Listado de Órdenes en Linux

Parte 2: Administración

por Pablo Meca

Recopilación de los comandos estudiados en la asignatura “Introducción a Sistemas Operativos” de segundo curso del grado en Ingeniería en Informática y otras órdenes de interés.

Listado de Órdenes en Linux. Parte 2: Administración *por Pablo Meca*

Contenido

TEMA 5: Gestión de usuarios en Linux

Conceptos básicos
id [usuario]
newgrp
who
su [usuario]

Gestión de las cuentas de usuarios
El fichero /etc/passwd
chfn
chsh
El fichero /etc/shadow
useradd [opciones] usuario
usermod [opciones] usuarios
userdel [OPCION] usuario
finger
passwd [opciones] [<nombre_usuario>]
chage [opciones] usuario

Gestión de grupos de usuarios
/etc/group
/etc/gshadow
groupadd grupo
groupdel grupo
gpasswd opciones usuarios grupo

Otros aspectos sobre gestión de usuarios
.bash_profile
.bashrc
.bash_logout
/etc/skel

Propietarios y permisos
chown [opciones] [propietario][: [grupo]] ficheros
chgrp [opciones] grupo ficheros

Permisos especiales
- t (sticky bit): chmod +t directorio
- suid: chmod u+s fichero
- sgid: chmod g+s fichero
- sgid: chmod g+s directorio

Tema 6: Gestión de discos

Particiones

Particionamiento DOS
fdisk -l /dev/<nombre de la partición>
fdisk /dev/<nombre>

Posicionamiento GPT
Las particiones GPT se crean también con fdisk, usando la orden g

Listar dispositivos de bloques
lsblk [opciones]

loop

Sistemas LVM
pvcreate /dev/<dispositivo>
pvdisplay [/dev/<dispositivo>]
vgcreate <nombre del grupo> <volúmenes>
vgdisplay [OPCIONES] <nombre del grupo>

Volúmen lógico lineal
lvcreate OPCIONES <VG donde se crea>
lvdisplay [OPCIONES] /dev/<VG>/<nombre>

Volumen lógico repartido
lvcreate OPCIONES <VG donde se crea>

Volumen lógico reflejado
lvcreate OPCIONES <VG donde se crea>

Volumen lógico raid5
lvcreate OPCIONES <VG donde se crea>

Cambio de tamaño de un sistema LVM
lvextend OPCIÓN [+]<tamaño> /dev/<VG>/<LV>
lvreduce OPCIÓN [-]<tamaño> /dev/<VG>/<LV>
vgextend
vgreduce
pvmove

Eliminación de elementos en un sistema LVM
lvremove /dev/<VG>/<LV>
vgremove <VG>
pvremove <PV>

Dispositivos de bloques loop
dd if=<fichero_in> of=<fichero_out> [bs=<tamaño>] [count=<nº bloques a copiar>]
losetup OPCIONES [<fich>]

Tema 7: Administración de Sistemas de Ficheros en Linux

mkfs OPCION /dev/<dispositivo>
mkfs.FS /dev/<dispositivo>

Redimensionamiento de un sistema de ficheros
resize2fs [OPCIONES] /dev/<VG>/<LV> [TAMAÑO]

Montaje y desmontaje
mount [[[OPCIONES] FicheroEspecialBloque] PuntoMontaje]
umount PuntoMontaje|FicheroEspecialBloque
fuser [OPCIONES]

Fichero /etc/fstab
FicheroBloques|UUID=<UUID>|LABEL=<etiqueta> PuntoMontaje Tipo OPCIONES dump_f pass_num
e2label
findmnt [directorio|fich_especial_disp]

Control y gestión del espacio de un sistema de ficheros
df [OPCIONES]
du [OPCIONES] [Directorio]…

Listado de Órdenes en Linux. Parte 2: Administración *por Pablo Meca*

Cuotas de disco

quotacheck -nm <sistfich>
quotaon <sistfich>
setquota -u <username> <blocks_soft> <blocks_hard> <inodes_soft> <inodes_hard> <sistfich>
repquota <sistfich>
quota

Creación de enlaces

ln [OPCIONES] Fichero NombreEnlace

Copias de seguidad

tar [OPCIONES] [<salida.tar>] [<fichero>…]
cpio [OPCIONES] [copia_seguridad [fichero]]
rsync [OPCIONES] <directorio_transferir> <maquina_receptora>:<directorio_receptor>

Tema 8: Control de los recursos del sistema

Administración de paquetes

rpm [OPCIONES]
dnf [OPCION]

Control y gestión de la CPU

ps [OPCIONES]
pstree [OPCIONES]
PID
top

Número nice y prioridad de procesos

nice -<numero> <orden_a_ejecutar>
renice <nueva_prioridad> <pid>

Envío de señales a procesos

kill [-señal] <pid>…
killall [-señal] <orden>

Programación puntual de tareas

at [hora] [fecha]

Programación de tareas periódicas

crontab [OPCIONES]

Control y gestión de la memoria

free
vmstat [intervalo número]

Espacio para paginación

Tema 9: Gestión de la E/S

Ficheros especiales de dispositivo

mknod primerdiscoduro b 8 0
mknod
udev

Versión del núcleo

uname -r

Módulos del núcleo

Dependencias

Listado de Órdenes en Linux. Parte 2: Administración *por Pablo Meca*

Carga de módulos

Herramientas para la gestión de módulos

lsmod
modinfo <modulo>
insmod <ruta del fichero .ko.xz del módulo>
rmmod <nombre del módulo, tal cual lo muestra lsmod>
modprobe [OPCIONES] <nombre del módulo >
depmod -a

Los sistemas de ficheros proc y sysfs

lshw
lsusb [OPCIONES]
lspci [OPCION]

udev

Planificadores de disco

Cambio de planificador

cat /sys/block/sda/queue/scheduler
echo <planificador> > /sys/block/sda/queue/scheduler
elevator=<planificador>

Prioridad de la E/S de disco

ionice [OPCIONES] [<clase>]
iotop [OPCION]

TEMA 5: Gestión de usuarios en Linux

Conceptos básicos

Las características principales de un usuario son: su nombre, su identificador (UID) y los grupos a los que pertenece. Las características de un grupo son: su nombre y su identificador (GID). Para cada usuario podemos encontrar dos tipos de grupos: el Primario (guardado en /etc/passwd) y los Secundarios (guardados en /etc/group).

id [usuario]: muestra UID, GID y los grupos a los que pertenece el usuario pasado como parámetro. En caso de no pasarse ningún usuario, muestra la información del usuario actual. Cuando la llamamos proporcionando un nombre, la información mostrada se corresponde con el momento en el que el usuario accedió al sistema, los cambios posteriores no se verán reflejados.

newgrp: lanza un nuevo Shell que cambia el grupo activo de un usuario (inicialmente es su grupo primario). Al ejecutar la orden **exit** se vuelve al Shell original con el grupo primario. Si se ejecuta **id** con un nombre de usuario, la información no cambia.

Cuando un usuario crea un fichero, se establece como grupo propietario de este fichero el grupo activo del usuario en ese momento. Cuando un usuario solicita realizar una operación sobre un fichero del que no es propietario, se le permite realizar esta operación si alguno de los grupos a los que pertenece tiene permiso.

who: muestra un listado de los usuarios con sesiones abiertas, la terminal a la que están conectados, la fecha y hora y la IP.

su [usuario]: ejecuta un nuevo Shell con los UID y GID del nuevo usuario indicado, podemos trabajar en el sistema como si fuésemos ese usuario. Se sale con **exit**. Por defecto (nombre vacío) es root.

> Si se ejecuta sin opciones, la mayoría de variables de entorno no cambian, salvo HOME y SHELL. Si el usuario al que cambiamos es distinto del root, también cambian USER y LOGNAME.
>
> **-l** o solamente **-** : se cambian todas las variables de entorno, equivale a hacer un login nuevo.

Gestión de las cuentas de usuarios

La información de las cuentas de usuario se guarda en los ficheros **/etc/passwd** y **/etc/shadow**.

El fichero /etc/passwd

Tiene el siguiente formato: `name:password:uid:gid:gecos:home:shell`, donde:

password tradicionalmente guardaba la contraseña encriptada, aunque actualmente se hace en /etc/shadow por seguridad.

uid puede valer 0 para el/los root, 1-999 para los usuarios del sistema y a partir de 1000 para los usuarios normales.

gecos contiene información que identifica al usuario (nombre, teléfono, etc), se puede cambiar la información con:

chfn

home contiene la ruta absoluta al directorio de trabajo del usuario, cuando este inicia sesión se sitúa aquí.

shell contiene la ruta del intérprete de órdenes usado para ese usuario, el valor se puede cambiar con:

chsh. El intérprete se puede escoger de entre los que aparecen en el fichero **/etc/shells**, si se elimina un shell del fichero, queda como prohibido y ya no se puede elegir, aunque los usuarios que ya lo tenían asignado pueden seguir usándolo.

Si un usuario no tiene asignado ningún intérprete, se usa el shell por defecto: **/bin/sh**.

Si se desea que el usuario no pueda entrar al sistema, se le debe asignar **/bin/false** o **/sbin/nologin**.

Se puede indicar que el Shell de un usuario es un fichero ejecutable específico, convirtiéndolo en una **cuenta restringida**.

El fichero /etc/shadow

En este fichero se guardan las contraseñas encriptadas de los usuarios. Es un fichero más seguro que /etc/passwd porque sus permisos son ---------. También guarda información para establecer restricciones. Su formato es:

```
name:password:changed:minlife:maxlife:warn:inactive:expired:unused
```

Donde:

password es la contraseña del usuario. Puede ser una contraseña encriptada o '*', '!!', lo que significa que la cuenta está bloqueada y no se puede usar.

changed indica la fecha del último cambio de contraseña.

minlife indica el número de días que tienen que pasar antes de poder cambiar la contraseña de nuevo.

maxlife indica el número de días máximo que puede estar con la misma contraseña sin cambiarla.

warn señala con cuántos días de antelación se debe avisar a un usuario de que su contraseña va a expirar y debe cambiarla.

inactive indica el número de días durante los que una contraseña sigue siendo válida después de que haya expirado. Transcurrido este tiempo, la cuenta se deshabilita.

expired es la fecha en la que expira la cuenta y se deshabilita automáticamente.

Herramientas de gestión de usuarios

useradd [opciones] usuario: crea cuentas de usuario.

-d HOME_DIR: el nuevo usuario se creará utilizando **HOME_DIR** como directorio personal. Por defecto se crea un directorio personal usando el nombre del usuario.

-g GROUP: indica el grupo principal del usuario, debe existir previamente. Por defecto el grupo principal es uno nuevo con su mismo nombre o el grupo por defecto (depende de la configuración del sistema).

-G GROUP [, GROUP2, … [, GROUPN]]: Para indicar una lista de grupos secundarios.

-m: para forzar la creación del directorio personal en caso de que no exista previamente. Por defecto depende del sistema.

-s SHELL: para indicar el shell inicial del usuario.

-c COMMENT: para establecer la información que identifica al usuario, es la que aparece en **gecos**.

usermod [opciones] usuarios: se utiliza para modificar las características de una cuenta de usuario ya existente.

-g GROUP: el grupo principal del usuario.

-G GROUP [, GROUP2, …[, GROUPN]] [-a]: Para indicar una lista de grupos secundarios, si se quiere añadir al usuario a los grupos sin quitarlo de los actuales, usar la opción **-a**.

-s SHELL: para cambiar el shell inicial del usuario. También se puede realizar el cambio con **chsh**.

-c COMMENT: para cambiar la información que identifica al usuario, también se puede cambiar con **chfn**.

-o: permite usar UIDs duplicados.

userdel [OPCION] usuario: para borrar un usuario.

-r: elimina también su directorio personal y el buzón de correo.

finger: muestra la información del usuario en multilíneas formateadas.

passwd [opciones] [<nombre_usuario>]: cuando se crea un usuario nuevo sin asignarle contraseña, su cuenta está bloqueada (no puede entrar al sistema, pero sí ejecutar procesos programados). Esta orden sirve para asignar/cambiar una contraseña.

vacía: la usa cualquiera para cambiar su contraseña.

<nombre_usuario>: la usa el root para cambiar la contraseña al usuario indicado. La nueva contraseña se pedirá después de ejecutar la orden.

--stdin <nombre_usuario>: igual que la anterior, pero la contraseña se lee de la entrada estándar, por lo que se pueden usar tuberías.

-e <nombre_usuario>: la usa el root para obligar al usuario a cambiar su contraseña en su siguiente acceso.

-l <nombre_usuario>: la usa el root para bloquear la contraseña al usuario.

-u <nombre_usuario>: la usa el root para desbloquear la contraseña al usuario.

