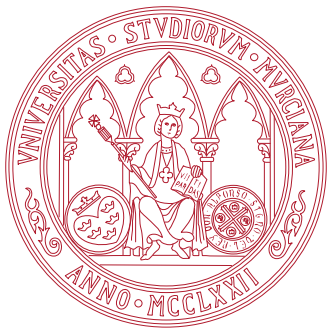

Universidad de Murcia

Facultad de Informática

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores

Prácticas de

Introducción a los Sistemas Operativos

2º de Grado en Ingeniería Informática

Boletín de Prácticas 7 – Administración de Sistemas de Ficheros en Linux

Curso 2021/2022

Índice

1. Introducción

Una vez estudiado en el boletín anterior la gestión de dispositivos de bloques y cómo estos se representan a través de ficheros especiales de bloques, veremos ahora cómo utilizar dichos dispositivos para almacenar información sobre ellos a través de los conceptos de fichero y directorio. Tal y como hemos visto en la parte teórica, el sistema de ficheros es la parte del sistema operativo encargado de hacerlo posible.

Aunque el propósito básico de un sistema de ficheros es organizar los dispositivos de almacenamiento, en él podemos encontrar muchos tipos de ficheros:

- Ficheros de datos de los usuarios, de configuración de aplicaciones o del sistema operativo, bases de datos, librerías,...
- El núcleo del sistema operativo y ficheros con sus parámetros.
- Canales para la comunicación entre procesos (tuberías o sockets).
- Ficheros que representan los dispositivos conectados, etc.

La administración de sistemas de ficheros es otra de las tareas importantes para el administrador de un sistema operativo, ya que a través de la misma se asegura que los usuarios pueden acceder a sus ficheros y que éstos están en perfecto estado y con la seguridad oportuna.

2. Objetivos

Al terminar el boletín el alumno debe ser capaz de:

- Crear sistemas de ficheros de diversos tipos.
- Redimensionar un sistema de ficheros ya existente para aumentar o reducir su tamaño en función de las características del dispositivo de bloques que lo va a contener.
- Montar sistemas de ficheros, con opciones de montaje según el tipo de sistema de ficheros, usando bien el fichero de configuración `/etc/fstab` o bien la orden `mount`.
- Desmontar un sistema de ficheros y, si no fuese posible tal acción, determinar el motivo por el que no se puede desmontar.
- Consultar la ocupación de un sistema de ficheros, el espacio libre que aún queda y el número de ficheros adicionales que se pueden crear.
- Establecer límites en cuanto al espacio de disco que un usuario puede consumir y/o el número de ficheros que puede crear en el sistema de ficheros.
- Crear enlaces físicos y simbólicos a un fichero y distinguir entre ambos tipos de enlaces.
- Realizar diferentes tipos de copias de seguridad (completas, diferenciales o incrementales; locales o remotas) utilizando para ello la herramienta más adecuada a cada caso.

3. Órdenes utilizadas

Las órdenes que usaremos para resolver estos boletines son:

- `mkfs`
- `resize2fs`
- `mount`, `umount`

- `fuser` y `lsof`
- `e2label`
- `findmnt`
- `df` y `du`
- `ln`
- `quotaon`, `quotacheck`, `repquota`, `setquota`
- `chmod`, `chown` y `chgrp`
- `tar`, `cpio` y `rsync`

En las páginas de manual de cada una de estas órdenes encontrarás información detallada de cómo usarlas.

4. Gestión básica de un sistema de ficheros

Analizaremos en primer lugar las órdenes necesarias para crear un sistema de ficheros en un dispositivo de bloques, y posteriormente, montarlo en el sistema de ficheros lógico global de Linux. Sin embargo, antes de comenzar, es conveniente hacer un poco de historia para poner en contexto los sistemas de ficheros con los que trabajaremos en esta práctica.

4.1. Sistemas de ficheros Ext4 y vFAT

Durante muchos años Ext2 fue el sistema de ficheros de Linux, hasta que apareció Ext3 (una versión mejorada de Ext2). Ext2 no es un sistema de ficheros transaccional, por lo que cuando el sistema cae, para comprobar y recuperar la consistencia del mismo hay que analizarlo por completo. No constituye esta una operación que como administradores (o simples usuarios) queramos tener que realizar frecuentemente, ya que puede llevar mucho tiempo si la partición es grande y el sistema de ficheros está bastante lleno. Ext3 tiene el mismo formato que Ext2, pero además es transaccional, ya que añade un registro o journal que permite recuperar rápidamente la consistencia tras una caída del sistema. Actualmente el sistema de ficheros estándar de facto en Linux es el Ext4, que es muy similar a Ext3, incluyendo además diversas mejoras que hacen que tenga un menor uso de CPU y mayor rapidez en los procesos de lectura y escritura que Ext3.

Por otro lado, FAT es una familia de sistemas de ficheros que fue introducida a finales de 1980 en el sistema operativo MS-DOS de Microsoft. La versión inicial, denominada FAT12 e ideada para ser usada en disquetes, fue después reemplazada por FAT16 para poder soportar discos duros que tenían mayores tamaños de almacenamiento. Con posterioridad, con la aparición de Windows 95, surge vFAT, que siendo compatible con las versiones anteriores, soportaba nombres de fichero largos (hasta 255 caracteres, frente a los 8 del nombre y 3 de extensión de las versiones anteriores) con espacios en blanco y puntos. Por último, FAT32, introducido en una revisión de Windows 95, permitió trabajar de forma más eficiente con discos duros más grandes que FAT16 y vFAT, y suele usar también el mecanismo de nombres largos definido en vFAT. En la actualidad, FAT, en algunas de sus modalidades, es la forma más sencilla de compartir, de manera limitada eso sí, ficheros entre Linux y el mundo Windows. No es de extrañar por tanto que su uso esté extendido en dispositivos tipo tarjetas de memoria y memorias USB. De entre todos los manejadores Linux para la gestión de sistemas de ficheros FAT, vfat es el más adecuado en la mayoría de las situaciones, y por lo tanto, el más usado, y utiliza direcciones de bloque de 32 bits (con un tamaño máximo de fichero de 4 GiB) y nombres largos.

Si bien en esta práctica trabajaremos con sistemas de ficheros Ext4 y vFAT, es preciso destacar la existencia de otros sistemas de ficheros como XFS y NTFS. El primero constituye el más antiguo de los sistemas de ficheros transaccionales y fue desarrollado por Silicon Graphics para ser usado en 1994 en su implementación de UNIX llamada IRIX. Posteriormente, en el año 2000, fue liberado bajo

licencia GNU y posteriormente soportado en Linux. El segundo es un sistema de ficheros que apareció en Windows NT y actualmente es el sistema de ficheros nativo del sistema operativo Windows (desde Windows 2000).

4.2. Creación de un sistema de ficheros

La creación de un sistema de ficheros en Linux podemos realizarla utilizando la orden `mkfs`, a la cual a través de la opción `-t` debemos indicarle el tipo de sistema de ficheros a crear (por ejemplo, `ext4` o `vfat`) y el fichero especial de bloque que representa al dispositivo en el cual queremos implantar el sistema de ficheros. Por ejemplo, crear un sistema de ficheros `ext4` sobre la primera partición del segundo disco de la máquina virtual (`/dev/sdb1`) es tan sencillo como ejecutar:

```
# mkfs -t ext4 /dev/sdb1
```

Es importante tener en cuenta que tras ejecutar dicha orden cualquier información que hasta el momento pudiese estar siendo almacenada en `/dev/sdb1` se perderá.

