

## 3. Actividades Administrativas Básicas

Copyright (c) 2025 Adrián Quiroga Linares Lectura y referencia permitidas; reutilización y plagio prohibidos

### 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 usuario

Una CPU puede constar de varios núcleos. El planificador de procesos se encarga de asignar los procesos a los núcleos en función de 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 puede 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**

Comandos	Función
<code>ps, pstree, top</code>	listan los procesos
<code>strace</code>	muestra las llamadas al sistema de un proceso
<code>fg / bg</code>	ejecución en primer o segundo plano
<code>jobs</code>	comandos lanzados desde el shell actual
<code>kill</code>	envía señales a procesos por PID
<code>CNTL-C, CNTL-Z</code>	envían señales de teclado
<code>pgrep</code>	busca procesos por nombre
<code>pkill, killall</code>	envían señales a procesos por nombre
<code>nohup</code>	lanza procesos ignorando la señal SIGHUP
<code>exec</code>	reemplaza un proceso por otro
<code>nice, renice</code>	fija/cambia la prioridad de un proceso
<code>ulimit</code>	controla los recursos del shell actual
<code>uptime</code>	información sobre el estado y carga del sistema
<code>w</code>	información sobre los usuarios y sus procesos
<code>free</code>	información sobre la memoria total, usada y libre
<code>/proc, /sys</code>	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

`ps`

#### ps (process status)

Lista los procesos con su **PID** (identificador de 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 la terminal actual y con el mismo EUID que el usuario que lo lanzó.

Tipos de opciones:

- **Opciones UNIX:** `ps -e`
- **Opciones BSD:** `ps ax`
- **Opciones largas GNU:** `ps --user tomas`
- `-e`: muestra todos los procesos
- `-u`: muestra los procesos de un usuario

- **-o**: permite definir el formato de salida

```
$ ps -o user,pid,state,comm
```

USER	PID	S	COMMAND
amo	1357	S	bash
amo	1992	S	gedit
amo	2279	R	ps

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

En cuanto al estado (código state), son posibles entre otros:

Código	significado
R	<i>Running</i> (ejecutándose o en cola de ejecución)
S	<i>interruptible Sleep</i> (dormido y esperando un evento, p.e, la entrada de un dato por teclado)
D	<i>uninterruptible sleep</i> (usualmente esperando IO, p.e., la copia de un fichero requiere que los datos lleguen desde el disco)
T	<i>sTopped</i> (detenido por una señal de control: kill, Ctrl-Z)
Z	<i>Zombie</i> (proceso terminado, pero que todavía está en la tabla de procesos porque su padre no lo liberó)

## 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

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 lleva el sistema encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, 15 mins
- Número total de tareas y resumen por estado
- estado de ocupación de la CPU y la memoria. En un sistema  $n$  núcleos el máximo uso de CPU es  $n \times 100\%$   
Por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU. Pulsando h mientras se ejecuta top, obtenemos una lista de comandos interactivos. Los campos de las columnas tienen un significado 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

```
$ 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 se añade &

```
$ sleep 10 &
```

- para terminar un proceso en foreground Ctrl-C
- para pasar un comando en foreground usar Ctrl-Z
  - **bg** pasa el proceso a background
  - **fg** lo devuelve a foreground

```
$ 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 identificando 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 lo que hace cada señal `man 7 signal`

La sintaxis de `kill`:

- `kill [señal] PID`
- `kill -9` es equivalente a `kill -SIGKILL`

Las señales más comunes son:

- **SIGTERM:** mata al proceso permitiéndole terminar correctamente
- **SIGKILL:** mata al proceso inmediatamente

- **SIGSTOP:** detiene temporalmente el proceso
- **SIGCONT:** continua si el proceso fue parado
- **SIGINT:** interrupción de teclado (Ctrl+C), mata el proceso
- **SIGTSTP:** stop de teclado (Ctrl+Z) para temporalmente el proceso

Algunas características:

- Excepto **SIGKILL** y **SIGSTOP**, las demás señales pueden ser ignoradas o gestionadas por el proceso
- La señal que se envía por defecto es **SIGTERM**
  - Se puede ignorar y no terminar
  - la señal equivalente de teclado es **SIGINT**
- **SIGSTOP** y **SIGTSTP** se utilizan para detener temporalmente un proceso, la primera desde un programa y la segunda desde teclado (Ctrl-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.

