`, obtenemos una lista de comandos interactivos
- los campos de las columnas tienen un significados similar a los del comando `ps` y se pueden seleccionar interactivamente pulsando `f`.
- para salir, `q`

strace

Muestra las llamadas al sistema realizadas por un proceso en ejecución

- Ejemplo de un `strace` sobre un `top` en ejecución

```
$ strace top
gettimeofday({1195811866, 763977}, {4294967236, 0}) = 0
open("/proc/meminfo", O_RDONLY)         = 3
fstat64(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE| ...
read(3, "MemTotal:      2066348 kB\nMemFre"..., 1024) = 728
close(3)                                = 0
munmap(0xb7f55000, 4096)                = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY) = 3
fstat64(3, {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
fcntl64(3, F_SETFD, FD_CLOEXEC)         = 0
getdents(3, /* 52 entries */, 1024)     = 1020
getdents(3, /* 64 entries */, 1024)     = 1024
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)          = 4
read(4, "1 (init) S 0 1 1 0 -1 4194560 44"..., 1023) = 185
close(4)
......
```

Ejecución en segundo plano

Por defecto, los comandos corren en primer plano (*foreground*): el shell espera a que termine el comando antes de aceptar uno nuevo

- para ejecutar un comando en segundo plano (*background*) se añade &

  ```
  $ sleep 10 &
  ```

- para terminar un proceso en foreground `Ctrl-C`
- para pausar un comando en foreground usar `Ctrl-Z`
  - `bg` pasa el proceso a background
  - `fg` lo devuelve a foreground
- para pasar un proceso de foreground a background: `Ctrl-Z` y luego `bg`
- Ejemplo:

```
$ sleep 20
Ctrl-Z
[3]+ Stopped     sleep 20
$ bg
[3]+ sleep 20 &
$ fg
sleep 20
```

- El comando `jobs` permite ver la lista de comandos (*jobs*) en background lanzados desde el shell, así como su estado (`fg` y `bg` pueden actuar sobre uno de los jobs identificándo por su número)

```
$ gedit nada.txt &; sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
$ fg 3
sleep 100
Ctrl-Z
[3]+ Stopped     sleep 100
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Stopped     sleep 100
$ bg 3
[3]+ sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
```

3.1.2. Señalización de procesos

El comando básico para enviar señales a un proceso es `kill`

- Si el proceso se lanzó en foreground se pueden enviar algunas señales desde teclado
- `kill -l` lista el conjunto de señales:

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      ....
```

- para ver la función de cada señal, ver `man 7 signal`

Sintaxis de `kill`

`kill` *`[señal]`* `PID`

- *señal* puede indicarse mediante el número o el código:
  - `kill -9` / `kill -SIGKILL` / `kill -KILL` son equivalentes

Las señales más comunes son:

- `SIGTERM` (15): mata el proceso permitiéndole terminar correctamente
- `SIGKILL` (9): mata el proceso inmediatamente (no puede ser ignorada)
- `SIGSTOP` (19): detiene temporalmente el proceso
- `SIGCONT` (18): continúa si el proceso fue parado
- `SIGINT` (2): interrupción de teclado (Ctrl-C), mata el proceso
- `SIGTSTP` (20): stop de teclado (Ctrl-Z), para temporalmente el proceso
- `SIGQUIT` (3): salida de teclado (Ctrl-\), similar a Ctrl-C, pero realiza un volcado de memoria (*core dumped*)
- `SIGHUP` (1): enviada en caso de cuelgue del terminal o muerte del proceso controlador, también se utiliza para reiniciar un *daemon*.

Algunas características de las señales:

- Excepto `SIGKILL` (matar) y `SIGSTOP` (parar), las demás señales pueden ser ignoradas o gestionadas por el proceso.
- La señal que se envía por defecto es `SIGTERM` (terminación)
  - los procesos pueden ignorar esta señal y no terminar
  - la señal equivalente de teclado es `SIGINT` (interrupción, Cntl-C)
- `SIGSTOP` y `SIGTSTP` se utilizan para detener temporalmente un proceso, la primera desde un programa y la segunda desde teclado (Cntl-Z)
- Cuando cerramos un shell o un terminal en un entorno gráfico, se envía un `SIGHUP` (*hang up*, cuelgue) a todos sus hijos, que suelen responder finalizando también.
- Por otro lado, la mayoría de los procesos de servicio del sistema (*daemons*) responden a la señal `SIGHUP` volviendo a leer sus ficheros de configuración.

Ejemplos:

```
$ yes >/dev/null &
[1] 9848
$ yes >/dev/null &
[2] 9849
$ ps
 PID TTY      TIME  CMD
9834 pts/7 00:00:00 bash
9848 pts/7 00:00:02 yes
9849 pts/7 00:00:01 yes
9850 pts/7 00:00:00 ps
$ kill -STOP 9849
[2]+ Stopped         yes >/dev/null
$ jobs
[1]- Running         yes >/dev/null &
[2]+ Stopped         yes >/dev/null
$ kill -CONT 9849
$ jobs
[1]- Running         yes >/dev/null &
[2]+ Running         yes >/dev/null &
$ kill -KILL 9848
```

```
$ kill -HUP 9849
[1]- Matado          yes >/dev/null
[2]+ Colgar          yes >/dev/null
```

Otros comandos

nohup normalmente, cuando salimos de un login shell (`logout`) o cerramos un terminal, se envía una señal `SIGHUP` a todos los procesos hijos[1]:

- si lanzamos un proceso en background y salimos de la sesión el proceso se muere al morir el shell desde el que lo iniciamos.

El comando `nohup` permite lanzar un comando ignorando las señales `SIGHUP`

`nohup` *`firefox`*

- la salida del comando se redirige al fichero `nohup.out`

pgrep busca en la lista de procesos para localizar el PID a partir del nombre (similar a `ps | grep`)

- Ejemplo:

```
$ pgrep sshd # devuelve el PID del proceso sshd de root
```

pkill permite enviar señales a los procesos indicándolos por nombre en vez de por PID

- Ejemplo:

  `$ pkill -KILL` *`firefox`*

- si hay varios procesos con el mismo nombre los mata a todos
- en vez de un nombre admite un patrón (p.e. pkill ´l.*´)
  - tener cuidado con su uso (es fácil matar procesos de forma errónea)

killall similar a pkill, pero no admite patrones en el nombre, y tiene otras opciones

[1] En bash en principio podemos fijar el comportamiento por defecto del shell modificando la opción `huponexit` con `shopt`.

exec ejecuta un comando reemplazando al shell desde el que se lanza
Ejemplos:

```
$ yes > /dev/null &
[1] 14724
$ yes > /dev/null &
[2] 14725
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14724 pts/3 00:00:02 yes
14725 pts/3 00:00:02 yes
14726 pts/3 00:00:00 ps
$ pgrep yes
14724
14725
$ pkill -KILL yes
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14730 pts/3 00:00:00 ps
[1]- Matado         yes > /dev/null
[2]+ Matado         yes > /dev/null
```

Más ejemplos:

```
$ nohup yes > /dev/null &
[1] 9620
$ kill -HUP 9620
$ ps
 PID TTY      TIME  CMD
8293 pts/5 00:00:00 bash
9620 pts/5 00:00:13 yes
9621 pts/5 00:00:00 ps
$ kill 9620
[1]+ Terminado         nohup yes > /dev/null
```

3.1.3. Manejo de la prioridad y recursos de un proceso

Cuando un proceso se ejecuta, lo hace con una cierta *prioridad*

- las prioridades van desde -20 (prioridad más alta) a 19 (la más baja)

- por defecto, los procesos se ejecutan con prioridad 0
  - un usuario normal solo puede asignar prioridades más bajas (números positivos)
  - `root` puede asignar prioridades más altas (números negativos)
- los comandos para manejo de prioridades son `nice` y `renice`
- Nota: NI (o NICE) denota el valor de la prioridad asignada al iniciar el proceso, mientras que PRI es la prioridad actual, ajustada por el planificador del kernel de Linux.

`nice`

Permite lanzar un comando con una cierta prioridad

- Sintaxis

  `nice -n` *`ajuste comando`*

  la prioridad inicial se fija a *`ajuste`*

- Ejemplo:

```
$ nice -n 10 openoffice &     # disminuye la prioridad a 10
$ ps -o pid,pri,ni,stat,cmd
  PID PRI  NI STAT CMD
 7133  24   0 Ss   bash
 7431  14  10 SN   openoffice
 7552  23   0 R+   ps -o pid,pri,ni,stat,cmd
$ nice -n -1 libreoffice &    # aumenta la prioridad a -1
nice: no se puede establecer la prioridad: Permiso denegado
```

`renice`

Permite cambiar la prioridad de un proceso que está en ejecución

- Sintaxis:

  `renice` *`pri`* `[-p` *`pid`*`] [-u` *`user`*`] [-g` *`pgrp`*`]`

- las opciones son:
  - `-p` *`pid`* cambia la prioridad para el proceso especificado

  - `-u` *`user`* cambia la prioridad para los procesos del usuario especificado
  - `-g` *`pgrp`* cambia la prioridad para los procesos ejecutados por los usuarios que pertenecen al grupo de gid *`pgrp`*
- Ejemplo:

```
$ abiword &
[1] 7681
$ renice 10 -p 7681
7681: old priority 0, new priority 10
$ renice 3 -u tomas
503: old priority 0, new priority 3
```

Control de los recursos de un proceso

El comando interno de bash `ulimit` permite controlar los recursos de los que dispone un proceso arrancado por el shell

- Sintaxis

  `ulimit [`*`opciones`*`] [`*`limite`*`]`

- Algunas opciones:
  - `-a` muestra los límites actuales
  - `-f` máximo tamaño de los ficheros creados por el shell (opción por defecto)
  - `-n` máximo número de ficheros abiertos
  - `-s` máximo tamaño de la pila
  - `-t` máximo tiempo de cpu
  - `-S`/`-H` usa los límites *soft* y *hard*
    - el usuario puede incrementar su límite *blando*, pero sin superar el límite duro
    - estos se pueden poner en `/etc/profile`, `/etc/bash` o `.bashrc`
- Para más información `help ulimit`
- Ejemplo: limitar el tamaño de los ficheros creados a 1 kbyte

  ```
  $ ulimit -f 1
  ```

3.1.4. Análisis básico del rendimiento del sistema

Además de `ps` y `top` existen comandos básicos que nos pueden mostrar el estado del sistema en cuanto a uso de CPU y consumo de memoria[2]

`uptime`

Muestra la hora actual, el tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos (lo mismo que en la primera línea de `top`)

- Ejemplo:

```
$ uptime
 20:25:03 up 25 days, 11:12, 13 users, load average: 3.00, 3.07, 3.08
```

`w`

Además de la información dada por `uptime`, el comando `w` muestra información sobre los usuarios y sus procesos

- Ejemplo:

```
$ w
 20:24:52  up 25 days, 11:11, 13 users,  load average: 3.10, 3.09, 3.08
USER     TTY      FROM     LOGIN@  IDLE    JCPU   PCPU  WHAT
paula    pts/1    godello  12:08pm  8:15m  8.39s  8.36s  ssh -p 1301 -X
paula    pts/2    godello  12:09pm  7:30   0.11s  0.11s  bash
pichel   pts/4    -        11:08am  7.00s  0.33s  0.33s  -bin/tcsh
pichel   pts/5    -         7:12pm 56:56   0.26s  0.16s  ssh www.usc.es
pichel   pts/6    -         4:35pm 21:15  16.61s  0.02s  /bin/sh ls
tomas    pts/8    jumilla   8:24pm  0.00s  0.05s  0.02s  w
```

- Definiciones:
  - `LOGIN@` la hora a la que se conectó el usuario
  - `IDLE` tiempo que lleva ocioso el terminal
  - `JCPU` tiempo de CPU consumido por los procesos que se ejecutan en el TTY
  - `PCPU` tiempo de CPU consumido por el proceso actual (el que aparece en la columna `WHAT`)

[2]Hay otros comandos de monitorización más complejos como `vmstat` o `sar`.

`free`

Muestra la cantidad de memoria libre y usada en el sistema, tanto para la memoria física como para el swap, así como los buffers usados por el kernel (similar a lo mostrado en la cabecera de `top`)

- Ejemplo:

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:          5,7Gi       1,0Gi       3,7Gi        20Mi       1,0Gi       4,5Gi
Swap:          18Gi          0B        18Gi
```

- la columna `shared` indica la memoria usada por `tmpfs`, `buffers` la memoria usada por los buffers del kernel y `cache` la memoria usada por los datos cacheados.
- Opciones:
  - `-b,-k,-m,-g` memoria en bytes/KBytes/MBytes/GBytes
  - `-t` muestra una línea con el total de memoria (física + swap)
  - `-s` ***delay*** muestra la memoria de forma continua, cada ***delay*** segundos

3.1.5. Los directorios `/proc` y `/sys`

Son dos diectorios que guardan la información que recopila el S.O.

- `/proc` ofrece información sobre los procesos (de ahí su nombre). También almacena todo tipo de información sobre los componentes principales del computador.
- `/sys` ofrece información detallada sobre los dispositivos.

Se caracterizan por:

- se inicializan durante el arranque cuando el kernel configura el sistema
- son *seudosistemas de ficheros* que están implementados en memoria y no se guardan en disco
- los comandos que muestran información del computador y de los procesos (`ps`, `top`, etc.) obtienen la información de estos directorios.

Algunos ficheros y directorios de `/proc` son:

- `cpuinfo`: información de la CPU
- `meminfo`: información de uso de la memoria
- `interrupts`: muestra las interrupciones usadas por IRQ
- `ioports`: lista los puertos de entrada salida usados en el sistema
- `net/`: directorio con información de red
- `bus/`: directorio con información de los buses PCI y USB
- `devices`: dispositivos del sistema, de tipo caracter (que operan caracter a carecter, por ejemplo, teclados) o bloque (por ejemplo, discos)
- `filesystems`: sistemas de ficheros soportados por el kernel
- `partitions`: información sobre las particiones
- `modules`: módulos del kernel

Además, en el directorio `/proc` existe un directorio por cada proceso, que se identifica con el PID del proceso, `/proc/`*PID*`/`, en el que se puede encontrar información sobre cada proceso, incluidos:

- el directorio desde que se invocó el proceso (enlace `cwd`)
- nombre del ejecutable (enlace `exe`) y la línea de comandos con la que fue invocado (fichero `cmdline`)
- entorno en que se ejecuta el proceso (fichero `environ`)
- estado del proceso (fichero `status`)
- descriptores de ficheros abiertos y archivos o procesos relacionados (directorio `fd` conteniendo enlaces simbólicos a los ficheros)
- mapa de memoria (fichero `maps`)

La información detallada de cada dispositivo se ofrece en el directorio `/sys`. Por ejemplo, la información sobre la carga de un portátil puede encontrarse en `/sys/class/power_supply/BAT1/uevent`:

```
POWER_SUPPLY_NAME=BAT0
POWER_SUPPLY_STATUS=Discharging
POWER_SUPPLY_PRESENT=1
POWER_SUPPLY_TECHNOLOGY=Li-ion
POWER_SUPPLY_CAPACITY=93
POWER_SUPPLY_CAPACITY_LEVEL=Normal
...
```

3.2. Gestión del sistema de ficheros

UNIX tiene múltiples comandos para trabajar con ficheros y directorios: `ls, rm, cp, mv, mkdir, rmdir, touch, etc.`

En esta sección trataremos:

- los diferentes tipos de ficheros y sus atributos
- los permisos de acceso para ficheros y directorios
- la creación de enlaces
- la localización de ficheros

3.2.1. Tipos de ficheros y operaciones

En Linux y Unix se consideran ficheros todo tipo de objetos que pueden ser operados de forma similar a un fichero. Por ejemplo, en un dispositivo la lectura de los datos que contiene puede asimilarse a la lectura de un fichero, mientras que el envío de órdenes puede interpretarse como una escritura. Teniendo en cuenta esto, en Linux se definen 7 tipos de ficheros:

Ficheros regulares son los usuales; se crean con distintos programas (`vi, cp, touch, etc.`) y se borran con `rm`

Directorios contiene referencias a otros ficheros y directorios; se crean con `mkdir` y se borran con `rmdir` o `rm -r`

Ficheros de dispositivos de caracteres o bloques permiten la comunicación con el hardware y los periféricos; se crean con `mknod` y se borran con `rm`

- caracteres: entrada/salida byte a byte
- bloques: entrada salida en bloques de datos

Tuberías con nombre (*named pipes*) también llamados ficheros FIFO, permiten la comunicación entre procesos; se crean con `mknod` y se borran con `rm`

Sockets comunican procesos en la red; se crean con `socket()` y se borran con `rm` o `unlink()`

Enlaces simbólicos también llamados enlaces *blandos*: apuntador a otro fichero; se crean con `ln -s` y se borran con `rm`

El comando `file` nos permite determinar el tipo de un fichero:

- para ficheros normales, distingue según contenido (fichero de imagen, pdf, ASCII, etc)

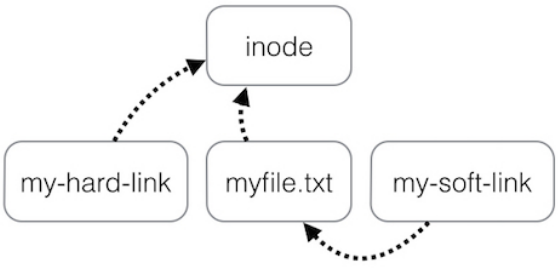
```
$ file /dev/xconsole
/dev/xconsole: fifo (named pipe)
$ file fichero1
fichero1: PDF document, version 1.2
$ file fichero2
fichero2: Microsoft Office Document
$ file fichero3
fichero3: PNG image data, 750 x 686, 8-bit/color RGB, ...
```

Creación de enlaces

Los enlaces permiten referirse a un fichero con otro nombre

Dos tipos:

- **Enlaces duros**: asignan otro nombre al fichero
  - crean una entrada en el directorio apuntando al mismo nodo-i que el fichero original
  - el fichero no se borra hasta que se borran todos sus enlaces duros
  - no se puede enlazar con ficheros de otra partición
- **Enlaces blandos**: un fichero que apunta al original
  - si el fichero se borra, el enlace permanece sin apuntar a nada
  - no tienen problema con las particiones



Comando `ln`

Permite crear enlaces

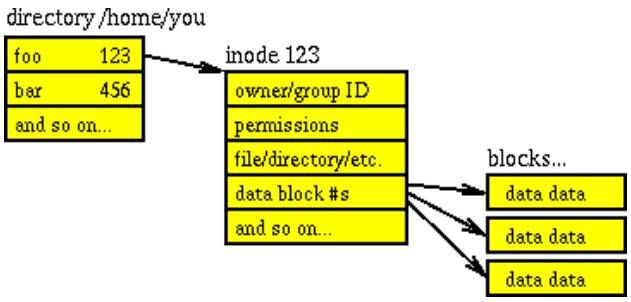
- Formato

  `ln [-s] [`*`opciones`*`] `*`destino`*` [`*`enlace`*`]`

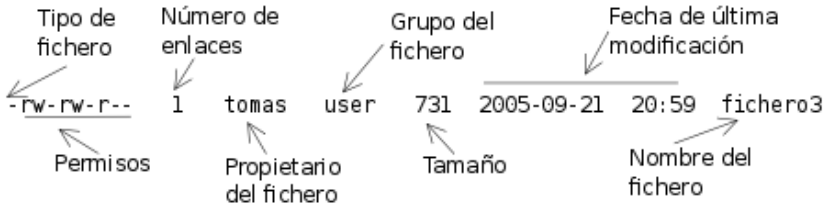
- por defecto crea enlaces duros; con `-s` se crean enlaces blandos
- si no se pone nombre del enlace se usa el del *`destino`*

```
$ ln /home/luis/fich1 fichhard
$ ln /home/luis/fich2 /home/luis/fich3 .
$ ls -l /home/luis/fich* ./fich*
-rw-r--r-- 2 luis luis 12 Sep 22 20:19 ./fich2
-rw-r--r-- 2 luis luis 12 Sep 22 21:18 ./fich3
-rw-r--r-- 2 luis luis 13 Sep 22 20:17 ./fichhard
-rw-r--r-- 2 luis luis 13 Sep 22 20:17 /home/luis/fich1
-rw-r--r-- 2 luis luis 12 Sep 22 20:19 /home/luis/fich2
-rw-r--r-- 2 luis luis 12 Sep 22 21:18 /home/luis/fich3
$ ln -s /home/luis/fich4 blando
$ ls -l /home/luis/fich4 blando
-rw-r--r-- 1 luis luis 12 Sep 22 21:22 /home/luis/fich4
lrwxrwxrwx 1 pepe pepe 17 Sep 22 21:22 blando -> /home/luis/fich4
```

