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
`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

Figure 1: Pasted image 20251013130143.png

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 son 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
```

Figure 2: Pasted image 20251013130524.png

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
```

Figure 3: Pasted image 20251013130913.png

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

Figure 4: Pasted image 20251013131040.png

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

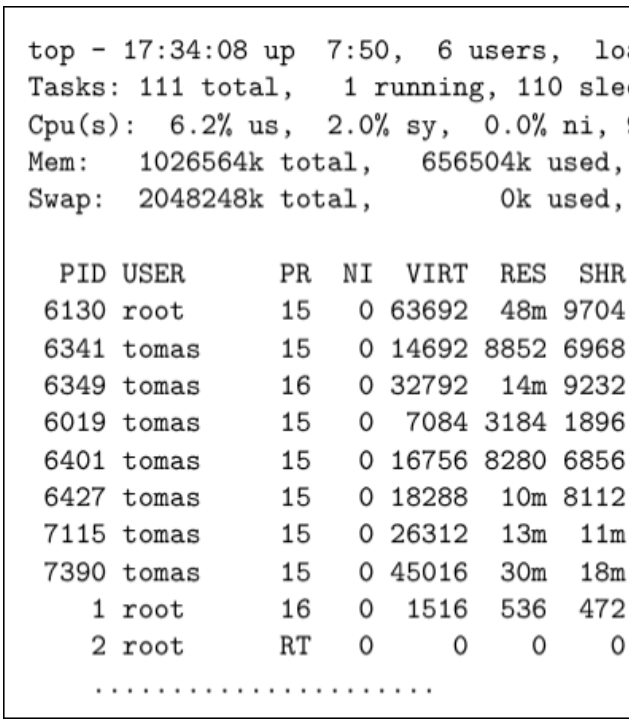
Código	
R	*Running* (ejecutándose
S	*interruptible Sleep* (dor
	entrada de un dato por
D	*uninterruptible sleep* (u
	de un fichero requiere
T	*sTopped* (detenido por
Z	*Zombie* (proceso termi
	de procesos porque su

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-+-a
        |         |         |       |                |      +-g
        ...
```

top



```
top - 17:34:08 up  7:50,  6 users,  lo
Tasks: 111 total,   1 running, 110 sle
Cpu(s):  6.2% us,  2.0% sy,  0.0% ni,
Mem:   1026564k total,   656504k used,
Swap:  2048248k total,        0k used,

  PID USER      PR  NI  VIRT  RES  SHR
 6130 root      15   0 63692  48m 9704
 6341 tomas     15   0 14692 8852 6968
 6349 tomas     16   0 32792  14m 9232
 6019 tomas     15   0  7084 3184 1896
 6401 tomas     15   0 16756 8280 6856
 6427 tomas     15   0 18288  10m 8112
 7115 tomas     15   0 26312  13m  11m
 7390 tomas     15   0 45016  30m  18m
    1 root      16   0  1516  536  472
    2 root      RT   0     0    0    0
    .....................
```

Nos da una lista de procesos actualizada periódicamente

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 significados similar a los del comando ps y se pueden seleccionar interactivamente pulsando f. Para salir, q.

strace

```
$ strace top
gettimeofday({1195811
open("/proc/meminfo",
fstat64(3, {st_mode=S
mmap2(NULL, 4096, PRO
read(3, "MemTotal:
close(3)
munmap(0xb7f55000, 40
open("/proc", O_RDONL
fstat64(3, {st_mode=S
fcntl64(3, F_SETFD, F
getdents(3, /* 52 ent
getdents(3, /* 64 ent
stat64("/proc/1", {st
open("/proc/1/stat",
read(4, "1 (init) S 0
close(4)
......
```

Muestra las llamadas al sistema realizadas por un proceso en ejecución

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

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