## 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:

- 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 de **SIGHUP**
- 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:

- **pgrep sshd** devuelve el PID de **sshd**

**pkill** permite enviar señales a otros procesos indicándolos por el nombre en vez por PID

- **pkill -KILL firefox**
- Si hay varios procesos con el mismo nombre los mata a todos
- admite también patrones en vez de nombres

**exec** ejecuta un comando reemplazando al shell desde el que se lanza

## 3.1.3 Manejo de la prioridad y recursos de un proceso

Cuando un proceso se ejecuta, lo hace con una cierta prioridad

- Van desde -20 (más alta) a 19 (más baja)
- Por defecto, los procesos se ejecutan con prioridad 0

- Los comandos para manejo de prioridades son **nice** y **renice**

**nice** permite lanzar un comando con una cierta prioridad.

- nice -n ajuste comando**

```
$ 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

- renice pri [-p pid] [-u user] [-g pgrp]**
- 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**

```
$ 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 **ulimit** permite controlar los recursos de los que dispone un proceso arrancado por el shell

- ulimit [opciones] [límite]**
- a** muestra los límites actuales
- f** máximo tamaño de los ficheros creados por el shell (opción por defecto)
- n** máximos número de ficheros abiertos
- s** tamaño máximo de pila
- t** máximo tiempo de cpu
- S/-H** usa los límites **soft** y **hard**

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 rendimiento del sistema en cuanto a uso de CPU y consumo de memoria

**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.

```
$ uptime
20:25:03 up 25 days, 11:12, 13 users, load average: 3.00, 3.07, 3.08
```

además de la información dada por **uptime**, el comando **w** muestra información sobre los usuarios y sus procesos

```
$ 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
```

- **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 TYY
- **PCPU** tiempo de CPU consumido por el proceso actual

**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

```
$ 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.
- **-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 en segundos

## 3.1.5 Los directorios /proc y /sys

Son dos directorios 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 ordenador.
- /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 carácter (que operan carácter a carácter, 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`

## 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.

### 3.2.1 Tipos de ficheros y operaciones

- **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, permite 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 **blandos**: apuntador a otro fichero; se crean con `ln -s` y se borran con `rm`.

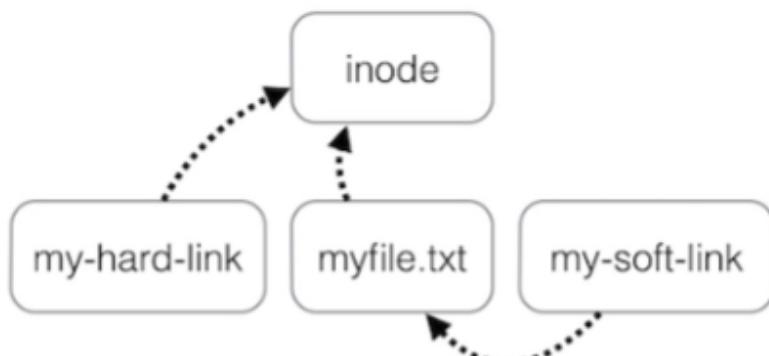
El comando `file` nos permite determinar el tipo de un fichero, para ficheros normales distingue según contenido (fichero de imagen, pdf, ASCII, etc):

```
$ 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 pueden 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



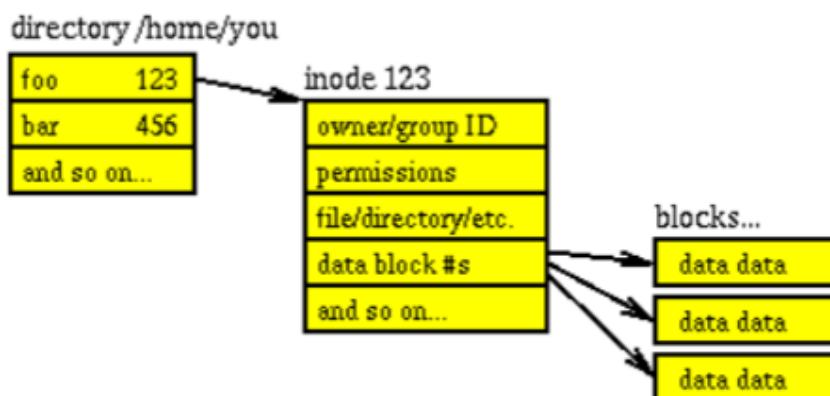
## Comando `ln`

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`

Tipo de fichero	Número de enlaces	Grupo del fichero	Fecha de última modificación
-rw-rw-r--	1 tomas	user 731	2005-09-21 20:59 fichero3