Atributos de un fichero



Podemos ver los atributos de un fichero con `ls -l`



Indicador de tipo el primer carácter nos indica el tipo del fichero

Carácter	Tipo
-	fichero regular
d	directorio
l	enlace simbólico
c	fichero de dispositivo de caracteres
b	fichero de dispositivo de bloques
p	tubería (pipe)
s	socket

Número de enlaces :

- indica el número de nombres (enlaces duros) del fichero
- en el caso de un directorio, esto corresponde con el número de subdirectorios (incluidos `.` y `..`)

Tamaño es el tamaño en bytes

- con `ls -lh` se ve el tamaño de forma más legible
- el tamaño máximo de un fichero depende del filesystem usado

Fecha especifica la fecha de última modificación del fichero

- podemos actualizarla con el comando `touch`

Nombre la longitud máxima del nombre es de 255 caracteres

- evitar el uso de espacios y caracteres especiales como `*,$, ?, ´, ", /, \`

Fecha

Linux guarda para cada fichero 3 tipos de fecha:

- `mtime`. Fecha de la última **m**odificación del fichero, esta es la fecha por defecto que se muestra. Por ejemplo, con `ls -l`
- `atime`. Fecha del último **a**cceso. Se muestra con `ls -l --time=atime`
- `ctime`. Fecha de la último **c**ambio de estado. Se muestra con `ls -l --time=ctime`

Permisos de ficheros y directorios

UNIX proporciona tres operaciones básicas para realizar sobre un fichero o directorio: lectura (`r`), escritura (`w`) y ejecución (`x`)

- Efecto sobre un fichero:
  1. lectura (`r`): permite abrir y leer el fichero
  2. escritura (`w`): permite modificar o truncar el fichero (para borrarlo, basta con que el directorio tenga permiso de escritura)
  3. ejecución (`x`): permite ejecutar el fichero (binario o script)
- Efecto sobre directorios:
  - ejecución (`x`): permite entrar en el directorio (pero no listar su contenido, ni crear ficheros o directorios)
  - lectura y ejecución (`rx`): permite listar el contenido del directorio (pero no crear ficheros o directorios)
  - escritura y ejecución (`wx`): permite crear, borrar o renombrar ficheros (pero no listar su contenido)
  - acceso total (`rwx`)

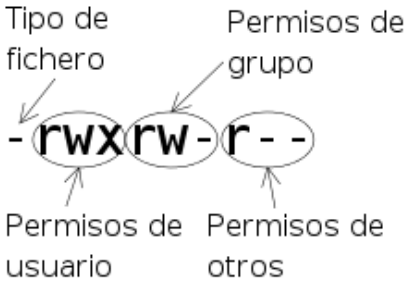
Los permisos se aplican en tres categorías:

- Permisos de usuario (`u`): propietario del fichero (por defecto, el usuario que lo creó)
- Permisos de grupo (`g`): grupo del fichero (por defecto, grupo principal del usuario que lo creó)
- Permisos de otros (`o`): resto de usuarios

Cada usuario cae en uno solo de estas categorías:

- p.e. al propietario se le aplican los permisos de usuario, aunque sean más restrictivos que los de grupo

Los permisos se identifican con 9 caracteres:



Cambio de permisos

El comando para modificar los permisos es `chmod`

- Formato

  `chmod [-R]` *`operación ficheros`*

- `-R` indica acceso recursivo
- solo el propietario del fichero (o `root`) puede cambiar los permisos

`operación` indica como cambiar los permisos, y puede especificarse mediante símbolos o números:

- Permisos simbólicos: formato *`quien op permisos`*

  quien especificado por `u`, `g`, `o` o `a` para todos

  op puede ser `+` para añadir permisos, `-` para quitar o `=` para establecer

  permisos especificados por `r`, `w`, `x`

  Ejemplos:

  - `chmod u+x temp.dat`
    añade permisos de ejecución para el usuario (manteniendo los permisos existentes)
  - `chmod ug=rw,o=r temp.dat`
    lectura y escritura para usuario y grupo y sólo lectura para el resto
  - `chmod -R =r *`
    pon, de forma recursiva, permisos sólo lectura para todos (ugo)
  - `chmod a-x *.bak`
    quita el permiso de ejecución para todos
  - `chmod g=u temp.dat`
    pon los permisos de grupo igual a los del usuario
  - `chmod a= *`
    quita los permisos a todos

- Permisos numéricos:

  - *`operación`* se representa por un número **octal** de tres dígitos, para `u`, `g`, y `o` respectivamente
  - cada dígito vale:

- 4 para `r`, 2 para `w` y 1 para `x`
- para combinaciones, se suman:
  p.e. `rw` es 6, `rx` es 5 y `rwx` es 7

Ejemplos:

- `chmod 750 temp.dat`
  permisos `rwx` para usuario, `rx` para grupo y ninguno para otros
- `chmod 043 temp.dat`
  ninguno para usuario, `r` para grupo y `wx` para otros

Permisos especiales

Además de `rwx` existen los permisos `setuid`/`setgid` (`s`) y `sticky bit` (`t`)

`setuid` y `setgid` están relacionados con los atributos de los procesos:

- cuando un proceso se crea se le asigna un UID/GID real y un UID/GID efectivo

  UID/GID real (RUID/RGUI) identificadores de usuario y grupo del usuario que lanzó el proceso (y que puede matarlo)

  UID/GID efectivos (EUID/EGUI) determinan las operaciones que el proceso puede hacer sobre los objetos del sistema

  - por ejemplo, un proceso con UID efectivo 0 (`root`) puede manipular todos los ficheros del sistema

- lo normal es que los UID/GID real y efectivo de un proceso coincidan y sean los del usuario que lanzó el proceso (y del grupo al que pertenece)

Podemos usar `ps` para ver los RUID/RGID y EUID/EGID

- `ps -eo ruid,rgid,euid,egid,cmd` para ver números
- `ps -eo ruser,euser,rgroup,egroup,cmd` para nombres

Los permisos `setuid`/`setgid` permiten que un proceso lanzado por un usuario se ejecute con EUID/EGID de otro usuario

IMPORTANTE: es peligroso ejecutar procesos con permisos de otro usuario, por lo debería evitarse poner `setuid`/`setgid` a los ficheros

- Ejemplo: el programa `passwd`

```
-rwsr-xr-x 1 root root 25872 2005-07-25 23:15 /usr/bin/passwd
```

Cuando un usuario ejecuta `passwd`, el proceso que se genera tiene el UID real del usuario, pero su UID efectivo es el de root (porque el ejecutable es propiedad de root y tiene activado el permiso `s`). Por lo tanto, puede modificar el fichero `/etc/shadow` propiedad de root.

Fijar `setuid/setgid`

- Formas simbólica y numérica

  fijar `setuid`: `chmod u+s` , `chmod 4`***`xyz`***
  fijar `setgid`: `chmod g+s` , `chmod 2`***`xyz`***
  siendo *x, y, z* los otros tres permisos (de 0-7)

Ejemplo:

```
$ ls -l temp
-rw-r-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+s temp; ls -l temp
-rwSr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+x temp; ls -l temp
-rwsr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod 6740 temp; ls -l temp
-rwsr-S---  1 tomas gac 0 2005-09-22 18:07 temp
```

Nótese que aparece `S` al poner el setuid, y luego cambia a `s` al añadir el permiso de ejecución.

sticky bit solo se usa en directorios

- permite crear ficheros en el directorio (si tiene permiso de escritura), pero solo los puede borrar:
  - el propietario del fichero
  - el propietario del directorio
  - el superusuario

Ejemplo:

```
$ ls -ld /tmp
drwxrwxrwt 15 root root 3072 2005-09-22 19:09 /tmp/
```

Para activar el `sticky bit`

- Forma simbólica: `chmod +t` ***`dir`***
- Forma numérica: `chmod 1`***`xyz dir`***

Permisos por defecto

Cuando se crea un fichero se ponen los permisos por defecto

- los permisos por defecto pueden fijarse con `umask`

  `umask [opciones] valor`

  donde *valor* son tres dígitos que especifican los permisos para `u`, `g`, `o` según la tabla

Octal	Permisos	Octal	Permisos
0	`rwx`	4	`-wx`
1	`rw-`	5	`-w-`
2	`r-x`	6	`--x`
3	`r--`	7	`---`

- Opciones (dependen de la versión de shell):
  - `-S` muestra los permisos por defecto
- Ejemplo[3]

```
$ umask 027
$ umask -S
u=rwx,g=rx,o=
```

Cambio de usuario/grupo

Los comandos `chown` y `chgrp` permiten cambiar el propietario y grupo de un fichero

- sólo `root` puede cambiar el propietario
- el grupo puede cambiarse a otro al que pertenezcamos

Formato:

```
chown [opciones] propietario ficheros
chgrp [opciones] grupo ficheros
chown [opciones] propietario:grupo ficheros
```

Algunas opciones

- `-R` recorre recursivamente los subdirectorios
- `-v` (*verbose*) indica las operaciones que realiza

[3] Comandos que crean ficheros como `touch` o `vi` no ponen permiso de ejecución aunque lo diga `umask`

3.2.2. Localización de ficheros

Una tarea común de administración es la búsqueda de ficheros que verifiquen ciertas propiedades

- buscar ficheros muy grandes
- buscar ficheros de un determinado usuario
- mostrar los ficheros que se hayan modificado en los últimos 2 días
- buscar ficheros `setuid`/`setgid`

El comando básico para hacer esto es `find`

Comando `find`

Busca a través de la jerarquía de directorios ficheros que cumplan determinado criterio

- Formato

  `find [`*`directorio_de_búsqueda`*`] [`*`expresión`*`]`

- Ejemplos:

  - Muestra desde el directorio actual todos los ficheros de forma recursiva

    ```
    find
    ```

  - Busca desde el directorio actual de forma recursiva los ficheros que terminen en `.txt`

    ```
    find -name "*.txt"
    ```

  - Busca desde `/etc` los ficheros de tipo directorio

    ```
    find /etc -type d
    ```

  - Busca desde `/etc` y `/usr/share` los ficheros que se llamen `magic` o `passwd`

    ```
    find /etc /usr/share -name magic -o -name passwd
    ```

La *`expresión`* tiene los siguientes componentes:

- opciones: modifican la forma de operación de `find`
- criterio de búsqueda
- acciones: especifica que hacer con los ficheros que encuentra
- operadores: permiten agrupar expresiones

Criterios de busqueda

- Búsqueda por nombre/path/tipo/usuario/grupo:

Criterio	Efecto
`-name` *patrón*	busca ficheros que coincidan con el patrón (pueden usarse comodines, pero deben ponerse entre comillas o escapados)
`-iname` *patrón*	igual que el anterior, pero no distingue mayúsculas/minúsculas
`-regex` *patrón*	igual pero usa expresiones regulares
`-path/-ipath` *patrón*	búsca el camino `path`
`-type` *tipo*	busca por tipo de fichero (`f, d, l, b, c, p, s`)
`-user, -group` *nombre*	busca por propietario/grupo
`-uid/-gid` *n*	busca por UID/GID
`-nouser/-nogroup`	busca ficheros con prop./grupo no válidos

NOTA: Cuando se usa el criterio *path* para la búsqueda, debemos usar el patrón tal como lo encuentra el comando `find`. Por ejemplo, si el path en el computador es *./dir1/dir2/dir3/dir4* y queremos buscar *dir2/dir3* hay que escribir "**/dir2/dir3*" (para que concida con el path encontrado por el comando, que es *./dir1/dir2/dir3*).

- Búsqueda por tamaño/permiso:

Criterio	Efecto
`-size` *n*`[ckMG]`	tamaño igual a *n* (`c`=caracteres(bytes), `k`=kbytes, `M`=Mbytes, `G`=Gbytes)
`-size -`*n*`[ckMG]`	tamaño menor a *n*
`-size +`*n*`[ckMG]`	tamaño mayor que *n*
`-perm` *permisos*	permisos exactos
`-perm -`*permisos*	permisos indicados (si se indica solo el permiso de lectura, no se miran los permisos de escritura y ejecución)
`-perm /`*permisos*	para los permisos exactos el prefijo / indica que es suficiente que se verifique uno de los permisos entre usuario, grupo y otros
`-perm /-`*permisos*	lo mismo para los permisos indicados

Más en detalle para los permisos:

permiso	encuentra	permiso	encuentra
0	0	-0	0,1,2,3,4,5,6,7
1 (x)	1	-1	1,3,5,7
2 (w)	2	-2	2,3,6,7
3 (wx)	3	-3	3,7
4 (r)	4	-4	4,5,6,7
5 (rx)	5	-5	5,7
6 (rw)	6	-6	6,7
7 (rwx)	7	-7	7

Por ejemplo, `-perm 643` encuentra 643 y `-perm -643` encuentra 643, 743, 653, 753, 663, 763, 673, 773, 645, 745, 655, 755, 665, 765, 675 y 773. Por su parte, `-perm /213` encuentra `2aa`, `a1a` y `aa3` siendo `a` cualquier permiso.

- Busqueda por atributos temporales:

Criterio	Efecto
`-atime [ /+/-]`n	busca ficheros cuya última fecha de acceso para lectura coincide (), es anterior (+) o es posterior (-) a n días
`-mtime [ /+/-]`n	lo mismo, pero con la fecha de última modificación del fichero
`-ctime [ /+/-]`n	lo mismo, pero con la fecha en que se cambió el estado del fichero
`-amin/-mmin/` `-cmin [ /+/-]`n	lo mismo, pero ahora n representa minutos
`-newer` *file*	busca ficheros modificados más recientemente que *file*
`-anewer` *file*	ficheros con último acceso más reciente que la modificación de *file*
`-cnewer` *file*	ficheros con cambio de estado más reciente que la modificación de *file*

Opciones de `find` normalmente se colocan al principio de la expresión

Opción	Efecto
`-maxdepth` n	desciende como máximo n directorios
`-daystart`	para medidas con tiempo, empieza desde el principio del día actual
`-mount`	no pasa a otras particiones

Operadores de `find` permiten agrupar expresiones

Operador	Descripción
expr1 *expr2*	AND (*expr2* no se evalúa si *expr1* es falsa). También puede ponerse: *expr1* `-a` *expr2*
expr1 `-o` *expr2*	OR (*expr2* no se evalúa si *expr1* es cierta)
`!` *expr1*	NOT (cierto si *expr* falsa). También puede ponerse: `not` *expr1*
`\(` *expr1* `\)`	agrupan expresiones (hay que escapar los paréntesis)

Acciones de `find` `find` permite realizar distintas acciones con los ficheros que encuentra

- mostrar su nombre (acción por defecto)
- mostrar otra información del fichero
- ejecutar un comando sobre el fichero

Acción	Descripción
`-print`	imprime el nombre de los ficheros que encuentra (acción por defecto)
`-ls`	imprime el nombre de los ficheros con formato de listado largo
`-exec` *comando* `{} \;`	ejecuta *comando* sobre los ficheros encontrados
`-ok` *comando* `{} \;`	igual que `-exec` pero pregunta antes de ejecutar *comando*
`-prune`	no desciende por el directorio indicado con la opción `path`

Los caracteres `{}` se refieren al fichero que `find` acaba de encontrar, mientras que `;` indica el fin del comando. La acción `-prune` se usa por ejemplo en la forma: `find . -path "*/dir" -prune -o -name file -print`.

Ejemplos con `find`

- `find . -maxdepth 1 -user david`
  busca ficheros, sólo en el directorio actual, propiedad de `david`

- `find / -name "*.html" -ls`
  busca, en todo el sistema de ficheros, ficheros terminados en `.html` y muestra un listado largo
- `find /home/httpd/html ! -name "*.html"`
  busca, desde `/home/httpd/html`, los ficheros que no acaben en `.html`
- `find /home -size +2500k -mtime -7`
  busca, desde `/home`, ficheros más grandes de 2500KB que hayan sido modificados en los últimos 7 días
- `find /home -iname "*.bak" -ok rm {} \;`
  busca ficheros terminados en `.bak` (sin distinguir mayúsculas/minúsculas) y pregunta si se quiere borrar
- `find /home -iname "*.bak" -exec mv {} /BAK \;`
  busca ficheros terminados en `.bak` y muevelos a `/BAK`
- `find / -path "/home" -prune -o -name "*.bak" -ls`
  busca excluyendo el directorio `/home`
- `find . -perm 022`
  encuentra ficheros con permisos `-----w--w-`
- `find . -perm -022`
  encuentra ficheros escribibles por `grupo` **y** `otros`
- `find . -perm -g=w,o=w`
  idéntico al anterior
- `find /home/httpd -name "*.html" -exec grep -l expiral {} \;`
  lista los nombres de los `.html` que contengan la palabra `expiral`

Este último ejemplo funciona, pero es muy ineficiente pues por que genera un proceso grep para cada fichero encontrado

- otra forma de hacer lo mismo

  ```
  grep -l expired $(find /home/httpd/html -name "*.html")
  ```

si el número de ficheros `.html` es muy grande `grep` puede tener problemas

- podemos usar `xargs`

  ```
  find /home/httpd/html -name "*.html" | xargs grep -l expired
  ```

Otros comandos para localizar ficheros

Existen otros comandos para la localización de ficheros: `which`, `whereis`, `locate`

Comando `which` muestra la localización de comandos buscando en $PATH

- Formato:

  `which [`*`opciones`*`] `*`comando`*

- Ejemplo:

```
$ which ls
/bin/ls
```

Comando `whereis` muestra la localización del binario, fuente y página de manual de un comando buscando en las localizaciones estándar

- Formato:

  `whereis [`*`opciones`*`] `*`comando`*

- Ejemplo:

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

- Para más opciones ver la página de manual

Comando `locate` localiza todo tipo de ficheros rápidamente

- utiliza una base de datos donde guarda la localización de los ficheros (`/var/cache/locate/locatedb`)
- esa base de datos la crea y actualiza el administrador con el comando `updatedb`
- Ejemplo:

```
$ locate "*.bak"
/root/.mozilla/firefox/bookmarks.bak
/var/backups/group.bak
/var/backups/inetd.conf.bak
```

- Ver página de manual para opciones

3.3. Gestión de discos y particiones

3.3.1. Particiones y sistemas de ficheros

Vimos en el primer capítulo como crear particiones y sistemas de ficheros en el momento de la instalación

- si añadimos un nuevo disco al sistema ya instalado deberemos crear las particiones y los sistemas de ficheros
- esta operación implica los siguientes pasos:
  1. creación de particiones (comando `fdisk`)
  2. creación de los sistemas de ficheros (comando `mkfs`)
  3. montado de los sistemas de ficheros (comando `mount` o `/etc/fstab`)

Comando	Operación
`fdisk`	crea o lista particiones
`blkid`	muestra el identificador único universal (UUID)
`mkfs.`*tipo*	crea sistemas de ficheros de tipo *tipo*
`fsck.`*tipo*	testea y repara sistemas de ficheros
`tune2fs`	ajusta parámetros de ext2/ext3/ext4
`dumpe2fs`	muestra información de ext2/ext3/ext4
`e2label`	fija la etiqueta de ext2/ext3/ext4
`mkswap`	crea un sistema de ficheros de tipo swap
`swapon`	activa la partición de swap
`mount` / `umount`	monta/desmonta particiones
`fstab` / `mtab`	ficheros relacionados con el montado
`df` / `du`	muestra espacio en particiones montadas / directorios