```
$ sleep 20
Ctrl-Z
[3]+ Stopped     sleep 20
$ bg
[3]+ sleep 20 &
$ fg
sleep 20
```

Figure 5: Pasted image 20251013132449.png

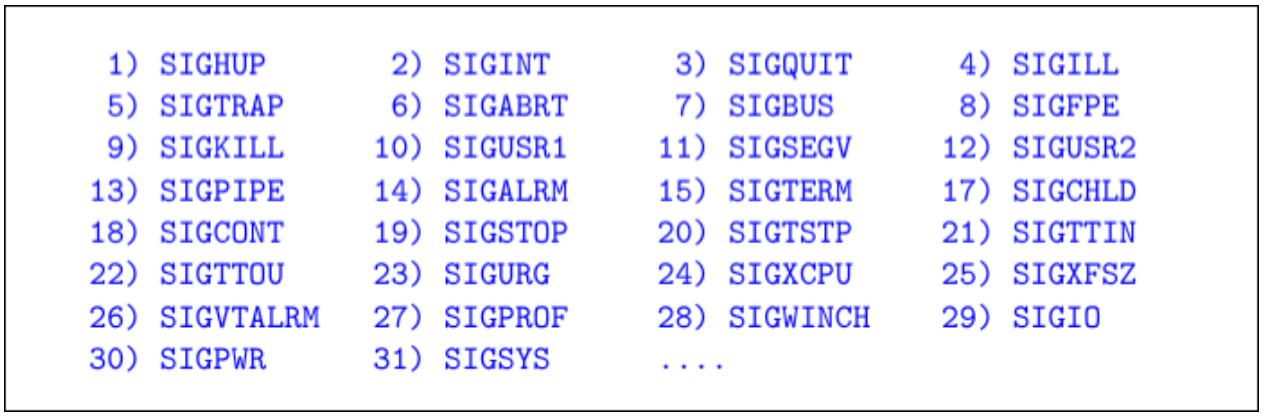
```
$ gedit nada.txt &; sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
$ fg 3
sleep 100
Ctrl-Z
[3]+ Stopped     sleep 100
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Stopped     sleep 100
$ bg 3
[3]+ sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
```

por su número)

3.1.2 Señalización de procesos

El comando básico para enviar señales a un proceso el `kill` - Si el proceso se lanzó en foreground se pueden enviar algunas señales desde teclado - `kill -l` lista el conjunto

de señales



- 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 carácterísticas: - 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 (Crtl-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 salimo sde 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] [limite]` - `-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`

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 cuento a uso de CPU y consumo de memoria

Ejemplo: limitar el tamaño de los ficheros creados a 1 kbyte

```
$ ulimit -f 1
```

Figure 6: Pasted image 20251013161000.png

`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
```