Permisos      Propietario del fichero      Tamaño      Nombre del fichero

**Indicador de tipo** el primer carácter nos indica el tipo de 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 nombre (enlaces duros) del fichero. En caso de un directorio, esto corresponde con el número de subdirectorios

**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`. Además linux guarda para cada fichero 3 tipos de fecha:

- `mtime`, fecha de la última modificación, opción por defecto.
- `atime`, fecha del último acceso, se muestra con `ls -l --time=atime`
- `ctime`, fecha del último cambio de estado, se muestra con `ls -l --time=ctime`

**Nombre:** la longitud máxima del nombre es de 255 caracteres.

## Permisos de ficheros y directorios

UNIX proporciona tres operaciones básicas para realizar sobre un fichero o directorio: lectura (`l`), escritura (`w`) y ejecución (`x`)

Efecto sobre un fichero:

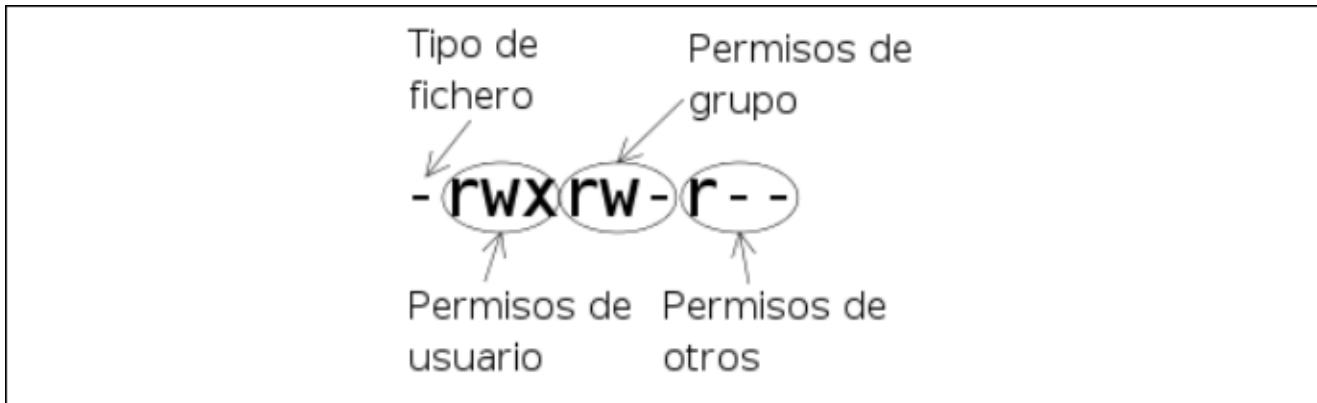
- **Lectura:** permite abrir y leer el fichero
- **Escritura:** permite modificar o truncar el fichero
- **Ejecución:** permite ejecutar el archivo (binario o script)

Efecto sobre directorios:

- **Ejecución:** permite entrar al directorio (pero no listar contenido ni crear ficheros ni directorios)
- **Lectura y ejecución:** permite listar el contenido del directorio (pero no crear ficheros ni directorios)
- **Escritura y ejecución:** permite crear, borrar o renombrar ficheros (pero no listar su contenido)
- **Acceso total**

Los permisos tienen tres categorías:

- Permisos de usuario (**u**): propietario del fichero (por defecto, 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



## Cambios de permisos

**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 por 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**

- **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. 4 para **r**, 2 para **w** y 1 para **x**, para combinaciones se suman (**rw=7**)
  - 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

Ademas de **rwx** existen los permisos **setuid/setgid(s)** (relacionados con los atributos de los procesos) y **sticky bit (t)**.