Creación de particiones

- `fdisk [`*`opciones`*`]` *`dispositivo`*
- *`dispositivo`* es el dispositivo del disco `/dev/sd`*`x`* para SATA o `/dev/nvme`*`x`* para NVME
- Debemos tener permiso de administrador para usarlo
- Opciones:
  - `-l` muestra la tabla de particiones del dispositivo

`fdisk` se usa mediante un menú:

```
# fdisk /dev/sdb
Command (m for help): m
Command action
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

1. Para crear una partición primaria de 5 GB usamos `n` (*new*):

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-20805, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-20805, default 20805): +5G
```

2. Para cambiar el tipo de partición se usa `t` (*type*)

   - Por defecto, las particiones que se crean son de tipo Linux (Id 83).
   - Con `l` (*list*) vemos el tipo de particiones soportadas

```
Command (m for help): t
Partition number (1-4): 1
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)
```

3. Mostramos la tabla de particiones:

```
Command (m for help): p
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
```

```
Device Boot  Start   End   Blocks    Id  System
/dev/sdb1    1      9689  4883224+  82  Linux swap / Solaris
```

4. Por último, para que se guarden los cambios debemos usar `w` (*write*)

   ⟶ Al escribir la tabla de particiones el contenido de las particiones modificadas se pierde

Otras herramientas para modificar las particiones:

cfdisk interfaz para el `fdisk` (también escribe la tabla de particiones)

parted programa de GNU que permite crear, borrar, cambiar el tamaño, chequear y copiar particiones

qtparted programa Linux para manejar particiones, con interfaz gráfico

Creación de los sistemas de ficheros

Sobre cada partición debemos crear sistemas de ficheros con el comando `mkfs`

- Formato básico:

  `mkfs.`*`tipo`* `[`*`opciones`*`]` *`dispositivo`*

- Opciones:
  - *`tipo`* es el tipo de sistema de ficheros a crear (ext2/3/4, xfs, etc.) Si no se especifica se crea el por defecto del sistema
  - `-V` verbose

`mkfs` es un *front-end* a distintos comandos, que permiten crear particiones de los tipos específicos:

- `mkfs.ext2`, `mkfs.ext3` y `mkfs.ext4` crean sistemas ext2 / ext3 /ext4.
- `mkfs.btrfs`, `mkfs.jfs` y `mkfs.xfs` crean sistemas btrfs, JFS y XFS, respectivamente
- `mkfs.vfat`, `mkfs.exfat` y `mkfs.ntfs` crean sistemas MS Windows.
- `mkswap` crea un sistema de ficheros de tipo Linux swap

Cada uno de estos comandos pueden tener distintas opciones

- ver las páginas de manual para más detalles

Comandos relacionados

- `tune2fs` permite ajustar parámetros en sistemas ext2/3/4
  - p.e. define el intervalo entre chequeos automáticos, fija el porcentaje de disco reservado para el sistema (por defecto el 5 %), etc.
- `dumpe2fs` muestra información de sistemas de ficheros ext2/3/4
  - información sobre inodos, bloques y grupos
- `e2label` cambia la etiqueta de un sistema ext2/3/4
- existen comandos similares para otros tipos de sistemas de ficheros, p.e. `btrfstune`, `jfs_tune`, etc.

Partición de intercambio

- Para crear una partición de intercambio primero la debemos formatear con `mkswap`

```
# mkswap /dev/sdb2
Setting up swapspace version 1, size = 5736919 kB
no label, UUID=a6c2849b-c33e-478e-8a8d-fecfe3f18f6d

# fdisk -l /dev/sdb
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Device Boot  Start     End    Blocks   Id  System
/dev/sdb1        1    9689   4883224+  83  Linux
/dev/sdb2     9690   20805   5602464   82  Linux swap / Solaris
```

- A continuación debemos activarla con `swapon`

```
# swapon /dev/sdb2
# swapon -s
Filename        Type          Size       Used    Priority
/dev/sdb7       partition     377488     0       -1
/dev/sdb2       partition     5602456    0       -2
```

- Finalmente, para que se active en el arranque, debe incluirse la entrada correspondiente en el fichero `/etc/fstab` (véase más adelante)

```
#(file system) (mount point) (tipo) (opciones) (dump) (pass)
/dev/sdb2       none          swap   sw         0      0
```

Montado de los sistemas de ficheros

Para poder acceder a los sistemas de ficheros debemos *montarlos*

- los comandos para montar y desmontar son: `mount` y `umount`

Comando `mount` Permite asociar (*montar*) directorios a sistemas de ficheros

- Formato

  `mount [`*`opciones`*`] [`*`disp.`*`] [`*`dir.`*`]`

- Ejemplo

  ```
  $ mount /dev/sdb1 /home2
  ```

- Si no se especifica el dispositivo, este se busca en `/etc/fstab`
- Algunas opciones:
  - `-a` monta todos los filesystems listados en `/etc/fstab`
  - `-o` *`opciones`* donde las opciones tienen el mismo formato que las usadas en el fichero `fstab` (véase el siguiente apartado)
  - Una opción muy importante es `-o remount`, que permite pasar un sistema de ficheros de modo solo lectura a lectura y escritura,

    ```
    $ mount -o rw,remount /dev/sdb1 /home2
    ```

Comando `umount` Desmonta los sistemas de ficheros

- Formato

  `umount [`*`opciones]`* *`directorio`*

- Ejemplo:

  ```
  $ umount /home2
  ```

- Algunas opciones
  - `-a` desmonta los filesystems listados en `/etc/mtab`
- Si hay algún proceso bloqueando el filesystem, este no se puede desmontar:
  - el comando `fuser -c` *`directorio`* nos da el PID del proceso

Fichero `/etc/fstab` Al iniciar el sistema se montan los filesystems listados en `/etc/fstab`

- cada línea del fichero tiene las siguientes columnas

```
(file system) (mount point) (tipo) (opciones)   (dump) (pass)
```

- Ejemplo:

```
/dev/sda1      /            auto    defaults      0      1
/dev/sda9      /home        ext4    auto,user,rw  0      2
```

- Alguna de las opciones son (se separan por comas y sin espacios):
  - `ro` monta en modo sólo lectura
  - `rw` monta en modo lectura+escritura
  - `auto`/`noauto` monta/no monta en el arranque
  - `user`/`nouser` puede/no puede montarlo un usuario (y el primer caso sólo el que lo monta puede desmontarlo)
  - `users` similar al anterior, pero puede montarlo/desmontarlo un usuario y el que desmonta no tiene que ser el que lo montó
  - `defaults` selecciona opciones por defecto (`rw`, `auto` y `nouser`)
- Filesystems específicos pueden tener opciones específicas:
  - ver el manual de `mount` para más detalles
- Si un directorio aparece listado en el `fstab` puede montarse sin especificar el dispositivo:

```
$ mount /home
```

- Campos `dump` y `pass`

  dump lo usa el comando `dump` para determinar de que filesystems hacer copias de seguridad
  - valor 1 o 0 según si la partición va a tener un backup controlado por `dump` o no (normalmente no se usa)

  pass lo usa el comando `fsck` para determinar el orden en que se chequean los filesystems al iniciar el sistema
  - si 0, el filesystem no se chequea

  - si > 0, los filesystems se chequean en el orden indicado por los números
    - si varios tienen el mismo número, se chequean en paralelo (si es posible)
    - normalmente `/` tendrá 1 y el resto 2
  - El número de montados que transcurre entre testeos se especifica para cada filesystem con el comando `tune2fs`.

Fichero `/etc/mtab` Contiene una lista de los filesystem que están montados en el sistema

- Ejemplo de fichero `/etc/mtab`

```
$ cat /etc/mtab
/dev/sda1 / ext4 rw,errors=remount-ro 0 0
proc /proc proc rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev/sda9 /home ext3 rw 0 0
/dev/sda8 /tmp ext3 rw 0 0
/dev/sda5 /usr ext3 rw 0 0
/dev/sda6 /var ext3 rw 0 0
usbfs /proc/bus/usb usbfs rw 0 0
/dev/sdb1 /home2 ext2 rw,nodev 0 0
```

Autofs Sistema que permite montar los filesystems "bajo demanda"

- cuando se accede al directorio, este se monta
- se desmonta automáticamente después de un tiempo de inactividad (por defecto, 5 minutos)
- suele usarse para montar sistemas remotos con NFS
- `/etc/auto.master` define los puntos de montado
  - por cada uno de los puntos definidos, se inicia un proceso `automount` usando los parámetros indicados
  - Ejemplo de `auto.master`:

```
/home     /etc/auto.home
/misc     /etc/auto.misc   --timeout 60
```

- Los ficheros le indican al `automount` los filesystems a montar
  - Ejemplo de `auto.misc`

    ```
    cdrom   -fstype=iso9660,ro   :/dev/cdrom
    windoz  -fstype=vfat         :/dev/sda1
    home    -rw,hard,intr        server:/export/home
    ```

  - esto monta el cdrom y la partición `/dev/sda1` en los directorios `/misc/cdrom` y `/misc/windoz`, y el directorio remoto `/export/home` del sistema `server` en `/misc/home`
- Para más información ver el manual de `autofs` y `automount`, el Autofs Automounter HOWTO o el Automount mini-Howto

Chequeo del sistema de ficheros

Periódicamente es necesario chequear o reparar los sistemas de ficheros

- Formato básico:

  `fsck.`*`tipo`* `[`*`opciones`*`]` *`dispositivo`*

- Al igual que `mkfs`, `fsck` es un front-end a comandos específicos para cada filesystem:
- `fsck.ext2`, `fsck.ext3` y `fsck.ext4`, chequean sistemas ext2/ext3/ext4
- `fsck.jfs`, `fsck.btrfs`, `fsck.xfs` para JFS, btrfs y XFS
- `fsck.vfat`, `fsck.exfat` y `ntfsfix` para sistemas MS Windows

Alguno de los errores que pueden aparecer se deben a:

- Varios ficheros que usan el mismo bloque
- Bloques marcados libres y ocupados simultáneamente
- Número de enlaces erróneo
- Nodos-i conteniendo información pero que no están en la entrada del directorio (la información se recupera en el directorio `lost+found` con el número de nodo-i)
- Entradas del directorio que apuntan a nodos-i ilegales o vacíos

- etc.

Algunas de las opciones de `fsck` son:

- `-y` repara el filesystem sin preguntar sobre los errores

Otras opciones dependen del filesystem particular

- ver las páginas de manual para cada caso

Otras utilidades

- `du`: muestra el espacio de disco usado por los ficheros y subdirectorios de un directorio
  - Formato:

    `du [`*`opciones`*`] [`*`directorio`*`]`
  - Algunas opciones:
    - `-a` (all) muestra valores para ficheros y directorios (por defecto, solo muestra directorios)
    - `-h` (humano) salida más legible
    - `-s` (space) muestra sólo la ocupación total
  - Ejemplo:

```
$ du -hs /home /usr
1,2G    /home
2,3G    /usr
```

- `df`: muestra el espacio de disco usado y disponible de los sistemas de ficheros montados
  - Formato:

    `df [`*`opciones`*`]`
  - Algunas opciones:
    - `-h` (humano) salida más legible
    - `-T` (tipo) muestra el tipo de sistema de ficheros
    - `-l` (local) sólo muestra filesystems locales
  - Ejemplo:

```
$ df -hTl
Filesystem   Tipo   Tamaño Usado  Disp Uso% Montado en
/dev/sda1    ext4      67M   50M   13M  80% /
tmpfs        tmpfs     63M     0   63M   0% /dev/shm
/dev/sda9    ext4     272M  8,1M  250M   4% /home
/dev/sda8    ext4      23M  1,1M   20M   5% /tmp
/dev/sda5    ext4     464M   90M  350M  21% /usr
/dev/sda6    ext4      74M   44M   27M  63% /var
```

Identificador único universal

El identificador único universal (UUID) permite referenciar a discos y a particiones de forma única.

- Tiene la ventaja con respecto a `/dev/sd*` en que permite distinguir diferentes discos y no depende del orden en que son iniciados (al primer disco se le asigna `/dev/sda`, al segundo `/dev/sdb`, etc).
- La correspondencia se puede obtener con el comando `blkid` (como superusuario):

```
$ blkid /dev/sda
/dev/sda: PTUUID="00097350" PTTYPE="dos"
$ blkid
/dev/sda1: UUID="b0f7f038-c762-40f4-aa9b-c718193e1db0" TYPE="ext4" ...
/dev/sda2: UUID="7556114f-e8ad-4777-96e3-35433b14124b" TYPE="swap" ...
/dev/sda3: UUID="b52618f6-657b-4560-a093-20bc7812428b" TYPE="ext4" ...
```

Este identificador se puede utilizar en muchos comandos que trabajan sobre discos y particiones,

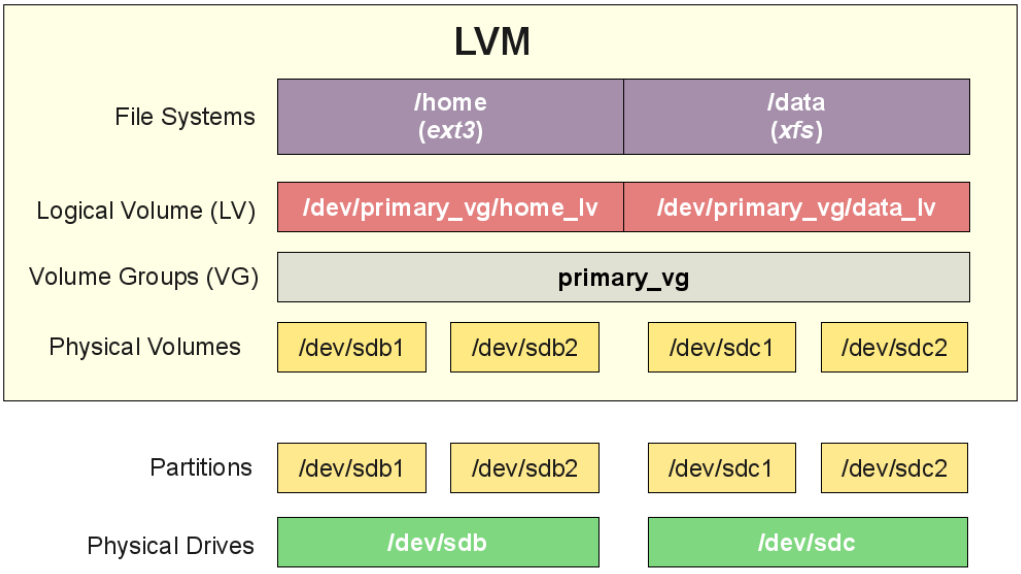
```
$ cat /etc/fstab
# /etc/fstab: static file system information.
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5)

# <file system>           <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda6 during installation
UUID=f727751c-a85d-485e-b681-901110f983a6  /      ext4  ro       0  1
# /home was on /dev/sda1 during installation
UUID=b0f7f038-c762-40f4-aa9b-c718193e1db0  /home  ext4  defaults 0  2
```

3.3.2. Sistemas de ficheros con LVM

En el tema 2 vimos como crear un sistema LVM; algunas de sus ventajas son:

- LVM proporciona mucha más flexibilidad a la hora de distribuir el espacio disponible
- LVM permite mover y cambiar de tamaño los volúmenes creados bajo su control
- Existen varios beneficios inmediatos por usar LVM:
  - Es posible aumentar y decrecer los volúmenes en caliente: esto permite redistribuir el espacio en las particiones según nos sea necesario; también se puede dejar espacio sin asignar e ir asignándolo según vaya siendo necesario
  - Es posible añadir espacio de almacenamiento al sistema de volúmenes: si se añade un nuevo disco a la máquina se puede añadir este espacio a LVM y hacer crecer los volúmenes que contiene para aprovechar el nuevo espacio de almacenamiento



La extructura de LVM es la siguiente:

1. Por el lado físico tenemos los discos que hemos dividido en particiones (*sda1*, *sdb2*, ...), a partir de las cuales se generan los **Volúmenes Físicos (PV)**.

2. A partir de los volúmenes físicos se construyen agrupaciones lógicas denominadas **Grupos de Volúmenes (VG)**. Hay que poner un nombre (etiqueta) a cada VG.

3. Los grupos de volúmenes se dividen de forma lógica en **Volúmenes Lógicos (LV)**. A cada LV ha de dársele una etiqueta.

4. Los volúmenes lógicos pueden cifrarse (véase un apartado posterior). A cada LV cifrado ha de dársele una etiqueta.

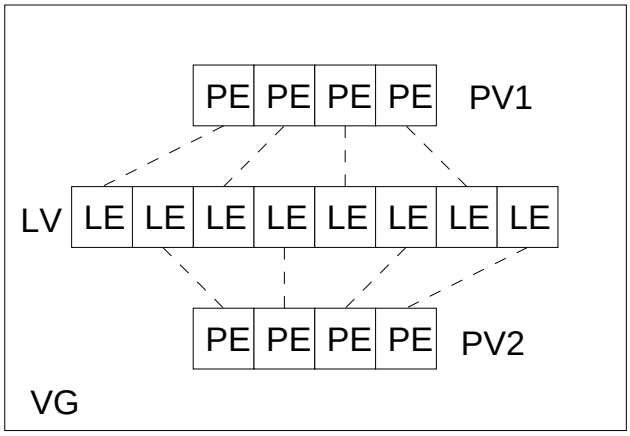
5. Para utilizar un volumen volumen lógico antes debe formatearse.

6. Por último, hay que asignar el volumen lógico formateado al sistema de ficheros (con el comando `mount` o añadiéndolo al fichero `fstab`)

El tamaño de los volúmenes físicos y lógicos puede medirse en bytes o en unidades lógicas o bloques:

- **Extensión física (PE)**: unidades básicas en las que se divide cada volumen físico
- **Extensión lógica (LE)**: unidades básicas en las que se divide cada volumen lógico; su tamaño coincide con el de las PEs de las que está formado

La siguiente figura muestra como se asignan las extensiones físicas a las lógicas mediante un mapeado de tipo *stripping* (otros mapeados posibles son *lineal* y *mirroring*).