`w` 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, 3.0
USER     TTY      FROM     LOGIN@  IDLE    JCPU   PCPU  WHAT
paula    pts/1    godello  12:08pm  8:15m  8.39s  8.36s  ssh -p
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/tc
pichel   pts/5    -         7:12pm 56:56   0.26s  0.16s  ssh www
pichel   pts/6    -         4:35pm 21:15  16.61s  0.02s  /bin/sh
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
```

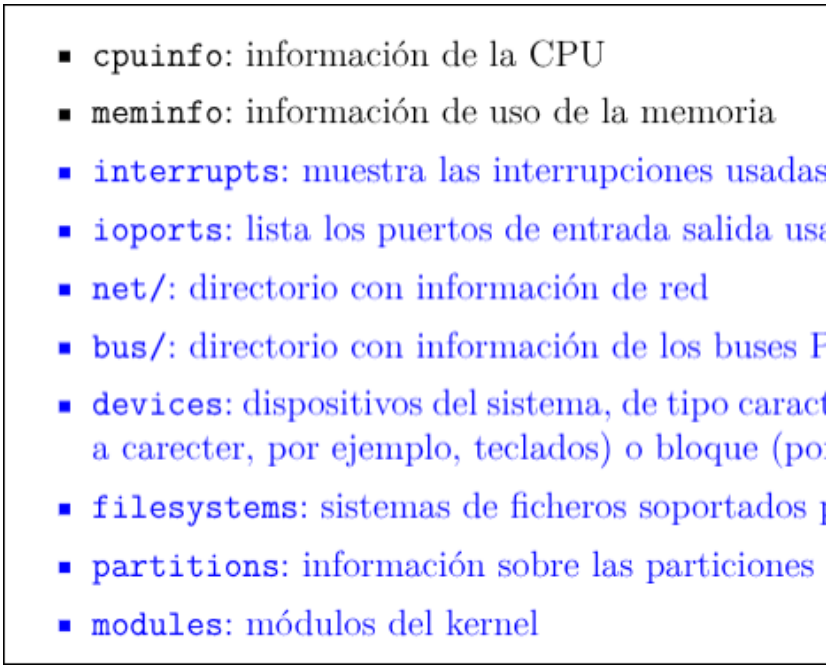
- La columna `shared` indica la memoria usada por tmpfs, `buffers` la memoria usada por los buffers del kernel y cache la memoria usada por los datos cacheados.
- `-b`, `-k`, `-m`, `-g` memoria en bytes/KBytes/MBytes/GBytes
- `-t` muestra una línea con el tatal de memoria (física+swap)
- `-s` delay muestra la memoria de forma continua, cada delay en segundos

3.1.5 Los directorios /proc y /sys

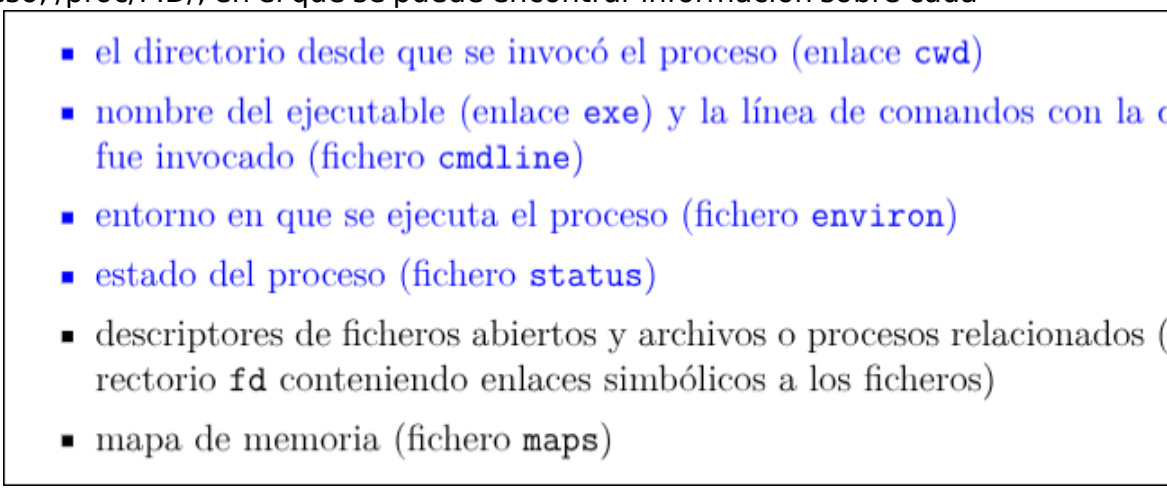
Son dos diectorios que guardan la información que recopila el S.O. - /proc ofrece información sobre los procesos (de ahí su nombre). También almacena todo tipo de información sobre los componentes principales del ordenador. - /sys ofrece información detallada sobre los dispositivos.

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:



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:



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 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
$ file fichero1
fichero1: PDF document, ve
$ file fichero2
fichero2: Microsoft Office
$ file fichero3
fichero3: PNG image data,
```