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. Un proceso con UID efectivo 0 (root) puede manipular todos los ficheros del sistema

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.

fijar <b>setuid</b> :	<b>chmod u+s</b> , <b>chmod 4xyz</b>
fijar <b>setgid</b> :	<b>chmod g+s</b> , <b>chmod 2xyz</b>
siendo x, y, z los otros tres permisos (de 0-7)	

**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

- Forma simbólica: `chmod +t dir`
- Forma numérica: `chmod 1xyz dir`

## Cambio de usuario/grupo

Los comandos `chown` y `chgrp` permiten cambiar el propietario y grupo de un fichero. Sólo root puede cambiar el propietario y el grupo puede cambiarse a otro al que pertenezcamos

```
chown [opciones] proprietario ficheros
chgrp [opciones] grupo ficheros
chown [opciones] proprietario:grupo ficheros
```

### 3.2.2 Localización de ficheros

#### Comando `find`

Busca a través de la jerarquía de directorios ficheros que cumplan determinado criterio.

```
find [directorio de búsqueda] [expresión]
```

- 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:

- operaciones: 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 búsqueda

### Búsqueda por nombre/path/tipo/usuario/grupo:

Criterio	Efecto
<code>-name <i>patrón</i></code>	busca ficheros que coincidan con el patrón (pueden usarse comodines, pero deben ponerse entre comillas o escapados)
<code>-iname <i>patrón</i></code>	igual que el anterior, pero no distingue mayúsculas/minúsculas
<code>-regex <i>patrón</i></code>	igual pero usa expresiones regulares
<code>-path/-ipath <i>patrón</i></code>	búsca el camino path
<code>-type <i>tipo</i></code>	busca por tipo de fichero (f, d, l, b, c, p, s)
<code>-user, -group <i>nombre</i></code>	busca por propietario/grupo
<code>-uid/-gid <i>n</i></code>	busca por UID/GID
<code>-nouser/-nogroup</code>	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
<code>-size <i>n</i>[ckMG]</code>	tamaño igual a <i>n</i> ( <i>c</i> =caracteres(bytes), <i>k</i> =kbytes, <i>M</i> =Mbytes, <i>G</i> =Gbytes)
<code>-size <i>-n</i>[ckMG]</code>	tamaño menor a <i>n</i>
<code>-size <i>+n</i>[ckMG]</code>	tamaño mayor que <i>n</i>
<code>-perm <i>permisos</i></code>	permisos exactos
<code>-perm <i>-permisos</i></code>	permisos indicados (si se indica solo el permiso de lectura, no se miran los permisos de escritura y ejecución)
<code>-perm /<i>permisos</i></code>	para los permisos exactos el prefijo / indica que es suficiente que se verifique uno de los permisos entre usuario, grupo y otros lo mismo para los permisos indicados
<code>-perm /-<i>permisos</i></code>	

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

- **Búsqueda por atributos temporales**

Criterio	Efecto
<code>-atime [ /+/-]n</code>	busca ficheros cuya última fecha de acceso para lectura coincide (), es anterior (+) o es posterior (-) a <i>n</i> días
<code>-mtime [ /+/-]n</code>	lo mismo, pero con la fecha de última modificación del fichero
<code>-ctime [ /+/-]n</code>	lo mismo, pero con la fecha en que se cambió el estado del fichero
<code>-amin/-mmin/</code> <code>-cmin [ /+/-]n</code>	lo mismo, pero ahora <i>n</i> representa minutos
<code>-newer file</code>	busca ficheros modificados más recientemente que <i>file</i>
<code>-anewer file</code>	ficheros con último acceso más reciente que la modificación de <i>file</i>
<code>-cnewer file</code>	ficheros con cambio de estado más reciente que la modificación de <i>file</i>

## Operadores de `find`

Operador	Descripción
<code>expr1 expr2</code>	AND ( <i>expr2</i> no se evalúa si <i>expr1</i> es falsa). También puede ponerse: <code>expr1 -a expr2</code>
<code>expr1 -o expr2</code>	OR ( <i>expr2</i> no se evalúa si <i>expr1</i> es cierta)
<code>! expr1</code>	NOT (cierto si <i>expr</i> falsa). También puede ponerse: <code>not expr1</code>
<code>\( expr1 \)</code>	agrupan expresiones (hay que escapar los paréntesis)