En este apartado veremos comandos para manejar sistemas LVM (nota: todos estos comandos tienen distintas opciones, véanse las páginas de manual)

`pvdisplay` o `pvs`
`vgdisplay` o `vgs`
`lvdisplay` o `lvs`
	información del volumen físico
	información del grupo de volúmenes
	información del volumen lógico

`pvcreate`	crear de un volumen físico

`vgcreate`
`vgremove`
`vgextend`
`vgreduce`
	crear un grupo de volúmenes
	borrar un grupo de volúmenes
	añadir un volumen físico a un grupo de volúmenes
	quitar un volumen físico de un grupo de volúmenes

`lvcreate`
`lvremove`
`lvextend`
`lvreduce`
	crear un volumen lógico
	borrar un volumen lógico
	aumentar de tamaño un volumen lógico
	reducir de tamaño un volumen lógico

`fsadm` `resize2fs`, `btrfs`, `xfs_growfs`	Comando genérico cambiar tamaño un filesystem Específicos para ext2/3/4, btrfs y XFS

- Información acerca de un volumen físico: `pvdisplay` o `pvs`

```
# pvdisplay /dev/sda2
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               GrupoVolumen
  PV Size               9,88 GB / not usable 0
  Allocatable           yes (but full)
  PE Size (KByte)       32768
  Total PE              316
  Free PE               0
  Allocated PE          316
  PV UUID               U6rMMw-5Z9U-fhBH-4R6G-reeJ-ZVha-K4xyHs
```

- Información acerca de un grupo de volúmenes: `vgdisplay` o `vgs`

```
# vgdisplay GrupoVolumen
  --- Volume group ---
  VG Name               GrupoVolumen
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                6
  Open LV               6
```

```
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               14,84 GB
  PE Size               32,00 MB
  Total PE              475
  Alloc PE / Size       473 / 14,78 GB
  Free  PE / Size       2 / 64,00 MB
  VG UUID               N2NjFx-7ISe-J7hH-EX03-231w-XfbS-eCYfv0
```

- Información acerca de un volumen lógico: `lvdisplay` o `lvs`

```
# lvdisplay /dev/GrupoVolumen/homelv
  --- Logical volume ---
  LV Name                /dev/GrupoVolumen/homelv
  VG Name                GrupoVolumen
  LV UUID                dI6BqF-LAeG-3fO9-jXcr-vNde-f7iF-y9Oa2l
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                1,00 GB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:1
```

Manejar volúmenes físicos y grupos de volúmenes

- Creación de un volumen físico (PV), sobre una partición de tipo LVM

```
# pvcreate /dev/sdc1
```

- Crear un grupo de volúmenes (VG), de nombre NuevoGrupo a partir de dos PVs

```
# vgcreate NuevoGrupo /dev/sda2 /dev/sdc1
```

- Borrar un VG

```
# vgremove NuevoGrupo
```

- Añadir el PV `/dev/sdc1` a un VG ya creado

```
# vgextend GrupoVolumen /dev/sdc1
```

- Quitar PVs de un VG

```
# vgreduce NuevoGrupo /dev/sda2
```

Trabajar con volúmenes lógicos

- Crear un volumen lógico (LV) de nombre testlv en el VG NuevoGrupo con un tamaño de 4.20 GB

```
# lvcreate -L4.20G -n testlv NuevoGrupo
```

- Borrar un volumen lógico (hay que desmontarlo primero)

```
# umount /dev/NuevoGrupo/otrotestlv
# lvremove /dev/NuevoGrupo/otrotestlv
```

NOTA: si hay procesos accediendo al sistema de ficheros del LV, antes hay que matarlos con `kill`, si no, nos dirá que el filesystem está en uso:

```
# lsof /path-to-filesystem
# kill -KILL número_de_proceso
```

- Agrandar un LV; se puede especificar el nuevo tamaño en bytes (`-L`) o LEs (`-l`), o la diferencia (`+/-`)

```
# lvextend -L12G /dev/GrupoVolumen/homelv
# lvextend -L+1G /dev/GrupoVolumen/tmplv
# lvextend -l+200 /dev/GrupoVolumen/tmplv
```

- Reducir un LV: `lvreduce` funciona igual que el `lvextend`

Dispositivo asignado a LVM

Como vimos en el apartado anterior, una vez creado el volumen lógico, los comandos que operan sobre el lo referencian a través del nombre del dispositivo. El sistema proporciona dos formas para acceder al dispositivo:

- Directamente

  `/dev/GrupoVolumen/VolumenLogico`

- A través del *mapeador de dispositivos (device mapper)*

  `/dev/mapper/GrupoVolumen-VolumenLogico`

El mapeador de dispositivos es un entorno proporcionado por el núcleo de Linux para el mapeo de dispositivos de bloque (discos y similares) físicos a dispositivos virtuales de nivel superior.

- Constituye la base de LVM2, RAID y del software de cifrado de disco dm-crypt, y ofrece características adicionales, tales como instantáneas (snapshots) del sistema de ficheros.
- Esta forma de acceso es un poco más cómoda cuando se trabaja con volumenes cifrados.

Asignar sistemas de ficheros

1. Una vez creados los volúmenes lógicos hay que crear los sistemas de ficheros y montarlos comandos (`mkfs` y `mount`)

   ```
   # mkfs.ext4 /dev/GrupoVolumen/homelv
   # mount /dev/GrupoVolumen/homelv /home
   ```

2. Si hemos agrandado un LV debemos agrandar el filesystem. El siguiente comando es un front-end para los comandos específicos de diferentes filesystems (ext2/3/4, btrfs, XFS):

   - `fsadm` chequea y redimensiona un sistema de ficheros

   Ejemplo: aumenta a 2G el tamaño del filesystem

   ```
   # fsadm resize /dev/mapper/GrupoVolumen-usrlv 2048M
   ```

   Si no se especifica tamaño, aumenta al máximo posible

3. Comandos específicos de redimensionado de ficheros. Permiten ampliación en caliente (sin desmontar), aunque para redución requieren el desmontado. Las características de cada comando dependen de cada filesystem particular:

   ext2/3/4 comando `resize2fs` (en Debian, paquete `e2fsprogs`):

   ```
   # resize2fs /dev/GrupoVolumen/homelv
   ```

   btrfs comando `btrfs` (en Debian, paquete `btrfs-tools`)

   XFS se usa el comando `xfs_growfs` (en Debian, paquete `xfsprogs`)

   JFS el filesystem se amplía de tamaño haciendo un remontado:

   ```
   # mount -o remount,resize=1048576 /home
   ```

3.3.3. Manejo de discos cifrados

El cifrado que se usa en el proceso de instalación es un cifrado de disco completo (*Full disk encryption*, FDE)

- Se cifran todos los bits del disco o de la partición
- Es diferente del cifrado a nivel de sistema de ficheros (*Filesystem-level encryption*, FLE) en el que se cifra el contenido de los ficheros, no los metadatos (nombre del fichero, fechas de modificación, etc.).

Precauciones:

- Si nos olvidamos de la clave de acceso o la borramos del sistema, no será posible acceder al filesystem.
- Una vez establecido el cifrado no hay una forma directa de eliminarlo. Quizás la forma más fácil sea, una vez descifrado el filesystem, copiar su contenido a una partición auxiliar con el comando `dd`, eliminar la configuración de cifrado y finalmente copiar de vuelta el contenido a la partición original.

El subsistema de cifrado FDE en Linux 4 es *dm-crypt*

- Parte de la infraestructura *device mapper*, utilizada también en LVM2 o RAID software, y puede colocarse por encima de estos
  - Permite cifrar discos completos, particiones, volúmenes lógicos o volúmenes RAI software
- Comando básico: `cryptsetup`
- Permite utilizar el estándar LUKS (*Linux Unified Key Setup*)
- Fichero `/etc/crypttab`: indica en el arranque como descifrar los discos

LUKS

Estándar para cifrado de disco en Linux

- Facilita la compatibilidad entre distribuciones
- Incluye soporte para múltiples claves
- Revocación de contraseña efectiva
- Uso mediante el comando *cryptsetup*, con *dm-crypt* como backend

cryptsetup

- Comando básico para el cifrado de disco. Puede cifrar sin LUKS (sistema sencillo que básicamente solo necesita la operación `create`) o con LUKS (más elaborado, con muchas operaciones posibles)

Operaciones del comando:

`create`	cifrado sin LUKS
`luksFormat`	iniciar el formato LUKS y añadir la primera clave
`luksOpen`	abrir, descifrar para acceder
`luksDump`	mostrar los slots de claves
`luksAddKey`	añadir una nueva clave
`luksRemoveKey`	borrar una clave
`luksKillSlot`	borrar un slot (sin conocer su clave)
`luksUUID`	obtener el identificador universal (UUID)
`resize`	redimensionar si se ha modificado el volumen lógico
`crypttab`, `fstab`	ficheros que se leen en el arranque

Son posibles los siguientes tipos de claves:

1. Clave aleatoria. Estas claves se obtienen de `/dev/random` (mejor) o de `dev/urandom` (peor).[4] Cada vez que arranca la máquina la partición se cifra de forma diferente, con los cual no es posible acceder a los datos del arranque anterior. Es adecuado para swap o para `/tmp`.

2. Clave desde fichero. La clave está previamente guardada en un fichero que se lee en el arranque. Para evitar que alguien pueda leer este fichero es aconsejable que esté contenido en una partición previamente cifrada.

3. Frase de contraseña. En el arranque nos pide la clave, que debemos teclear.

- NOTA: En los siguientes ejemplos se considera que no se usa LVM. En caso contrario ha de usarse */dev/mapper/GrupoVolumen-VolumenLogico*.

[4] `/dev/random`: genera números aleatorios basándose en la entropía (ruido) del sistema; `/dev/urandom`: genera números seudoaleatorios usando la entropía como semilla. `/dev/random` genera números de mayor calidad, pero es lento y puede bloquearse si la entropía del sistema es baja; `/dev/urandom` es un poco menos seguro, pero puede ser adecuado

Cifrado sin LUKS

Se requieren los siguientes pasos:

1. Cifrado de la partición. Se realiza con el comando `cryptsetup`, que requiere indicar una etiqueta para la partición cifrada (usualmente se añade el sufijo `_crypt` a la partición original), el dispositivo de la partición original y la clave

   ```
   cryptsetup create sda7_crypt /dev/sda7 --key-file /dev/urandom
   ```

   La partición ya está cifrada. Ahora hay que decirle al ordenador que la descifre en el arranque.

2. El fichero `/etc/crypttab` especifica en el arranque cómo se han de descifrar las particiones. El fichero consta de líneas con cuatro campos:

   - nombre de la partición una vez descifrada.
   - dispositivo de la partición original.
   - clave de descifrado.
   - opciones.

   Suponemos que se va a tratar de una partición de *intercambio*

   ```
   #<descifrado>  <original>   <clave descifrado>   <opciones>
   sda7_crypt     /dev/sda7    /dev/urandom         ...
       ... cipher=aes-cbc-essiv:sha256,size=256,swap
   ```

3. Añadir a `fstab` el volumen lógico cifrado

   ```
   (file system)   (mount point) (tipo) (opciones) (dump) (pass)
   /dev/sda7_crypt  none          swap  sw          0     0
   ```

Cifrado con LUKS

- En LUKS, por cada partición encriptada, puede haber hasta ocho claves (slots) diferentes. Se puede optar por tener cualquier número de claves.
- Cualquiera de las ocho claves puede abrir la partición cifrada.

Disponemos de los siguientes comandos:

- Para ver los slots asignados:

```
# cryptsetup luksDump /dev/sdb1
Key Slot 0: ENABLED
Key Slot 1: ENABLED
Key Slot 2: DISABLED
...
```

- Para añadir una nueva clave (nos pregunta una clave antigua):

```
# cryptsetup luksAddKey /dev/sdb1
Enter any passphrase:
Enter new passphrase for key slot:
Verify passphrase:
```

- Para añadir una clave desde un fichero (hay que responder con cualquiera de las claves existentes):

```
cryptsetup luksAddKey /dev/sdb1 masterkeyfile
Enter any passphrase:
```

- Para cambiar la clave del slot 1 que ya tenemos:

```
cryptsetup luksAddKey /dev/sdb1 -S 1
```

- Para borrar una clave (no hay que especificar el slot, basta con la clave):

```
cryptsetup luksRemoveKey /dev/sdb1
```

- Para borrar el slot 2 sin conocer su clave:

```
# cryptsetup luksKillSlot /dev/sdb1 2
Enter any remaining LUKS passphrase:
```

Extensión de un volumen cifrado

Si extendemos un volumen lógico y este está cifrado, a continuación debemos extender también el dispositivo cifrado. Para ello se utiliza también el comando `cryptsetup`:

```
# cryptsetup resize /dev/mapper/GrupoVolumenes-VolumenLogico_crypt
```

Ejemplos con LUKS

1. Cifrar y activar una partición usando formato LUKS y una contraseña como clave (todos los datos se pierden y debemos reiniciar el filesystem)

```
# Desmontar la partición
umount /dev/sda8
# Se podría sobreescribir, pero puede ser muy lento
dd if=/dev/urandom of=/dev/sda8
# Formatea la partición, cifrando con LUKS
# nos va a pedir establecer la primera clave
cryptsetup luksFormat /dev/sda8
# Abre (descifra) la partición en el
# dispositivo /dev/mapper/sda8_crypt
cryptsetup luksOpen /dev/sda8 sda8_crypt
# Reinicia el sistema de ficheros
mkfs.tipo /dev/mapper/sda8_crypt
# Obtiene el UUID luks
crypsetup luksUUID /dev/sda8
```

2. Usar un fichero de clave para una partición cifrada inicialmente con contraseña, eliminando la contraseña inicial

```
mkdir /etc/keys
# Crea un fichero aleatorio para usar como clave
dd if=/dev/urandom of=/etc/keys/sda6.luks bs=1 count=4096
# Cambia los permisos del fichero (debe ser de root)
chown root:root /etc/keys/sda6.luks
chmod 700 /etc/keys
chmod 400 /etc/keys/sda6.luks
# Añade el fichero sda6.luks como clave
cryptsetup luksAddKey /dev/sda6 /etc/keys/sda6.luks
# Comprueba que existen dos claves (slots)
cryptsetup luksDump /dev/sda6
# Si se desea, se puede borrar el slot 0 con la clave original
# (impide usar esa clave, hacerlo solo después de comprobar que
el fichero luks funciona como clave)
cryptsetup luksKillSlot /dev/sda6 0 --key-file /etc/keys/sda6.luks
# Comprueba que el slot se ha borrado
cryptsetup luksDump /dev/mapper/sda6_crypt
```

```
# Por último, modifica el fichero crypttab tal como se muestra en
el siguiente apartado para indicar que se use el fichero luks
```

3. Extiende el dispositivo cifrado `grupovol-homelv_crypt`, después de haber extendido el volumen lógico sobre el que está definido

```
# cryptsetup resize /dev/mapper/grupovol-homelv_crypt
```

Fichero /etc/crypttab

Especifica en el arranque como se deben descifrar los discos
Ejemplo:

```
#<descifrado>  <original>  <clave descifrado>   <opciones>
sda8_crypt     /dev/sda8   none                 luks
sda6_crypt     /dev/sda6   /etc/keys/sda6.luks  luks
```

Línea 1 partición con cifrado LUKS; al poner `none` en el campo de clave se indica que se pide una contraseña en el arranque

Línea 2 partición con cifrado LUKS y clave obtenida desde fichero

NOTA: a pesar de que la etiqueta sugiere lo contrario, el dispositivo original está cifrado, mientras que la partición que tiene el prefijo `_crypt` está descifrada.

Para evitar problemas (posibles cambios en el nombre de las particiones), es preferible substituir el nombre del dispositivo original por su UUID obtenido usando `crypsetup luksUUID`

```
# cryptsetup luksUUID /dev/sda8
0e44dcc3-2b1f-4349-9fb3-db82b547acd1
# cat /etc/crypttab
.....
sda8_crypt UUID=0e44dcc3-2b1f-4349-9fb3-db82b547acd1 none luks
.....
.....
```

3.4. Gestión de usuarios

Todo usuario de un sistema UNIX debe tener una cuenta para poder acceder.

Cuenta UNIX: colección de características lógicas que especifican quien es el usuario y lo que puede hacer en el sistema. Estas características incluyen:

- nombre de usuario (*login* o *user name*) e identificador numérico (UID)
- la contraseña (*passwd*)
- grupo o grupos a los que pertenece, con un identificador numérico del grupo por defecto (GID)
- un directorio *home*
- un *login shell*
- una colección de ficheros de inicio

Entre las cuentas asociadas a usuarios podemos encontrar diferentes tipos:

- cuentas normales de usuario
- cuenta del administrador (*root*)
- cuentas especiales de los servicios (*nobody*, *lp*, *bin*, etc.), usadas por servicios internos del sistema, aumentan la seguridad, al permitir que servicios del sistema no se ejecuten como *root*

Comandos y ficheros disponibles en Linux:

`/etc/passwd`, `/etc/shadow`	información de usuarios
`/etc/group`, `/etc/gshadow`	información de grupos
`/etc/skel`, `/etc/adduser.conf`	plantillas para crear las cuentas
`chown`, `chgrp`	cambio del propietario de ficheros/dir
`chmod`	cambio de permisos para ficheros/dir
`passwd`, `vipw`	cambio contraseña/información de usuarios
`gpasswd`	gestión de grupos (clave, miembros) por root
`chage`	cambio de expiración
`newgrp`	cambio entre grupos por el propio usuario
`useradd`, `userdel`, `usermod`	creación de usuarios (bajo nivel)
`groupadd`, `groupdel`, `groupmod`	creación de grupos (bajo nivel)
`adduser`, `deluser`	creación de usuarios (alto nivel)
`addgroup`, `delgroup`	creación de grupos (alto nivel)
`newusers`, `chpasswd`	operación múltiple
`mkpasswd`	obtiene la versión cifrada de una clave
`su`, `sudo`, `/etc/sudoers`	pasar/ejecutar como root/otro usuario

3.4.1. Ficheros de información de los usuarios

La información de usuarios y grupos está incluida en los archivos:

- `/etc/passwd` mantiene la información principal de cada cuenta: nombre de usuario, UID, GID, *login shell*, directorio *home*, contraseña (en sistemas antiguos), ...
- `/etc/shadow` en sistemas actuales, fichero sin permiso de lectura que guarda las contraseñas encriptadas
- `/etc/group` información sobre los grupos definidos en el sistema. nombre del grupo, GID y miembros del mismo
- `/etc/gshadow` contraseñas para grupos

Fichero `/etc/passwd`

Ejemplo de líneas de `/etc/passwd`:

```
root:x:0:0:root:/root:/bin/bash
pepe:x:1002:1002:Pepe Perez,dept.EC,desp.1:/home/pepe:/bin/bash
```

donde se indican (si aparecen `::` seguidos, el campo está vacío):

- `pepe`: identificación de usuario en el sistema, que deberían tener las siguientes características
  - únicos en toda la organización (no sólo en la máquina local)
  - preferiblemente corto, en minúsculas y sin caracteres acentuados (para evitar problemas)
  - fácil de recordar
  - de formato fijo para todos los usuarios (p.e. nombre+apellido)
- `x`: contraseña encriptada
  - si aparece una `x` la contraseña está en el fichero `/etc/shadow`
- `1002`: UID número identificador del usuario
  - para usuarios normales, número entre 1000 y 65535
  - números por debajo de 1000 para usuarios especiales del sistema (`root` usualmente número 0)

  - el UID para un usuario debería ser único, y el mismo para todas las máquinas
  - se debe evitar reutilizar un UID, para evitar problemas de pertenencia de archivos
- `1002`: GID código del grupo principal al que pertenece el usuario
- `Pepe Perez,dept.EC,desp.1`: información GECOS
  - cualquier cosa, usualmente el nombre completo del usuario y información adicional (n. de despacho, teléfono, etc.)
- `/home/pepe`: directorio personal del usuario
- `/bin/bash`: shell interactivo que utilizará el usuario

Fichero `/etc/shadow`

Fichero de acceso restringido que almacena las contraseñas encriptadas:

```
pepe:$1$.QKDPc5E$SWlkjRWexrXYgc98F.:12825:0:90:5:30:13096:
```

Contiene las contraseñas encriptadas y otros campos separados por `:`

- día en que la contraseña se cambió por última vez
  - si vale 0 se fuerza a que el usuario cambie su contraseña la primera vez que se conecta
  - se cuenta como número de días a partir del 1/1/1970 (también conocido como *epoch*)
- número de días que deben pasar hasta que pueda ser cambiada
- número de días de validez de la contraseña. Si el plazo expira, el usuario podrá entrar en la cuenta, pero será forzado a cambiar la contraseña.
- número de días de antelación del aviso de caducidad de la contraseña
- número de días en que se deshabilitará la cuenta, una vez expirada
- día en que la cuenta se inhabilitará (contado desde el 1/1/1970)
  - si no aparece nada, la cuenta no se inhabilita nunca
- un campo reservado

Fichero `/etc/group`

Información sobre los grupos de usuarios

```
users:x:100:pepe,elena
```

donde tenemos

- nombre del grupo
- contraseña del grupo (no suele usarse)
  - si `x`, se guarda en el fichero `/etc/gshadow`
- GID identificador numérico del grupo
- lista de usuarios que pertenecen al grupo

Algunos grupos

`users`	son los usuarios normales
`sudo`	puede ejecutar cualquier comando (son superusuarios)
`http`	puede gestionar los servicios web
`games`	puede gestionar los juegos
...	

Cuando se crea un nuevo usuario por defecto se le da su propio grupo.

Fichero `/etc/gshadow`

- Si existe, contiene las contraseñas de los grupos y una copia de la lista de miembros
- El administrador puede fijar/cambiar la contraseña de cada grupo con el comando `gpasswd`

Cambio de grupo

- un usuario puede cambiar de grupo con `newgrp`
  - si el usuario es miembro no se le preguntará la contraseña
  - si el usuario no es miembro y el grupo tiene contraseña se le preguntará al usuario
  - si el usuario no es miembro y el grupo no tiene contraseña se le denegará el cambio.
  - si el grupo existe en `/etc/gshadow` se usará la lista de miembros y la clave de este fichero en vez de `/etc/group`

Otros ficheros

El administrador debe crear los ficheros de inicio para los usuarios, especificando los paths de ejecución, variables del sistema, etc.