Antes de realizar el cambio, la orden comprueba que la contraseña elegida cumpla los parámetros de seguridad.

chage [opciones] usuario: establece restricciones de tiempo o envejecimiento respecto a la validez de una cuenta y su contraseña. Las restricciones se guardan en /etc/shadow, el fichero /etc/login.defs tiene los valores por defecto.

-M número: indica los días que puede estar un usuario sin cambiar su contraseña.

-W número: indica cuántos días antes de que expire la contraseña se le empezará a avisar al usuario.

-I número: indica el tiempo de inactividad, es decir, una vez expirada la contraseña, cuántos días tiene el usuario para cambiarla antes de que su cuenta se bloquee.

-E <fecha en formato aaaa-mm-dd>: indica cuándo expira la cuenta, independientemente de la validez de la contraseña.

-l usuario: para consultar la información.

Este comando puede ser interactivo si ponemos solamente el nombre del usuario. Es recomendable si vamos a aplicar varias cosas.

Gestión de grupos de usuarios

La información relativa a los grupos se almacena en dos ficheros:

/etc/group: guarda información relativa a los grupos definidos y los usuarios miembros.

```
name:x:gid:list_of_users
```

/etc/gshadow: guarda información relativa a la seguridad de los gruos (grupo, contraseña y relación administradores y miembros).

```
name:password:list_of_administrators:list_of_members
```

Órdenes:

groupadd grupo: crea un nuevo grupo.

groupdel grupo: elimina un grupo.

gpasswd opciones usuarios grupo: para administrar la información contenida en los ficheros. Los usuarios se escriben separados por comas, sin espacios (pepe,paco,luis).

Cada grupo puede tener usuarios administradores, miembros y una contraseña de acceso. Es el usuario root el que establece la lista de usuarios miembros y administradores de un grupo.

-A: define la lista de usuarios que serán administradores.

-M: define la lista de usuarios miembros del grupo.

Un administrador de grupo puede utilizar también esta orden para gestionar el grupo.

-a: añadir a un usuario al grupo.

-d: para sacar a un usuario del grupo.

Otros aspectos sobre gestión de usuarios

El fichero `/etc/issue` contiene un mensaje que se muestra cuando nos conectamos a un terminal antes de pedirnos el *login*. Se pueden incluir caracteres de control que se sustituirán por su valor correspondiente (p.ej: `\l` se sustituye por el nombre de la terminal de texto). El mensaje del día `/etc/motd` (inicialmente vacío) se muestra siempre que un usuario entra al sistema, se puede modificar para incluir mensajes recordatorios. Estos cambios no se aplican al abrir la consola en modo gráfico, porque, en realidad, no estamos haciendo un *login* en el sistema.

Ficheros de inicialización

Son guiones shell que tiene un usuario en su directorio `$HOME`:

> **`.bash_profile`**: se ejecuta automáticamente cuando un usuario inicia sesión (local o remotamente).
>
> **`.bashrc`**: se ejecuta automáticamente cuando un usuario abre un nuevo terminal, o incluso si desde dentro de un shell se inicia una nueva instancia.

Cada usuario también cuenta con un fichero de finalización en su `$HOME`:

> **`.bash_logout`**: se ejecuta automáticamente cuando el usuario sale del sistema.

Algunas de las labores que pueden llevar a cabo los ficheros de inicialización son:

- Crear o modificar variables de entorno que el usuario puede usar o necesitar.
- Establecer los alias (dar un nuevo nombre a una orden utilizada con unos determinados argumentos y opciones, p.ej: `alias llcol='ls -l –color=auto'`). Se suele realizar en `–bashrc` para que afecte únicamente al nuevo intérprete.

`/etc/skel` directorio en el que hay unas versiones por defecto de estos ficheros. Cuando se usa la orden `useradd` para crear un usuario, se copian todos los ficheros de `/etc/skel` al `$HOME` del nuevo usuario.

Frecuentemente se incluye una llamada para ejecutar `.bashrc` desde `.bash_profile` para tener, por ejemplo, todas las definiciones de alias disponibles desde el principio en todo el sistema. Tras iniciarse el sistema, el usuario podrá realizar modificaciones en su `.bashrc` que afectarán a los intérpretes de órdenes que abra desde ese momento.

Cuentas del sistema

Un usuario es una entidad que ejecuta programas o posee ficheros. Una cuenta del sistema ejecuta programas o posee ficheros para que determinados servicios funcionen, pero NO está asociada a una persona. Por ejemplo, la cuenta `apache` ejecuta los servicios necesarios de un servidor Web (en concreto, `httpd`).

Propietarios y permisos

EL permiso para acceder a los ficheros está organizado en dos grados de propiedad: usuario y grupo.

`chown [opciones] [propietario][: [grupo]] ficheros`: cambia el usuario propietario, solo puede ejecutar esta orden el superusuario.

`chgrp [opciones] grupo ficheros`: cambia el grupo propietario. El propietario puede ejecutar esta orden, pero tiene que pertenecer al nuevo grupo. También vale para un sistema de ficheros.

En ambos casos, con **`-R`** se hace de forma recursiva.

Además, en la orden **`chmod`** podemos referirnos a todos los permisos en octal, o de forma específica indicando el/los bloques (usuario (`u`), grupo (`g`), otros (`o`)), la operación (activar (`+`), desactivar (`–`)) y el permiso sobre el que actuar (`r`, `w`, `x`). Por ejemplo, `chmod –R g+w`

`apuntes` activa el permiso de escritura para los usuarios del grupo propietario del directorio apuntes, así como de todos los ficheros y subdirectorios que este contenga, recursivamente.

Permisos especiales

Nos permiten dar respuesta a ciertas situaciones, todos ellos se aplican sobre `chmod`:

- **t (sticky bit): `chmod +t directorio`**
  Si un usuario tiene permiso de escritura en el directorio, puede crear ficheros en su interior, pero solo puede borrar los que le pertenecen. Si un directorio no tiene activado este bit, puede borrar cualquier fichero (le pertenezca o no) teniendo el permiso de escritura.
- **suid: `chmod u+s fichero`**
  Al activarse sobre un fichero ejecutable, se produce un cambio de dominio a nivel de usuario, durante la ejecución el usuario efectivo del proceso es el propietario del fichero, no el usuario que lo ejecutó.
- **sgid: `chmod g+s fichero`**
  Al activarse sobre un fichero ejecutable, se produce un cambio de dominio a nivel de grupo, durante la ejecución, el grupo efectivo del proceso es el grupo propietario del fichero, no el grupo del usuario que lo crea.
- **sgid: `chmod g+s directorio`**
  Al activarse sobre un directorio, al crear un fichero en su interior, el grupo propietario del nuevo fichero es el grupo del directorio, no el activo del usuario que ejecuta la orden.

Tema 6: Gestión de discos

Los *ficheros especiales de bloques* sirven para dar nombre a los dispositivos de bloques y, especialmente, a los discos. Los ficheros especiales de bloques se distinguen del resto porque sus permisos comienzan por la letra `b`, se encuentran habitualmente en `/dev`.

Particiones

Normalmente un disco tiene al menos una partición; una partición es un conjunto de sectores consecutivos dentro de un disco. Al crear una partición, estamos creando un dispositivo de bloques dentro de otro ya existente.

Las particiones solo existen para nuestro sistema operativo, son componentes lógicos que el sistema gestiona mediante una estructura de datos que almacena en el propio disco.

Particionamiento DOS

Es el formato de particiones original de MS-DOS, se diseñó inicialmente para gestionar hasta cuatro particiones en un disco. Para poder aumentar este número, se dotó de una naturaleza especial a una de esas cuatro particiones, pasándose a denominar *partición extendida*, mientras que a las restantes particiones iniciales se les denominó *particiones primarias*. Una partición extendida no soporta un sistema de ficheros directamente, dentro de ella se pueden crear *particiones lógicas* (o *secundarias*) que son las formateables para contener un sistema de ficheros.

Para obtener información general sobre las particiones existentes en un dispositivo, usamos:

`fdisk -l /dev/<nombre de la partición>`, aquellas en las que ponga *extendida* se tratan como un nuevo disco, teniendo un nuevo MBR con su tabla de particiones.

Actualmente, el hueco hasta el sector 2048 se usa para contener un gestor de arranque. Los sectores se numeran desde 0, siendo 0 el primer sector del disco. Al número que recibe cada sector se le denomina LBA (*Logical Block Addressing*), y constituye su dirección. Este esquema de particionamiento requiere que el número de sector de un disco quepa en 32 bits, lo que nos limita a discos de 2TiB, ya que el sector suele tener un tamaño de 512 bytes.

Para gestionar las particiones usamos:

`fdisk /dev/<nombre>` (p.ej. `sdb`)

Cuando se llama a la orden aparece una sencilla guía para gestionarlas, en la primera línea podemos usar:

> **`n`**: para crear una nueva partición
>
> **`w`**: para guardar en disco.
>
> **`p`**: para ver la tabla de particiones.
>
> **`d`**: para borrar una partición.
>
> **`q`**: para salir sin guardar los cambios.
>
> **`t`**: para cambiar el tipo de una partición (las vemos todas con L, nos quedamos con el número de la izquierda y cerramos con q, luego escribimos ese número, p.ej. 30 para LVM).

El tamaño se puede indicar en bytes, así no hace falta hacer cálculos del número de sector.

El tipo de una partición por defecto es *Linux*, con código hexadecimal 83. El tipo indica lo que la partición *va a contener*, no *lo que contiene*. Hay que formatear cada partición antes de usarla. Otros tipos son *Linux swap*, código 82, cuando la partición se usa como espacio de intercambio de la memoria virtual, y *Linux LVM*, código 8e, cuando se va a usar como volumen físico de un grupo de volúmenes.

Posicionamiento GPT

El esquema GPT (*GUID Partition Tabl*e) permite discos mucho mayores, es posible hasta 8 ZiB y admite un número prácticamente ilimitado, aunque los sistemas operativos lo suelen limitar a 128 para mantener la tabla de particiones en un tamaño razonable de 16 KiB.

Se utiliza un GUID (*Globally Unique Identifier*) para identificar unívocamente un disco o partición. Es un número de 16 bytes (32 dígitos hexadecimales) dividido en 5 campos. Puede ser pseudoaleatorio en todos o algunos de sus campos. La idea es garantizar que sea altamente improbable tener dos recursos con el mismo identificador en el mismo sistema, aún cuando un recurso se mueva de un sistema a otro. GPT también utiliza un GUID para identificar el tipo de una partición, aunque ahora no es pseudoaleatorio, sino que tiene un valor predeterminado y conocido, habitualmente decidido por el fabricante/sistema operativo.

Las particiones GPT se crean también con **`fdisk`**, usando la orden **`g`**. El resto de órdenes son las mismas. La principal diferencia es que desaparece la distinción entre particiones, todas son iguales.

Listar dispositivos de bloques

`lsblk [opciones]`: lista los dispositivos de bloques disponibles y sus particiones. Sin argumentos muestra un listado en forma de árbol en la que los hijos de un nodo son los dispositivos que contiene.

> **`-p`**: muestra la ruta absoluta del fichero especial de bloque.
>
> **`-f`**: muestra información del sistema de ficheros contenido en cada dispositivo de bloques. Muestra el **`UUID`**.
>
> **`loop`**: permite ver las particiones del fichero.

Sistemas LVM

Para poder tener un almacenamiento flexible, surgen los gestores de volúmenes lógicos o sistemas LVM, que *virtualizan* el almacenamiento para crear discos lógicos y proporcionar métodos de asignación de espacio en disco más flexibles que las simples particiones.

Un gestor de volúmenes permite combinar particiones de distintos discos para crear particiones lógicas o virtuales que los administradores pueden mover o redimensionar mientras el sistema se sigue usando. También permiten crear dispositivos lógicos de bloques con más capacidad, mayor rendimiento y/o mayor fiabilidad que un simple disco.

Arquitectura de un sistema LVM

Un gestor de volúmenes parte de *volúmenes físicos* (PVs) que pueden ser discos enteros, particiones u otros dispositivos de bloques. Cada volumen físico está dividido en bloques llamados *extensiones físicas* (PE), todos del mismo tamaño (de forma predeterminada, 4 MiB).

Podemos convertir cualquier dispositivo de bloques en un volumen físico, para que pueda ser usado en un sistema LVM, mediante la orden:

pvcreate /dev/<dispositivo>

Una vez creado, podemos consultar las características de cada volumen físico con:

pvdisplay [/dev/<dispositivo>]

Si no indicamos un volumen dísico como parámetro, nos aparecerá la información de todos los dispositivos disponibles.