Realmente en lugar de `mkfs` suele ser más cómodo ejecutar las órdenes concretas para cada tipo de sistema de ficheros, las cuales siguen el patrón `mkfs.`*sf*, donde *sf* representa al sistema de ficheros concreto. De esta forma, simplemente tecleando `mkfs.` y pulsando a continuación el tabulador para autocompletar, podríamos obtener la lista de todos los sistemas de ficheros disponibles. La siguiente orden tendría el mismo efecto que la del ejemplo previo:

```
# mkfs.ext4 /dev/sdb1
```

4.3. Redimensionamiento de un sistema de ficheros

En el boletín de prácticas anterior se vio la posibilidad de aumentar o reducir el tamaño de un volumen lógico a través de las órdenes `lvextend` y `lvreduce` respectivamente. Hay que tener en cuenta que si dicho volumen lógico contiene un sistema de ficheros, debemos también proceder a redimensionarlo para que ocupe el nuevo tamaño del volumen lógico. Es decir, una cosa es el tamaño de un volumen lógico y otra el tamaño del sistema de ficheros que pueda contener. Por ejemplo, si aumentamos el tamaño de un volumen lógico pero no redimensionamos el sistema de ficheros, este no podrá aprovechar el espacio adicional añadido al volumen lógico.

En general, nos podemos encontrar situaciones en las que el tamaño del sistema de ficheros no se corresponda con el del dispositivo de bloques. En estos casos debemos proceder a redimensionar el sistema de ficheros, de forma que su tamaño se ajuste al de este último. Para sistemas de ficheros Ext2, Ext3 o Ext4 podemos usar la orden `resize2fs` para conseguir tal propósito, para otros sistemas de ficheros habría que usar una orden adecuada (por ejemplo, `ntfsresize` para sistemas de ficheros NTFS) . La sintaxis para los usos más comunes de esta orden es la siguiente:

```
resize2fs [opciones] FicheroEspecialBloque [Tamaño]
```

En particular, la orden `resize2fs` lleva el sistema de ficheros al tamaño indicado, aumentándolo si se indica un tamaño mayor que el que actualmente tiene o reduciéndolo en otro caso. Si no indicamos ningún tamaño, el sistema de ficheros ocupará todo el espacio disponible en el dispositivo de bloques, con lo que será la forma de utilizar esta orden habitualmente tras haber extendido el tamaño de un dispositivo de bloques:

```
# resize2fs /dev/vg-iso/lv-repartido
```

Una operación de expansión del sistema de ficheros (hacerlo más grande) se puede realizar incluso mientras este se encuentra montado (siempre que el sistema de ficheros lo permita). Sin embargo, para reducir el tamaño del sistema de ficheros habremos de desmontarlo previamente.

Alternativamente, y para el caso concreto de los volúmenes lógicos con sistemas de ficheros Ext2, Ext3 y Ext4[1], las órdenes `lvextend` y `lvreduce` disponen de la opción `-r|--resizefs`, que permite redimensionar el sistema de ficheros de un volumen lógico al tiempo que este se aumenta y reduce de tamaño respectivamente.

[1] También para sistemas de ficheros ReiserFS and XFS.

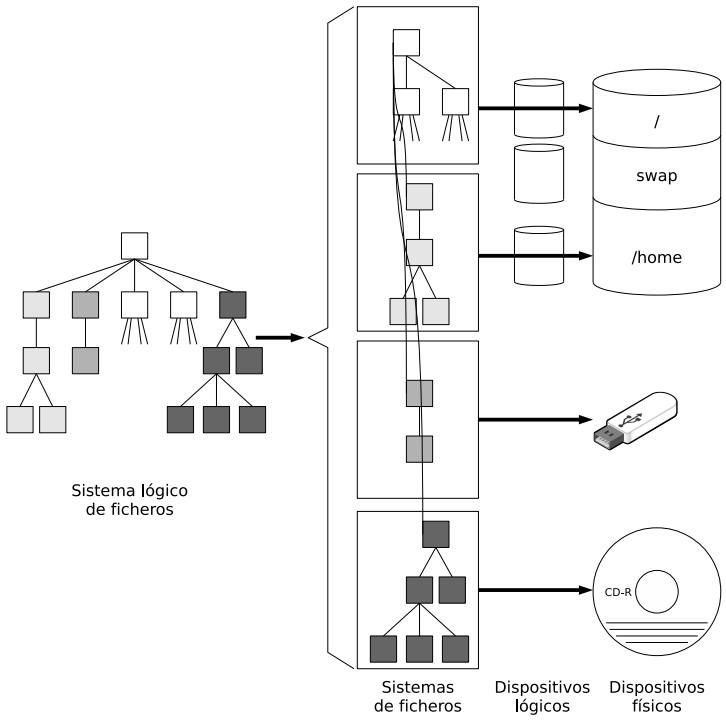

Figura 1: Relación entre sistema de ficheros lógico, sistemas de ficheros físicos y puntos de montaje.

4.4. Montar-Desmontar

Una vez creado el sistema de ficheros, hay que montarlo para habilitar el acceso al mismo. En Linux hay un único sistema de ficheros lógico (una única jerarquía de directorios) del que cuelgan todos los dispositivos de almacenamiento disponibles en cierto momento. De manera que el proceso de montar un sistema de ficheros conlleva que el contenido esté disponible, ya que con este montaje se une este sistema de ficheros al sistema de ficheros lógico global. Los datos de este sistema de ficheros montado se podrán acceder a partir de un directorio dentro del sistema de ficheros lógico global, que hace de punto de montaje (figura 1). Al desmontar un sistema de ficheros, éste 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, por tanto, es el directorio en el que se monta un sistema de ficheros, por lo que al acceder a ese directorio se accede a los datos de ese sistema de ficheros. En principio, un sistema de ficheros puede ser montado en cualquier directorio, lo que trae como consecuencia que temporalmente no se pueda acceder al contenido que previamente hubiera en ese directorio.

El sistema de ficheros raíz, donde se instala el sistema operativo, siempre está montado en el directorio `/` y no se puede desmontar. Durante el proceso de arranque, primero se monta el sistema de ficheros raíz y después el resto.

4.4.1. Órdenes

Las órdenes relacionadas con el montaje y desmontaje de sistemas de ficheros son:

- `mount`: Para montar un sistema de ficheros. Sintaxis:

  ```
  mount [opciones] FicheroEspecialBloque PtoMontaje
  ```

  Siendo las opciones más habituales:

  `-t tipo_sf`: tipo de sistema de ficheros.

  `-r`: montaje en modo solo lectura.

`-w`: montaje en modo lectura/escritura.

`-o`: opciones de montaje (`ro`, que es equivalente a `-r`; `rw`, que es equivalente a `-w`; `remount`, para poder cambiar algunas de las opciones de montaje; y otras como `nosuid`, `exec`, ...). Analizaremos varias de estas opciones de montaje en el siguiente apartado, cuando expliquemos el fichero de configuración `/etc/fstab`.

Ejemplos:

```
# mount -w -o remount /dev/sdb1 /media/disk
# mount -o usrquota,noexec,nosuid /dev/sda2 /home
```

- `umount`: Para desmontar un sistema de ficheros. Solamente se podrá desmontar si no está siendo utilizado (no está ocupado). Sintaxis:

```
umount PtoMontaje
umount FicheroEspecialBloque
```

- `fuser`: Para saber qué ficheros se están usando y qué procesos los usan (`f`: fichero abierto, `c`: directorio de trabajo, `e`: ejecutando un fichero, etc.). De entre las distintas opciones disponibles, a través de la opción `-v` podemos indicarle que nos dé una salida detallada y con la opción `-m` el punto de montaje a considerar.