- durante la creación de una cuenta, los ficheros de inicio del nuevo usuario se copian del directorio `/etc/skel`
- también pueden usarse los ficheros `/etc/profile` o `/etc/bash.bashrc` (ver Tema 2, *Ficheros de inicialización de Bash*)

3.4.2. Creación manual de una cuenta

Implica los siguientes pasos.

1. Editar el fichero `/etc/passwd` y añadir una nueva línea para la cuenta
   - para evitar corrupción del fichero usar el comando `vipw`
     - este comando solo grabará el fichero si el formato es correcto
     - nos permite seleccionar el editor a usar (`vi`, `nano`, etc.)
   - Si el sistema usa `shadow` (lo normal), poner `x` en el campo de contraseña
2. Editar el fichero `/etc/shadow`
   - para evitar corrupción del fichero usar el comando `vipw -s`
   - la contraseña debe escribirse cifrada (lo hace el comando `passwd`)
3. Editar `/etc/group` para añadir un nuevo grupo (si es necesario) o para añadir el usuario a los grupos que deseemos
4. Si es necesario, editar `/etc/gshadow`
   - poner la contraseña cifrada (! o * si no queremos ninguna)
   - añadir también aquí la lista de miembros del grupo
5. Crear el directorio del usuario
6. Copiar los ficheros de `/etc/skel` al directorio del usuario
7. Usar `chown`, `chgrp` y `chmod` para fijar el propietario, grupo y permisos del directorio
8. Fijar la contraseña con `passwd`
   - el usuario debe cambiar la contraseña tan pronto como sea posible
   - puede forzarse con la opción `-e`

Comando `passwd`

Permite fijar, cambiar la contraseña de un usuario o sus propiedades.

- Formato:

  `passwd [`*`opciones`*`] [`*`username`*`]`

- Opciones:
  - `-e` fuerza a que el usuario cambie la contraseña al siguiente login
  - `-d` borra la contraseña (e impide el acceso a la cuenta)
  - `-l`/`-u` bloquea/desbloquea la cuenta
  - `-m MIN_DAYS` número mínimo de días entre cambios de contraseña
  - `-x DÍAS_MAX` número de días de validez de la contraseña
  - `-w DÍAS_AVISO` número de días de aviso de caducidad
  - `-i INACTIVO` número de días en que se deshabilitará la cuenta una vez expirada la contraseña
  - `-S` indica el estado de la contraseña (`L`: bloqueada, `NP`: sin contraseña o `P`: con contraseña válida) junto que información de la expiración

Comando `chage`

Para cambiar la información de expiración de la contraseña también puede utilizarse el comando `chage`

- Formato:

  `chage [`*`opciones`*`] [`*`username`*`]`

- Algunas opciones:
  - `-l` muestra información de expiración

- Ejemplo:

```
$ chage -l debian
Último cambio de contraseña                                : ago 09, 2016
La contraseña caduca                                       : nunca
Contraseña inactiva                                        : nunca
La cuenta caduca                                           : nunca
Número de días mínimo entre cambio de contraseña           : 0
Número de días máximo entre cambio de contraseña           : 99999
Número de días de aviso antes de que caduque contraseña : 7
```

3.4.3. Comandos para gestión de cuentas

Comandos simples de manejo de cuentas

- `useradd` añade un nuevo usuario al sistema. Uso:

  `useradd [`*`opciones`*`] `*`username`*

  - por defecto, sólo modifica los ficheros `passwd` y `shadow`, no crea el directorio home ni le pone contraseña (cuenta inhabilitada)
  - varias opciones:
    - `-c` *`geckos`* especifica los comentarios
    - `-m` crea el directorio *home* y copia los ficheros de `/etc/skel`
    - `-d` *`home`* especifica el directorio *home* del usuario si no queremos ponerle el por defecto
    - `-g` *`grupo`* especifica el grupo principal
    - `-G` *`grupo`* especifica los grupos adicionales (los grupos se separan por comas, sin espacios entre ellos)
    - `-s` *`shell`* especifica la shell a utilizar
    - `-p` *`clave`* pone la contraseña (debe estar cifrada con `mkpaswd`)
    - `-e` *`fecha`* fecha de expiración de la cuenta (YYYY-MM-DD)
  - Ejemplo:

    ```
    useradd -c "Aitor Tilla" -m -g staff -s /bin/bash
    -e 2006-11-02 aitor
    ```

- `userdel` borra un usuario del sistema
- `usermod` modifica las cuentas de usuario
  - Puede utilizarse para cambiar cualquier parámetro de la cuenta, por ejemplo, puede cambiar el grupo principal del usuario (opción `-g`), cambiar los grupos adicionales (opción `-G`) o añadir grupos adicionales (con las opciones combinadas `-G -a`).
  - También puede usarse `gpasswd` para añadir o eliminar un usuario a un grupo (además de para poner la clave a un grupo).
- `groupadd` (crea un nuevo grupo en el sistema), `groupdel` (borra un grupo), `groupmod` (modifica un grupo existente)

- `mkpasswd` obtiene la versión cifrada (usando la función `crypt`) de una cadena para usar como contraseña (admite un `salt`, que funciona como una semilla y fortalece el cifrado):

```
mkpasswd -m sha-512 -s salt mypassword
```

Comandos de alto nivel para el manejo de cuentas

- Comandos `adduser`, `addgroup`:
  - hacen de front-end a los comandos de bajo nivel anteriores `useradd`, `groupadd` y `usermod`
  - crean los usuarios/grupos en función de la configuración especificada en el fichero `/etc/adduser.conf`

Comandos de gestión de múltiples cuentas

- `newusers` permite crear varias cuentas a partir de un fichero con nombres de usuario y contraseñas
  - las líneas del fichero deben tener el mismo formato que las del fichero `/etc/passwd`, con la contraseña sin encriptar
  - Si el usuario existe, se modifican los parámetros
  - Se crea el directorio *home* especificado, si este no existe
- `chpasswd` lee líneas en el formato `user_name:password` (sin encriptar) y actualiza las contraseñas de usuarios existentes:

```
echo "pepe:pepepassword" | chpasswd
```

Otros comandos relacionados

- `su`: comando que permite cambiar de usuario o pasar a administrador

```
su [opciones] [-] username
```

  - Si no se especifica el *username* pasa a administrador (root)
  - Nos pedirá la clave del usuario destino (salvo que seamos root)
  - Algunas opciones:
    - `-` inicia un login shell (igual al usuario entrando en su cuenta)
    - `-p` preserva el entorno (no ejecuta el `.bashrc` del usuario)

- `sudo` ejecuta un comando como administrador o con la identidad de otro usuario.

  `sudo [-u username] commando`

  - Si no se encuentra instalado en el sistema, puede instalarse con `apt-get install sudo`
  - Si no se especifica el *username*, el comando se ejecuta como root
  - La clave que pide es la del usuario que lo ejecuta (no la de root)
  - Un usuario del grupo *sudo* puede ejecutar cualquier comando
  - En caso contrario se comprueba el fichero `/etc/sudoers` para ver si el usuario está la lista de los permitidos a usar el comando

- `/etc/sudoers` permite dar permisos a ciertos usuarios para ejecutar ciertos comandos privilegiados.

  - Muchas veces es necesario otorgar a un usuario distintos permisos para que pueda hacer uso de comandos propios del administrador.
  - Es impensable que un administrador "preste" a un usuario la contraseña de root y tampoco sirve el comando `su` pues de nuevo necesita la contraseña de root.
  - La mejor alternativa es hacer uso de `sudo` controlado por el fichero de configuración `/etc/sudoers`
  - Este fichero de configuración se puede editar con el comando `visudo`, que solo permite guardar los cambios si el formato del fichero es el adecuado.

  Ejemplo básico de configuración:

```
#usuario  host=(user:group)  opciones: comando
debian    ALL=(ALL:ALL)      NOPASSWD: /usr/sbin/vipw
```

  - Indicamos el usuario que puede ejecutar el comando. Podemos poner una lista de usuarios separados por comas.
  - `ALL=` indica que el permiso es válido en todos los hosts. Esto nos permite copiar el mismo fichero en todas las máquinas de una red e indicar en qué máquina o lista de máquinas el permiso es válido. Para esto último, aquí se pondría una lista de máquinas separadas por comas.

- `(ALL:ALL)` indica que el usuario puede escalar privilegios a cualquier otro usuario y grupo existente.
- `PASSWD` y `NOPASSWD` indican si se preguntará o no la clave al ejecutar el comando (la clave del propio usuario, no la del administrador).

  La autentificación es válida durante algunos minutos, por lo que en este intervalo la clave no se volverá a preguntar. Es posible especificar la duración de este intervalo en una de las opciones globales de configuración.
- Por último se escribe el comando. Es recomendable indicar el path completo y si es necesario las opciones que puede ejecutar.

- Para hacer uso de los provilegios, el usuario debe ejecutar el comando con `sudo` *comando*.

Otro ejemplo de `/etc/sudoers`. Para facilitar la escritura se permiten establecer alias para grupos de usuarios, grupos de máquinas y grupos de comandos.

```
# lista de usuarios administradores de red
User_Alias NETOPS = marcela, andrea

# lista de usuarios webmasters
User_Alias WEBMAS = cristina, juan

# lista de maquinas servidores web
Host_Alias WEBSERVERS = 10.0.1.100, 10.0.1.101

# lista de comandos de red permitidos
Cmnd_Alias REDCMDS = /sbin/ifconfig, /sbin/iptables

# listas de comandos de apache
Cmnd_Alias APACHECMDS = /usr/sbin/apache, /sbin/service httpd *

# definicion de reglas
# admin. de red, en todos los equipos, ejecutar comandos de red
NETOPS ALL = REDCMDS

# webmasters, en los servidores web con los comandos indicados
# y sin necesidad de contraseña reiniciar los servidores.
WEBMAS WEBSERVERS = APACHECMDS, NOPASSWD: /sbin/reboot
```

3.4.4. Módulos de autentificación

PAM (*Pluggable Authentication Module*) es una biblioteca de autentificación genérica que cualquier aplicación puede utilizar para validar usuarios, utilizando por debajo múltiples esquemas de autentificación alternativos (ficheros locales, claves de un solo uso, DNI electrónico, Kerberos, LDAP, etc.) PAM utiliza módulos de varios tipos:

- Módulos de autentificación (`auth`): para la identificación del usuario (por ejemplo, contraseña, tarjeta de identificación, características biométricas, etc).
- Módulos de cuentas (`account`): controlan las condiciones para que la autenticación sea permitida (por ejemplo, que la cuenta no haya caducado, que el usuario tenga permiso para iniciar sesiones a esa hora del día, etc.)
- Módulos de contraseña (`password`): condiciones y procedimientos para el cambio de contraseñas
- Módulos de sesión (`session`): configuran y administran sesiones de usuarios (tareas adicionales que son necesitadas para permitir acceso, como el montaje de directorios, actualización del fichero `lastlog`, etc.)

La configuración se realiza de la siguiente forma:

- Existe un fichero de configuración para cada servicio que usa PAM.
- También existen ficheros comunes que son incluidos por los ficheros de configuración: `common-auth`, `common-account`, `common-password` y `common-session`
- Una vez modificados hay que ejecutar el comando `pam-auth-update`

Claves de un solo uso

A modo de ejemplo de uso de PAM instalaremos el módulo de autentificación que permite las claves de un solo uso (OTPW - One Time PassWord).

1. Instalamos los paquetes con

   ```
   # apt-get install otpw-bin libpam-otpw
   ```

2. Configuramos PAM. Para ello incluimos en fichero de configuración `/etc/pam.d/common-auth` las dos primeras lineas que indicamos:

```
# Anadimos estas dos lineas para usar OTPW
auth       sufficient pam_otpw.so
session    optional   pam_otpw.so
# deben colocarse antes de las dos lineas ya existentes
auth  [success=1 default=ignore] pam_unix.so nullok_secure
auth  requisite                  pam_deny.so
```

El orden de las líneas es importante para que primero nos pida la clave de un solo uso y solo en caso de fallo se pida la contraseña normal.

3. Para que funcione `ssh` con OTPW:

   - Editamos la configuración del servidor ssh: `/etc/ssh/sshd_config` y podemos a *yes* las opciones, que ya se encuentran en el fichero:

     ```
     UsePAM yes
     KbdInteractiveAuthentication yes
     ```

   - Reiniciamos el servicio con: `systemctl reload ssh`

Cada usuario deberá generar las claves OTPW con el comando `otpw-gen`

1. El usuario genera las claves (conviene que las imprima)

   ```
   $ otpw-gen
   Generating random seed ...
   Enter new prefix password:
   Reenter prefix password:
   Creating '~/.otpw'.
   Generating new one-time passwords ...
   ```

2. Nos pide un prefijo que deberemos incluir antes de la clave generada cuanto intentemos el acceso.

3. La salida es del tipo:

   ```
   000 ENBC GjEr  056 a5Lo ePue  112 8Ysy FNGH  168 GJiL Xy7M  224 DPTd HUqR
   001 5SBa dA%f  057 xxCF pm7a  113 M8k: Cw=k  169 ZCfi oUof  225 vLgE krI4
   002 e5KX W=4a  058 idW4 R4A+  114 Tbp6 bopC  170 EP+J whvv  226 Xng3 hM5b
   ```

4. Cuando accedamos:

   ```
   $ ssh -p 2222 debian@localhost
   Password 057:   # respondemos con: {prefix}xxCF pm7a
   ```

3.4.5. Cuotas de disco

Algunos filesystems permiten limitar el uso del disco a los usuarios y grupos

- Evitan que los usuarios monopolicen el disco
- Pueden causar problemas a los usuarios:
  - preferible instalar más disco o avisar a los usuarios que consuman demasiado

Límites de cuotas:

- **Límite débil**: si la cuenta del usuario o del grupo supera el límite débil, se impondrá un *período de gracia* en el que el usuario podrá reducir la ocupación
- **Límite duro**: se deniega cualquier intento de escribir datos después de este límite
- **Período de gracia**: tras superar el límite débil, si el usuario no resuelve el problema borrando archivos, la cuenta se bloquea

Cuotas de usuario y de grupos

- **Usuario**: fija un máximo al espacio de todos los ficheros del usuario
- **Grupo**: fija un máximo al espacio de todos los ficheros del grupo (cuenta el espacio consumido por los ficheros de varios usuarios)

Instalación de cuotas de disco en Debian

Los comandos y ficheros involucrados son los siguientes:

`/etc/fstab`	para indicar los filesystems que tendrán cuotas
`quotacheck`	construye el índice y testea la integridad
`quotaon/quotaoff`	activa/desactiva las cuotas
`edquota`	ajusta las cuotas de usuarios o grupos
`repquota`	genera informes de uso
`quota`	informa a un usuario del estado de sus cuotas

Los pasos a seguir son:

1. Instalar el paquete `quota`
2. Cambiar las opciones de `/etc/fstab` para marcar los filesystems que tendrán cuotas:

```
(file system) (mount point) (tipo) (opciones) (dump) (pass)
/dev/hda9     /home    ext4  defaults,usrquota,grpquota 0 1
```

3. Remontar el filesystem que hemos modificado

   ```
   mount -vo remount /home
   ```

4. Crear los índices de las cuotas.

   ```
   quotacheck -vguma
   ```

   Este comando construye el índice y verifica la integridad de las bases de datos de las cuotas

   - se ejecuta en el script de inicio del sistema de cuotas
   - la primera vez que se ejecuta puede dar un mensaje indicando que el índice todavía no existe. No lo dará si repetimos el comando
   - debe ejecutarse con las cuotas desactivadas

5. Activar las quotas:

   ```
   quotaon -va
   ```

   En cualquier momento, podemos usar `quotaon`/`quotaoff` para activar/desactivar el sistema de cuotas

6. Reiniciar la máquina y usar el comando `edquota` para editar las cuotas de usuarios y grupos

- Sintaxis:

  ```
  edquota [opciones] [usuario|grupo]
  ```

- Opciones:

  - `-u` *`usuario`* configura las cuotas del usuario
  - `-g` *`grupo`* configura las cuotas para un grupo
  - `-f` *`filesystem`* realiza las operaciones sobre un filesystem concreto (por defecto, lo hace sobre todos los filesystems que admitan cuotas)
  - `-t` configura el período de gracia

  - `-p` ***user1 usuarios*** copia la configuración de cuotas de ***user1*** a los usuarios indicados

- Al ejecutar `edquota` se abre el editor indicado en la variable `EDITOR` (`nano`, `vi` u otros) para modificar las cuotas:
  - se muestran los bloques de 1k en uso, así como los límites *soft* y *hard* (también para i-nodos o ficheros)
    - si un límite está a 0 no se aplica
  - esta información se guarda en los dos ficheros `aquota.user` y `aquota.group` en el directorio base del filesystem

Otros comandos

Existen otros comandos para la gestión de las cuotas:

- `repquota` genera un informe del uso de las cuotas

```
# repquota /home
*** Report for user quotas on device /dev/hda9
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   34920       0       0              6     0     0
tarabelo  --     728       0       0             31     0     0
tomas     *-     108     100     200  7days       8     0     0
```

- `quota` permite al usuario ver el estado de sus cuotas
  - Algunas opciones:
    - `-g` muestra información sobre las cuotas del grupo del usuario
    - `-v` imprime información incluso para los filesystem sin límite en la cuota
    - `-q` imprime un mensaje si se ha superado la cuota
  - Ejemplo

```
$ quota
Disk quotas for user tomas (uid 1001):
Filesystem blocks  quota limit grace files quota limit  grace
 /dev/hda9    108*   100   200 6days     9     0     0
 /dev/hda8      1     10    20           1     0     0
```

- Para más información sobre cuotas ver Quota mini-HOWTO

3.5. Gestión de redes de área local

Linux soporta múltiples protocolos y hardware de red:

- Diversos protocolos como TCP/IP, IPX/SPX, PPP, etc.
- Soporta hardware para redes Ethernet, Wi-Fi, MPLS, etc
- Diferentes NICs (*Network Interface Cards*) implican diferentes dispositivos de comunicación:
  - En el esquema clásico los interfaces se denominan `eth`***x*** para Ethernet, `wlan`***x*** para Wifi, `ppp`***x*** para PPP, etc, donde $x = 0, 1, 2, \ldots$, numera los interfaces.
  - En las nuevas versiones se utiliza un esquema de nomenclatura de interfaces más concreta que depende del firmware, localización física, dirección MAC, etc. Por ejemplo, `enp3s0 = ethernet network pci 3 slot 0`.
  - Además, existe el dispositivo de *loopback* `lo`
    - Funciona como un circuito cerrado en el que cualquier datagrama que se le pase como parámetro es inmediatamente devuelto a la capa de red del sistema
    - Se utiliza para realizar pruebas, y para un par de aplicaciones de red
- En Linux se crean dinámicamente por software y no requieren los ficheros de dispositivos
  - En cambio, en muchos UNIX estos dispositivos aparecen en `/dev`
- En Linux puede ser necesario incluir los módulos del kernel adecuados para cada dispositivo

En esta sección trataremos la configuración de TCP/IP en redes Ethernet; para más información ver:

- Administración de red en Linux: Linux Network Administrators Guide 2 ed., Olaf Kirch y Terry Dawson
- Linux Networking-HOWTO
- Dispositivos de red soportados en Linux: Linux Hardware Compatibility HOWTO - Network adapters

Ficheros y comandos de configuración de red

`networking`	systemctl (servicio) de arranque de red
`/etc/network/interfaces`	configuración de interfaces
`/etc/resolv.conf`	configuración del DNS
`ip_forward` (variable)	controla el rutado entre interfaces
`/etc/hostname`	fichero con el nombre del host
`hostname, dnsdomainname`	comandos del nombre/dominio del host
`/etc/hosts`, `/etc/networks`	bases locales de hosts/redes
`/etc/nsswitch.conf`	control central de las bases de datos
`/etc/dhcp/dhcpd.conf`	configuración del servidor de DHCP
`dhclient`	comando de obtención de IP por DHCP
`ifconfig`, `ifup/ipdown`	configuración de interfaces
`route`	configuración de rutas
`netstat`	información de red
`ip`	unifica `ifconfig/ifup/route/netstat`
`iwconfig`	configuración de interfaces wireless
`ping`, `traceroute`	testeo de comunicaciones
`host`, `dig`	consultas al DNS
`arp`	gestiona la tabla ARP
`mii-tool`, `ethtool`	configuración de la tarjeta Ethernet

3.5.1. Ficheros de configuración de red

Durante el proceso de arranque, la red se establece del siguiente modo:

- La red es iniciada por `systemctl` mediante la invocación del servicio de `networking` (en otras distribuciones, el servicio es `NetworkManager`).