Los volúmenes físicos se agrupan en *grupos de volúmenes* (VGs), las extensiones físicas de todos ellos conforman el almacén o repositorio de extensiones físicas del grupo de volúmenes. Para crear un grupo de volúmenes:

vgcreate <nombre del grupo> <volúmenes>

Si creamos un grupo de volúmenes sin haber hecho antes un pvcreate para cada volúmen, vgcreate lo hará automáticamente. LVM usa 1 de las extensiones físicas para guardar metadatos, por lo que no estará disponible para nuestro uso. Una vez creado, podemos conocer sus características con:

vgdisplay [OPCIONES] <nombre del grupo>

> **vacío**: Muestra la información de todos los grupos de volúmenes existentes.
>
> **-v o --verbose:** permite ver qué volúmenes físicos forman parte de cada grupo de volúmenes.

Donde:

> **VG Size** señala el tamaño del grupo.
>
> **PE Size** señala el tamaño de cada extensión física.
>
> **Total PE** señala el número total de extensiones físicas.

Para poder usar el almacenamiento disponible en un grupo debemos crear dispositivos de bloques llamados *volúmenes lógicos* (LVs). Un LV es como cualquier otro dispositivo de bloques, por lo que se puede formatear para crear un sistema de ficheros de cualquier tipo, o para formar parte de otro grupo de volúmenes. Un LV también se divide en bloques llamados *extensiones lógicas* (LEs), todas del mismo tamaño, que coincide con el de las PEs.

Cuando se crea un volumen lógico, sus LEs se hacen corresponder con PEs libres. *Normalmente*, a cada extensión lógica le corresponde solo una extensión física, de manera que cualquier operación de L/E que se haga con una extensión lógica se terminará realizando realmente sobre su correspondiente extensión física.

Volúmen lógico lineal

A cada extensión lógica le corresponde una única extensión física. No hay control en cómo se seleccionan las extensiones físicas.

lvcreate OPCIONES <VG donde se crea>

Orden para crear un volumen lógico lineal.

> **-L <tamaño>**: indica el tamaño, p.ej: 100M son 100 MiB.
>
> **-n <nombre>**: indica el nombre.
>
> **-l**: indica el tamaño en extensiones lógicas. También permite indicar un incremento en función del espacio libre, ejemplos: un 10% del grupo de volúmenes (-l +10%VG), o un 10% del espacio libre en ese momento (-l +10%FREE).

lvdisplay [OPCIONES] /dev/<VG>/<nombre>

Orden para consultar las características del nuevo LV.

> **-m o --maps**: para ver también el *mapa* que muestra cómo se corresponden las LE con las PE.

Los volúmenes lógicos que creemos en un grupo se representarán mediante ficheros en /dev/<grupo>, que son enlaces simbólicos a ficheros especiales de bloques en /dev.

Volumen lógico repartido

También asigna a cada LE una única PE, pero ahora las PEs de dos LEs consecutivas se almacenan en PVs diferentes. Esto hace que el rendimiento de un LV sea mayor que en el de un único disco, ya que las L/E se realizan usando varios discos a la vez; pero también hace que su fiabilidad sea menor, ya que la probabilidad de que falle uin LV es aproximadamente la suma de las probabilidades de fallo de los discos individuales.

Para crearlo se utiliza la misma orden, pero añadiendo un comando:

lvcreate OPCIONES <VG donde se crea>

> **-i número**: número de volúmenes físicos que se van a usar (también se puede poner --stripes en vez de -i). Cada PV tendrá capacidad/número de espacio.
>
> El resto de opciones son las mismas.

Las extensiones de un volumen lógico repartido se dividen todavía en bloques más pequeños llamados ***porciones***. El *tamaño de porción* (o *stripe size*) indica cada cuántos bytes consecutivos (de forma predeterminada son 64 KiB) tenemos que cambiar de extensión lógica y, por tanto, de extensión física.

> **-I número**: cambia el *stripe size*.

Si al crear un LV repartido se le da un tamaño cuyo cociente no da LEs exactas, se redondea hacia arriba (si ponemos 300M, nos hace 304M, p.ej.).

Volumen lógico reflejado

A cada extensión lógica le corresponden dos o más PE, todas ellas con la misma información. Estas extensiones se almacenan en PV distintos pertenecientes a discos distintos, consiguiendo así tolerancia a fallos, ya que, si un PV falla y sus PE se pierden, la información de cada LE todavía está disponible en el resto de PE.

En cuanto al rendimiento, en un LV reflejado las escrituras tardan lo mismo que en un disco físico individual, ya que se tiene que hacer una copia en todos los discos, pero se mejora el rendimiento de las lecturas, ya que la lectura de una LE puede ir a un volumen físico en un disco y la de otra al otro volumen físico en otro disco.

`lvcreate OPCIONES <VG donde se crea>`

> **`-m número:`** número de copias adicionales (y no el total de PV). También se puede poner `--mirrors` en vez de `-m`.
>
> El resto de opciones son las mismas.

Para implementar este tipo de LV, el propio LVM de Linux usa volúmenes lógicos lineales.

Lo habitual es que un LV reflejado use solo dos LV en discos diferentes, pero nada impide usar más. Se mejora así la fiabilidad y el rendimiento de las lecturas, pero no se consigue aumentar ni la capacidad del volumen lógico ni el rendimiento.

Volumen lógico raid5

Funciona como un LV repartido al que se le añade paridad, de manera que, aunque un PV deje de funcionar, el LV seguirá funcionando. Este tipo de volúmenes **necesita al menos 3 PVs en dispositivos distintos**.

`lvcreate OPCIONES <VG donde se crea>`

> **`--type raid5`**
>
> **`-i número:`** número de volúmenes físicos que se van a usar, siempre se usarán **`N+1 PVs independientes.`**
>
> El resto de opciones son las mismas.

Se usa la mitad de LVs para guardar datos y paridad y la otra mitad para guardar los metadatos necesarios para este tipo de volúmenes lógicos. Respecto a las PE, se usa una para cada uno de los LV que almacenan metadatos.

Un raid5 ofrece tolerancia a fallos, mayor capacidad y mejor rendimiento. Un raid5 también utiliza porciones.

Cambio de tamaño de un sistema LVM

Los volúmenes lógicos se pueden agrandar concatenándoles nuevas extensiones lógicas, o encoger quitándoles extensiones lógicas cuyas extensiones físicas se liberan y ser devuelven al repositorio de extensiones libres dentro del VG. Al agrandar un LV, las nuevas LEs pueden estar asociadas a PE que no tienen por qué ser contiguas ni tienen por qué encontrarse en los mismos PV que las ya utilizadas. Esto permite a los LV crecer sin tener que mover las LEs que ya poseen. Es posible cambiar el tamaño de un LV mientras está en uso.

lvextend OPCIÓN [+]<tamaño> /dev/<VG>/<LV>

Cambia el tamaño de un LV, si se indica el +, se incrementa en el tamaño indicado, si no se indica, el nuevo tamaño del LV será exactamente el que indicamos, y no tamaño actual + tamaño indicado. El dispositivo debe haberse añadido al grupo primero usando vgextend.

-L: indica el tamaño en K, M, G, etc.

-l: indica el tamaño en extensiones lógicas. También permite indicar un incremento en función del espacio libre, ejemplos: un 10% del grupo de volúmenes (-l +10%VG), o un 10% del espacio libre en ese momento (-l +10%FREE).

-r: también redimensiona el sistema de ficheros del LV al mismo tiempo. Alternativamente se puede usar --resizefs. Solo para sistemas Ext[234], ReiserFS y XFS.

Una cosa es el tamaño de un LV y otra el tamaño del sistema de ficheros que pueda contener, si incrementamos el tamaño de un LV que contiene un sistema de ficheros, no redimensionamos este, el sistema seguirá teniendo el mismo tamaño y, por tanto, no aprovechará todo el espacio disponible en el LV.

lvreduce OPCIÓN [-]<tamaño> /dev/<VG>/<LV>

Reduce el tamaño de un LV eliminando las LEs del LV indicado, HAY QUE DESMONTARLO PRIMERO.

Las opciones son las mismas.

Si el LV contiene un sistema de ficheros, habrá que reducir primero el tamaño de este antes de reducir el del LV. Es posible aumentar el tamaño de un VG añadiendo más PV con la orden:

vgextend

O reducir su tamaño, eliminando volúmenes físicos, con la orden:

vgreduce, en este último caso, si se están usando PEs del volumen a eliminar, será necesario mover su contenido a otras PEs libres de otros PVs.

pvmove: mueve las PE de un volumen a otro (tienen que estar en el mismo grupo, si no, hacer vgextend). Normalmente, el gestor de LVs puede hacer este movimiento en vivo.

Eliminación de elementos en un sistema LVM

lvremove /dev/<VG>/<LV>: elimina el LV (siempre que no se esté usando).

vgremove <VG>: elimina el VG entero junto a todos los LV que haya dentro.

pvremove <PV>: para eliminar PV del sistema LVM. Tras esta orden, el dispositivo de bloques seguirá existiendo, aunque ya no se tendrá en cuenta como posible PV.

Dispositivos de bloques *loop*

Los dispositivos de bloques *loop* permiten lo contrario, que un fichero regular sea visto como un dispositivo de bloques. Esto permite experimentar con dispositivos de bloques aún cuando no tenemos discos o particiones disponibles en nuestro ordenador.

Creamos el fichero usando:

dd if=<fichero_in> of=<fichero_out> [bs=<tamaño>] [count=<n° bloques a copiar>]

> **if=**: indica el fichero de entrada, o de origen, del que queremos leer la información.
>
> **of=**: establece el fichero de salida, o destino, de la copia.
>
> **bs=**: fija el tamaño de los bloques a copiar del origen al destino.
>
> **count=**: indica cuántos de esos bloques queremos copiar.

Una vez creado el fichero, debemos asociarlo a un dispositivo *loop*, lo que hará que se cree también el fichero especial de bloques que lo represente:

losetup OPCIONES [<fich>]: siempre que queramos asociar el fichero al *loop* hay que usar -fP.

> **-f**: hace que se utilice el primer dispositivo *loop* libre. En lugar de este comando podemos escribir detrás de -P un dispositivo *loop* concreto, como /dev/loop0. fich es fichero_out aquí.
>
> **-P**: para que, si se crean particiones, también se creen los ficheros especiales de bloques que las representan y así poder trabajar con ellas. fich es fichero_out aquí.
>
> **-a**: muestra los dispositivos de loop actuales (no hay que poner ningún fich en este comando).
>
> **-d**: para eliminar el dispositivo *loop*. fich es el dispositivo que queremos eliminar aquí.
>
> **-D**: para eliminar todos los dispositivos *loop* (no hay que poner ningún fich en este comando).

Tema 7: Administración de Sistemas de Ficheros en Linux

mkfs OPCION /dev/<dispositivo>: sirve para crear un sistema de ficheros (formatear).

-t <FS>: tipo de sistema de ficheros a crear (ext4 o vfat).

Suele ser más cómodo directamente usar:

mkfs.FS /dev/<dispositivo>: integra directamente el sistema de ficheros, si tecleamos mkfs. y pulsamos TAB podemos ver una lista de todos los sistemas de ficheros disponibles.

Tras ejecutar esta orden, cualquier información almacenada en el dispositivo se perderá.

Redimensionamiento de un sistema de ficheros

A la hora de aumentar o reducir un LV, hay que hacer lo propio también con su sistema de ficheros para que ocupe el nuevo tamaño.

resize2fs [OPCIONES] /dev/<VG>/<LV> [TAMAÑO]: para sistemas Ext[234], lleva el sistema de ficheros al tamaño indicado, aumentándolo si se indica un tamaño mayor, o reduciéndolo en otro caso. Si no se indica ningún tamaño, el sistema de ficheros ocupará todo el espacio disponible en el dispositivo de bloques.

Para otros sistemas de ficheros hay que usar la orden adecuada (p.ej., ntfsresize para sistemas NTFS).

Una operación de expansión del sistema de ficheros se puede realizar incluso montado, pero para reducir el tamaño **hay que desmontarlo primero**.

Para el caso concreto de Ext[234], también se puede usar la opción r de lvextend y lvreduce.

Montaje y desmontaje

Después de crear un sistema de ficheros, hay que montarlo para habilitar el acceso al mismo. E Linux hay un único sistema de ficheros lógico del que cuelgan todos los dispositivos de almacenamiento disponibles en cierto momento. Al desmontar un sistema de ficheros, este deja de estar disponible directamente desde el punto de montaje del sistema de ficheros lógico global, quedando sus datos en estado consistente, oportunamente modificados y guardados en su dispositivo de almacenamiento secundario.

El punto de montaje es el directorio en el que se monta un sistema de ficheros. En principio, el sistema de ficheros puede ser montado en cualquier directorio, lo que trae como consecuencia que temporalmente no se pueda acceder al contenido que había en ese directorio.