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

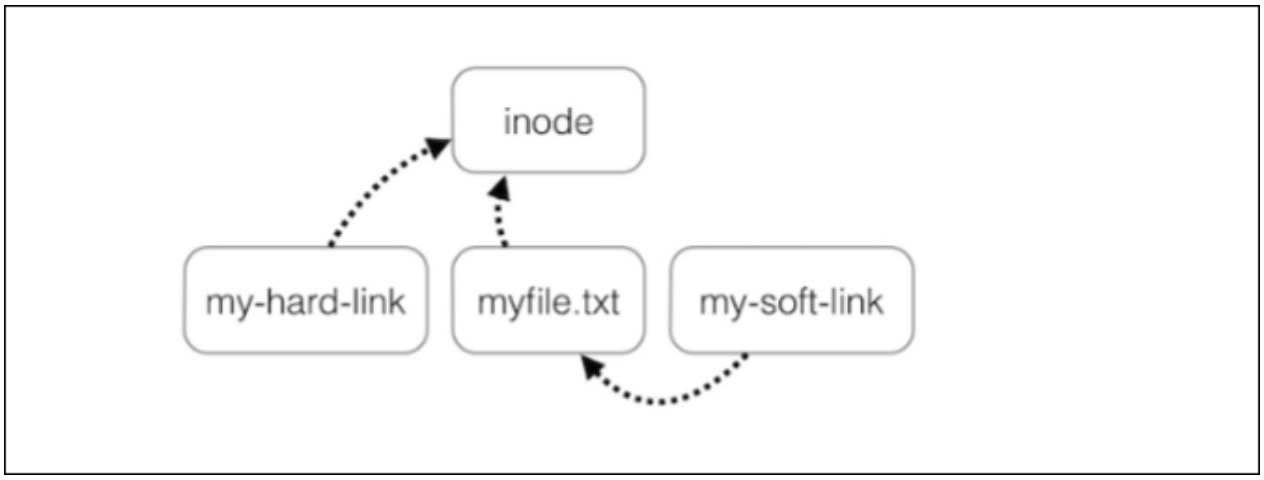


Figure 7: Pasted image 20251031110000.png

Comando `ln`

Permite crear enlaces - Formato `ln [-s] [opciones] destino [enlace]`

- por defecto crea enlaces dos; 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
```

Figure 8: Pasted image 20251031110501.png

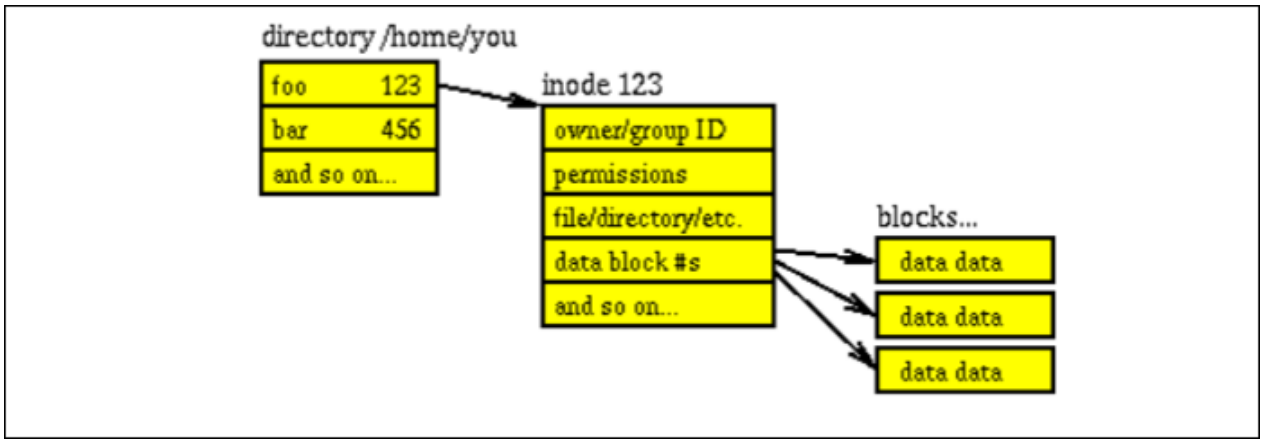
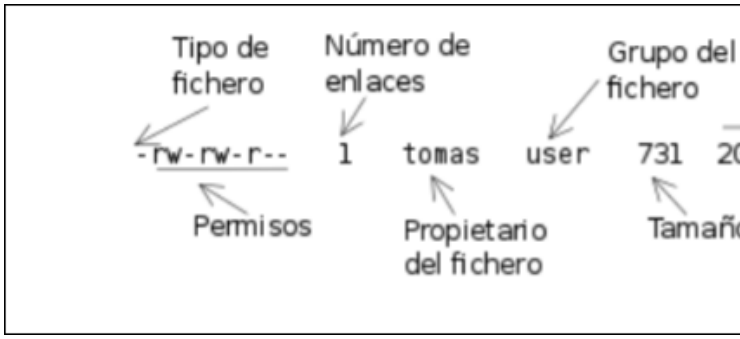


Figure 9: Pasted image 20251031110532.png

Atributos de un fichero



Podemos ver los atributos de un fichero con `ls -l`

Carácter	
-	fichero
d	director
l	enlace s
c	fichero
b	fichero
p	tubería
s	socket

Indicador de tipo el primer carácter nos indica el tipo de fichero

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

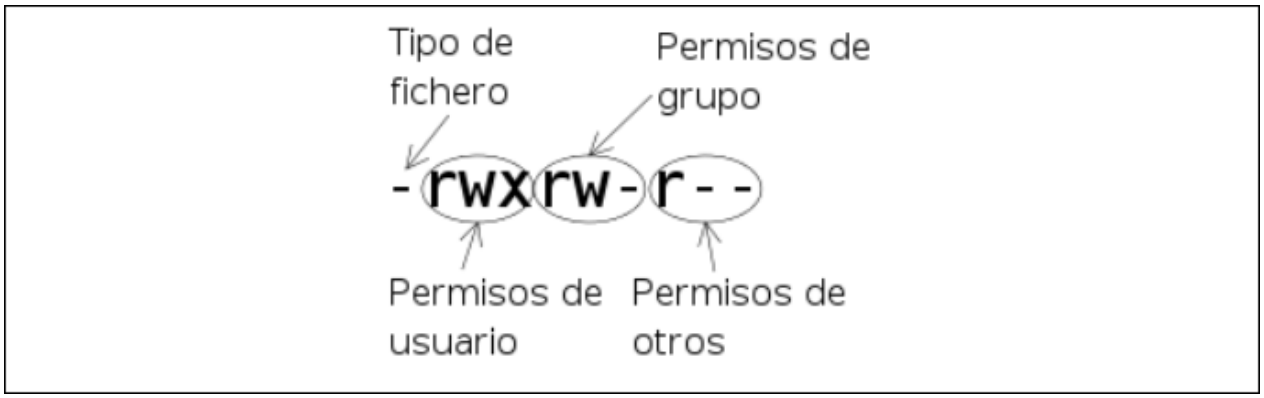


Figure 10: Pasted image 20251031172301.png

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`

- Permisos numéricos:

- `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

Figure 11: Pasted image 20251031173133.png