## Acciones de `find`

Acción	Descripción
<code>-print</code>	imprime el nombre de los ficheros que encuentra (acción por defecto)
<code>-ls</code>	imprime el nombre de los ficheros con formato de listado largo
<code>-exec comando {} \;</code>	ejecuta <code>comando</code> sobre los ficheros encontrados
<code>-ok comando {} \;</code>	igual que <code>-exec</code> pero pregunta antes de ejecutar <code>comando</code>
<code>-prune</code>	no desciende por el directorio indicado con la opción <code>path</code>

- `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

## Otros comandos para localizar ficheros

**which** muestra la localización de comandos buscando en **\$PATH**

- **which [opciones] comando**

```
$ which ls
/bin/ls
```

**whereis** muestra la localización del binario, fuente y página del manual de un comando buscando en las localizaciones estándar

- **whereis [opciones] comando**

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

**locate** localiza todo tipo de ficheros rápidamente. Utiliza una bd donde guarda la localización de los ficheros. Esa base de datos la crea y actualiza el administrador con el comando **updatedb**.

## 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:

- Creación de particiones (comando **fdisk**)
- Creación de los sistemas de ficheros (comando **mkfs**)
- Montado de los sistemas de ficheros (comando **mount** o **/etc/fstab**)

Comando	Operación
<code>fdisk</code>	crea o lista particiones
<code>blkid</code>	muestra el identificador único universal (UUID)
<code>mkfs.<i>tipo</i></code>	crea sistemas de ficheros de tipo <i>tipo</i>
<code>fsck.<i>tipo</i></code>	testea y repara sistemas de ficheros
<code>tune2fs</code>	ajusta parámetros de ext2/ext3/ext4
<code>dumpe2fs</code>	muestra información de ext2/ext3/ext4
<code>e2label</code>	fija la etiqueta de ext2/ext3/ext4
<code>mkswap</code>	crea un sistema de ficheros de tipo swap
<code>swapon</code>	activa la partición de swap
<code>mount / umount</code>	monta/desmonta particiones
<code>fstab / mtab</code>	ficheros relacionados con el montado
<code>df / du</code>	muestra espacio en particiones montadas / directorios

## Creación de particiones

- `fdisk [opciones] dispositivo`
- `dispositivo` es el dispositivo del disco `/dev/sdx/` para SATA o `/dev/nvmeX` para NVME
- Debemos tener permiso de administrador para usarlo
- `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
<code>/dev/sdb1</code>		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

## Creación de los sistemas de ficheros

Sobre cada partición debemos crear sistemas de ficheros con el comando `mkfs`

- `mkfs.tipo [opciones] dispositivo`
- `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

`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

## 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.

`mount` permite asociar (montar) directorios a sistemas de ficheros:

- `mount [opciones] [disp.][dir.]`

```
$ mount /dev/sdb1 /home2
```

- 

`umount` desmonta los sistemas de ficheros:

- `umount [opciones] directorio`

```
$ umount /home2
```

- 

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)
```

/dev/sda1	/	auto	defaults	0	1
/dev/sda9	/home	ext4	auto,user,rw	0	2

- Algunas opciones son:

- `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`)

- Si un directorio aparece listado en el fstab puede montarse sin especificar el dispositivo:

```
$ mount /home
```

Fichero `/etc/mtab`. Contiene una lista de los filesystem que están montados en el sistema

```
$ 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
```

## Chequeo del sistema de ficheros

- `fsck.tipo [opciones] dispositivo`
- La opción `-y` repara el filesystem sin preguntar sobre los errores

## Otras utilidades

`du` muestra el espacio de disco usado por los ficheros y subdirectorios de un directorio

- **du [opciones] [directorio]**

- **-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