El sistema de ficheros raíz donde se instala el SO siempre está montado en el directorio / y no se puede montar; durante el proceso de arranque, primero se monta el sistema de ficheros raíz y después el resto.

mount [[[OPCIONES] FicheroEspecialBloque] PuntoMontaje]

Para montar el sistema de ficheros.

Si se llama sin argumentos, muestra información general sobre todos los sistemas de ficheros que están montados.

El FicheroEspecialBloque suele ser /dev/sd[abcd][12…] y el PuntoMontaje puede ser cualquier sitio. **Se puede omitir el FicheroEspecialBloque si en el fichero /etc/fstab hay una línea para ese sistema de ficheros**, en este caso, basta con indicar el punto de montaje y se usarán las opciones predeterminadas.

-t tipo_fs: tipo de sistema de ficheros.

-r: montaje en modo solo lectura.

-w: montaje en modo lectura/escritura.

-o: opciones de montaje (ro, equivalente a -r; rw, equivalente a -w; remount, para poder cambiar algunas opciones de montaje; más opciones en la página siguiente).

-a: para montar todos los sitemas de ficheros indicados en /etc/fstab salvo los que ya estén montados o usen la opción noauto.

Si se hace mount usando una etiqueta, en lugar del directorio, hay que poner LABEL= delante.

umount PuntoMontaje|FicheroEspecialBloque: para desmontar un sistema de ficheros. Solo se puede desmontar si no está siendo utilizado (no está ocupado).

fuser [OPCIONES]: para saber qué ficheros se están usando y qué procesos los usan (f: fichero abierto, c: directorio de trabajo, e: ejecutando un fichero, etc).

-v: para indicarle que nos dé una salida detallada.

-m PuntoMontaje: el punto de montaje a considerar.

Ejemplo de uso: /sbin/fuser -mv /media/disk/

Fichero /etc/fstab

Contiene información sobre sistemas de ficheros a montar o disponibles, así como de las zonas de intercambio a activar. El formato de cada una de sus líneas de texto es:

FicheroBloques|UUID=<UUID>|LABEL=<etiqueta> PuntoMontaje Tipo OPCIONES dump_f pass_num

FicheroBloques|UUID=|LABEL=: fichero especial de bloques, UUID o una etiqueta, sirve para referirse al dispositivo. Aunque las etiquetas pueden ser más representativas que el UUID, también pueden repetirse y dar problemas, el UUID no.

PuntoMontaje: directorio que sirve de punto de montaje.

tipo: tipo de sistema de ficheros.

OPCIONES: las opciones para el montaje (separadas por comas y sin espacios). Son las mismas que pueden usarse en la opción -o de mount. Debajo se enumeran algunas.

dump_f: frecuencia para hacer una copia de seguridad del SF con la orden dump, actualmente no se usa este campo.

pass_num: en el tiempo de arranque, indica en qué orden hay que verificar los sistemas de ficheros (ejecutar fsck para comprobar su estado). Si vale 0 indica que no se chequea, si es 1 es que se chequea el primero (solo debe tener valor 1 el SF raíz), cualquier otro número señala el orden.

Posibles opciones de montaje de un sistema de ficheros:

- **rw**: lectura-escritura.
- **ro**: solo lectura.
- **suid**/**nosuid**: nosuid impide que los bits SETUID y SETGID tengan efecto en aquellos ficheros ejecutables del sistema de ficheros que los tengan activos.
- **auto**/**noauto**: determina si se monta o no automáticamente durante el arranque.
- **exec**/**noexec**: permite o no la ejecución de ficheros.
- **usrquota, grpquota**: permite o no las cuotas de usuario y de grupo. Esta opción solo está disponible si el sistema de ficheros soporta un mecanismo de cuotas.
- **users**: permite que un usuario normal pueda montar el sistema de ficheros y que cualquier usuario normal lo pueda desmontar. Esta opción implica que, cuando un usuario normal decida montar el sistema de ficheros, las opciones de montajes sean noexec, nosuid y **nodev** (que prohíbe el acceso a cualquier fichero especial de dispositivo que pueda haber en el sistema de ficheros). Si se quiere combinar esta orden con otra (p.ej.: exec), primero debe ponerse esta y luego la otra, si no, la otra quedaría anulada.
- **defaults**: marca las opciones de montaje por defecto, principalmente se incluyen rw, suid, exec, auto, nouser y dev (que permite el acceso como tales a los ficheros especiales de dispositivo que pueda haber en el sistema de ficheros).
- **uid=X, gid=Y**: solo sirve en sistemas **vfat**, define el propietario (x) y el grupo propietario (y) de los ficheros del sistema de ficheros.
- **umask=XXX**: solo sirve en sistemas **vfat**, establece los permisos que los directorios y ficheros, existentes o que se creen en el sistema de ficheros, van a tener mientras dicho sistema de ficheros esté montado. Lo indicado en el campo umask es el inverso de los permisos a establecer (137 son permisos 640). Se podría establecer esta máscara únicamente para directorios con la opción dmask, o solo para ficheros ordinarios con fmask.

Podemos utilizar etiquetas para referirnos al sistema de ficheros dentro de un dispositivo de bloques. Si el sistema es Ext[234] usamos:

e2label, si es vfat, fatlabel, y si es NTFS, ntfslabel. Ponemos primero el dispositivo y luego la etiqueta que queramos utilizar.

Tras el montaje de un sistema de ficheros de tipo Ext[234], el punto de montaje tomará los permisos, propietario y grupo del directorio raíz del propio sistema de ficheros que se monta

(normalmente el propietario y el grupo será el `root`). Una vez montado, podemos modificar dicha información para permitir el acceso a otros usuarios.

Durante el arranque del sistema, se leen del fichero `/etc/fstab` los sistemas de ficheros a montar y las opciones de montaje, pero solo se montarán aquellos que tengan la opción `auto`.

findmnt [directorio|fich_especial_disp]: muestra información en forma de árbol de los sistemas de ficheros que están montados, alternativa a la llamada a mount sin argumentos. Si se le pasa un directorio, indicará si hay algún sistema de ficheros montado en él. Si se le pasa un fichero especial de dispositivo, informará sobre si está o no montado, y dónde en caso afirmativo.

Control y gestión del espacio de un sistema de ficheros

df [OPCIONES]: informa de la capacidad de un sistema de ficheros, del espacio libre y el usado, y de su punto de montaje. Por defecto, la primera columna muestra el nombre de la partición tal y como aparece en ele directorio /dev. Las columnas siguientes muestran el espacio total, bloques asignados y bloques disponibles.

> **-h:** muestra los tamaños en formato **más legible** (KiB, MiB, GiB…).
>
> **-i:** lista información de nodos-i en vez de uso de bloques.
>
> **-T:** muestra el tipo de sistema de ficheros.

Es importante recalcar que el sistema de ficheros raíz no debe quedarse sin espacio o sin nodos-i libres, ya que, de ocurrir esto, el sistema tendría problemas para funcionar, hasta el punto de no poder arrancar.

du [OPCIONES] [Directorio]…: se usa para informar acerca del espacio en disco que ocupa un fichero o directorio. Si no se indica directorio, por defecto es el actual. Si se indica un directorio, lista también el espacio ocupado por todos los subdirectorios encontrados a partir de él. Se muestra, por omisión, en unidades de 1 KiB.

Si se utiliza sin opciones informa del tamaño de cada subdirectorio de cada uno de los directorios especificados, así como el tamaño total del directorio.

> **-a:** cuando se indica un directorio, muestra también el uso de espacio de cada fichero.
>
> **-s:** en vez de las salida por defecto, informa solo del tamaño total ocupado por los directorios especificados (si se indican ficheros, el resultado de la salida no cambia con esta opción).
>
> **-h:** muestra los tamaños en un formato más legible (KiB, MiB, GiB).

Cuotas de disco

Aquellos sistemas de ficheros que tienen soporte para cuotas permiten establecer ciertos límites a cada usuario sobre la cantidad de espacio que puede utilizar. Los sistemas de ficheros Ext[234] permiten limitar el número de bloques y/o el número de nodos-i que un usuario puede usar en una partición. Estos límites también se pueden establecer para grupos de usuarios.

- **Límite hard**: el usuario no puede sobrepasarlo. Cuando lo alcance ya no podrá usar más bloques de datos o crear más ficheros.
- **Límite soft**: se puede sobrepasar durante cierto tiempo, pero sin llegar al límite hard. Pasado ese tiempo es como si se hubiese superado el límite hard. El **periodo de gracia** es ese tiempo durante el que se puede sobrepasar el límite soft. En este periodo se informa al usuario de que ha superado el límite y que debe liberar espacio o nodos-i, según el caso.

Los límites se establecen de forma independiente para bloques y nodos-i, solo tienen sentido en sistemas de ficheros donde pueden escribir los usuarios.

Pasos a seguir para activar cuotas para los usuarios en el sistema de ficheros contenido en `/dev/sdb1` y con la etiqueta `nuevodisco`, el sistema se encuentra ya montado en /media/disk:

1. Activar la opción `usrquota` en el sistema de ficheros al que se le pretenden asignar, por ejemplo, añadiéndola en `/etc/fstab`:
   `LABEL=nuevodisco /media/disk ext4 defaults,`**`usrquota`** `1 0`

   También podríamos hacerlo directamente en el paso siguiente haciendo `remount,usrquota`.
2. Remontar la partición para que se active la opción `usrquota`:
   `mount -o remount <sistfich>`
3. Crear el fichero de control de cuotas, **`aquota.user`**, con **`quotacheck`**:

   `quotacheck -nm <sistfich>`

4. Activar las cuotas

   `quotaon <sistfich>`

5. Para cada usuario, establecer su cuota

   `setquota -u <username> <blocks_soft> <blocks_hard> <inodes_soft> <inodes_hard> <sistfich>`

   Si queremos dejar algún límite sin establecer, usamos el valor 0. Por defecto los valores de los bloques se interpretan como número de bloques de 1 KiB, podemos especificar los tamaños añadiendo K, M, G y T. Los valores de los nodos-i son interpretados de forma **literal**, pudiendo añadir también los símbolos k, m, g y t (minúsculas).
6. Establecer el periodo de gracia (**en segundos**) para bloques y nodos-i a través de la opción `-t` de la orden `setquota`:
   `setquota -t <blocks_grace> <inodes_grace> <sistfich>`
   El período aplica a todos los usuarios, no se puede especificar uno.

A través de la orden:

repquota <sistfich> el administrador puede obtener una estadística de las cuotas para todos los usuarios. Un usuario puede utilizar la orden:

quota (sin parámetros) para obtener información acerca de las cuotas que le han sido establecidas sobre distintos sistemas de ficheros.

Creación de enlaces

Un enlace es una conexión entre un nombre de fichero y el contenido del mismo. En Linux podemos manejar dos tipos de enlaces: físicos y simbólicos. El primer tipo está referido a a conexión entre una entrada y el nodo-i correspondiente, el segundo se refiere a ficheros especiales cuyo contenido es la ruta de acceso a fichero que se enlaza.

ln [OPCIONES] Fichero NombreEnlace: sirve para crear los enlaces. Si se llama sin opciones, creará un **enlace físico**, que contiene el mismo número de nodo-i que /bin/ls. La segunda columna nos muestra el número de enlaces a dicho nodo-i.

> **-s:** para crear un enlace simbólico. Otra alternativa es escribir **--symbolic**. A la hora de hacer un ls -l, podemos ver que el enlace simbólico está marcado con una **l** delante de los permisos. El contenido es simplemente la ruta al fichero que hemos indicado.

Copias de seguidad

Pueden ser:

- Completas: se copian todos los ficheros del sistema de ficheros, tarda mucho tiempo y la recuperación de un único fichero puede no ser inmediata.
- Diferenciales: copia únicamente los ficheros que han sido creados y/o modificados desde la última copia completa. Esto es, una copia diferencial después de otra diferencial tomará la misma base, la completa, no la diferencial que le precede.
- Incrementales: se copian solo aquellos ficheros que han cambiado o se han creado desde la última copia de seguridad (del tipo que sea). Se usa en conjunto con la primera, así se tiene una copia completa realizada cada mucho tiempo, y muchas copias incrementales para los cambios pequeños hechas con mucha frecuencia.

Normalmente, las copias diferenciales ocupan más espacio que las incrementales debido a que parten de la base de un único punto fijo en el tiempo.