  Ejemplo:

```
# mount -w /dev/sdb1 /media/disk
# cd /media/disk
# umount /media/disk
umount: /media/disk: device is busy.
# /sbin/fuser -mv /media/disk/
                     USER        PID ACCESS COMMAND
/media/disk:         root     kernel mount /media/disk
                     alumno     2246 ..c.. bash
```

  Observa como se monta primero en el directorio `/media/disk` el sistema de ficheros contenido en `/dev/sdb1` para lectura/escritura. A continuación se cambia el directorio actual a dicho punto de montaje, para, seguidamente, tratar de desmontar sin éxito el sistema de ficheros (se nos informa que está ocupado). Utilizando la orden `fuser` podemos ver que el proceso `bash` está utilizando dicho directorio como directorio de trabajo (`c`), por lo tanto, impidiendo que pueda ser desmontado el sistema de ficheros.

4.4.2. Fichero `/etc/fstab`

El fichero de configuración `/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:

```
fi_bl|UUID=|LABEL=   pto_montaje   tipo   opciones   dump_f   pass_num
```

Siendo el significado de cada campo:

- `fi_bl`: fichero especial de bloques. Alternativamente se puede utilizar el UUID del sistema de ficheros, precediéndolo de `UUID=`, o incluso podríamos definir (mediante las órdenes `e2label` o `fatlabel`, dependiendo del tipo del sistema de ficheros) una etiqueta para referirnos al dispositivo e indicarla aquí anteponiendo `LABEL=`. Observa que aunque las etiquetas podrían ser más representativas que el UUID, podrían repetirse y crear situaciones problemáticas. El UUID, por el contrario, es único.

- `pto_montaje`: directorio que sirve de punto de montaje.
- `tipo`: tipo de sistema de ficheros (Ext3, Ext4, vfat, iso9660, swap,...).
- `opciones`: las opciones para el montaje (separadas por comas y sin espacios). Observa que estas mismas opciones podrían usarse con la opción `-o` de la orden `mount`.
- `dump_f`: frecuencia para hacer una copia de seguridad del SF con la orden `dump` (actualmente este campo no se usa).
- `pass_num`: en tiempo de arranque, en qué orden hay que chequear los sistemas de ficheros (ejecutar `fsck` para comprobar su estado). Sus posibles valores son:
  - `0`: indica que no se chequea.
  - `1`: indica que se chequea el primero (sólo el SF raíz debe tener este valor).
  - `2,3,4, ...`: indica que será el segundo, tercero, cuarto, ..., en chequear.

Algunas de las posibles opciones de montaje de un sistema de ficheros y que, por tanto, se pueden incluir en el campo `opciones` de cada línea del fichero `/etc/fstab` son:

- `rw`: Lectura-escritura.
- `ro`: Sólo lectura.
- `suid/nosuid`: Si se selecciona la opción `nosuid` se impide que los bits `SETUID` y `SETGID` tengan efecto en aquellos ficheros ejecutables del sistema de ficheros que los tengan activos. La utilidad de estos permisos «especiales» se vio en el boletín 5.
- `auto/noauto`: Montar automáticamente durante el arranque, o no montar automáticamente.
- `exec/noexec`: Permitir o no la ejecución de ficheros.
- `usrquota`, `grpquota`: Permitir o no cuotas de usuario y de grupo. Obviamente esta opción estaría disponible únicamente si el sistema de ficheros soporta un mecanismo de cuotas, cosa que ocurre, por ejemplo, en los sistemas de ficheros nativos de Linux (hablaremos sobre las cuotas en el apartado 5).
- `users`: se 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 con la orden `mount`, las opciones de montaje (las que se especifican en la orden `mount` tras `-o`) sean `noexec`, `nosuid` y `nodev` (esta última prohibe el acceso a cualquier fichero especial de dispositivo que pueda haber en el sistema de ficheros). De esta forma, si queremos combinarla, por ejemplo, con la opción `exec`, la opción `users` habrá de ir primero (es decir, `users,exec`). De lo contrario, si se pone primero `exec`, quedaría anulada por la opción `noexec` que `users` implica.
- `defaults`: Marcar las opciones de montaje por defecto. Entre otras se incluyen `rw`, `suid`, `exec`, `auto`, `nouser` (el sistema de ficheros no puede ser montado por un usuario distinto al superusuario) 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 para sistemas de ficheros vfat**): con sistemas de ficheros vfat, sirve para definir el propietario (`X`) y grupo propietario (`Y`) de los ficheros del sistema de ficheros respectivamente. Recordemos que, en los sistemas de ficheros Ext2, Ext3 y Ext4 para cada fichero existe un nodo–i donde se guardan sus atributos. Entre los atributos del fichero está la información del propietario y el grupo propietario. Sin embargo, en un sistema de ficheros vfat los ficheros no tienen asociados ni propietario ni grupo propietario, por lo que al montar este tipo de sistemas de ficheros en Linux es necesario establecer este propietario global para todos

los ficheros del sistema de ficheros, de cara a gestionar el sistema de protección de Linux para cada fichero, mediante los permisos habituales que conlleva la lista de control de acceso de dicho fichero.

- `umask=XXX` (**solo para sistemas de ficheros vfat**): en un sistema de ficheros vfat los ficheros no tienen asociados permisos, por lo que al montar este tipo de sistemas de ficheros en Linux es necesario establecerlos de forma global para todos los ficheros del sistema de ficheros a través de los tres dígitos octales indicados en esta opción. Es decir, con esta opción se establecen 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. A modo de ejemplo, si `XXX` es `137`, los permisos quedarían como `640`, pues lo indicado en el campo `umask` es el inverso de los permisos a establecer. De igual forma se podría establecer esta máscara únicamente para directorios, con la opción `dmask`, o bien, únicamente para ficheros ordinarios, con la opción `fmask`.

Como decíamos, podemos utilizar etiquetas para referirnos al sistema de ficheros dentro de un dispositivo de bloques. Para ello, si el sistema de ficheros es Ext2, Ext3 o Ext4 usaríamos la orden `e2label`, mientras que para los sistemas de ficheros vfat y NTFS usaríamos `fatlabel` y `ntfslabel`, respectivamente, para crear la etiqueta. Por ejemplo, si suponemos que el dispositivo `/dev/sdb1` contiene un sistema de ficheros Ext4, podríamos crear una etiqueta `nuevodisco` de la siguiente forma:

```
# e2label /dev/sdb1 nuevodisco
```

Una vez definida, la etiqueta se puede usar en `/etc/fstab` para referirnos al dispositivo como se ha explicado.

Ejemplos de líneas en el fichero `/etc/fstab`:

```
/dev/mapper/fedora-root                   /              ext4    defaults  1 1
UUID=92adbe3c-835a-4fe5-ac2f-64440f7e7c4c /boot          ext4    defaults  1 2
/dev/mapper/fedora-swap                   swap           swap    defaults  0 0
LABEL=nuevodisco                          /media/disk    ext4    defaults  1 0
```

Recuerda que a través de la orden `lsblk -f` podemos obtener información acerca de los dispositivos de bloques disponibles, entre otras cosas el UUID y el tipo del sistema de ficheros que cada uno de ellos contiene.