  Este script también puede ser invocado por el administrador:

  ```
  systemctl restart networking
  ```

  Con las opciones habituales: *start, stop, restart, status, ...*

- A continuación se lee el fichero de configuración de la red,

  `/etc/network/interfaces`

- Y la configuración del servicio de nombres se lee del fichero

  `/etc/resolv.conf`

Configuración de los interfaces

En Debian en el arranque se usa el fichero `/etc/network/interfaces` (en otras distribuciones Linux se utilizan otros ficheros).

- Con configuración estática (manual) debemos indicar los parámetros:

```
auto eth0
iface eth0 inet static
   address 193.144.84.77
   netmask 255.255.255.0
   network 193.144.84.0
   broadcast 193.144.84.255
   gateway 193.144.84.1
```

- Con configuración dinamica (DHCP) se obtendrán de un servidor:

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

Configuración del servicio de nombres

- El fichero `/etc/resolv.conf` especifica el dominio y los servidores DNS

```
search usc.es etse.usc.es
nameserver 193.144.75.9
nameserver 193.144.75.12
```

- si buscamos por un hostname (sin dominio) le añade `usc.es` y si no aparece busca por `etse.usc.es`
- pueden añadirse hasta tres servidores de DNS

Otras formas de establecer los servidores de nombres

- En los hosts que usan DHCP para establecer la red, este fichero se genera automáticamente con los datos obtenidos por el protocolo DHCP
- Otra posibilidad es incluir los servidores de nombres en el fichero `/etc/network/interfaces` (véase en el manual el correspondiente formato de las líneas que hay añadir).
- Si tenemos instaladas aplicaciones de red dinámicas o en el espacio de usuario, estas pueden sobreescribir el fichero `/etc/resolv.conf`.

  Por ejemplo, en las distribuciones que usan una configuración dinámica mediante la aplicación `resolvconf` este fichero se genera automáticamente, pero podemos escribir su contenido en el fichero alternativo `/etc/resolvconf/resolv.conf.d/base`

  También podemos evitar la sobreescritura del fichero con `chattr +i /etc/resolv.conf`

Otros ficheros de configuración

Fichero `/etc/hosts` fichero que asocia nombres de hosts a direcciones IP

- Podemos usar nombres si necesidad de usar un DNS externo
- Solo se usa internamente, pero es más rapido que acceder al DNS
- Ejemplo de `/etc/hosts`:

```
127.0.0.1     localhost.localdomain localhost
193.144.84.77 servidor.usc.es servidor
```

Nombre de la máquina y del dominio

- El anterior fichero permite poner el nombre, el dominio de la máquina y los alias
- El nombre también debe ponerse en el fichero `/etc/hostname`
- El nombre y el dominio pueden obtenerse mediante los comandos `hostname` y `dnsdomainname`

Fichero `/etc/networks` fichero de texto que asocia nombres a redes

- No es imprescindible
- Ejemplo de `/etc/networks`

```
red1 172.16.1.0
red2 172.16.2.0
```

Fichero `/etc/nsswitch.conf` configuración del *Name Service Switch*

- centraliza la información de diferentes servicios para la resolución de nombres, indicando las acciones a realizar para acceder a las diferentes bases de datos del sistema: hosts, contraseñas, servicios, etc.
- Ejemplo de `nsswitch.conf`

```
hosts:          dns files
networks:       files
```

indica que un host se busque primero en el DNS y en caso de fallo en el fichero `/etc/hosts`, mientras que una red se busca sólo en `/etc/networks`

3.5.2. Configuración de red en Ubuntu

La configuración de red mediante ficheros en Ubuntu ha cambiado mucho. Ahora usa un entorno conocido como `netplan`. El fichero a configurar es `/etc/netplan/01-network-init.yaml`. Tenemos las siguientes posibilidades de configuración:

- `renderer: networkd`. Es el valor predeterminado si no se especifica ningún renderer. Utiliza `systemd` para configurar las interfaces de red. Recomendado para servidores y sistemas sin interfaz gráfica.
- `renderer: NetworkManager`. Utiliza `NetworkManager` para gestionar las conexiones de red. Más apropiado para sistemas de escritorio y portátiles. Ofrece una interfaz gráfica para la gestión de redes.

Los distintos valores de los parámetros de red se ponen de la sguiente forma:

Opción	Ejemplo	Descripción
ethernets	enp3s0	interface de red
addresses	192.168.1.2/24	IP
routes	to: default via 192.168.1.1	pasarela
nameservers	[8.8.8.8, 8.8.4.4]	servidores de nombres
dhcp4	true	usar DHCP para IPv4
dhcp6	false	no usar DHCP para IPv6

Las listas de IPs pueden ponerse entre corchetes o en distintas líneas comenzando con un guion.

Veamos tres casos para este fichero:

1) Para una configuración estática asignamos los datos:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      addresses:
        - 172.16.253.82/24
      routes:
        - to: default
          via: 172.16.253.1
      nameservers:
```

```
        search: [usc.es,  etse.usc.es]
        addresses: [193.144.75.9, 8.8.4.4]
```

Validamos la configuración (para detectar errores) y la aplicamos:

```
$ netplan try
$ netplan apply
```

2) En el caso de que queramos configuración por DHCP

```
network:
  version: 2
  renderer: networkd
  ethernet:
    enp3s0:
      addresses: []
      dhcp4: true
      optional: true
```

Podemos obtener manualmente los datos DHCP con el comando `netplan ip leases enp3s0`.

3) Por último, podemos delegar la responsabilidad de configuración de la red en *NetworkManager*. En este caso el contenido del fichero será:

```
network:
  version: 2
  renderer: NetworkManager
```

3.5.3. Configuración de DHCP

DHCP (*Dynamic Host Configuration Protocol*) permite configurar automáticamente la red de los sistemas a partir de un servidor DHCP

- La información de IPs, DNS, etc. se mantiene centralizada en el servidor
- Si utilizamos nombres para alguna máquina (como *pasarela.dominio.com*) el servidor deberá poder acceder a la traducción (por DNS o `/etc/hosts`)
- Al iniciarse, los clientes se conectan al servidor (por broadcast) y cargan su configuración
- En `/var/lib/dhcp/dhcpd.leases` al final estarán las IPs asignadas

Configuración del servidor

- En el fichero `/etc/default/isc-dhcp-server` debemos especificar el interfaz por el que servimos DHCP
- El fichero de configuración es `/etc/dhcp/dhcpd.conf`. Por ejemplo:

```
# Nombre de Dominio que vamos a dar a todos los clientes
option domain-name "midominio.com";

# Servidores de Nombres que vamos a indicar a los clientes
option domain-name-servers 10.0.2.3, 193.144.75.9;

# Tiempo por defecto que dura una asignación
default-lease-time 600;

# Duración máxima de una asignación
max-lease-time 7200;

# Rango de IPs a repartir, dirección de broadcast y gateway
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.10 192.168.0.200;
  option broadcast-address 192.168.0.255;
  option routers 192.168.0.1; # esta es la gateway
}

# Configuramos un host concreto
host marte {
  hardware ethernet 52:54:00:12:34:70;
  fixed-address 192.168.0.201; }
```

Configuración del cliente

- Para que el cliente se configure por DHCP debemos indicarlo fichero de configuración de red (en el caso de Debian, `/etc/network/interfaces`):

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

- El administrador puede forzar la obtención de los datos con el comando:

  ```
  dhclient eth0
  ```

3.5.4. Comandos de configuración de red

Los comandos clásicos más importantes para configurar la red son:

- `ifconfig`: configuración del interfaz de red
- `route`: configuración de rutado
- `netstat`: información de la red

Si no se encuentran estos comandos, instalar el paquete `net-tools`. Debe tenerse en cuenta que la configuración no se mantiene al reiniciar el sistema.

Comando `ifconfig`

Muestra y configura una interfaz de red:

```
$ /sbin/ifconfig eth0
eth0   Link encap:Ethernet  HWaddr 00:12:43:A6:05:5C
       inet addr:193.144.84.77  Bcast:193.144.84.255  Mask:255.255.255.0
       inet6 addr: fe80::211:43ff:fea6:55c/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1035446 errors:0 dropped:0 overruns:0 frame:0
       TX packets:1053062 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:196973192 (187.8 MiB)  TX bytes:270128587 (257.6 MiB)
```

- En las tres primeras líneas se muestran las direcciones hardware (Ethernet), IP e IPv6, así como las máscaras y la dirección de broadcast.
- La cuarta línea indica que el interface está activado (UP) y funcionando (RUNNING) y que soporta *broadcast* y *multicast*.
- MTU es el máximo tamaño de las tramas medido en bytes (1500 bytes para las tramas Ethernet) y la métrica indica que hay que atravesar un interface.
- Las dos siguientes líneas indican el número de paquetes recibidos y transmitidos, así como el número de errores que se han producido.
- *txqueuelen* es la longitud de la cola de transmisión medida en bytes
- En la última línea se muestra el número de bytes recibidos y transmitidos
- Sintaxis:

```
ifconfig [opciones] [interfaz] [configuración] [up|down]
```

- Opciones de visualización:
  - `-a` muestra todas las interfaces, incluso las inactivas, que por defecto no se muestran
- En las opciones de configuración se indica entre otras cosas la IP, máscara de red y dirección de broadcast:

```
# ifconfig eth0 193.144.84.77 netmask 255.255.255.0 \
      broadcast 193.144.84.255 up
```

- `ifconfig` permite también configurar el estado del interfaz, por ejemplo, cambiar el MTU, poner modo promiscuo, activar/desactivar ARP, cambiar su dirección hardware (si el dispositivo lo permite), etc.

```
# ifconfig eth0 mtu 2000
# ifconfig eth0 promisc
# ifconfig eth0 -arp
# ifconfig eth0 hw ether 52:54:00:12:34:56
```

- ver el manual de `ifconfig` para más información

Otros comandos relacionados

Otros comandos de configuración de interfaz son:

- `ifup/ifdown` activan/desactivan un interfaz de red

```
# ifdown eth0
```

- `iwconfig` configura un interfaz wireless

```
# iwconfig eth1 essid "Mi Red"
```

Comando `route`

Permite modificar la tabla de routing, mostrando, añadiendo o borrando rutas

- muestra las rutas definidas
- permite añadir/borrar rutas estáticas

- permite definir un gateway de salida por defecto para conectarnos al exterior
- permite configurar el sistema para que actúe como un `router`

Mostrar una tabla de routing

Se usa `route [-n]` (equivale a `netstat -r`)

```
$ /sbin/route
Destination  Gateway      Genmask        Flags Metric Ref Use Iface
192.168.0.0  *            255.255.255.0  U     0      0     0 eth1
10.0.2.0     *            255.255.255.0  U     0      0     0 eth0
172.16.0.0   192.168.0.1  255.255.255.0  UG    1      0     0 eth1
default      10.0.2.2     0.0.0.0        UG    0      0     0 eth0
```

- Opciones:
  - `-n` usa direcciones IP en vez de nombres

```
$ /sbin/route -n
Destination  Gateway      Genmask        Flags Metric Ref Use Iface
192.168.0.0  0.0.0.0      255.255.255.0  U     0      0     0 eth1
10.0.2.0     0.0.0.0      255.255.255.0  U     0      0     0 eth0
172.16.0.0   192.168.0.1  255.255.255.0  UG    1      0     0 eth1
0.0.0.0      10.0.2.2     0.0.0.0        UG    0      0     0 eth0
```

- Los flags indican el estado de la ruta
  - `U` la ruta es accesible (Up)
  - `H` el destino es una estación (Host). Si este flag no está presente podemos asumir que el destino es una red.
  - `G` la ruta usa una pasarela (Gateway). Si este flag no está presente podemos asumir que se trata de un destino directamente conectado.
  - `D` la ruta fue creada dinámicamente por un demonio de encaminamiento (RIP, OSPF, ...) o un mensaje ICMP de redirección
  - `M` la ruta fue modificada dinámicamente
  - `!` ruta rechazada
- De las siguientes columnas, algunas no se usan
  - `Metric` distancia (normalmente en saltos) al destino
  - `Ref` número de referencias a la ruta (no usado en Linux)
  - `Use` número de consultas para la ruta

Añadir/borrar rutas estáticas

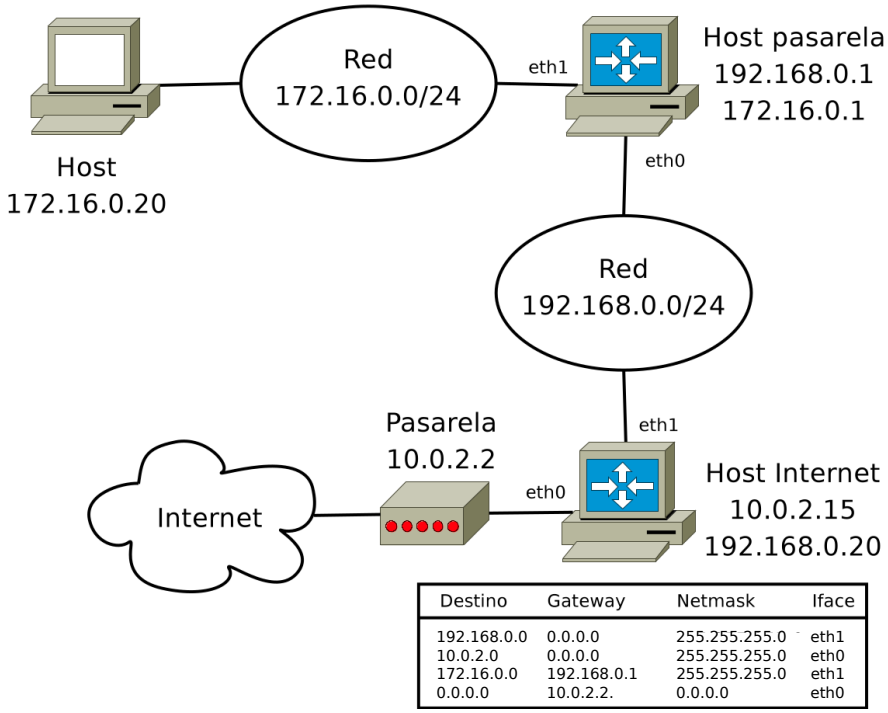
Se usa

> `route [add|del] [default] [-net|-host]` ***`destino`*** `[netmask` ***`máscara`***`] [gw` ***`pasarela`***`] [`***`opciones`***`] [dev` ***`interface`***`]`

Ejemplo: suponer que tenemos la configuración del dibujo y queremos configurar la tabla de rutas para el host Internet (abajo a la derecha)

Destino	Gateway	Netmask	Iface
172.16.0.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	172.16.0.1	0.0.0.0	eth0

Destino	Gateway	Netmask	Iface
192.168.0.0	0.0.0.0	255.255.255.0	eth0
172.16.0.0	0.0.0.0	255.255.255.0	eth1
0.0.0.0	192.168.0.20	0.0.0.0	eth0



Destino	Gateway	Netmask	Iface
192.168.0.0	0.0.0.0	255.255.255.0	eth1
10.0.2.0	0.0.0.0	255.255.255.0	eth0
172.16.0.0	192.168.0.1	255.255.255.0	eth1
0.0.0.0	10.0.2.2.	0.0.0.0	eth0

- Las ruta a las redes directamente conectadas (192.168.0.0/24 y 10.0.2.0/24) se ponen automáticamente al asignar las IPs a los interfaces.
- Añadimos una ruta para la red 172.16.0.0/24, usando como pasarela el host con IP 192.168.0.1

  ```
  route add -net 172.16.0.0 netmask 255.255.255.0 gw 192.168.0.1
  ```

- Añadimos la ruta por defecto

```
route add default gw 10.0.2.2
```

Nota: El interface de salida usado en la ruta no es necesario indicarlo si no existe ambigüedad, aunque también se puede hacer.

IMPORTANTE: Las pasarelas ayudan a nuestra máquina a encaminar los paquetes hacia otras redes, por lo que deben cumplir dos propiedades:

1. No pueden ser una las IPs de la propia máquina (las pasarelas no son la propia máquina).
2. Deben estar en una de las redes directamente conectadas a la máquina. Hay que recordar que las pasarelas se utilizan para salir de la propia red y si se ponen pasarelas de redes exteriores, no se van a poder alcanzar.

- El host pasarela tiene que permitir routing entre sus interfaces; pasa eso debemos activar el *ip_forward*:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Información de la red: comando `netstat`

`netstat` muestra todo tipo de conexiones de red y estadísticas

- Formato:

```
netstat [tipo de información] [opciones]
```

- Algunos tipos de información:
  - Sin parámetros muestra la lista de sockets abiertos

```
$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp     0      0    jumilla:58946 aiff:telnet     ESTABLISHED
tcp     0      0    jumilla:43658 ulla:1301       ESTABLISHED
tcp     0      0    jumilla:35346 cesga:ssh       ESTABLISHED
tcp     0      0    jumilla:ssh   ulla:1688       LAST_ACK
tcp     0      0    jumilla:ssh   teneguia:35161  CLOSE_WAIT
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags Type   State     I-Node Path
unix  8      [ ]   DGRAM            15368  /dev/log
unix  2      [ ]   DGRAM            194110 @/org/kernel/udev/udevd
unix  2      [ ]   STREAM CONNECTED 15671  @/tmp/.X11-unix/X0
```

Los valores de estado indican lo siguiente:

- `ESTABLISHED` hay una conexión establecida
- `LISTEN` socket a la espera de solicitudes de conexión
- `CLOSE` socket cerrado
- `SYN_SENT`, `SYN_RECV` inicio de una conexión
- `FIN_WAIT1`, `FIN_WAIT2`, etc, en proceso de desconexión
- `UNKNOWN`, estado del socket desconocido

Por defecto se muestra todo tipo de sockets. Otras opciones:

- `-t|--tcp`, `-u|--udp`, `-r|--raw` muestra solo sockets de ese tipo.
- `-i` información de interfaces (igual que `iconfig`)
- `-r` información de rutas (igual que `route`)
- `-n` información numérica para hosts y puertos en vez de nombres

- Estadísticas
  - `-s` muestra estadísticas para todo tipo de protocolo

```
Ip:
    37529 total packets received
    2 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    37527 incoming packets delivered
    40521 requests sent out
    44 outgoing packets dropped
Icmp:
    93 ICMP messages received
    0 input ICMP message failed.
    ...
Tcp:
    1089 active connections openings
    0 passive connection openings
    2 failed connection attempts
    150 connection resets received
    7 connections established
    28102 segments received
```

```
    29440 segments send out
    ...
Udp:
    8886 packets received
    427 packets to unknown port received.
    0 packet receive errors
    6725 packets sent
...
```

Comando `ip`

Muestra y modifica dispositivos y rutas, alternativa a `ifconfig`, `route` y `netstat`, más potente y complejo

- Operaciones sobre el interface: para activarlo (es necesario siempre) y mostrar la configuración:

```
# ip link set up dev eth0
# ip link show
```

- Operaciones sobre las direcciones del interface: añadir datos IP, borrarlos (si se añaden otros nuevos, los antiguos permanecen) y mostrar datos:

```
# ip address add 192.168.10.2/24 broadcast 255.255.255.0 dev eth0
# ip address del 192.168.10.2/24 broadcast 255.255.255.0 dev eth0
# ip address show
```

- Operaciones sobre ruta (hacia una red, la pasarela por defecto y mostrar datos de rutas):

```
# ip route add 192.168.10.0/24 via 192.168.10.1 dev eth0
# ip route add default via 192.168.20.1 dev eth1
# ip route show
```

- Mostrar estadísticas:

```
# ip -s link
# ip -s address
# ip -s route
```

IP forwarding

Linux permite convertir un PC de sobremesa en un router. Para ello hay que configurar el routado entre interfaces, es decir, que un paquete que entra por un interface pueda salir por otro (por defecto, esto está desactivado). Esto se denomina IP forwarding (retransmisión de paquetes IP). La conversión puede hacerse de varias formas:

- Temporalmente, con la variable del directorio `/proc/sys/net/ipv4`

  ```
  # echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

- Temporalmente, con el comando `sysctl`
- De forma permanente, en el fichero `/etc/sysctl.d/99-custom.conf`

  ```
  net.ipv4.ip_forward=1
  ```

Opciones de IP

Además del IP forwarding, Linux permite configurar diversas opciones sobre el tráfico IP en el fichero `/etc/sysctl.conf`. Por ejemplo,

- `ip_default_ttl` el tiempo de vida por defecto de los paquetes (por defecto 64 ms)
- algunas de estas opciones tienen un 0 (opción desactivada) o un 1 (opción activada). Otras pueden tener un valor.