Se realizan utilizando la orden dd de forma similar a los dispositivos loop, p.ej., si /dev/sde1 es la ruta de un pedrive, podemos hacer la copia de seguridad y guardarla en respaldo-pendrive.img:

```
# dd if=/dev/sde1 of=respaldo-pendrive.img status=progress
```

Luego podríamos acceder al contenido de la copia de seguridad a través de un dispositivo *loop*:

```
# losetup /dev/loop0 respaldo-pendrive.img
# mount /dev/loop0 /media/disk
```

También podríamos recuperar la copia de seguridad en una partición de otro pendrive, siempre que la partición destino (/dev/sdf1 en este caso) tenga al menos el mismo tamaño que el fichero imagen:

```
# dd if=respaldo-pendrive.img of=/dev/sdf1 status=progress
```

dd realiza una copia *a ciegas*, sin importar si el sistema de ficheros está casi vacío, por lo que es bastante lento.

tar [OPCIONES] [<salida.tar>] [<fichero>…]: Realiza copias de seguridad de ficheros, directorios (de forma recursiva) o sistemas de ficheros. Por defecto solo empaqueta, pero también puede comprimir. Por defecto hace copias completas, para hacer copias diferenciales o incrementales se pueden usar las opciones:

> **--newer=DATE-OR-FILE, --after-date=DATE-OR-FILE, -N:** compara las fechas usando **ctime** (fecha de último cambio de estado) con la fecha de la última copia de seguridad completa o incremental realizada.
>
> **--newer-mtime=DATE-OR-FILE:** compara las fechas usando **mtime** (fecha de modificación) con la fecha de la última copia de seguridad completa o incremental realizada.
>
> **c:** crea un fichero contenedor.
>
> **x <salida.tar>:** extrae ficheros de un fichero contenedor. A la hora de recuperar un fichero en específico, hay que indicar la ruta con la que tar lo almacenó, no basta con el nombre.
>
> **t <salida.tar>:** muestra una tabla de contenidos, es decir, lista los ficheros almacenados en un fichero contenedor.
>
> **r <salida.tar> <fichero>…:** añade ficheros al final de un fichero contenedor (siempre que no se use compresión).
>
> **h:** en caso de un enlace simbólico, incluye el fichero al que apunta.
>
> **v:** modo verboso (muestra la información detallada).
>
> **f <salida.tar>:** permite especificar el nombre del fichero contenedor.
>
> **z:** comprime o descomprime mediante gzip.
>
> **p:** conserva los permisos de los ficheros.
>
> **P:** guarda los ficheros con ruta absoluta.

cpio [OPCIONES] [copia_seguridad [fichero]]: realiza copias de seguridad de conjuntos de ficheros. **Lee de la entrada estándar**, por lo que normalmente se usa enlazada con otras órdenes mediante tuberías, especialmente find. Guarda los nombres de los ficheros tal cual se le pasan (ruta absoluta o relativa), y así es como se restauran después.

Se pueden hacer copias diferenciales o incrementales con la opción -newer de find.

-o: para copiar hacia afuera, **crear** la copia de seguridad.

-i: para copiar hacia adentro, **desempaquetar** una copia de seguridad. Se puede indicar la ruta de ficheros concretos (relativa o absoluta, depende de cómo se guardasen) si solo se quiere extraer esos.

-t: para crear una tabla de contenidos, **mostrar** el contenido de la copia.

-A <contenedor> <fichero>…: para **añadir** nuevos ficheros a un contenedor ya existente.

-d: para crear **directorios** al desempaquetar.

-m: para **conservar fecha y hora** de los ficheros al desempaquetar.

--no-absolute-filename: si se creó el respaldo usando rutas absolutas, al desempaquetar extrae los ficheros relativos al directorio actual.

-v: modo verboso (detallado).

-F <nombre>: nombre del fichero a usar como contenedor de la copia de seguridad. Se puede reemplazar por los operadores **<** o **>**, dependiendo de si el flujo sale o entra.

rsync [OPCIONES] <directorio_transferir> <maquina_receptora>:<directorio_receptor>: permite realizar copias de ficheros de manera local en una máquina o bien hacia/desde otro ordenador (sshd debe estar activo en la máquina remota). Utiliza un algoritmo que reduce la cantidad de datos que hay que transmitir por la red de interconexión ya que realiza el envío únicamente de las diferencias que encuentra entre los ficheros fuente y los ficheros que existan en el destino.

La orden *sincroniza* el contenido del directorio destino con el de origen, tras la ejecución, el directorio destino tendrá, al menos, los mismos ficheros que el directorio origen, incluyendo contenido de cualquier subdirectorio existente.

> **-a**: modo archivo, hace un recorrido recursivo para buscar los ficheros a copiar y preserva las características más importantes de los mismos.
>
> **-v**: modo verboso, informa de los ficheros que se van transmitiendo.
>
> **-z**: los datos a enviar son comprimidos durante la comunicación para reducir el tiempo de transmisión al destino.
>
> **-n**: no realiza ninguna operación de copia o sincronización. Combinada con -v puede ser muy útil para ver el efecto que produciría la ejecución de la orden.
>
> **--delete**: si hemos borrado un fichero del directorio origen, cuando se sincronice con el directorio remoto se realizará también el borrado del fichero en el destino.
>
> **--backup --backup-dir=DIR**: para tener en el ordenador remoto el directorio de backup incremental DIR, donde se guardarán las versiones antiguas de los ficheros reemplazados y/o eliminados al sincronizar.

Si ponemos la máquina receptora como *localhost*, el ordenador *remoto* será nuestro propio equipo.

Tema 8: Control de los recursos del sistema

Administración de paquetes

Un paquete se encuentra en un fichero comprimido, conteniendo información del producto, ficheros de programa, iconos, documentación y scripts de administración. Las aplicaciones de administración utilizan estos ficheros para ubicar, instalar, actualizar y eliminar software de una forma segura.

Un paquete puede ser compilable si se basa en código fuente, o instalable si lo hace en binarios compilados ya para una determinada arquitectura o plataforma.

El sistema de paquetes utilizado por Fedora es RPM, heredado de Red Hat.

Normalmente el nombre de fichero de un paquete tiene este formato: [nombrepaquete-versión-revisión.[*arquitectura* | src | noarch].rmp]. La arquitectura hardware señalada por este nombre de fichero es el mínimo tipo de máquina requerida para utilizar el paquete, posibles arquitecturas:

- `noarch.rpm`: Apropiado para cualquier tipo de arquitectura.
- `i386.rpm`: Apropiado para cualquier procesador compatible con Intel.
- `i586.rpm`: Apropiado para cualquier procesador compatible con Intel Pentium y superior.
- `i686.rpm`: Apropiado para cualquier procesador compatible con Intel Pentium Pro y superior.
- `x86 64.rpm`: Apropiado para cualquier procesador compatible con Intel de 64-bit

Un fichero src.rpm contiene código fuente, por lo que tendrá que ser compilado usando la orden `rpmbuild` para poder hacer uso de este.

Una vez instalado, podemos referenciar a un paquete usando simplemente su nombre de paquete, no será necesario indicar ni siquiera la versión. Pueden existir dependencias entre paquetes, de forma que para instalar un paquete sea necesario instalar antes otros paquetes, o que al querer desinstalar, no se pueda porque haya dependencias con otros que ya hay instalados. Estas dependencias son controladas por el sistema RPM.

`rpm [OPCIONES]:` `proporciona la funcionalidad básica para la instalación, consulta y eliminación de paquetes.`

> **`-i <ficheropaquete.rpm>:`** `instala un paquete. Si se intenta instalar un paquete que necesita de otro, la operación no se realiza y devuelve un mensaje de error indicando la dependencia.`
>
> **`-U <ficheropaquete.rpm>:`** `actualiza un paquete. En caso de que exista un fichero de configuración modificado de la versión anterior, lo guarda como <ficheropaqueteanterior.rpmsave>. Si un fichero de configuración fue modificado, y el nuevo fichero de configuración ha cambiado de formato y no puede ser adaptado, se deja el fichero antiguo y se crea uno nuevo que se llamará <ficheropaquete.rpmnew>. El administrador debe adaptar el fichero al nuevo formato.`

-e <paquete>: elimina el paquete. Si se intenta borrar un paquete que es necesario para algún otro paquete, la operación no se realizará y se recibirá un mensaje de error indicando dichas dependencias.

-q [OPCIONES]: es una opción de consulta, se le adjuntan subopciones correspondientes a lo que queremos consultar (-i para información general de un paquete, -l para ver el listado de ficheros que lo componen, -f para saber a qué paquete pertenece un fichero…).

-V <paquete>: para verificar el estado de un paquete.

dnf [OPCION]: permite actualizar de forma automática o interactiva paquetes. A diferencia de rpm, cuenta con resolución automática de dependencias, de forma que descarga e instala los paquetes extras necesarios durante la instalación de un paquete. A la hora de desinstalar un paquete, también desinstala todos los paquetes que dependan del paquete que se ha ordenado desinstalar de cara a no dejarlos inconsistentes.

check-update: avisa de los paquetes con actualizaciones pendientes.

update <paquete>: para actualizar un paquete (si existe actualización).

update: para actualizar todos los paquetes actualizables.

install <paquete1>…: descarga e instala la última versión del paquete/s indicado/s.

download <paquete1>…: descarga la última versión del paquete/s indicado/s, pero sin llegar a instalarlo/s.

remove/erase <paquete1>…: ambas sirven para eliminar el paquete/s que se indica.

clean [packages | headers | all]: permite limpiar los ficheros que se acumulan en el directorio caché de dnf.

-h: para obtener más información.

`dnf` usa como fichero de configuración `/etc/dnf/dnf.conf`. Para saber de dónde descargar los paquetes parte del directorio `/etc/yum.repos.d/`, que contiene varios repositorios repo, que son ficheros que se dividen en tres secciones, una para los paquetes normales, otra para los paquetes de depuración y una tercera para los paquetes fuentes. generalmente habrá varias copias de un paquete de distribución disponible en varias ubicaciones (o espejos) denominados repositorios. El fichero repo le informa a `dnf` dónde puede encontrar la última lista de espejos para cada sección.

Podemos añadir nuevos ficheros repo de cara a poder instalar algún paquete que no está recogido en ninguno de los repositorios que tenemos en un momento dado, esto lo podemos hacer instalando el repositorio con `dnf install`.

`dnf` también puede simplemente instalar paquetes previamente descargados, con la ventaja sobre rpm de que añade la resolución automática de dependencias. Hay que pasarle como argumento una ruta hasta ese fichero de paquete que queremos instalar.

Control y gestión de la CPU

Hay tres juegos de opciones diferentes:

- Opciones de BSD: Estas opciones de un único carácter se pueden agrupar y no van precedidas por ningún guión.
- Opciones de Unix98: Estas opciones de un único carácter se pueden agrupar y van precedidas de un único guión.
- Opciones GNU largas: Estas son opciones multi carácter y nunca van juntas. Están precedidas por dos guiones.

ps [OPCIONES]: muestra información de la actividad de los procesos en ejecución. Si no se añade ninguna opción ni parámetro, mostrará únicamente los procesos iniciados por el usuario desde la consola actual. Admite los tres tipos de opciones, incluso mezclarlas, aunque es desaconsejable. Usaremos opciones BSD:

> **a:** para listar procesos de todos los usuarios. Si no está acompañada de otras opciones, se mostrarán únicamente los procesos con terminal asociada.
>
> **x:** para listar todos los procesos, tengan o no una terminal asociada. Si no está acompañada de otras opciones, se mostrarán únicamente los procesos del usuario que ejecuta la orden.
>
> **u:** para mostrar la información más importante de cada proceso.
>
> **l:** si se usa en lugar de la u, podremos ver más campos de interés como el identificador de proceso padre (**PPID**), el **UID** y los valores de los campos de prioridad (**PRI** y **NI**). El campo **F** es un campo de flags (su valor es la suma de 1 si el proceso ha nacido en un fork, pero no ha realizado un exec, y 4 si es un proceso con privilegios de root), y el campo **WCHAN** muestra la dirección de la función del núcleo de aquellos procesos que están bloqueados en alguna.
>
> **o <campo>…:** si se usa en lugar de u, muestra solamente los campos que hemos solicitado (escribirlos en minúsculas).

Si ejecutamos `# ps axu`, los campos son:

> **USER:** el usuario efectivo del proceso.
>
> **PID:** identificador del proceso.
>
> **%CPU:** porcentaje de utilización de la CPU por parte del proceso. Se calcula con el tiempo de CPU transcurrido desde que inició el proceso dividido por el tiempo en el que el proceso ha estado ejecutándose, expresado en porcentaje.
>
> **%MEM:** fracción de memoria RAM consumida (es una estimación).

VSZ: tamaño de la memoria virtual usada por el proceso (suma de los tamaños de las regiones del mapa de memoria del proceso), en KiB.

RSS: memoria RAM usada, en KiB.

TTY: terminal asociado con el proceso.