A la hora de montar un sistema de ficheros con la orden `mount`, se pueden escoger directamente las opciones de montaje:

```
# mount -t ext4 -r -o noexec /dev/sdb1 /media/disk
```

o bien, dejar que se utilicen las opciones detalladas para este sistema de ficheros en el fichero de configuración `/etc/fstab` (en el caso de que exista una línea para dicho sistema de ficheros):

```
# mount /media/disk
```

Es importante notar que tras el montaje de un sistema de ficheros de tipo Ext2, Ext3 o Ext4, el punto de montaje tomará los permisos, propietario y grupo del directorio raíz del propio sistema de ficheros que se monta (usualmente el propietario y el grupo será el `root`). Una vez montado, podemos modificar dicha información para permitir acceso a otros usuarios (ajustando los permisos de la forma adecuada).

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

A pesar de que se asigne permisos de montaje a los usuarios (con la opción `users` por ejemplo), estos no pueden especificar opciones extra. De manera que, según el ejemplo anterior, un usuario normal (sin privilegios de superusuario) no podría ejecutar:

```
# mount -o noexec /media/disk
```

Para montar todos los sistemas de ficheros indicados en `/etc/fstab` (salvo los ya montados y los que usen la opción `noauto`) ejecutaremos:

```
# mount -a
```

Para obtener información general sobre todos los sistemas de ficheros que están montados en un momento dado podemos ejecutar la orden `mount` sin argumentos ni opciones. Otra orden interesante que nos muestra esta información en forma de árbol es `findmnt`, cuya salida puede verse a continuación:

```
# findmnt
TARGET                                SOURCE     FSTYPE  OPTIONS
/                                     /dev/mapper/fedora-root ext4 rw,relatime,seclabel,data=orde
├─/sys                                sysfs      sysfs   rw,nosuid,nodev,noexec,relatim
│ ├─/sys/kernel/security              securityfs securit rw,nosuid,nodev,noexec,relatim
│ ├─/sys/fs/cgroup                    tmpfs      tmpfs   ro,nosuid,nodev,noexec,seclabe
│ │ ├─/sys/fs/cgroup/systemd          cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/net_cls,net_prio cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/blkio            cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/perf_event       cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/freezer          cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/cpuset           cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/memory           cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/devices          cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/cpu,cpuacct      cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ ├─/sys/fs/cgroup/pids             cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ │ └─/sys/fs/cgroup/hugetlb          cgroup     cgroup  rw,nosuid,nodev,noexec,relatim
│ ├─/sys/fs/pstore                    pstore     pstore  rw,nosuid,nodev,noexec,relatim
│ ├─/sys/fs/selinux                   selinuxfs  selinux rw,relatime
│ ├─/sys/kernel/debug                 debugfs    debugfs rw,relatime,seclabel
│ ├─/sys/kernel/config                configfs   configf rw,relatime
│ └─/sys/fs/fuse/connections          fusectl    fusectl rw,relatime
├─/proc                               proc       proc    rw,nosuid,nodev,noexec,relatim
│ ├─/proc/sys/fs/binfmt_misc          systemd-1  autofs  rw,relatime,fd=27,pgrp=1,timeo
│ └─/proc/fs/nfsd                     nfsd       nfsd    rw,relatime
├─/dev                                devtmpfs   devtmpf rw,nosuid,seclabel,size=100893
│ ├─/dev/shm                          tmpfs      tmpfs   rw,nosuid,nodev,seclabel
│ ├─/dev/pts                          devpts     devpts  rw,nosuid,noexec,relatime,secl
│ ├─/dev/hugepages                    hugetlbfs  hugetlb rw,relatime,seclabel
│ └─/dev/mqueue                       mqueue     mqueue  rw,relatime,seclabel
├─/run                                tmpfs      tmpfs   rw,nosuid,nodev,seclabel,mode=
│ ├─/run/user/42                      tmpfs      tmpfs   rw,nosuid,nodev,relatime,secla
│ └─/run/user/1000                    tmpfs      tmpfs   rw,nosuid,nodev,relatime,secla
│   └─/run/user/1000/gvfs             gvfsd-fuse fuse.gv rw,nosuid,nodev,relatime,user_
├─/tmp                                tmpfs      tmpfs   rw,seclabel
├─/boot                               /dev/sda1  ext4    rw,relatime,seclabel,data=orde
├─/var/lib/nfs/rpc_pipefs             sunrpc     rpc_pip rw,relatime
└─/media/disk                         /dev/sdb1  ext4    rw,nosuid,nodev,noexec,relatim
```

Además, a `findmnt` podemos darle como parámetro un directorio y en este caso nos indicará si hay algún sistema de ficheros montado en el mismo, o un fichero especial de dispositivo y nos informará sobre si está montado (y dónde) o no.

4.5. Control y gestión del espacio de un sistema de ficheros

Una vez creado el sistema de ficheros es preciso disponer de una serie de órdenes a través de las cuales se pueda obtener información acerca de la ocupación del mismo, para saber por ejemplo, cuánto espacio hay aún disponible o la cantidad de espacio que un directorio concreto puede estar ocupando. Para ello vamos a ver en detalle las órdenes `df` y `du`.

4.5.1. La orden `df`

La orden `df` informa de la capacidad de un sistema de ficheros, del espacio libre y el usado, y de su punto de montaje. En su ejecución por defecto, la primera columna muestra el nombre de la partición

tal como aparece en el directorio `/dev`. Las columnas siguientes muestran el espacio total, bloques asignados y bloques disponibles. Por ejemplo:

```
# df

S.ficheros                              bloques de 1K  Usados Disponibles Uso% Montado en
...
/dev/mapper/fedora_localhost--live-root       8187320 5759784     1991928  75% /
/dev/sda1                                      999320  188484      742024  21% /boot
...
```

La sintaxis de uso de esta orden es:

```
df [opciones]
```

siendo las opciones más importantes:

- `-h`: Muestra los tamaños en formato más legible (es decir, en KiB, MiB o GiB, por ejemplo). Así, la salida que obtendríamos en el caso anterior utilizando esta opción sería:

  ```
  # df -h

  S.ficheros                             Tamaño Usados  Disp Uso% Montado en
  ...
  /dev/mapper/fedora_localhost--live-root   7,9G   5,5G  1,9G  75% /
  /dev/sda1                                 976M   185M  725M  21% /boot
  ...
  ```

- `-i`: Lista información de nodos-i en vez de uso de bloques. De nuevo, para el ejemplo anterior obtendríamos:

  ```
  S.ficheros                            Nodos-i NUsados NLibres NUso% Montado en
  ...
  /dev/mapper/fedora_localhost--live-root  524288  164602  359686   32% /
  /dev/sda1                                 65536     452   65084    1% /boot
  ...
  ```

- `-T`: Muestra el tipo de sistema de ficheros.

Llegados a este punto hay que hacer mención especial al sistema de ficheros raíz. Es importante tener en cuenta que si este sistema de ficheros se quedase sin espacio, o bien sin nodos-i libres, el sistema tendría problemas para funcionar normalmente, ya que, por ejemplo, no podría arrancar.

4.5.2. La orden `du`

La orden `du` se usa para informar de cuánto espacio en disco ocupa un fichero o directorio. Si se indica un directorio, se lista, además, el espacio ocupado por todos los subdirectorios encontrados a partir de él. Esta información se muestra, por omisión, en unidades de 1 KiB.