```
$ 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

- **df [opciones]**

- **-h** (humano) salida más legible
- **-T** (tipo) muestra el tipo de sistema de ficheros
- **-l** (local) sólo muestra filesystems locales

```
$ 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 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,

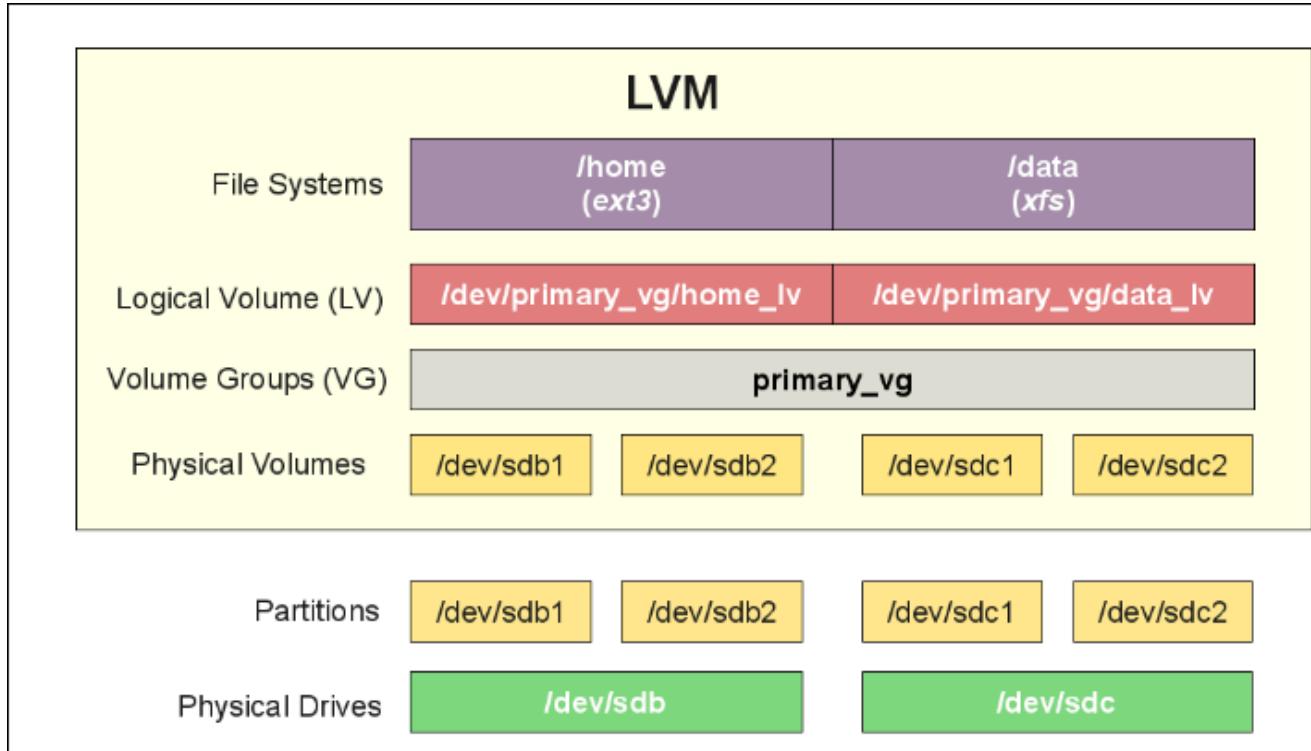
```
$ cat /etc/fstab
# /etc/fstab: static file system information.
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5)

# <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 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



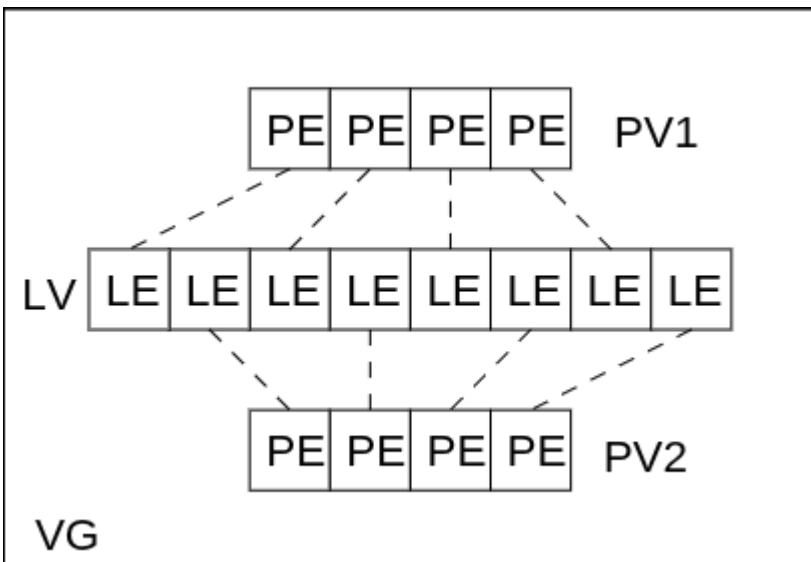
La estructura de LVM es la siguiente:

- 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)**
- A partir de los volúmenes físicos se construyen agrupaciones lógicas denominadas **Grupos de Volúmenes (VG)**. Hay que poner un nombre (etiqueta) a cada VG.
- Los grupos de volúmenes se dividen de forma lógica en **Volúmenes Lógicos (LV)**. A cada LV ha de dársele una etiqueta.
- Los volúmenes lógicos pueden cifrarse (véase un apartado posterior). A cada LV cifrado ha de dársele una etiqueta.
- Para utilizar un volumen volumen lógico antes debe formatearse.
- 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).

<code>pvdisplay o pvs</code>	información del volumen físico
<code>vgdisplay o vgs</code>	información del grupo de volúmenes
<code>lvdisplay o lvs</code>	información del volumen lógico
<code>pvcreate</code>	crear de un volumen físico
<code>vgcreate</code>	crear un grupo de volúmenes
<code>vgremove</code>	borrar un grupo de volúmenes
<code>vgextend</code>	añadir un volumen físico a un grupo de volúmenes
<code>vgreduce</code>	quitar un volumen físico de un grupo de volúmenes
<code>lvcreate</code>	crear un volumen lógico
<code>lvremove</code>	borrar un volumen lógico
<code>lvextend</code>	aumentar de tamaño un volumen lógico
<code>lvreduce</code>	reducir de tamaño un volumen lógico
<code>fsadm</code>	Comando genérico cambiar tamaño un filesystem
<code>resize2fs, btrfs, xfs_growfs</code>	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
```

- 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-3f09-jXcr-vNde-f7iF-y90a21
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` y `vremove /dev/NuevoGrupo/otrotestlv`
- Agrandar un LV; se puede especificar el nuevo tamaño en bytes (-L) o LEs (-I), o la diferencia (+/-): `vextend -L12G /dev/GrupoVolumen/homelv`, `vextend -L+1G`

```
/dev/GrupoVolumen/tmp_lv, vextend -l+200 /dev/GrupoVolumen/tmp_lv.
```

- 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 él 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 volúmenes cifrados.

## Asignar sistemas de ficheros

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
```

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
```

Comandos específicos de redimensionado de ficheros. Permiten ampliación en caliente (sin desmontar), aunque para reducción requieren el desmontado. Las características de cada comando dependen de cada filesystem particular.

## 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 tipo:

- 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 /etc/group, /etc/gshadow /etc/skel, /etc/adduser.conf	información de usuarios información de grupos plantillas para crear las cuentas
chown, chgrp chmod	cambio del propietario de ficheros/dir cambio de permisos para ficheros/dir
passwd, vipw gpasswd chage newgrp	cambio contraseña/información de usuarios gestión de grupos (clave, miembros) por root cambio de expiración cambio entre grupos por el propio usuario
useradd, userdel, usermod groupadd, groupdel, groupmod adduser, deluser addgroup, delgroup newusers, chpasswd mkpasswd	creación de usuarios (bajo nivel) creación de grupos (bajo nivel) creación de usuarios (alto nivel) creación de grupos (alto nivel) operación múltiple 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, ...

- **/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

```
root:x:0:0:root:/root:/bin/bash
pepe:x:1002:1002:Pepe Perez,dept.EC,desp.1:/home/pepe:/bin/bash
```

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
  - preferiblemente corto, minúsculas y sin caracteres acentuados
  - fácil de recordar
  - de formato fijo para todos los usuarios (por ejemplo 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
  - número por debajo de 1000 para usuarios especiales del sistema
  - el **UID** para un usuario debería ser único, y el mismo para todas las máquinas
  - se debe evitar utilizar 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 e información adicional
- **/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 :

## Fichero /etc/group

Información sobre los grupos de usuarios.