- `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

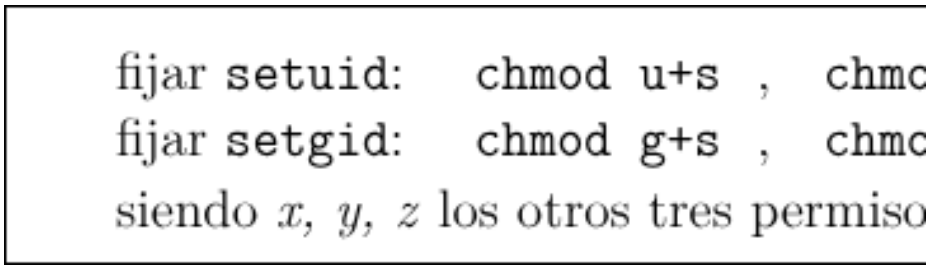
Ademas de `rwx` existen los permisos `setuid/setgid(s)` (relacionados con los atributos de los procesos) y `sticky bit (t)`.

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 ma- nipular todos los ficheros del sistema

Podemos usar ps para ver los RUID/RGID y EUID/EGID

- `ps -eo ruid,rgid,euid,egid,cmd`
- `ps -eo ruser,euser,rgroup,egro`

Los permisos `setuid/setgid`D permiten que un proceso lanzado por un usua- rio se eje-



cute con EUID/EGID de otro usuario.

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

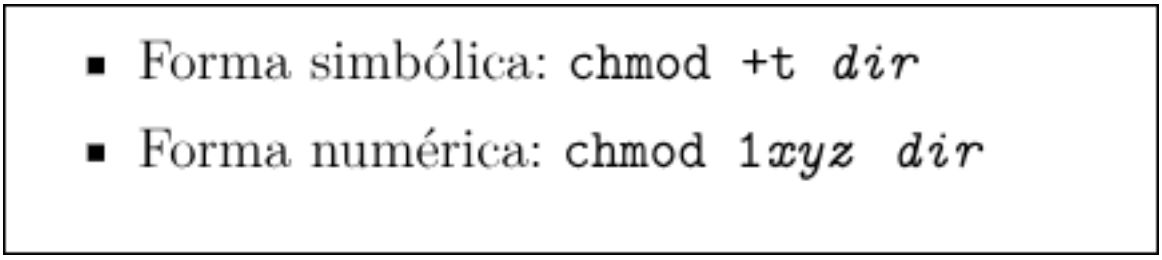


Figure 12: Pasted image 20251031174206.png

Cambio de usuario/grupoç

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

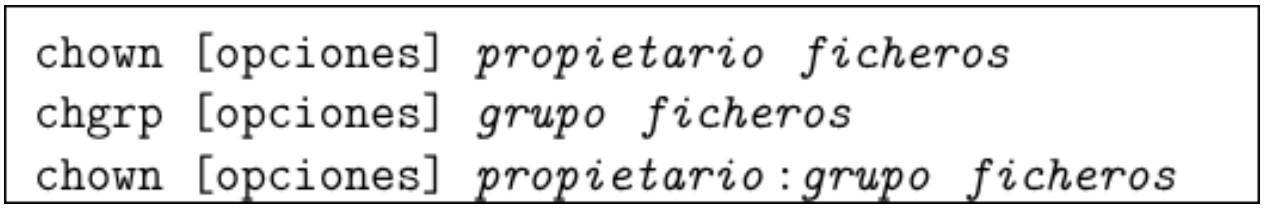


Figure 13: Pasted image 20251031174434.png

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]`

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

- 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
  ```