La sintaxis de uso de esta orden es:

```
du [Opciones] [Directorio]...
```

Si se utiliza sin opciones, nos informa acerca del tamaño de cada subdirectorio de cada uno de los directorios especificados (el actual si no se indica nada), así como el tamaño total del directorio. Algunas de las opciones de uso más frecuente son:

- `-a`: Cuando se indica un directorio, muestra también el uso de espacio de cada fichero.
- `-s`: En vez de la salida por defecto, informa sólo 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 formato más legible (es decir, en KiB, MiB o GiB, por ejemplo).

5. 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. Las cuotas de disco en los sistemas de ficheros nativos de Linux (Ext2, Ext3 y Ext4) permiten limitar el número de bloques y/o número de nodos-i que un usuario puede usar en una partición. Estos límites también se pueden establecer para grupos de usuarios. Hay dos tipos de límites:

- Límite hard: el usuario no puede sobrepasarlo. Cuando lo alcance ya no podrá usar más bloques de datos o crear más ficheros (nodos-i).
- Límite soft: es inferior al límite hard y 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 le 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 para nodos-i. Obviamente, sólo tienen sentido en sistemas de ficheros donde pueden escribir los usuarios.

A continuación se enumeran los pasos a seguir para activar cuotas para los usuarios en el sistema de ficheros contenido en `/dev/sdb1` y al que anteriormente asignamos la etiqueta `nuevodisco`. El sistema de ficheros 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
   ```

2. Remontar la partición para que se active la opción `usrquota`:

   ```
   # mount -o remount LABEL=nuevodisco
   ```

3. Crear el fichero de control de cuotas, `aquota.user`, con `quotacheck`:

   ```
   # quotacheck -nm /media/disk
   ```

4. Activar las cuotas:

   ```
   # quotaon /media/disk
   ```

5. Para cada usuario, establecer su cuota con la orden `setquota` de la siguiente forma:

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

   Si queremos dejar algún límite sin establecer, usaríamos el valor 0. Por defecto, los valores `blocks_soft` y `blocks_hard` se interpretan como número de bloques de 1 KiB. Alternativamente, podemos especificar los tamaños añadiendo al valor numérico los símbolos `K`, `M`, `G` y `T`. Los valores `inodes_soft` y `inodes_hard` son interpretados de forma literal, pudiendo añadirles también los símbolos `k`, `m`, `g` y `t` para especificarlos como múltiplos de 10^3, 10^6, 10^9 y 10^{12}, respectivamente.

   Por ejemplo, para establecer para el usuario `alumno` los límites hard y soft de bloques de datos en 70 MiB y 50 MiB respectivamente, y los límites hard y soft de nodos-i en 15.000 y 10.000 respectivamente, habría que ejecutar:

   ```
   # setquota -u alumno 50M 70M 10k 15k  /media/disk
   ```

6. Establecer el periodo de gracia (especificado en segundos) para bloques y nodos-i a través de la opción `-t` de la orden `setquota` de la siguiente forma:

   ```
   setquota -t <blocks_grace> <inodes_grace> <sistfich>
   ```

   Para el ejemplo anterior, si queremos establecer un periodo de gracia de 5 días ($5 \times 24 \times 60 \times 60 = 432\,000$ segundos), habría que ejecutar:

   ```
   # setquota -t 432000 432000  /media/disk
   ```

Finalmente, a través de la orden `repquota <sistfich>` el administrador puede obtener una estadística de las cuotas para todos los usuarios. Un usuario, por su parte, puede utilizar la orden `quota` (sin parámetros) para obtener información acerca de las cuotas que le han sido establecidas sobre distintos sistemas de ficheros.

6. Creación de enlaces

En general, podemos decir que un enlace es una «conexión» entre un nombre de fichero y el contenido del mismo (los datos reales que hay en el disco). Según vimos en el tema 4 de la parte de teoría, en un sistema de ficheros Linux podemos manejar dos tipos de enlaces: enlaces físicos (o *hard links*) y enlaces simbólicos (o *soft links*). El primer tipo de enlace está referido a la conexión entre una entrada de directorio y el nodo-i correspondiente (podríamos tener varias entradas de directorio apuntando al mismo nodo-i y sería el caso de un fichero compartido a través de enlaces físicos). Por el contrario, el segundo tipo de enlaces son ficheros especiales, cuyo contenido es la ruta de acceso al fichero que se enlaza.

Para la creación de los enlaces físicos y simbólicos disponemos en Linux de la orden `ln`, cuya sintaxis de uso más habitual es:

```
ln [Opciones] FicheroEnlazado NombreEnlace
```

Si utilizamos `ln` sin opciones, se creará un enlace físico. Por ejemplo:

```
# ln /bin/ls enlace_a_ls
# ls -l enlace_a_ls
-rwxr-xr-x. 2 root root 133168 abr 28  2017 enlace_a_ls
```

Vemos cómo se ha creado la entrada de directorio `enlace_a_ls`, que contiene el mismo número de nodo-i que `/bin/ls`, y cómo la segunda columna nos muestra el número de enlaces a dicho nodo-i (el contador de enlaces con valor 2 en este caso).

Por el contrario, si queremos crear un enlace simbólico debemos usar la opción `-s` (o `--symbolic`). Por ejemplo:

```
# ln -s /bin/ls enlace_a_ls_simbolico
# ls -l enlace*
-rwxr-xr-x. 2 root root 133168 abr 28  2017 enlace_a_ls
lrwxrwxrwx. 1 root root      7 may 15 13:18 enlace_a_ls_simbolico -> /bin/ls
```

Vemos cómo esta vez se ha creado un fichero de tipo *enlace simbólico* (date cuenta de la letra `l` que aparece antes de los permisos) y cuyo contenido es simplemente la ruta a `ls` que hemos puesto (7 caracteres, que son los 7 bytes que en este caso tiene de tamaño el fichero).

7. Copias de seguridad

Para prevenir una posible pérdida de información, el administrador del sistema debe planear e implementar una política de copias de seguridad que periódicamente guarde los ficheros del sistema. También es responsabilidad del administrador hacer que las copias de seguridad se hagan en el momento oportuno, y que los datos guardados están en un lugar seguro. Hay que considerar que el esfuerzo que se dedique a planear e implementar la estrategia de copias de seguridad no es en balde, ya que evitará la posible pérdida de datos en el futuro.

7.1. Tipos de copias de seguridad

Podemos hablar de tres estrategias generales para abordar las copias de seguridad de un sistema de ficheros:

1. Copia de seguridad completa de un sistema de ficheros: En una copia completa se copian todos los ficheros del sistema de ficheros, por lo que este tipo de copia tarda mucho tiempo en realizarse y, además, una vez obtenida la copia de seguridad, la recuperación de un único fichero concreto puede no ser inmediata. Por todo ello, esta estrategia de copia está justificada solamente si todos los ficheros cambian mucho y son vitales para el trabajo de mucha gente, o bien, ante grandes cambios: nuevo software, nuevo sistema operativo... Sin embargo, si los ficheros no cambian muy a menudo, o cambian con frecuencia solo unos pocos ficheros, no es la estrategia más aconsejable.

2. Copia diferencial de un sistema de ficheros: La copia diferencial copia únicamente los ficheros que han sido creados y/o modificados desde la última copia completa. Esto viene a suponer que si el lunes hemos realizado una copia completa y el martes ejecutamos una copia diferencial, únicamente se copiarán los ficheros creados o modificados durante el martes (es decir, desde el instante en que se realizó la copia completa hasta ahora). Este mismo comportamiento se efectuará si la lanzamos el miércoles, tomando la copia completa del lunes como base e incluyendo todos los ficheros creados o modificados durante el martes o el miércoles.

3. Copia incremental de un sistema de ficheros: En las copias incrementales se copian solo aquellos ficheros que han cambiado o se han creado desde la última copia de seguridad, por lo que se deben realizar con mucha más frecuencia, incluso a diario. Esta estrategia se usa en conjunto con la primera, de forma que se tiene una copia completa del sistema realizada con poca frecuencia, y si se producen cambios muy pequeños se irán haciendo copias incrementales para tenerlos recogidos. Para el ejemplo anterior, la segunda copia de seguridad, si se hace incremental, incluiría solamente los ficheros creados o modificados durante el miércoles (desde la anterior copia incremental). 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 (la copia completa inicial).

7.2. Herramientas de copias de seguridad

A través de la orden `dd`, que ya vimos en el boletín anterior, podríamos hacer una copia completa de un sistema de ficheros. Por ejemplo, si tenemos insertado un pendrive con una única partición en la máquina virtual, podríamos respaldar el contenido de su sistema de ficheros en un fichero a través de la ejecución de la orden:

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

Obviamente, podríamos acceder al contenido de la copia de seguridad a través de un dispositivo *loop* como también vimos en el boletín anterior, asignándolo al fichero creado y montándolo posteriormente:

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

Podríamos recuperar también la copia de seguridad en una partición de otro pendrive ejecutando de nuevo la orden `dd`:

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

Obviamente, la partición destino (`/dev/sdf1`) debería tener, al menos, el mismo tamaño que el fichero imagen.

Observa que `dd` realiza una copia «a ciegas», es decir, byte a byte, sin importar que el sistema de ficheros pudiese estar casi vacío. Proceder de esta forma resulta, por tanto, bastante lento. En Linux se dispone, sin embargo, de otras herramientas más sofisticadas con las que podremos efectuar copias de seguridad diferenciales e incrementales y que se examinan a continuación.

7.2.1. La orden `tar`

La herramienta `tar` realiza copias de seguridad de ficheros, directorios (operando de forma recursiva) o sistemas de ficheros. Por defecto, `tar` sólo empaqueta, aunque también puede comprimir usando internamente `gzip` o `bzip`, entre otros.

Por defecto realiza copias completas. Para implementar una estrategia de copias diferenciales o incrementales se pueden usar las opciones `--newer=DATE-OR-FILE` (`--after-date=DATE-OR-FILE` o -N) y `--newer-mtime=DATE-OR-FILE`, de cara a seleccionar los ficheros, comparando sus respectivas fechas[2] (`ctime` en el primer caso y `mtime` en el segundo) con la fecha de la última copia de seguridad completa o incremental realizada (indicando el fichero que almacena dicha última copia de seguridad a través de alguna de las opciones anteriores). A la hora de especificar la fecha, se pueden utilizar varios fomatos, como por ejemplo `--newer="2020-10-1"` o `--newer="1 week ago"`. Se podría añadir también información sobre la hora, por ejemplo `--newer="2020-10-1 10:00:00"` (si se indica solamente información sobre la hora se usará la fecha actual). Si queremos que se tome la fecha y hora de un fichero existente, tendremos que indicar el nombre del fichero comenzando por «.» o «/», como por ejemplo `--newer-mtime="/home/alumno/copias_de_seguridad/20200901.tgz"`.

Algunas de las opciones más comunes que se usan con esta orden son:

- `c`: Crea un fichero contenedor.
- `x`: Extrae ficheros de un fichero contenedor.
- `t`: Muestra una tabla de contenidos, es decir lista los ficheros almacenados en un fichero contenedor.
- `r`: 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.
- `f`: 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.

A continuación se muestran algunos ejemplos de uso:

- `tar czvf practicas.tgz prac_iso`

  copia el directorio `prac_iso` en el fichero `practicas.tgz`
- `tar tzvf practicas.tgz`

  lista el contenido de la copia de seguridad realizada en el fichero

[2]Recuerda que cuando se explicó la orden `find` en el boletín 1, distinguimos entre la fecha de modificación de un fichero (`mtime`), la fecha del último acceso (`atime`) y la fecha del último cambio de estado, por ejemplo, de los permisos (`ctime`).

- `tar xzvf practicas.tgz`

  restaura la copia de seguridad

- `tar xzvf practicas.tgz prac_iso/boletin1.pdf`

  recupera el fichero `boletin1.pdf` (observa que hay que indicar la ruta con la que `tar` lo almacenó)

7.2.2. La orden `cpio`

La herramienta `cpio` realiza copias de seguridad de conjuntos de ficheros. Esta orden lee de la entrada estándar los nombres de los ficheros a guardar, por lo que normalmente se usa enlazada con otras órdenes mediante tuberías, especialmente con la orden `find`. La orden `cpio` guarda los nombres de los ficheros tal cual se le pasan (con la ruta dada) y serán así como se restauren después.

Al igual que `tar`, esta herramienta se puede emplear para realizar copias completas, o copias diferenciales o incrementales. Esto último puede realizarse utilizando, por ejemplo, la opción `-newer` de la orden `find`, de forma que podrían seleccionarse (y respaldarse) los ficheros cuya fecha de modificación fuese posterior a la fecha de modificación de la última copia de seguridad completa o incremental realizada (habrá que pasar la ruta del fichero que contiene dicha última copia a la opción `-newer`).