STAT: estado del proceso:

- R: en ejecución o listo para ejecutarse.
- N: prioridad baja (valor mayor que 0).
- <: prioridad alta (valor menor que 0).
- T: parado por una señal de stop.
- t: parado por el depurador.
- Z: proceso zombie (ha completado su ejecución, pero aún tiene una entrada en la tabla de procesos, estando a la espera de que el proceso padre ejecute wait).
- S: durmiendo (bloqueado).
- D: durmiendo (bloqueado) ininterrumpiblemente, el proceso está realizando una llamada al sistema, no pudiendo ser interrumpido (o eliminado) hasta que esta llamada al sistema se complete.
- l: el proceso tiene varios hilos.
- +: proceso ejecutándose en primer plano.
- s: proceso líder de sesión.
- I: hilo del núcleo desocupado.

START: momento en que se inició el proceso. Hace menos de 24 horas, el formato en que se muestra es HH:MM (hora y minuto), en otro caso, será MmmDD (iniciales del mes y el día).

TIME: tiempo acumulado de uso de CPU.

COMMAND: nombre del proceso.

pstree [OPCIONES]: muestra los procesos del sistema en una estructura de árbol útil para ver las relaciones padre-hijo, comenzando por el ancestro de todos, **systemd**. Cuando existen varios procesos idénticos, hijos del mismo padre, se simplifica la notación, colocándose una sola rama con el nombre del proceso entre corchetes junto al contador de repeticiones. Los hilos que se crean desde un proceso se muestran como una rama a partir de la del proceso creador, con el nombre del proceso entre llaves junto al contador de repeticiones.

-p: muestra el PID de cada proceso.

PID: para ver el árbol de un proceso.

top: proporciona una visión dinámica de la actividad del sistema en tiempo real, mostrando las tareas que más uso hacen de la CPU con una interfaz interactiva para manipular procesos. Cada cierto tiempo preestablecido esta información se actualiza, reflejando lo ocurrido en el intervalo transcurrido desde el último refresco. Por defecto se ordena decrecientemente por uso de la CPU, actualizándose la lista normalmente cada 3 segundos.

Una ventaja de top respecto a **ps** es que podemos tener la información actualizada en una pantalla en todo momento. Una desventaja es que no muestra la información de todos los procesos en ejecución en el sistema.

Permite realizar tareas sobre procesos y la información mostrada de ellos pulsando:

r: para cambiar la prioridad de un proceso.

k: para matar o enviar una señal a un proceso.

M: para ordenar según uso de memoria.

P: para ordenar según uso de CPU.

N: para ordenar según el PID.

T: para ordenar por tiempo.

n: para cambiar el número de procesos que se muestran.

L: para buscar un texto en lo que se muestra en pantalla.

H: para mostrar información de cada hilo en lugar de cada proceso y viceversa.

1: para mostrar la información de la CPU por cada núcleo en lugar de todas en conjunto y viceversa.

q: para salir.

Las cinco primeras líneas muestran información general del sistema:

1. La hora actual, cuánto tiempo lleva en marcha el sistema, el número de terminales de usuario abiertas, y la carga media del sistema en forma de número promedio de procesos en los últimos 1, 5 y 15 minutos (se expresa en tanto por uno, sin estar normalizado al número de cores de la CPU, por lo que en CPU multicores es habitual valores mayores que 1). Los procesos que se tienen en cuenta son tanto los que están en estado ejecutable (estado R), como los que están en estado de bloqueado no interrumpible a la espera de una operación de E/S de corta duración, normalmente de disco (estado D).
2. Estadísticas sobre los procesos del sistema.
3. El estado de la CPU:
    - us: porcentaje de uso de la CPU para tareas a nivel de usuario.
    - sy: porcentaje de uso de la CPU para tareas del sistema.
    - ni: porcentaje de uso de la CPU para tareas a nivel de usuario con prioridad base mayor que 0.

- `id`: porcentaje en que la CPU está ociosa.
- `wa`: porcentaje de tiempo que la CPU ha estado esperando, desocupada, a que el sistema finalice operaciones de E/S (normalmente de disco).
- `hi`: porcentaje de tiempo que la CPU ha estado tratando interrupciones hardware.
- `si`: porcentaje de tiempo que la CPU ha estado tratando interrupciones software.
- `st`: porcentaje de tiempo de una máquina virtual que ha sido *robado* por el *hypervisor* de dicha máquina.

4. La memoria: la última columna (`buff/cache`) indica la utilizada para caché de disco y memoria caché de páginas.
   El último valor mostrado en línea siguiente (`avail Mem`) realmente está relacionado con la información de la línea actual, pues es una estimación de la cantidad de memoria disponible que se podría usar, tanto por los procesos existentes como para crear nuevos procesos, por lo que incluye la memoria que no está siendo usada en absoluto (la mostrada en la columna `free`) y otras zonas de memoria que también se pueden utilizar dinámicamente cuando sea oportuno (como parte de la mostrada en la columna `buff/cache`).
5. El espacio de swap.

Los datos de la parte inferior son similares a `ps` en su mayoría, algunos cambian ligeramente:

- `VIRT`: se corresponde con VSZ de ps.
- `RES`: se corresponde con RSS de ps.
- `SHR`: la cantidad de memoria que podría ser compartida con otros procesos.
- `S`: columna STAT de ps.
- `TIME+`: columna TIME de ps.

Número nice y prioridad de procesos

Inicialmente, al lanzar un proceso de usuario se le asigna un valor de prioridad base, el rango va de -20 (máxima prioridad) a 19 (mínima prioridad). Por defecto cada proceso hereda la prioridad base de su padre. Asignar un valor que mejore la prioridad base del proceso **solo puede hacerlo el root**.

La prioridad final del proceso de usuario se calcula a partir de la prioridad base, teniendo en cuenta determinados factores. Normalmente esta prioridad final suele ser igual al valor `nice` + 20, quedando un rango de 0 a 39. Este rango queda mapeado en el rango absoluto de 100 a 139, mientras que los procesos a nivel de núcleo conforman el rango de prioridades de tiempo real (`rt`) desde 0 a 99.

`nice -<numero> <orden_a_ejecutar>`: `ejecuta el proceso empeorando la prioridad (numero es positivo) o mejorándola (numero es negativo).`

`renice <nueva_prioridad> <pid>`: `fija el valor nice en nueva_prioridad.`

Envío de señales a procesos

Una señal es un aviso que puede enviar un proceso a otro proceso. Cuando un proceso recibe una señal, se pone inmediatamente a ejecutar la función de tratamiento de esta señal. Cuando acaba el tratamiento, continúa su ejecución normal (si la señal no era de terminación). Todas las señales tienen una función de tratamiento por defecto, un proceso puede decidir capturar una señal, cambiando esta función de tratamiento por otro código que él disponga.

Se pueden enviar señales a los procesos para pararlos, con la señal SIGSTOP (19), para eliminarlos con la señal SIGKILL (9), para hacer que continúen con SIGCONT (18), etc. Se envían mediante la orden:

`kill [-señal] <pid>`…

La señal que se envía por defecto es SIGTERM (15), que ordena al proceso receptor que termine su labor de forma correcta y controlada, puede ser capturada. Si la señal que se envía es SIGKILL (9), tendremos la seguridad de que el proceso receptor finalizará.

`killall [-señal] <orden>`: hace un envío más genérico usando el nombre del proceso, en lugar de su PID. De esta forma, nos aseguramos de que todos los procesos con este nombre reciban la señal.

Hay procesos que no mueren a pesar de recibir la señal SIGKILL: los zombies (porque ya están muertos, estado Z), y los que esperan una petición de E/S hecha a un disco (estado D), porque están realizando una llamada al sistema y no pueden ser interrumpidos (o eliminados) hasta que esta llamada se complete.

Programación puntual de tareas

`at [hora] [fecha]`: se utiliza para ejecutar tareas a una determinada hora y/o día. De forma predeterminada, pide las órdenes a ejecutar mediante un prompt especial (at>), se finaliza con **`Ctrl+D`**. Podemos ejecutar comandos en directorios concretos así: /<directorio>/<comando>.

> **`-f <fichero>`**: para recibir como parámetro un fichero de texto con las órdenes a ejecutar.

Para establecer el momento de ejecución podemos indicar una hora absoluta, o una hora relativa a partir del momento actual. También podemos establecer una fecha absoluta o relativa.

Ejemplos:

- El domingo a las 4pm: `# at 4pm sunday`
- Dentro de 3 días a las 11am: `# at 11am +3days`
- A las 3am de nochebuena de 2021: `# at 3am 12/24/2021`
- El próximo viernes: `# at Friday`
- En 5 horas: `# at now +5hours`

El servicio encargado de ejecutar órdenes es **`atd.service`**, si no está activo, no se lanzarán. Para consultar su estado debemos ejecutar `systemctl status atd.service` y para activarlo `systemctl start atd.service`. La orden `atq` sirve para consultar la lista de órdenes pendientes, con `atrm` podremos eliminar algunas de estas órdenes.

Programación de tareas periódicas

Usaremos el servicio `crond.service` (igual que con `atd.service`, si no está activo, no se lanza). Para consultar su estado debemos ejecutar `systemctl status crond.service` y activarlo con `systemctl start crond.service`.

`crontab [OPCIONES]:` `para ejecutar tareas periódicamente.`

> **`-e:`** `para añadir/modificar tareas. Al ejecutarlo se abre un editor (por defecto vim). En cada línea podemos indicar la periodicidad de un tarea siguiendo este formato:`
>
> `minutos hora día_mes mes día_semana tarea`
>
> **`-l:`** `para listar todas las tareas programadas.`
>
> **`-r:`** `para eliminar todas las tareas programadas.`

Poner un valor en lugar de un asterisco introduce una restricción (podemos poner un número, un rango para los días, p.ej.: 1-5, o el día de la semana: sun). Si ponemos una fecha y una hora, estamos limitando la ejecución a ese día y esa hora, pero, si ponemos un día del mes y un día de la semana, se ejecutará ese día del mes y, **además**, esos días de la semana del mes. SI queremos poner más restricciones deberemos incluirlas en la orden que se ejecute, por ejemplo, comprobando con `date` si es o no domingo.

Otra opción para indicar periodicidad consiste en sustituir los 5 primeros campos de la línea por uno de estos *nicknames*:

- `@reboot: Se ejecuta justo después de rearrancar.`
- `@yearly: Se ejecuta una vez al año.`
- `@monthly: Se ejecuta una vez al mes.`
- `@weekly: Se ejecuta una vez a la semana.`
- `@daily: Se ejecuta una vez al día.`
- `@hourly: Se ejecuta una vez cada hora.`

El fichero `/etc/cron.d/0hourly` es el fichero para el servicio cron del sistema. Está preparado para ejecutar una serie de tareas cada hora. Para ello, usa el script `run-parts`, que se encarga de ejecutar por orden alfabético todos los ficheros con permiso de ejecución que encuentra en el directorio que se le pasa como parámetro. Las tareas para ejecutar se copian como ficheros ejecutables en el directorio `/etc/cron.hourly/`. Si vemos las tareas planificadas para ejecutarse cada hora encontramos al ejecutable `anacron`, que es un planificador de tareas periódicas que no asume que la máquina tenga que estar encendida continuamente. La configuración de `anacron` está establecida en el fichero `/etc/anacrontab`.

Las tareas que queramos planificar para ejecutarse cada día, semana y mes se deben copiar como fichero ejecutables en los directorios `/etc/cron.daily/`, `/etc/cron.weekly/` y `/etc/cron.monthly/`, respectivamente. Cada hora se pondrá a funcionar `anacron`, que leerá la lista de trabajos especificados, comprobará si se han ejecutado en el último período establecido y, en caso negativo, los pondrá a funcionar utilizando el script `run-parts`.

Control y gestión de la memoria

La paginación y el intercambio son técnicas que Linux usa para distribuir la memoria disponible entre los procesos. El intercambio consiste en llevar un proceso a disco para liberar la memoria que está utilizando, posteriormente el proceso volverá a memoria para continuar su ejecución. La paginación implica llevar parte de la memoria de los procesos, en unidades llamadas páginas, a disco para liberar memoria. Estas páginas se traerán de vuelta a memoria cuando se vuelvan a necesitar.

free: presenta información sobre la ocupación de la memoria principal y del espacio de swap del que se disponga (por defecto se muestra la información en KiB). La información mostrada es similar a la que aparece en las 2 últimas líneas de la cabecera de top.

El campo **free** representa la memoria libre completamente, el **buff/cache** la usada en buffers y en memoria caché de páginas, y el **available** es una estimación de la cantidad de memoria disponible que se podría usar, se incluye en free. El campo **shared** puede ser ignorado pues presenta un valor obsoleto.

vmstat [intervalo número]: presenta información sobre el uso de la memoria principal, las lecturas/escrituras de/en el espacio de swap, la actividad del sistema operativo en cuanto a interrupciones y cambios de proceso, así como uso de la CPU.

intervalo: cada cuántos segundos debe mostrar los datos.

número: cuántos muestreos se solicitan.