Figure 14: Pasted image 20251031175551.png

Criterios de búsqueda

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

- **Búsqueda por tamaño/permiso**

Criterio	Efecto
`-size` *n*`[ckMG]`	tamaño igual a *n* (`c`=ca
	`k`=kbytes, `M`=Mbytes, `G`=
`-size -`*n*`[ckMG]`	tamaño menor a *n*
`-size +`*n*`[ckMG]`	tamaño mayor que *n*
`-perm` *permisos*	permisos exactos
`-perm -`*permisos*	permisos indicados (si s
	permiso de lectura, no se
	misos de escritura y ejecu
`-perm /`*permisos*	para los permisos exactos
	ca que es suficiente que se
	los permisos entre usuari
`-perm /-`*permisos*	lo mismo para los permis

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

Figure 15: Pasted image 20251031175727.png

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*).

Figure 16: Pasted image 20251031175838.png

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

Figure 17: Pasted image 20251031175958.png

Criterio	Efect
`-atime [ /+/-]`*n*	busca ficheros cuya última f tura coincide (), es anterior *n* días
`-mtime [ /+/-]`*n*	lo mismo, pero con la fecha del fichero
`-ctime [ /+/-]`*n*	lo mismo, pero con la fech estado del fichero
`-amin/-mmin/` `-cmin [ /+/-]`*n*	lo mismo, pero ahora *n* rep
`-newer` *file*	busca ficheros modificados *file*
`-anewer` *file*	ficheros con último acceso modificación de *file*
`-cnewer` *file*	ficheros con cambio de esta modificación de *file*

- **Búsqueda por atributos temporales**

Operadores de `find`

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)

Figure 18: Pasted image 20251031180041.png

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`

Figure 19: Pasted image 20251031180102.png

Acciones de `find`

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`.

- `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`

Figure 20: Pasted image 20251031180124.png

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
`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

Figure 21: Pasted image 20251031181804.png

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 /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, defa
```

- `fdisk` se usa mediante un menú

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

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
```

`unmount` desmonta los sistemas de ficheros: - `unmount [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)
```

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

Figure 22: Pasted image 20251031182956.png

- 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 prim caso sólo el que lo monta puede desmontarlo)
  - `users` similar al anterior, pero puede montarlo/desmontarlo 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 especi- ficar el dispositivo:

```
$ mount /home
```

Fichero `/etc/mtab`. Contiene una lista de los filesystem que están monta- dos 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 su- perusuario):