```
users:x:100:pepe,elena
```

Donde tenemos nombre del grupo, contraseña del grupo (si **x**, se guarda en el fichero **/etc/shadow**), **GID** identificador número del grupo y lista de usuarios que pertenecen al grupo.

Algunos grupos:

<b>users</b>	son los usuarios normales
<b>sudo</b>	puede ejecutar cualquier comando (son superusuarios)
<b>http</b>	puede gestionar los servicios web
<b>games</b>	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**

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 miembro 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**

### 3.4.2 Creación manual de una cuenta

- Editar el fichero **/etc/passwd** y añadir una nueva línea para la cuenta.

- Para evitar la corrupción del fichero usar el comando `vipw`, este comando solo grabará el fichero si el formato es correcto y nos permite seleccionar el editor a usar
- Si el sistema usa `shadow`, poner `x` en el campo de contraseña
- Editar el fichero `/etc/shadow`
  - para evitar la corrupción del fichero usar el comando `vipw -s`
  - la contraseña debe escribirse cifrada (lo hace el comando `passwd`)
- Editar `/etc/group` para añadir un nuevo grupo (si es necesario) o para añadir el usuario a los grupos que deseemos
- Si es necesario, editar `/etc/gshadow`
  - poner la contraseña cifrada (`!` o `*` si no queremos ninguna)
  - añadir también aquí la lista de miembro del grupo
- Crear el directorio del usuario
- Copiar los ficheros de `/etc/skel` al directorio del usuario
- Usar `chown`, `chgrp` y `chmod` para fijar el propietario, grupo y permisos del directorio
- Fijar la contraseña con `passwd`
  - el usuario debe cambiar la contraseña tan pronto como sea posible
  - puede迫使se con la opción `-e`

## Comando passwd

Permite fijar, cambiar la contraseña de un usuario o sus propiedades

```
passwd [opciones] [username]
```

Opciones:

- `-e` fuerza a que el usuario cambie la contraseña al siguiente login
- `-d` borra la contraseña
- `-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: 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`

```
chage [opciones] [username]
```

Opciones:

- **-l** muestra información de expiración

```
$ 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: **useradd [opciones] username**
  - **-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 *mfpaswd*)
  - **-e fecha** fecha de expiración de la cuenta (YYYY-MM-DD)

```
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 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 un front-end a los comandos de bajo nivel anterior `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 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
  - Si no se especifica el `username` pasa a administrador
  - Nos pedirá la clave del usuario destino
  - `-` inicia un login shell (igual al usuario entrando en su cuenta)
  - `-p` preserva el entorno (no ejecuta el `.bashrc` del usuario)

**su [opciones] [-] username**

- **sudo** ejecuta un comando como administrador o con la identidad de otro usuario
  - si no se encuentra instalado en el sistema, se puede instalar 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
  - Un usuario del grupo **sudo** puede ejecutar cualquier comando
  - EN caso contrario se comprueba el fichero **/etc/sudoers** para ver si el usuario está en la lista de los permitidos a usar el comando

**sudo [-u username] 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.

```
#usuario    host=(user:group)    opciones: comando
debian      ALL=(ALL:ALL)        NOPASSWD: /usr/sbin/vipw
```

```

# 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** es una biblioteca de autentificación genérica que cualquier aplicación puede utilizar para validar usuarios, utilizando por debajo múltiples esquemas de autentificación alternativos. PAM utiliza módulos de varios tipos:

- Módulos de autentificación (auth): para la identificación del usuario (por ejemplo, contraseña, tarjeta de identificación, características bio- métricas, 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.)

Existe un fichero de configuración para cada servicio que usa PAM.

### 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).

- Instalamos los paquetes con `apt-get install otpw-bin libpam-otpw`
- 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_otp.so
session   optional  pam_otp.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
```

- 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`

### 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
- 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

<code>/etc/fstab</code>	para indicar los filesystems que tendrán cuotas
<code>quotacheck</code>	construye el índice y testea la integridad
<code>quotaon/quotaoff</code>	activa/desactiva las cuotas
<code>edquota</code>	ajusta las cuotas de usuarios o grupos
<code>repquota</code>	genera informes de uso
<code>quota</code>	informa a un usuario del estado de sus cuotas

Para más información sobre cuotas ver Quota mini-HOWTO