Concretamente, los campos que muestran información relacionada con la gestión de la memoria del sistema son:

Procesos (procs):

r: número de procesos ejecutables (en ejecución o listos para ejecutarse) (R de ps y top).

b: número de procesos bloqueados (D en ps y top).

Memoria (memory): información semejante a la cabecera de top y la orden free.

swpd: espacio de intercambio en KiBs.

free: memoria RAM sin usar en KiBs.

buff: memoria RAM empleada como buffers para E/S en KiBs.

cache: memoria RAM empleada como caché de páginas en KiBs.

Espacio de intercambio (swap):

si: memoria traída del espacio de intercambio a memoria en KiB/s.

so: memoria intercambiada al disco en KiB/s.

Entrada/Salida (io):

bi: información recibida desde un dispositivo de bloques en KiB/s.

bo: información enviada a un dispositivo de bloques en KiB/s.

La orden muestra la información relativa al instante actual para la memoria, y la información relativa a cada período de tiempo establecido (siendo la información que muestra en la primera línea la que corresponde al intervalo de tiempo desde que se arrancó el sistema hasta el momento actual) para el swap y la E/S. En caso de que no se le pasen argumentos, se muestra únicamente esta primera línea.

Espacio para paginación

Se puede utilizar una partición de intercambio o bien un fichero de paginación. El rendimiento usando un fichero de paginación es menor que con la partición debido a la sobrecarga de la gestión que se precisa por parte del sistema de ficheros. Además, a la partición no le afectan los problemas de fragmentación que puede tener cualquier fichero individual, incluido un fichero de intercambio.

Se puede asignar valores de prioridad de uso mediante la opción `pri=valor` (números altos significan mayor prioridad). En tiempo de arranque se activan todas las zonas de intercambio o paginación indicadas en el fichero `/etc/fstab`. Así, por ejemplo, podemos activar en el arranque una partición de intercambio y un fichero de intercambio, indicando que se use la partición prioritariamente, si tenemos en el `/etc/fstab`:

```
/dev/mapper/fedora-swap    none  swap  defaults,pri=5  0 0
/.fichero_swap             swap  swap  defaults,pri=3  0 0
```

Con la orden **swapon** se activa una partición de intercambio, mientras que con **swapoff** se desactiva. Una vez desactivada una partición, esta se podrá usar para otras cosas pues tendremos la garantía de que toda la información de la memoria virtual que maneja cada proceso que está en funcionamiento se encuentra ya definitivamente almacenada en la memoria secundaria asignada a dicho proceso.

La creación de un fichero de intercambio se realiza en varios pasos:

```
# dd if=/dev/zero of=/.fichero_swap bs=1048576 count=1024
# mkswap /.fichero_swap
# sync
# swapon /.fichero_swap
```

Tema 9: Gestión de la E/S

Ficheros especiales de dispositivo

Existen otros *ficheros especiales* que se utilizan para referenciar a diferentes elementos del sistema. Entre estos encontramos los *ficheros especiales de dispositivo* que representan dispositivos de E/S. Dentro tenemos los ficheros especiales de bloques y los ficheros especiales de caracteres. Todos se encuentran en el directorio `/dev` o en alguno de sus subdirectorios.

Los *ficheros especiales de bloques* sirven para dar nombre a los dispositivos de bloques, como los discos duros, las unidades de CD/DVD, las memorias flash… Se distinguen del resto porque sus permisos comienzan por la letra **b**. Si vemos los ficheros de `/dev` veremos ficheros especiales de bloques para los dispositivos *loop*, un disco duro (`sda`) y sus particiones (`sda1`, `sda2`, etc), y una unidad de DVD (`sr0`).

Por otro lado, los *ficheros especiales de caracteres* representan a dispositivos de caracteres. Los permisos de estos ficheros comienzan por la letra **c**. Entre los ficheros encontramos el del *frame buffer* (`fb0`), un puerto paralelo (`lp0`), terminales (ficheros `tty*`), el sumidero de capacidad infinita (`null`), la fuente inagotable de ceros (`zero`) y un fichero para generar butes aleatorios (`random`).

Los ficheros no tienen un tamaño asociado, sino un par de números que son utilizados internamente por el núcleo de Linux para referirse a los dispositivos que representan. El primero de estos es el *número mayor* (*major number*) que identifica con qué driver está asociado el fichero (es decir, qué driver maneja el dispositivo). El segundo número es el *número menor* (*minor number*) que identifica al dispositivo concreto asociado al fichero (también se llama *número de unidad* o *unit number)*. En general, dispositivos iguales o similares están representados por ficheros que tienen el mismo número mayor, pero un número menor distinto para cada uno.

A la hora de referirse a un dispositivo, lo importe es su número mayor y su número menor. El fichero especial de dispositivo es solo una forma sencilla y cómoda de referirse a él. Si ejecutamos:

`mknod primerdiscoduro b 8 0`, se creará un fichero especial de dispositivo a través del cual podremos hacer referencia al primer disco duro (igual que con `/dev/sda`). Si ajustamos los permisos, podemos usar de forma indistinta uno u otro. b es el tipo de dispositivo (bloques(b)/caracteres(c)), 8 es un ejemplo del número mayor y 0 del número menor.

Algunos de estos ficheros son creados a mano con la herramienta

`mknod` durante el arranque del ordenador, otros, los crea automáticamente la herramienta

`udev` que, además, crea y borra estos ficheros dinámicamente cuando los dispositivos correspondientes se conectan y desconectan.

Un fichero especial de dispositivo representa a un dispositivo de manera genérica, sin tener en cuenta detalles, por lo que `/dev/sda` representa al primer disco duro del ordenador, si se cambia, representará al nuevo.

Al ser ficheros, podemos abrirlos para leer de y escribir en ellos. La diferencia con un fichero normal es que realmente estamos leyendo o escribiendo a bajo nivel sobre el dispositivo que

representan. Es importante vigilar los permisos, propietarios y grupos, comandos como `mkfs` usan este fichero especial de bloques para acceder al dispositivo.

Versión del núcleo

`uname -r`: para obtener la versión del núcleo de Linux en ejecución en un instante determinado.

Módulos del núcleo

El núcleo de Linux sigue una arquitectura monolítica, si bien el diseño es modular. Esto significa que, aunque el núcleo puede ser visto como un único programa en ejecución, hay porciones de código (los módulos) que se pueden cargar/descargar en caliente para añadir/eliminar código al/del núcleo de forma dinámica, sin reiniciar el sistema, y, por tanto, para añadir/eliminar soporte para ciertas características.

Cuando un módulo se carga, su código pasa a ser parte del propio núcleo y se ejecuta en el modo núcleo (o supervisor) del procesador, con acceso sin restricciones a todas las funciones del núcleo, a todas las funciones exportadas por módulos previamente insertados y a todo el hardware de la máquina.

Los módulos proporcionan varias ventajas respecto a un núcleo basado en un único ejecutable: facilitan el diseño y la implementación del sistema operativo, permiten crear núcleos más pequeños y flexibles que se pueden adaptar al hardware subyacente, se pueden descargar cuando ya no son útiles, liberando así recursos, etc.

Podemos considerar a los módulos como los *drivers* de Linux. Se almacenan en ficheros objeto con extensión `.ko.xz` bajo el directorio `/lib/modules/<versión_kernel>`, por lo que cada versión del núcleo tiene sus propios módulos. Dentro de este directorio, la mayor parte de los módulos se encuentran en el subdirectorio `kernel` que, a su vez, está organizado en subdirectorios. Algunos de los subdirectorios dentro de `kernel` son:

`drivers`: contiene módulos para la gestión de los dispositivos hardware.

`fs`: contiene módulos que proporcionan soporte para distintos sistemas de ficheros.

`net`: contiene módulos para el soporte de diferentes protocolos de red.

`sound`: contiene módulos para la gestión de distintas tarjetas de sonido.

Aunque la mayor parte del código de Linux se encuentra en los módulos, hay código que, por su importancia, forma parte del núcleo de Linux de forma permanente. Es posible incluir el código de un módulo de forma permanente en el núcleo de Linux, indicándolo así en la configuración del núcleo y recompilando el mismo.

Dependencias

Cuando se implementa un módulo se usan funciones que ya están definidas dentro del núcleo del sistema operativo. Para cargar un módulo es necesario que el núcleo tenga exportadas todas las funciones externas que usa el módulo. Algunas pueden estar implementadas en otros módulos por lo que, para que sea posible la carga de nuestro módulo, es necesario que otros módulos se hayan cargado previamente. Esto hace que surjan dependencias entre los módulos.

Las dependencias entre los módulos de un núcleo determinado se encuentran en el fichero /lib/modules/<versión_kernel>/modules.dep, que indica qué otros módulos tienen que estar cargados previamente para que cada módulo con dependencias se pueda cargar.

Carga de módulos

Los módulos se pueden cargar en distintos momentos:

- Al ejecutar el script `init` del disco RAM. El contenido, usado durante el arranque del sistema de operativo, se obtiene de un fichero que, entre otras cosas, almacena algunos módulos que son necesarios para que el sistema operativo pueda acceder, al menos, al sistema de ficheros raíz.
- Durante la ejecución de los distintos scripts y programas por parte del proceso `systemd`.
- Al conectar un nuevo dispositivo. En este caso, `udev` detectará el dispositivo y cargará el módulo correspondiente.
- A mano, en cualquier momento, por parte del administrador.

Herramientas para la gestión de módulos

lsmod: permite ver los módulos cargados. Para cada módulo, muestra su nombre (Module), su tamaño (Size), un contador de usos y los módulos que lo usan (Used by). Obtiene toda esta información del fichero /proc/modules.

modinfo <modulo>: muestra la información sobre un módulo del núcleo concreto.

insmod <ruta del fichero .ko.xz del módulo>: para cargar los módulos. **No resuelve dependencias,** por lo que un módulo solo se podrá cargar si ya están cargados los módulos de los que depende.

rmmod <nombre del módulo, tal cual lo muestra lsmod>: para descargar un módulo. Un módulo no se puede descargar si está siendo usado. **Tampoco tiene en cuenta las dependencias.**

modprobe [OPCIONES] <nombre del módulo >: carga un módulo **teniendo en cuenta sus dependencias,** cargando previamente los módulos de los dependa. Respecto al parámetro, a diferencia de insmod, basta con el nombre, no hace falta indicar la ruta del fichero .ko.

> **-r:** para descargar un módulo. **También tiene en cuenta las dependencias,** si el módulo descargado dependía de otros que ya no se necesitan, también se descargarán.
>
> **--show-depends:** dado un módulo, muestra todos aquellos módulos de los que depende.

Podemos cambiar el comportamiento de `modprobe` creando un fichero de configuración en `/etc/modprobe.d`, puede tener cualquier nombre, pero su extensión debe ser `.conf`. Entre las líneas que podemos encontrar en un fichero de este tipo tenemos:

- `alias <nombre alias> <nombre módulo>`: permite asignar un alias (nombre alternativo) a un módulo.
- `options <nombre módulo> <opciones ...>`: permite configurar las opciones de un módulo (si las tiene). Estas opciones se usan cuando se carga el módulo.
- `install <nombre módulo> <orden a ejecutar>`: ejecuta la orden indicada en lugar de insertar el módulo, como se haría normalmente.

- `remove <nombre módulo> <orden a ejecutar>`: es similar a la línea anterior, pero cuando se elimina un módulo.
- `blacklist <nombre módulo>`: impide que se ponga en marcha un módulo de forma automática. El módulo podría ser cargado sin embargo si algún otro que depende de él y que no aparece en `blacklist` se carga. También podría cargarse manualmente. Para forzar que un módulo no pueda ser cargado de ningún modo habría que hacer uso de `install`, indicando como orden a ejecutar `/bin/false`. Observa que en este caso se estaría impidiendo la carga del módulo en cuestión y de cualquier otro que dependiese de él.

depmod -a: para actualizar el fichero /lib/modules/<versión kernel>/modules.dep, en el que se guardan las dependencias entre módulos.

Los sistemas de ficheros proc y sysfs

Los directorios `/proc` y `/sys` son dos puntos de montaje en los que solemos encontrar montados respectivamente los sistemas de ficheros virtuales `proc` y `sysfs`, son virtuales porque no se almacenan en ningún disco ni dispositivo similar, su información solo se almacena en RAM y existe mientras el núcleo de Linux esté en ejecución. Proporcionan una interfaz sencilla y conocida para leer y modificar información almacenada dentro del núcleo de Linux.

Respecto a la E/S, nos permiten acceder a la información que el núcleo tiene sobre los dispositivos hardware disponibles, buses, módulos, etc. También son utilizados por ciertas herramientas para la gestión del hardware. `udev` usa `sysfys` para obtener información sobre los dispositivos que se conectan y así poder cargar los módulos y crear los ficheros especiales de dispositivo correspondientes.