```
$ 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
```

Figure 23: Pasted image 20251031183615.png

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 volú- menes: 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: - 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

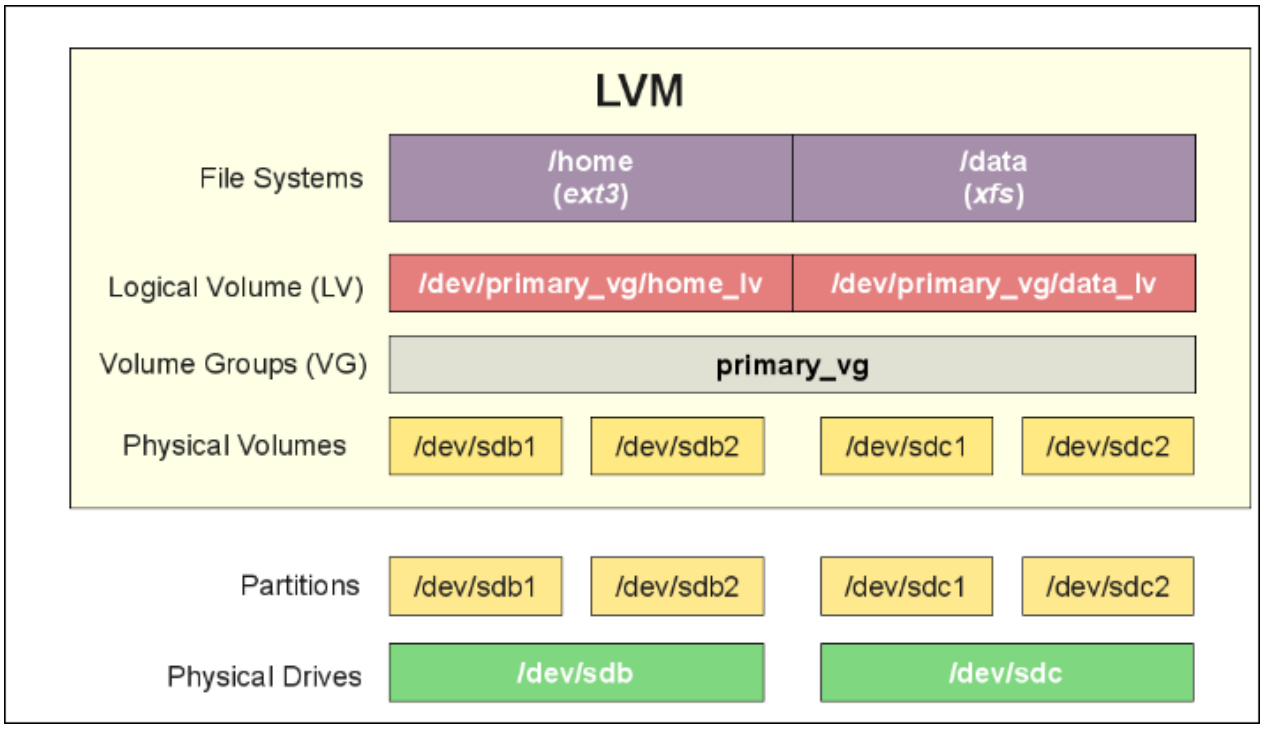


Figure 24: Pasted image 20251031200811.png

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).

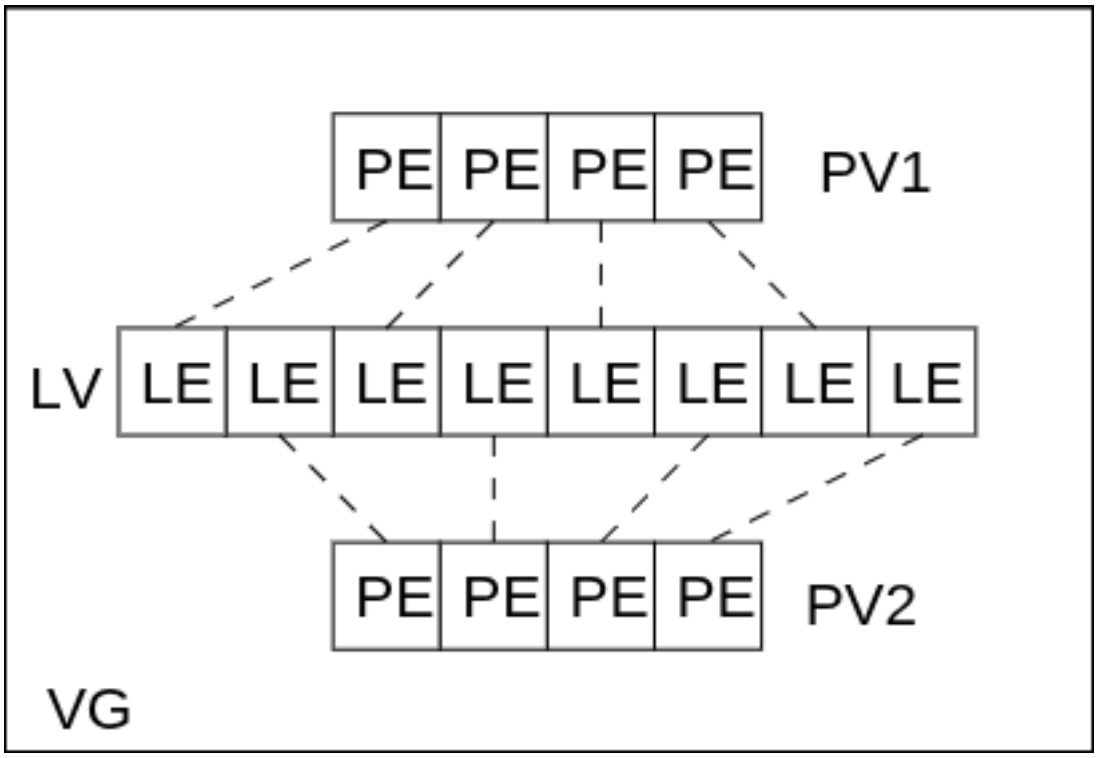


Figure 25: Pasted image 20251031201106.png

`pvdisplay` o `pvs`	información del volumen físico
`vgdisplay` o `vgs`	información del grupo de volúmenes
`lvdisplay` o `lvs`	información del volumen lógico
`pvcreate`	crear de un volumen físico
`vgcreate`	crear un grupo de volúmenes
`vgremove`	borrar un grupo de volúmenes
`vgextend`	añadir un volumen físico a un grupo de volúmenes
`vgreduce`	quitar un volumen físico de un grupo de volúmenes
`lvcreate`	crear un volumen lógico
`lvremove`	borrar un volumen lógico
`lvextend`	aumentar de tamaño un volumen lógico
`lvreduce`	reducir de tamaño un volumen lógico
`fsadm`	Comando genérico cambiar tamaño un filesystem
`resize2fs`, `btrfs`, `xfs_growfs`	Específicos para ext2/3/4, btrfs y XFS

Figure 26: Pasted image 20251031201205.png

- Información acerca de un volumen físico: pvdisplay o pvs