Otros comandos de red

Comando `ping`

- Muestra la disponibilidad de conexión y la velocidad de transmisión con un host remoto:

```
$ ping 193.144.84.1
PING 193.144.84.1 (193.144.84.1) 56(84) bytes of data.
64 bytes from 193.144.84.1: icmp_seq=1 ttl=255 time=0.420 ms
64 bytes from 193.144.84.1: icmp_seq=2 ttl=255 time=0.396 ms
64 bytes from 193.144.84.1: icmp_seq=3 ttl=255 time=0.368 ms
```

- `ping` envía paquetes ICMP (ECHO_REQUEST) al destino y espera respuesta, midiendo el RTT (tiempo de ida y vuelta)

- muchos firewalls bloquean el tráfico ICMP por lo que el `ping` puede que no funcione

Algunas opciones:

- `-b` permite `ping` a una dirección de broadcast
- `-c` *`COUNT`* envía solo *`COUNT`* paquetes
- `-s` *`packetsize`* especifica el tamaño del paquete (por defecto 56 bytes)

Comando `traceroute`

- Muestra la ruta que sigue un paquete hasta llegar al destino

```
$ traceroute www.elpais.es
traceroute to a17.akamai.net (130.206.192.32), 30 hops max, 40 byte p
 1  rutfis (193.144.64.1) 1.070 ms 0.688 ms 0.927 ms
 2  * * *
 3  10.56.5.1 (10.56.5.1) 57.463 ms 2.021 ms 1.923 ms
 4  193.144.79.72 (193.144.79.72) 2.507 ms 16.280 ms 2.080 ms
 5  GE2-0-0.EB-Santiago0 (130.206.204.21) 25.681 ms 2.068 ms 1.965 ms
 6  GAL.SO2-0-0.EB-IRIS4 (130.206.240.33) 10.959 ms 10.665 ms 10.710 ms
 7  130.206.220.59 (130.206.220.59) 20.277 ms 10.781 ms 10.470 ms
 8  a130-206.akamai.com (130.206.192.32) 11.011 ms 23.482 ms 12.185 ms
```

- `traceroute` envía paquetes UDP y utiliza el campo TTL de la cabecera IP para limitar el número de routers por los que puede pasar el paquete
- cada vez que el paquete pasa por un router el tiempo de vida disminuye
- cuando un router obtiene un tiempo de vida igual a cero devuelve un mensaje ICMP TIME_EXCEEDED al host que envió el paquete
- a continuación al programa incrementa en una unidad el tiempo excedido para obtener el siguiente router
- los sistemas que no envian mensajes de tiempo excedido aparecen `*`
- si los firewalls bloquean el tráfico ICMP no veremos nada
- otros programas similares:
  - `traceproto`: permite especificar el protocolo a usar (TCP, UDP, ICMP) y el puerto a tracear (por defecto 80)
  - `tcptraceroute`: envía paquetes TCP SYN para evitar problemas con firewalls

Comandos `host` y `dig`

- Permiten obtener la dirección IP de un sistema a partir del nombre o viceversa:

```
$ host www.elpais.es
www.elpais.es is an alias for elpais.es.edgesuite.net.
elpais.es.edgesuite.net is an alias for a1749.g.akamai.net.
a1749.g.akamai.net has address 130.206.192.38
a1749.g.akamai.net has address 130.206.192.32
```

- `nslookup` está desaprobado (*deprecated*) y no se recomienda su uso

Comando `arp`

- `arp` manipula la cache de ARP:
  - muestra la tabla ARP
  - borra / añade entradas manualmente

```
# arp
Address                  HWtype  HWaddress           Flags Mask  Iface
almansa.dec.usc.es       ether   00:0D:56:6F:E6:90   C           eth0
193.144.84.1             ether   00:E0:63:93:26:E5   C           eth0
teneguia.dec.usc.es      ether   00:C0:4F:A1:5D:89   C           eth0
```

Comando `mii-tool`

- Permite ver y/o configurar el estado de la unidad MMI (*Media Independent Interface*) de la tarjeta de red. Ethernet usa MII para autonegociar la velocidad de enlace y el modo duplex

```
# mii-tool eth0
eth0: 100 Mbit, half duplex, link ok
# mii-tool -v eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 16 rev 0
  basic mode:   autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
# mii-tool --force=100baseTx-HD eth0
```

- Comando parecido (más complejo): `ethtool`

Capítulo 4

Servicios básicos de servidor a cliente

En este tema veremos los siguientes servicios de intranet:

1. Acceso remoto y transferencia de ficheros (SSH)
2. Sistemas de ficheros de red (NFS)
3. Compartición Windows/Linux (Samba)
4. Servicio de directorio (LDAP)

4.1. Acceso remoto y transferencia de ficheros (SSH)

SSH: Shell seguro (todos los datos viajan encriptados)

- SSH incluye tres programas de conexión: `ssh`, `sftp` y `scp`
- `ssh` permite conectarnos a otro sistema y abrir sesiones encriptando toda la información
- Reemplazo de `rlogin`, `telnet` o `ftp`
- `scp`, `sftp` permiten la transferencia de ficheros de forma encriptada
- `scp` similar a `cp` y `sftp` similar a `ftp`
- La implementación open-source es OpenSSH

Paquetes Debian:

- Ciente: `openssh-client`
- Servidor: `openssh-server`

Modos de autenticación mediante SSH

SSH soporta 4 modos de autenticación:

1. Host remoto mediante lista. Si el nombre del host remoto desde el cual un usuario se conecta al servidor está listado en los ficheros del servidor `~/.rhosts`, `~/.shosts`, `/etc/hosts.equiv` o `/etc/shosts.equiv` el usuario remoto puede entrar sin contraseña
   - Método absolutamente desaconsejado
   - Deshabilitado por defecto
2. Host remoto mediante clave. Igual que el anterior pero la clave pública del host remoto debe aparecer en el servidor en `/etc/ssh_known_hosts` o `~/.ssh/known_hosts`
   - No demasiado seguro (si el host remoto se ve comprometido, el servidor local queda comprometido)
   - Deshabilitado por defecto
3. Usuario mediante clave pública. La clave del usuario remoto se coloca en el servidor `~/.ssh/authorized_keys`
   - El usuario remoto debe tener acceso a su clave privada
   - Permitido por defecto
4. Usuario mediante contraseña (modo por defecto)
   - Menos seguro que el anterior
   - Permitido por defecto

Las opciones para permitir o prohibir los modos de acceso se encuentran en el fichero de configuración del servidor: `/etc/ssh/sshd_config`

Protocolos SSH

Existen dos versiones del protocolo SSH, denominadas 1 y 2. La versión 1 hace uso de muchos algoritmos de cifrado patentados (algunas de estas patentes han expirado) y es vulnerable a un agujero de seguridad que potencialmente permite a un intruso insertar datos en la corriente de comunicación. Cuando se inicia una conexión se prueba primero el protocolo 2, pero si falla, se intenta con el protocolo 1 si este no se encuentra deshabilitado en el fichero de configuración del servidor.

Opciones de configuración del servidor

Otras opciones en `/etc/ssh/sshd_config`

Opción	Defecto	Significado
Port	22	Puerto (puede ser interesante cambiarlo)
Protocol	2,1	Protocolo aceptado (más seguro sólo 2)
ListenAddress	Todas	Dirección local aceptada para conexión
PermitRootLogin	`no`	Permite acceder a root
X11Forwarding	`no`	Permite ejecutar aplicaciones gráficas en el servidor

Para más opciones `man sshd_config`

Opciones para el cliente

Ficheros de configuración del cliente: `/etc/ssh/ssh_config` o `~/.ssh/config`

- Algunas de estas opciones se pueden especificar en el momento de ejecutar el comando, p.e.

```
$ ssh -p port servidor # Indica otro puerto
$ ssh -X servidor # Permite aplicaciones gráficas
```

- En el fichero de configuración se especifican opciones para los comandos `ssh`, `scp` o `sftp`

Opción	Defecto	Significado
Hosts		Host para los que se aplican las opciones (`*` implica todos)
Port	22	Puerto por defecto
Protocol	2,1	Protocolo usado por defecto
Cipher[s]		Mecanismos de cifrado usados
ForwardX11	`no`	Permite ejecutar aplicaciones gráficas del servidor en el cliente

Para más opciones `man ssh_config` y `man ssh`

Hosts conocidos

La primera vez que nos conectamos a un servidor, SSH nos pregunta si queremos añadirlo a la lista de hosts conocidos (`~/.ssh/known_hosts`):

```
The authenticity of host 'server.usc.es (216.9.132.134)' can't be established.
RSA key fingerprint is 53:b4:ad:c8:51:17:99:4b:c9:08:ac:c1:b6:05:71:9b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server.usc.es' (RSA) to the list of known hosts.
```

Si en una conexión posterior a la misma máquina la clave no coincide, no se permitirá la conexión y SSH nos devolverá el mensaje:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

Si estamos seguros de que no se trata de un ataque, podemos arreglarlo eliminado la clave del host antiguo con `ssh-keygen -R host` o simplemente borrando el fichero `~/.ssh/known_hosts` (aunque así perdemos todas las claves de hosts almacenadas).

Generación de claves públicas/privadas

Se utiliza el comando:

```
ssh-keygen [-t tipo]
```

Donde *tipo* puede ser `rsa` para la versión 1 del protocolo y `rsa` o `dsa` para la versión 2, entre otras. La salida del comando es la siguiente:

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
93:58:20:56:72:d7:bd:14:86:9f:42:aa:82:3d:f8:e5 user@usc.es
```

- El comando nos pregunta por una frase de longitud arbitraria para almacenar las claves generadas, que debe ser diferente a la contraseña del usuario.
- El comando nos genera dos ficheros, una clave pública (con sufijo `.pub`) y una clave privada.
- La razón por la que se utiliza un fichera de claves es para aumentar la seguridad de la sesión SSH al no utilizar la contraseña del sistema.

Instalación de la clave pública en el servidor

La instalación de la clave pública en un host remoto puede realizarse mediante un comando o de forma manual. Mediante un comando:

- `ssh-copy-id user@servidor.usc.es`
- Ahora podemos conectarnos al servidor escribiendo la frase que hemos puesto.

De forma manual:

1. Incluimos el fichero `~/.ssh/id_dsa.pub` (o `id_rsa.pub`) en el fichero del servidor `~/.ssh/authorized_keys`
2. Cambiamos los permisos del fichero que hemos copiado:

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Agente de autentificación

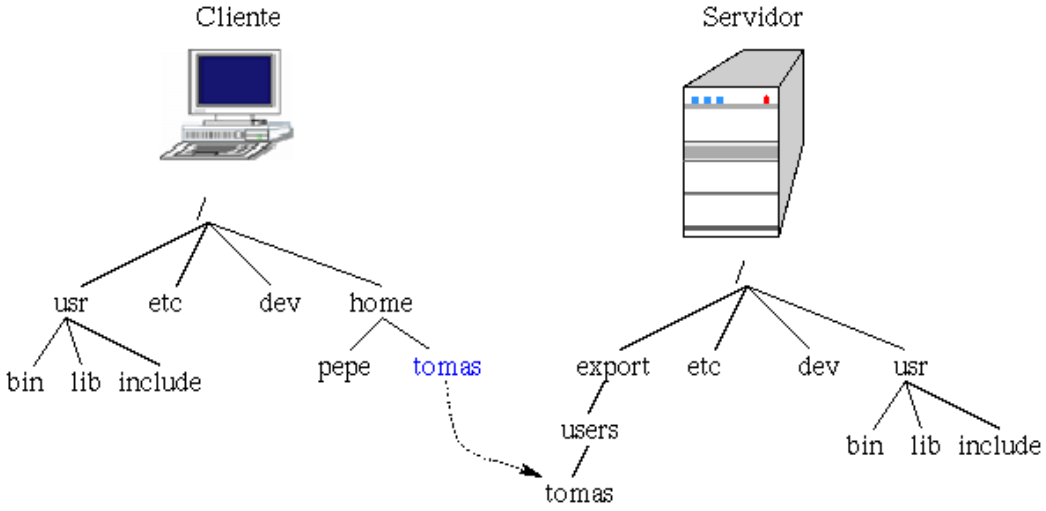
- `ssh-agent`
  - Mantiene en memoria la clave privada
  - Evita tener que escribir la *passphrase* cada vez que usemos ssh
  - Habitualmente, si entramos en X11 se activa automáticamente
  - Tecleamos (como usuario): `eval $(ssh-agent)`
- `ssh-add`
  - Añade las claves privadas al agente
  - Pueden añadirse múltiples claves. En una conexión se prueban las diferentes claves hasta que coincide
  - Tecleamos `ssh-add`
  - El agente nos pide la frase y una vez introducida nos responde con

    `Identity added: /home/user/.ssh/id_dsa`
  - A partir de ahora podemos entrar en el servidor sin teclear ni la contraseña ni la frase.
  - Por defecto añade los ficheros `~/.ssh/id_rsa`, `~/.ssh/id_dsa` y `~/.ssh/identity`, pidiendo las correspondientes *passphrases*

4.2. Sistemas de ficheros de red (NFS)

NFS (*Network File System*) permite compartir sistemas de ficheros en la red de forma totalmente transparente. El usuario no nota diferencias entre los sistemas de ficheros remotos y locales.

Ejemplo de funcionamiento:



4.2.1. Servidor NFS

Veremos como instalar un servidor y un cliente NFSv4 en Debian. Los pasos son los siguientes:

- Instalar los paquetes y reiniciar el servicio
- Configurar los directorios a exportar: `/srv/nfs4`
- Escribir el fichero de configuración: `/etc/exports`
- Comando para realizar la exportación: `exportfs`
- Mostrar los directorios exportados: `showmount`

Instalación del servicio

- Los paquetes a instalar son: `nfs-kernel-server` y `nfs-common` (este último suele estar instalado por defecto)
- El servicio se inicia con:

```
# systemctl start nfs-kernel-server
```

Configuración de directorios

- Los directorios a exportar de NFSv4 por el servidor deben residir en un directorio, donde se montan con la opción `--bind`. La opción `bind` permite montar un directorio en otro punto (se podrá acceder a el desde las dos localizaciones). Por ejemplo para incluir `/home` entre los directorios a exportar por el servidor:

```
# mkdir /srv/nfs4
# mkdir /srv/nfs4/home
# mount --bind /home /srv/nfs4/home/
```

- Para que este montado permanezca entre arranques, añadir a `/etc/fstab` la siguiente línea:

```
#(filesystem) (mount point) (tipo) (opc.) (dump) (pass)
/home       /srv/nfs4/home/  none   bind     0      0
```

Fichero `/etc/exports` en NFSv4

La primera línea nos indica que los directorios a exportar cuelgan de `/srv/nfs4` (opción `fsid=0`) y que cuando se monten no sea necesario indicar esta parte de la ruta (opción `crossmnt`).

```
#Directorios a exportar | red que puede acceder(opciones)
/srv/nfs4 192.168.2.0/24(rw,sync,fsid=0,crossmnt,no_subtree_check,no_root_squash)
/srv/nfs4/home  192.168.2.0/24(rw,sync,no_subtree_check,no_root_squash)
```

Opciones de la exportación:

- `rw/ro` exporta el directorio en modo lectura/escritura o sólo lectura
- `sync/async` modo síncrono/asíncrono: requiere/no requiere que todas las escrituras se completen. El asíncrono es más rápido, pero puede provocar pérdida de datos en una caída.
- `fsid=0` designa este path como la raíz de los directorios exportados por NFSv4
- `crossmnt` permite que los directorios debajo del raíz se muestren adecuadamente (alternativamente, se puede poner la opción `nohide` en cada uno de esos directorios)

- `subtree_check/no_subtree_check` con `subtree_check`, si se exporta un subdirectorio (no un filesystem completo) el servidor comprueba que el fichero solicitado por el cliente esté en el subdirectorio exportado; con `no_subtree_check` (opción por defecto) se deshabilita ese chequeo
- `no_root_squash` permite al administrador del sistema remoto escribir y hacer modificaciones en el filesystem
- Para más opciones `man exports`

Actualización

- Cada vez que se modifica este fichero se debe ejecutar el comando `exportfs` para actualizar el servidor

```
# exportfs -ra
```

  - ver `man exportfs` para opciones del comando

- A continuación reiniciamos el servicio

```
# systemctl restart nfs-kernel-server
```

Comprobar que funciona y mostrar los directorios exportados

- `showmount` muestra información de un servidor NFS: directorios que exporta, directorios montados por algún cliente y clientes que montan los directorios

```
# showmount --exports localhost
```

- Podemos ver las estadísticas del servidor NFS con

```
nfsstat
```

4.2.2. Cliente NFS

El cliente NFS en Linux está integrado en el nivel del Sistema de Ficheros Virtual (VFS) del kernel. Para la instalación:

1. Instalar (si no está ya instalado) el paquete `nfs-common`
2. Montar los directorios remotos con `mount -t nfs4`, o añadir una entrada en `fstab`

- Ejemplo de uso con `mount` (IP servidor NFS 192.168.2.1):

```
# mkdir /mnt/home
# mount -t nfs4 192.168.2.1:/home /mnt/home
```

- Ejemplo de entrada en `fstab`

```
#(server:dir) (mount point) (tipo) (opc.) (dump) (pass)
192.168.2.1:/home  /mnt/home  nfs4  rw,auto   0    0
```

- Automount se usa frecuentemente con NFS (ver la sección del apartado 3.2.4, *Montado de los sistemas de ficheros: Autofs*)

Opciones particulares de montado con NFS:

- `hard` el programa accediendo al sistema de ficheros remoto se colgará cuando el servidor falle; cuando el servidor esté disponible, el programa continuará como si nada (opción más recomendable)
- `soft` cuando una petición no tiene respuesta del servidor en un tiempo fijado por `timeo=`*t* el cliente devuelve un código de error al proceso que realizó la petición (puede dar problemas)
- Para ver más opciones, ver `nfs(5)`

Para ver los directorios NFSv4 montados en el cliente:

```
mount -t nfs4
```

4.2.3. Consideraciones de seguridad en NFS

NFS no fue diseñado pensando en la seguridad:

- Los datos se transmiten en claro
- Usa el UID/GID del usuario en el cliente para gestionar los permisos en el servidor:
  - El usuario con UID *n* en el cliente obtiene permisos de acceso a los recursos del usuario con UID *n* en el servidor (aunque sean usuarios distintos)
  - Un usuario con acceso a root en un cliente podría acceder a los ficheros de cualquier usuario en el servidor (no a los de root, si se usa la opción `root_squash`)
- Puede conseguirse más seguridad en NFS añadiendo tunelización sobre SSH o usando versiones de NFS seguras.

4.3. Compartición Linux-Windows: Samba

Samba permite a un sistema UNIX conversar con sistemas Windows a través de la red de forma nativa

- el sistema Unix aparece en el "Entorno de red" de Windows
- los clientes Windows pueden acceder a sus recursos de red e impresoras compartidas
- el sistema UNIX puede integrarse en un dominio Windows, bien como Controlador Primario del Dominio (PDC) o como miembro del dominio
- Samba ofrece los siguientes servicios
  - Servicios de acceso remoto a ficheros e impresoras
  - Autenticación y autorización
  - Servicio de resolución de nombres

Samba implementa los protocolos NetBIOS y SMB

- NetBIOS: protocolo de nivel de sesión que permite establecer sesiones entre dos ordenadores
- SMB (*Server Message Block*): permite a los sistemas Windows compartir ficheros e impresoras (llamado *Common Internet File System*, CIFS por Microsoft)

4.3.1. Instalación básica de Samba

Veremos una instalación básica de Samba en nuestro sistema Debian:

- permitirá desde un Windows acceder a los directorios de usuarios

Instalación de los paquetes

El paquete básico a instalar es `samba` que incluye los demonios de Samba

- instala también el paquete `samba-common`, que incluye utilidades como `smbpasswd` y `testparm`

Sólo instalaremos `samba` y `samba-common`

- la instalación nos pide un nombre de Grupo de Trabajo/Dominio: indicar un nombre, que debemos usar en el sistema Windows

La configuración de Samba se realiza en el fichero `/etc/samba/smb.conf`

- establece las características del servidor Samba, así como los recursos que serán compartidos en la red. Ejemplo sencillo:

```
[global]
  workgroup = MIGRUPO
[homes]
  comment = Home Directories
[printers]
  path = /usr/spool/public
[pub]
  path = /espacio/pub
```

Estructura del fichero `smb.conf`

El fichero `/etc/samba/smb.conf` se encuentra divido en secciones, encabezados por una palabra entre corchetes

- En cada sección figuran opciones de configuración, de la forma `etiqueta = valor`, que determinan las características del recurso exportado por la sección
- Existen tres secciones predefinidas: `global`, `homes` y `printers`
- Otras secciones (como `pub` en el ejemplo anterior) definen otros recursos para compartir

Secciones predefinidas:

`[global]` define los parámetros de Samba a nivel global del servidor, por ejemplo, el programa utilizado para que un usuario pueda cambiar su clave (`passwd program`)

`[homes]` define automáticamente un recurso de red por cada usuario conocido por Samba; este recurso, por defecto, está asociado al directorio *home* del usuario en el ordenador en el que Samba está instalado

`[printers]` define un recurso compartido por cada nombre de impresora conocida por Samba

Niveles de Seguridad

Samba ofrece dos modos de seguridad referidos a la compartición de recursos:

- Modo *share*: cada vez que un cliente quiere utilizar un recurso de Samba, debe suministrar una contraseña de acceso asociada a dicho recurso
- Modo *user*: el cliente establece una sesión con el servidor Samba (mediante usuario y contraseña); una vez Samba valida al usuario, el cliente obtiene permiso para acceder a los recursos ofrecidos por Samba

El nivel de seguridad se especifica con la opción `security`, la cual pertenece a la sección `[global]`

```
security = share | user | server | domain | ADS
```

Los niveles `user`, `server`, `domain` y `ADS` corresponden todos ellos al modo de seguridad `user`

- Nivel `user`: el encargado de validar al usuario es el sistema Unix donde Samba se ejecuta; es necesario que existan los mismos usuarios y con idénticas contraseñas en los sistemas Windows y en el servidor Samba
- Nivel `server`: Samba delega la validación del usuario en otro ordenador, normalmente un sistema Windows 2000 (método no recomendado)
- Nivel `domain`: el ordenador en el que se delega la validación debe ser un Controlador de Dominio (DC), o una lista de DCs; el sistema Samba actúa como miembro de un dominio
- Nivel `ADS`: en Samba-3 permite unirse a un dominio basado en Active Directory como miembro nativo

El modo por defecto es `user`

Otros comandos Samba

- `testparm` permite chequear el fichero `smb.conf` para ver si es correcto
- `net` herramienta básica para administrar Samba y servidores SMB remotos; funciona de forma similar al comando `net` de DOS
- `smbpasswd` permite cambiar la contraseña usada en las sesiones SMB; si se ejecuta como root también permite añadir y borrar usuarios del fichero de contraseñas de Samba
- `smbstatus` muestra las conexiones Samba activas
- `smbclient` permite a un usuario de un sistema Unix conectarse a recursos SMB y listar, transferir y enviar ficheros

4.4. Servicio de directorio: LDAP

LDAP: Protocolo Ligero de Acceso a Directorios (*Lightweight Directory Access Protocol*)

- Uno de sus usos es almacenar los recursos para la autentificación de usuarios de forma centralizada.
- LDAP es un estándar que está implementado sobre múltiples S.O.
- Existe la implementación libre OpenLDAP

Un directorio es una base de datos optimizada para lectura, navegación y búsqueda

- la información se almacena de manera jerárquica
- generalmente no se soportan transacciones complejas ni sistemas de recuperación
- las actualizaciones son cambios simples
- proporcionan respuestas rápidas a grandes volúmenes de búsquedas
- el directorio puede estar replicado y/o distribuido entre varios sistemas (p.e. DNS)

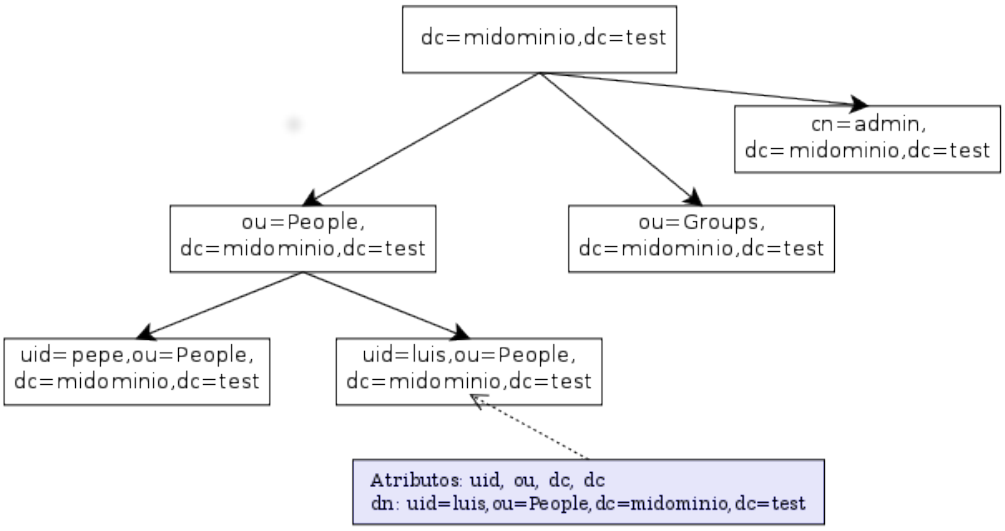
LDAP organiza el directorio como una estructura jerárquica de entradas (nodos) en forma de árbol

- Cada nodo debe poseer un nombre único: *nombre distintivo* o `dn` (*distinguished name*), que identifica de forma unívoca a cada objeto en la base de datos
- Cada nodo se caracteriza por un conjunto de atributos, algunos de los cuales forman parte del `dn`
  - por ejemplo: `dn: uid=pepe,ou=people,dc=midominio,dc=test`
  - los atributos son normalmente palabras nemotécnicas, como:

`uid` (identificador de usuario)	`ou` (organization unit)
`dc` (*domain component*)	`cn` (*common name*)
`c` (*country*)	`o` (*organization*)

  - los atributos pertenecen a clases, las cuales definen diferente tipo de información: clases para personas, para equipos, administrativas, etc.
  - las clases se definen mediante ficheros de *esquema* (*schema*)

Ejemplo: árbol de usuarios y grupos en LDAP, basado en nombres de dominios de Internet:



- cada nodo puede tener varios atributos, p.e. el nodo *uid=pepe* podría tener los siguientes atributos:

```
dn: uid=pepe,ou=people,dc=midominio,dc=test
objectClass: account
cn: Jose Pena
sn: Pena
description: alumno
mail: pepe@midominio.test
```

- el formato en el que se muestran los atributos del objeto se denomina LDIF (*LDAP Data Interchange Format*)
  - formato de intercambio de datos para importar y exportar datos a un servidor LDAP

4.4.1. Instalación de un servidor LDAP

Describiremos como montar un servidor LDAP simple que nos permita la gestión de usuarios y grupos

- el servidor mantendrá la lista de usuarios y grupos del dominio
- en los clientes la autenticación de usuarios y los permisos se basará en el servidor LDAP

Instalación del servidor LDAP

- Instalar los paquetes:

  `slapd` (servidor OpenLDAP)

  `ldap-utils` (utilidades del paquete OpenLDAP: `ldapsearch`, `ldapadd`)

  - En la configuración, elegir una contraseña para el administrador del LDAP (no tiene que ser la contraseña de root del sistema)
  - Esto hace una instalación básica, tomando como `DN` base (*suffix*) el dominio DNS de nuestro sistema (lo que devuelve `dnsdomainname`)
  - También crea un administrador (cn=admin) de LDAP
  - Para cambiar la clave u otros parámetros se puede reconfigurar el paquete `slapd` con: `dpkg-reconfigure slapd`

- El comando `slapcat` permite ver la estructura creada

```
dn: dc=nombre,dc=apellido1,dc=apellido2
objectClass: top
objectClass: dcObject
objectClass: organization
o: nombre.apellido1.apellido2
...
```

- Ficheros de configuración:

  - Fichero de opciones del demonio `/etc/default/slapd`
    Permite, entre otras cosas, especificar las interfaces donde se desea que escuche ldap (por defecto, todas las interfaces usando TCP puerto 389, `URI ldap://389`). Si se modifica tenemos que reiniciar el servicio: `systemctl restart slapd`
  - Ficheros de configuración del servidor: `/etc/ldap/sldap.conf`
    Permite especificar la base por defecto, el servidor LDAP, etc.

Creación de la estructura de la base de datos

Crearemos una estructura para almacenar como nodos los grupos y usuarios del sistemas. Para ello necesitaremos tres ficheros:

1. Crear el siguiente fichero `base.ldif` en formato LDIF que generará los nodos de los que cuelguen los grupos y los usuarios:

```
# Definimos la estructura superior de LDAP
dn: ou=groups,dc=nombre,dc=apellido1,dc=apellido2
objectClass: organizationalUnit
ou: groups

dn: ou=people,dc=nombre,dc=apellido1,dc=apellido2
objectClass: organizationalUnit
ou: people
```

2. Crear un o más ficheros `group.ldif` que contengan los grupos que queramos incorporar a LDAP:

```
# Definimos el grupo empleados
dn: cn=empleados,ou=groups,dc=nombre,dc=apellido1,dc=apellido2
objectClass: top
objectClass: posixGroup
cn: empleados
gidNumber: 2000
```

   Este fichero define el grupo `empleados` con información similar a la aparece en el fichero `/etc/passwd`.

3. Por crearemos uno o más ficheros `people.ldif` que contengan los usuarios que queramos incorporar a LDAP:

```
# Definimos el usuario pepe
dn: cn=pepe,ou=people,dc=nombre,dc=apellido1,dc=apellido2
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: pepe
cn: Jose Pena
uid: pepe
uidNumber: 2000
gidNumber: 2000
userPassword: pepe
homeDirectory: /home/pepe
loginShell: /bin/bash
gecos: Jose Pena, Despacho 22,,
```

Este fichero define un usuario `pepe` con información similar a la aparece en el fichero `/etc/passwd`.

4. Añadiremos los nodos a la base de datos:

```
# ldapadd -x -D cn=admin,c=nombre,dc=apellido1,dc=apellido2 \
    -W -f base.ldif
# ldapadd -x -D cn=admin,c=nombre,dc=apellido1,dc=apellido2 \
    -W -f group.ldif
# ldapadd -x -D cn=admin,c=nombre,dc=apellido1,dc=apellido2 \
    -W -f people.ldif
```

   - `-x` autenticación simple sin SASL
   - `-D` nombre distintivo con el que nos conectamos a LDAP (ponemos el del administrador)
   - `-W` pide la contraseña de forma interaciva
   - `-f` fichero a cargar

5. Si todo es correcto, la salida debe ser:

```
adding new entry "ou=people,dc=nombre,dc=apellido1,dc=apellido2"
adding new entry "ou=groups,dc=nombre,dc=apellido1,dc=apellido2"
adding new entry "cn=empleados,ou=groups,dc=nombre,dc=apellido1,dc=apellido2"
adding new entry "cn=pepe,ou=people,dc=nombre,dc=apellido1,dc=apellido2"
```

6. También podemos leer los nodos a modo de comprobación:

```
# ldapsearch -x -b dc=nombre,dc=apellido1,dc=apellido2
# ldapsearch -x -b dc=nombre,dc=apellido1,dc=apellido2 ou=people
# ldapsearch -x -b dc=nombre,dc=apellido1,dc=apellido2 ou=group
```

7. Añadir una contraseña al usuario: puede hacerse directamente metiendo un campo `userPassword` en el fichero anterior, pero es preferible hacerlo mediante el comando `ldappasswd`

```
# ldappasswd -x cn=pepe,ou=people,dc=nombre,dc=apellido1,dc=apellido2 \
    -D cn=admin,dc=nombre,dc=apellido1,dc=apellido2 -W -S
```

8. Por último, también podemos querer instalar el servidor como cliente, por lo que debemos seguir los pasos indicados en la sección de instalación de un cliente

4.4.2. Instalación de un cliente LDAP

Describiremos como como configurar un cliente para que acceda a la información almacenada en el directorio de LDAP del servidor

Tres pasos:

1. Indicar en el fichero `/etc/ldap/ldap.conf` la información sobre el servidor LDAP y el URI
2. Instalar y configurar el *Name Service Switch* (fichero de configuración `/etc/nsswitch.conf`)
3. Instalar y configurar el módulo de autenticación (PAM, *Pluggable Authentication Modules*)

Configuración de LDAP

- Instalar el paquete `ldap-utils`.
- En el fichero `/etc/ldap/ldap.conf` debemos introducir la información del servidor LDAP:

```
BASE    dc=nombre,dc=apellido1,dc=apellido2
URI     ldap://servidor
```

- Comprobar que tenemos conexión con el servidor

```
ldapsearch -x -b dc=nombre,dc=apellido1,dc=apellido2
```

La salida nos deberá dar el grupo (*empleados*) y el usuario (*pepe*) introducidos.

Configurar el *Name Service Switch*

El NSS se encarga, entre otras, de realizar la correspondencia entre los números y nombres de usuario

- permite gestionar los permisos de acceso de usuarios a ficheros
- se configura a través del fichero `/etc/nsswitch.conf` (cuyo formato vimos en el tema anterior)

Pasos:

1. En el fichero `nsswitch.conf` añadir la palabra `ldap` a las líneas `passwd`, `group`, `shadow` y `gshadow`

```
passwd:    files systemd ldap
group:    files systemd ldap
shadow:    files systemd ldap
gshadow:    files systemd ldap
```

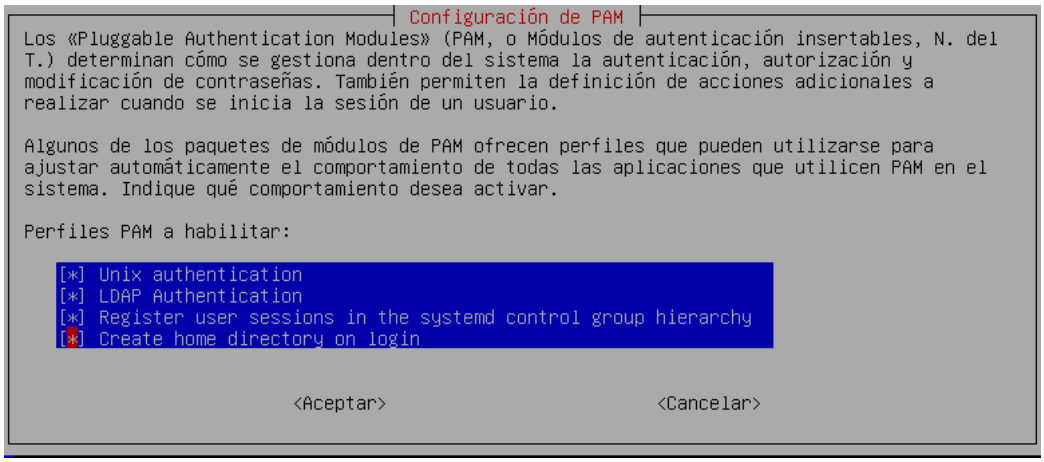
   - esto indica al NSS que busque la información primero en los ficheros locales y, si no la encuentra, en el servidor LDAP
   - NOTA: Esta operación de edición del fichero `/etc/nsswitch.conf` en las nuevas versiones se hace automáticamente durante la instalación de los módulos PAM asociados y ya no es necesario modificar manualmente este fichero.

Instalación de los módulos PAM

Como hemos comentado en el anterior tema, PAM (*Pluggable Authentication Module*) es una biblioteca de autentificación genérica que cualquier aplicación puede utilizar para validar usuarios, utilizando por debajo múltiples esquemas de autentificación alternativos (ficheros locales, Kerberos, LDAP, etc.)

1. Instalar los paquetes `libnss-ldapd` y `libpam-ldap`
   - Como dependencia se instala también el paquete `nscd` (*Name Service Cache Daemon*)
   - Hace caché para los nombres leidos del servidor LDAP para aumentar la eficiencia
2. En la configuración indicar
   - URI: `ldap://servidor`
   - DN: `dc=nombre,dc=apellido1,dc=apellido2`
   - Versión de LDAP: 3
   - Cuenta LDAP para root: `cn=admin,dc=nombre,dc=apellido1,dc=apellido2`
   - La clave de LDAP.

- Marcamos la opción de que las cuentas se creen automáticamente en el primer acceso del usuario.



3. En la reconfiguración podemos comprobar los datos que hemos introducido:

```
# dpkg-reconfigure ldap-utils
# dpkg-reconfigure libpam-ldap
# dpkg-reconfigure libnss-ldapd
```

4. Activamos los cambios en PAM con el comando `pam-auth-update` (con la opción `-force` si es necesario).

5. Por último, reiniciamos el cliente con `reboot`.

4.4.3. Configuración de LDAP con múltiples servidores

Podemos configurar varios servidores LDAP que mantengan imágenes sincronizadas de la información del directorio

- equilibran la carga de las consultas, y mejora la tolerancia a fallos

Esquema de maestro único y múltiples esclavos

- el *maestro* mantiene la copia principal sobre la que se hacen los cambios
  - si un cliente intenta hacer un cambio en un esclavo, este lo redirige automáticamente al maestro
- cada vez que se produce un cambio en el directorio del maestro, el servicio `slapd` escribe dicho cambio, en formato LDIF, en un fichero de log

- el demonio `slurpd` lee dichos cambios e invoca las operaciones de modificación correspondientes en todos los esclavos

4.4.4. Herramientas de administración de LDAP

Administrar LDAP desde línea de comandos resulta muy engorroso

- Se pueden programar scripts que nos hagan el trabajo. Por ejemplo:

  `https://www.server-world.info/en/note?os=Debian_9&p=openldap&f=2`

  `https://gist.github.com/plugandplay/1401980`

- Existen numerosas herramientas visuales que facilitan la gestión de LDAP. Algunas de ellas son:

  1. `phpldapadmin` - interfaz basada en web para administrar servidores LDAP
  2. `gosa` - herramienta de administración, basada en PHP, para gestión de cuentas y sistemas en LDAP
  3. `ldap-account-manager` - webfrontend para gestión de cuentas en un directorio LDAP
  4. `gq` - cliente LDAP basado en GTK+/GTK2 (bastante simple)
  5. `cpu` - herramientas de gestión para consola: proporciona comandos tipo `useradd`/`userdel` para usar con LDAP