Respecto a `/proc`, podemos encontrar muchos ficheros y directorios con información interesante sobre el sistema, entre los ficheros relacionados con la E/S tenemos:

- `devices`: dispositivos de bloques y de caracteres actualmente configurados.
- `interrupts`: interrupciones tratadas y dispositivos asociados.
- `iomem`: rangos de memoria usados por los dispositivos para la E/S mapeada a memoria.
- `ioports`: puertos registrados para el acceso a los dispositivos mediante instrucciones específicas de E/S.
- `modules`: listado de todos los módulos cargados actualmente.

Aunque `/proc` contiene ciertos datos sobre el hardware, la información más completa y detallada se encuentra en `/sys`.

lshw: nos da información detallada sobre la configuración hardware de la máquina.

lsusb [OPCIONES]: muestra información sobre todos los buses USB del sistema y todos los dispositivos conectados a ellos.

> **-v:** imprime información detallada de los dispositivos.
>
> **-d <vendor>:<product>:** para obtener información solo sobre el dispositivo indicado, donde <vendor>:<product> son los valores hexadecimales mostrados tras ID al ejecutar la orden sin opciones.

`lspci [OPCION]:` `muestra información sobre todos los buses PCI del sistema y todos los dispositivos conectados a ellos.`

`-v:` `imprime información detallada de los dispositivos.`

udev

Gran parte de la gestión de periféricos en Linux se hace en espacio de usuario mediante *udev*, que es un gestor de dispositivos en espacio de usuario.

El objetivo principal es que, al conectar un dispositivo, el usuario pueda utilizarlo directamente, sin intervenir en el proceso de preparación del dispositivo. Otro objetivo es proporcionar un directorio de dispositivos `/dev` dinámico, que contenga solo los ficheros de los dispositivos conectados en un momento dado.

El proceso que se encarga de realizar las tareas es el demonio `systemd-udevd`, utiliza información de `/sys` y de los *ficheros de reglas* (*rules*), almacenados en `/lib/udev/rules.d/` y `/etc/udev/rules.d/`, para saber exactamente qué hacer: qué nombre asignar al fichero especial de dispositivo, qué permisos y propietario darle, qué programa ejecutar (si es necesario), qué módulos se han de cargar, etc.

Cuando se conecta o desconecta un dispositivo, `systemd-udevd` procesa todos los ficheros de reglas en orden alfabético. Los nombres de estos ficheros suelen ser de la forma `<dos dígitos>-<nombre descriptivo>.rules`. Si hay una regla con el mismo nombre en `/etc/udev/rules.d/` y `/lib/udev/rules.d/`, la primera tiene preferencia.

1. Recibe una notificación del núcleo, que le indica que un nuevo dispositivo ha sido conectado.
2. Mira en `/sys` si el *driver* correspondiente proporciona un fichero `dev` que contenga el número mayor y menor para el fichero especial de dispositivo a crear.
3. Usa las reglas para preparar el dispositivo. En concreto:
    - Crea el fichero especial de dispositivo y los enlaces simbólicos al mismo con los nombres alternativos propuestos (si los hay).
    - Establece los permisos y los propietarios del fichero especial de dispositivo creado
    - Si es necesario, carga los módulos y el *firmware* para poder trabajar con el dispositivo.
    - Si alguna regla lo indica, ejecuta los programas especificados para inicializar el dispositivo, etc.

Cuando un dispositivo se desconecta, el funcionamiento de `systemd-udevd` es similar, si bien las reglas determinan qué hacer para liberar los recursos usados por el dispositivo.

Planificadores de disco

Existen diversos planificadores de E/S que determinan el orden en el que se atienden las peticiones de lectura y escritura de disco. En particular, el objetivo de estos planificadores es aumentar la productividad reordenando las peticiones en función de las direcciones lógicas de los bloques asociados a las peticiones e intentando agruparlas. Aunque esto puede aumentar la productividad general, puede conllevar también que algunas peticiones esperen demasiado tiempo, causando problemas de latencia.

De esta forma, los planificadores de E/S intentan equilibrar la necesidad de alta productividad con una atención lo más equitativa posible de las peticiones de disco de los distintos procesos.

A partir de la 5.0, se dispone de diversos planificadores multicola, cada uno potenciando unas metas. Estos planificadores distribuyen las peticiones en varias colas, las cuales son gestionadas por hilos del kernel que idealmente se ejecutan en diferentes núcleos del procesador. En concreto, los planificadores de los que disponemos son los siguientes:

bfq (Budget Fair Queuing): es un planificador diseñado para proporcionar una buena respuesta interactiva, especialmente para discos lentos. Trata de repartir el tiempo de uso del disco entre los distintos procesos de forma equitativa. A cada proceso que realiza peticiones de lectura y escritura sobre un dispositivo de almacenamiento, se le asocia un peso y una cola. Otorga acceso exclusivo al dispositivo a una de las colas cada vez (a un proceso), definiendo para cada cola, en función de su peso, una cantidad máxima de accesos (*budget*), medida en número de sectores. No es recomendable su uso con CPUs lentas o discos de alto rendimiento (SSD modernos y NVMe).

kyber: más simple y, por lo tanto, más rápido, está inspirado en las técnicas de gestión de colas activas empleadas en encaminamiento en redes. Las peticiones de disco se distribuyen en dos colas: una para las peticiones síncronas y otra para las peticiones asíncronas. Se establecen límites estrictos en el número de peticiones enviadas a cada cola, lo que en teoría limita el tiempo de espera de las peticiones para ser atendidas, y por lo tanto debería proporcionar un tiempo de finalización corto para las peticiones de alta prioridad.

mq-deadline: Asigna plazos de tiempo a las peticiones de las distintas colas e intenta evitar que dichos plazos expiren. Cuando todas las peticiones están dentro de sus plazos, estas se atienden según un algoritmo SCAN. Si hay peticiones con plazos cumplidos, entonces son las primeras en ser atendidas según un algoritmo FCFS.

none: funciona como un planificador FCFS (no se reordenan las peticiones de disco).

El uso de uno u otro planificador depende del tipo de dispositivo de almacenamiento y de la carga de trabajo que reciba. Para SSD o NVMe, hay poca diferencia, por lo que la mejor opción es utilizar *none*. Para HDDs es importante ordenar las peticiones teniendo en cuenta las direcciones de disco de las mismas, esto ayuda a reducir el tiempo de búsqueda, lo que minimiza la latencia media de las peticiones de disco, mejorando la productividad. Para servidores con gran carga de peticiones de disco, *mq-deadline* suele ser la mejor opción. Para sobremesa, *bfq* es capaz de cargar algunas aplicaciones de manera más rápida.

Cambio de planificador

Cada dispositivo de almacenamiento tiene su propio planificador. El planificador se puede cambiar en caliente, sin reiniciar el sistema. Podemos consultar y modificar el planificador usado por un disco a través de `/sys`.

Si queremos conocer el planificador utilizado para el primer disco duro:

```
cat /sys/block/sda/queue/scheduler
```

El planificador se muestra entre corchetes. Para cambiarlo:

```
echo <planificador> > /sys/block/sda/queue/scheduler
```

Muchas veces se usa *bfq* para un SSD o una flash por ser el planificador por defecto. Podemos cambiar el planificador por defecto pasando al núcleo de Linux el parámetro

elevator=<planificador> a través de GRUB.

Prioridad de la E/S de disco

Por defecto, las peticiones de disco de los procesos tienen la misma prioridad. Cuando utilizamos el planificador *bfq*, esta prioridad se puede cambiar de dos formas.

La primera forma, y también la más adecuada, es utilizar la orden `ionice` a la hora de ejecutar un proceso, esta orden define las siguientes clases o categorías de planificación (un proceso solo puede pertenecer a una):

Desocupado (*idle*): las peticiones solo se atenderán cuando ningún otro proceso haya enviado peticiones de E/S durante un cierto tiempo. El impacto de un proceso de esta clase sobre la actividad normal de un sistema debería ser 0.

Mejor esfuerzo (*best-effort*): define 8 niveles de prioridad, desde 0 hasta 7, donde un número menor significa mayor prioridad. Mismo nivel de prioridad, dentro de esta clase, se atienden siguiendo un algoritmo *round-robin.*

Tiempo real (*realtime*): las peticiones de los procesos de esta clase son las primeras que se envían a disco, sin importar que haya otros procesos accediendo a disco. Esta clase debe usarse con precaución, ya que un proceso puede impedir que otros usen el disco. Se definen 8 niveles de prioridad, que indican no qué peticiones se atienden primero sino cómo de grande es la porción de tiempo de disco que recibirá cada proceso, en función del nivel en el que se encuentre.

La planificación de las peticiones de disco es similar a una planificación de procesos multinivel sin realimentación. Solo el superusuario puede ejecutar procesos dentro de la clase de tiempo real. El resto de usuarios tiene que usar las otras dos clases.

ionice [OPCIONES] [<clase>]

Si no se indica ninguna clase, el proceso que se cree pertenecerá al nivel de prioridad 4 de la clase *mejor esfuerzo*.

-p <pid>: una vez en ejecución, se puede cambiar la clase (y el nivel de prioridad, si es el caso) a la que pertenece un proceso.

-c <clase>: para indicar la clase a la que debe pertenecer el proceso. 0 para *ninguna*, 1 para *tiempo real*, 2 para *mejor esfuerzo* y 3 para *desocupado*. También se puede poner --class <clase>.

-n <nivel>: para indicar el nivel dentro de una clase con niveles. También se puede poner --classdata <nivel>.

-p <PID>: para especificar el PID del proceso en ejecución al que establecer la prioridad o del que obtener su prioridad de E/S. También se puede poner --pid <PID>.

A diferencia de renice, donde un usuario normal solo puede disminuir la prioridad de un proceso, con ionice es posible tanto aumentar como disminuir la prioridad en cualquier momento.

Si un programa se ejecuta sin usar ionice, entonces pertenecerá a la clase *ninguna* (o *none*). Esta clase es, en realidad, la clase *mejor esfuerzo*, donde el nivel o prioridad al que pertenece un proceso depende de su valor *nice* de CPU, siguiendo la fórmula:

$$prioridad\_E\_S = \frac{valor\_nice + 20}{5}$$

La segunda forma de cambiar la prioridad de E/S es cambiando el valor *nice*, siempre que el programa correspondiente no se haya ejecutado usando ionice. Ya que el valor de *nice* de un proceso de usuario normal varía entre 0 y 19 (siendo 0 el valor por defecto), la fórmula anterior nos dice que la prioridad de E/S de uno de estos procesos puede estar entre 4 y 7. Solo el superusuario puede ejecutar procesos con un valor *nice* negativo y, por tanto, solo él podrá ejecutar procesos pertenecientes a los niveles 0, 1, 2 o 3, dentro de la clase *mejor esfuerzo*.

iotop [OPCION]: proporciona una funcionalidad similar a la de la orden top, pero apicada a la E/S. Muestra varias columnas con información sobre la actividad de E/S de cada proceso, y permite ordenar los procesos en base a esas columnas.

-P: muestra procesos y no los hilos que se muestran de forma predeterminada.

Para cada línea encontramos: identificador del hilo (TID), su clase y el nivel de prioridad de E/S, si es aplicable a la clase (PRIO), propietario (USER), tasa de transferencia en lecturas (DISK READ) y en escrituras (DISK WRITE), porcentaje de tiempo que el hilo pasa esperando a que se lean las páginas para las que ha producido fallos (SWAPIN), porcentaje de tiempo que el hilo espera a que se atiendan sus peticiones de E/S de disco (IO) y línea de órdenes correspondiente (COMMAND).

La información está ordenada por la columna IO, de mayor a menor. Esto lo indica el símbolo **>** a la derecha de IO. Si pulsamos la tecla **r**, se cambia a **<** y la información se ordena en orden ascendente. SI volvemos a pulsar **r**, se volverá al orden descendente. Podemos cambiar de columna usando los cursores izquierdo y derecho.

Una vez identificado el proceso que produce una mayor actividad de disco, podemos bajar su prioridad usando ionice, la mejor opción es moverlo a la clase *desocupado*.

ANEXO:

Quita la partición del grupo de volúmenes y utiliza en su lugar otra partición:

Hacer vgdisplay -v para ver si hay extensiones físicas utilizadas.

Si las hay, hay que moverlas primero:

Hacemos vgextend para añadir la nueva particion

Hacemos pvmove para mover las extensiones

Y lo eliminamos del grupo de volúmenes con vgreduce

Si se pide, borramos el volumen físico con pvremove.

¿Podemos forzar los permisos de todos los ficheros que se creen en la partición en Ext4?

En sistemas Ext4, cada fichero tiene su propio nodo-i y, por tanto, no hay una opción de montaje que permita que los permisos de todos los ficheros sean comunes. Solo se puede hacer en sistemas Fat.