Algunas de las opciones más frecuentemente usadas con la orden `cpio` son:

- `-o`: Para copiar hacia fuera (out), es decir, para crear la copia de seguridad.
- `-i`: Para copiar hacia dentro (in), es decir para desempaquetar una copia de seguridad.
- `-t`: Para crear una tabla de contenidos, es decir, para mostrar el contenido de la copia.
- `-A`: Para añadir nuevos ficheros a un contenedor de copia seguridad ya existente.
- `-d`: Para crear directorios al desempaquetar.
- `-m`: Para conservar fecha y hora de los ficheros al desempaquetar.
- `--no-absolute-filenames`: Si se creó el respaldo usando rutas absolutas, al desempaquetar una copia de seguridad extrae los ficheros relativos al directorio actual.
- `-v`: Modo verbose.
- `-F`: Seguido del nombre del fichero a usar como contenedor de la copia de seguridad.

A continuación se muestran algunos ejemplos de uso:

- `find /home | cpio -o -F copia_home.cpio`

  copia todos los ficheros del directorio `/home` en el fichero `copia_home.cpio`. Los nombres de los ficheros se guardan con las rutas absolutas devueltas por `find`.

- `find /home | cpio -o >copia_home.cpio`

  Ídem a la anterior.

- `find . -mtime -7 | cpio -o >semana.cpio`

  copia a partir del directorio actual los ficheros modificados la última semana en el fichero `semana.cpio`. Observa que ahora los ficheros se guardan con rutas relativas.

- `cpio -t <copia_home.cpio`

  lista el contenido de la copia de seguridad realizada en el fichero `copia_home.cpio`.

- `cpio -i <copia_home.cpio`

  restaura la copia de seguridad contenida en el fichero `copia_home.cpio`.

- `cpio -i -F copia_home.cpio alumno/practicas/doc.pdf`

  restaura sólo el fichero `doc.pdf` de la copia almacenada en el fichero `copia_home.cpio`.

7.2.3. La orden `rsync`

La herramienta `rsync` permite realizar copias de ficheros de manera local en una máquina o bien hacia/desde otro ordenador (en este caso, el servicio `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. A diferencia de las órdenes `tar` y `cpio`, que crean un fichero que contiene la copia de seguridad, `rsync` «sincroniza» el contenido del directorio destino con el del directorio origen, de tal manera que, tras la ejecución de la orden, el directorio destino tendrá, al menos, los mismos ficheros que el directorio origen, incluyendo el contenido de cualquier subdirectorio existente.

Algunas opciones de esta orden son:

- `-a`: Modo archivo. Indica que se quiere hacer un recorrido recursivo para buscar los ficheros a copiar y que además se quieren preservar las características más importantes de los mismos: enlaces simbólicos, atributos, permisos, etc.
- `-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 la opción `-v` puede ser muy útil para ver el efecto que produciría la ejecución de la orden al usar determinadas opciones, ficheros a copiar, ordenador remoto, etc.
- `--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.

Algunos ejemplos de uso serían:

- `rsync -avz $HOME/datos remoto.inf.um.es:$HOME/datosremoto`

  transfiere el directorio `$HOME/datos`, incluyendo recursivamente todo su contenido, al directorio `$HOME/datosremoto/datos` de la máquina `remoto.inf.um.es`. Los ficheros son transmitidos en modo «archivo», lo que asegura que se preservan los enlaces simbólicos, atributos, permisos,... Además, se comprimen los datos a transmitir.

- `rsync -avz` **`--delete`** `$HOME/datos remoto.inf.um.es:$HOME/datosremoto`

  como el ejemplo anterior, pero además se borrarán del directorio remoto `$HOME/datosremoto/datos` aquellos ficheros que previamente hubieran sido borrados en el directorio origen.

- `rsync -avz --delete` **`--backup --backup-dir=$HOME/datosviejos`** `$HOME/datos remoto.inf.um.es:$HOME/datosremoto`

  como el ejemplo anterior, pero además se usará en el ordenador remoto el directorio de backup incremental `$HOME/datosviejos` para guardar las versiones antiguas de los ficheros reemplazados y/o eliminados del directorio remoto `$HOME/datosremoto/datos` al realizar la sincronización.

En los ejemplos anteriores, si sustituimos `remoto.inf.um.es` por `localhost`, el ordenador «remoto» será nuestro propio equipo local, y por tanto, podremos hacer pruebas sin necesidad de disponer de otra máquina.

Como ya hemos comentado, antes de realizar la sincronización es necesario que nos aseguremos de que el servicio `sshd` se encuentre activo en el ordenador remoto (si usamos `localhost` deberá estar activo en nuestro ordenador local). Para ello bastará con ejecutar `systemctl status sshd`. Si estuviese inactivo, podemos activarlo ejecutando como superusuario `systemctl start sshd`.

También es importante hacer notar que, por defecto, las versiones más recientes de Fedora impiden, por motivos de seguridad, iniciar una sesión ssh con la cuenta root. Dado que `rsync` emplea ssh para sincronizar el contenido de un directorio local con una máquina remota, es importante que deshabilitemos dicha restricción si queremos, como administradores, realizar copias de seguridad utilizando la cuenta de superusuario. Para ello habrá que editar el fichero `/etc/ssh/sshd_config` como superusuario y establecer a «yes» la opción `PermitRootLogin` (por ejemplo, añadiendo la línea `PermitRootLogin yes` al final de dicho fichero). Para que el cambio en la configuración surta efecto, seguidamente será necesario reiniciar el servicio `sshd` a través de la ejecución de la siguiente orden `systemctl restart sshd` como superusuario.

8. Ejercicios

8.1. Gestión básica de un sistema de ficheros

1. Crea un sistema de ficheros `Ext4` en la primera partición del disco `/dev/sdb` de la máquina virtual. Para ello, crea primero una partición que ocupe todo el espacio disponible en el disco.

2. Asocia la etiqueta `sfext4` con el nuevo sistema de ficheros (usa para ello la orden `e2label`).

3. Define en el fichero `/etc/fstab` una entrada para montar el sistema de ficheros de la partición que acabas de crear (es decir `/dev/sdb1`) utilizando la etiqueta `sfext4` y con las siguientes características:

   - Sistema de ficheros ext4.
   - Modo de lectura/escritura.
   - El montaje no se realizará automáticamente.
   - Se permite el montaje a los usuarios.
   - No se permitirá la ejecución de ficheros.
   - El punto de montaje debe ser el directorio `/miparticion` (hay que crear este directorio previamente)

4. Tras esto y como superusuario, monta el sistema de ficheros creado y dale permiso de escritura a todos los usuarios para asegurarte de que estos pueden crear ficheros dentro de él (deberás cambiar los permisos de `/miparticion` una vez montado el sistema de ficheros). Desmonta a continuación el sistema de ficheros.

5. Con el usuario `alumno` monta el sistema de ficheros de la partición `/dev/sdb1`, usando la entrada creada para él en `/etc/fstab`. Lista ahora todas los sistemas de ficheros montados (recuerda que hay distintas formas de obtener esta información).

6. A continuación crea un fichero dentro del directorio `/miparticion` de 200 MiB (usa la orden `dd` para ello) y obtén seguidamente cuánto espacio libre y ocupado (en MiB) hay en el sistema de ficheros creado en `/dev/sdb1` (y montado en `/miparticion`).

7. Dentro de `/miparticion`, crea dos directorios, uno llamado `copia-b` y otro llamado `copia-d`. Copia dentro del primero todos los ficheros del directorio `/bin` cuyo nombre comience por la letra «b», y dentro del segundo, todos los ficheros del directorio `/bin` cuyo nombre comience por la letra «d». Mediante una sola orden, obtén el tamaño del directorio `/miparticion` y de

todos sus subdirectorios (en KiB y/o MiB). ¿Qué porcentaje del espacio del sistema de ficheros queda aún libre? ¿Y de nodos-i?

8. Dentro de `/miparticion/copia-b` se ha copiado el intérprete de órdenes `bash`. Trata de ejecutarlo mediante `/miparticion/copia-b/bash`. ¿Lo permite? (Si no indicas la ruta absoluta, ejecutará el fichero `/bin/bash` que es el primero que encuentra al recorrer los directorios indicados en `PATH`). Si no lo permite, ¿qué habría que cambiar para solucionarlo? Hazlo (como superusuario) y comprueba que, efectivamente, ahora sí puedes.