```
# pvdisplay /dev/sda2
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               GrupoVolu
  PV Size               9,88 GB /
  Allocatable           yes (but
  PE Size (KByte)       32768
  Total PE              316
  Free PE               0
  Allocated PE          316
  PV UUID               U6rMMw-52
```

- Información acerca de un grupo de volúmenes: vgdisplay o vgs

```
# vgdisplay GrupoVolumen
  --- Volume group ---
  VG Name               (
  System ID
  Format                1
  Metadata Areas        2
  Metadata Sequence No  7
  VG Access             r
  VG Status             r
  MAX LV                (
  Cur LV                (
  Open LV               (
```

- Información acerca de un volumen lógico: lvdisplay o lvs

```
# lvdisplay /dev/GrupoVolumen/home
  --- Logical volume ---
  LV Name               /dev/Grup
  VG Name               GrupoVolu
  LV UUID               dI6BqF-LA
  LV Write Access       read/writ
  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 (-l), o la diferencia (+/-): `vextend -L12G /dev/GrupoVolumen/homelv`, `vextend -L+1G /dev/GrupoVolumen/tmplv`, `vextend -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

Una vez creados los volúmenes lógicos hay que crear los sistemas de ficheros y montar-los 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
```

Figure 27: Pasted image 20251031202451.png

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