9. Como superusuario, usa un editor (por ejemplo, `gedit` para crear un fichero, llamado `datos`, en `/miparticion`, y escribe en su interior una pequeña frase. No cierres el editor. ¿Qué tamaño (en bytes) tiene el fichero (usa la orden `ls`)? ¿Cuántos bloques se le han asignado? ¿Qué tamaño de bloque se está usando? Puedes usar la orden `tune2fs -l /dev/sdb1` para comprobar las características del sistema de ficheros (entre otras, el tamaño de bloque).

10. Estando dentro de `/miparticion`, intenta desmontar el sistema de ficheros. ¿Por qué no deja desmontarlo?

11. Con la orden `fuser` averigua qué procesos están haciendo uso de ese sistema de ficheros y qué ficheros se están usando.

12. Cierra los procesos que sean necesarios y sal del directorio `/miparticion` de forma que sea posible desmontarlo. A continuación, desmonta el sistema de ficheros.

13. Como superusuario, crea ahora un sistema de ficheros `vfat` en un dispositivo *loop* asociado a un fichero de 40 MiB llamado `mifichero.img`. Para ello crea primero el fichero en `/home/alumno` y asócialo al fichero de dispositivo `/dev/loop0`. Es decir, en el directorio `/home/alumno` debes tener un fichero denominado `mifichero.img` y que habrás vinculado con `/dev/loop0`.

14. Define en el fichero `/etc/fstab` una entrada para montar el sistema de ficheros que acabas de crear (es decir el asociado con `/dev/loop0`) utilizando su UUID y con las siguientes características:

    - Modo de lectura/escritura.
    - El montaje se realizará automáticamente.
    - El propietario de los ficheros será el usuario `alumno` y el grupo propietario `users`.
    - Los permisos serán `rwxrw----` (ten en cuenta que se ponen mediante la opción de máscara, `umask`, indicando, en octal, la inversa de los permisos que se desean).
    - Punto de montaje: `/mifichero` (hay que crear este directorio previamente).

15. Como administrador, monta el sistema de ficheros y crea dentro del directorio `/mifichero` un fichero y comprueba qué propietario y grupo se han asignado, así como los permisos con los que se ha creado.

16. Sin cambiar nada en el fichero `/etc/fstab`, re-monta el sistema de ficheros como solo lectura y comprueba que aún con privilegios de superusuario no puedes crear un nuevo fichero en él.

17. Desmonta el sistema de ficheros.

18. Ejecuta la orden `reboot` y reinicia la máquina virtual. Verás que después de un rato intentando arrancar, da un error y se inicia en modo emergencia. Introduce la contraseña de «root». El problema está en la última línea que hemos introducido en el fichero `/etc/fstab`, y es que el sistema operativo es incapaz de encontrar a quién corresponde el UUID que añadimos (al haber perdido la vinculación con `/dev/loop0`). Podemos solucionarlo fácilmente, indicando al comienzo de esa línea la ruta absoluta del fichero donde está el sistema de ficheros a montar (es decir, quitamos del comienzo de la última línea el `UUID=...`, y en su lugar ponemos

`/home/alumno/mifichero.img`). Seguidamente, tecleamos la orden `reboot` y el sistema debe iniciarse ahora sin problemas.

19. ¿Se han montado los dos sistemas de ficheros creados en este apartado? Si no es así, haz los cambios necesarios para lograrlo cada vez que se arranque la máquina. Reinicia la máquina y asegúrate de que los dos ficheros quedan montados.

20. Vamos a practicar ahora con el redimensionamiento de un sistema de ficheros. Realiza para ello los siguientes ejercicios como superusuario:

    a) En el directorio raíz, crea un fichero de 100 MiB llamado `datospersonales.img`.

    b) Asocia el dispositivo *loop* `/dev/loop1` con dicho fichero y crea a través de él un sistema de ficheros `ext4`.

    c) Monta el sistema de ficheros creado en `/mnt` y comprueba el espacio que hay libre en el mismo.

    d) Desmonta dicho sistema de ficheros y haz una copia completa del mismo en el disco `/dev/sdc` de la máquina virtual (para hacer esto último, recuerda lo visto al comienzo del apartado 7.2).

    e) Monta ahora el sistema de ficheros copiado en el disco `/dev/sdc` y comprueba el tamaño de libre del mismo. ¿Se ha utilizado toda la capacidad del disco?

    f) Por último, redimensiona el sistema de ficheros de `/dev/sdc` para que ocupe todo el disco. Una vez hecho, comprueba que efectivamente el tamaño total del mismo ha crecido hasta cubrir el de `/dev/sdc`

8.2. Cuotas de disco

21. Activa las cuotas de disco en el sistema de ficheros que creaste en `/dev/sdb1` y que tienes montado en el directorio `/miparticion` de la máquina virtual[3], asignándole al usuario `alumno` 500 MiB de límite *soft* y 600 MiB de límite *hard*, así como un periodo de gracia de 11 días.

22. Entra al sistema con el usuario `alumno` y realiza los siguientes ejercicios:

    a) Usando la orden `quota`, obtén la información de la cuota asignada al usuario.

    b) Crea un fichero que exceda el límite *soft* asignado, para ello puedes usar la orden:

    ```
    $ dd if=/dev/zero of=/miparticion/fichero bs=1M count=X
    ```

    donde X es el número de bloques de 1 MiB que tendrá el fichero a crear. Usando la orden `quota`, comprueba si has logrado el objetivo.

    c) Crea ahora otro fichero a través del cual se trate de exceder el límite *hard* fijado al usuario y comprueba que no te deja sobrepasarlo.

    d) Intenta crear un nuevo fichero, por ejemplo con la orden "`echo hola > /miparticion/a`" y verifica que, aunque el fichero se crea, no es posible escribir nada en él.

    e) Borra los ficheros creados anteriormente para volver a dejar el consumo por debajo del límite *soft* y comprueba de nuevo el estado de la cuota asignada.

23. Como administrador, obtén un listado completo del estado de las cuotas de `/dev/sdb1` asignadas a todos los usuarios del sistema.

[3]Recuerda que las cuotas de disco sólo tiene sentido activarlas en los sistemas de ficheros donde los usuarios del sistema pueden escribir.

8.3. Enlaces físicos y simbólicos

24. En este ejercicio vamos a trabajar con enlaces físicos. Como usuario `alumno`, realiza lo siguiente:

    a) Crea, en el directorio `/miparticion`, un fichero llamado `origen` con el siguiente contenido: "Este es el fichero origen".

    b) Haz un enlace físico a ese fichero llamado `efisico` en ese mismo directorio. Recuerda que la orden `ln` es la que tienes que usar para crear el enlace físico; consulta la ayuda para saber cómo usar la orden.

    c) Con la orden `ls -il` comprueba los datos de los ficheros y verifica que son enlaces físicos: mismo nodo–i y, por tanto, mismas propiedades.

    d) Edita el fichero `efisico` y añade la línea "Modificando el enlace físico".

    e) Visualiza el fichero `origen` y comprueba que se ha añadido la línea del apartado anterior.

    f) A continuación, borra el fichero `origen`. Comprueba si puedes trabajar con el enlace físico. ¿Por qué sucede esto?

    g) Trata, finalmente, de crear el enlace físico `/home/alumno/efisico2` al fichero `/miparticion/efisico`. ¿Puedes? ¿Por qué sucede esto?

25. Ahora vamos a estudiar los enlaces simbólicos. De nuevo como usuario `alumno` haz lo siguiente:

    a) Crea un enlace simbólico llamado `esim` al fichero `efisico` creado en el ejercicio 24. Trabaja también en el directorio `/miparticion`.

    b) Edita el fichero `esim` y añade la línea "Modificando el enlace simbólico".

    c) Visualiza el fichero `efisico` y comprueba que se ha añadido la línea del apartado anterior.

    d) A continuación borra el fichero `efisico`. ¿Puedes trabajar con el enlace simbólico? ¿Puedes leer el fichero? ¿Qué sucede y por qué?

    e) Vamos a crear un nuevo fichero llamado `efisico`, ejecutando:

    ```
    $ echo "Creando de nuevo efisico" > /miparticion/efisico
    ```

    f) Comprueba que, aunque el fichero `efisico` es distinto del original, el enlace simbólico apunta a ese fichero. Así cuando se ejecuta:

    ```
    $ cat /miparticion/esim
    ```

    se muestra el nuevo fichero `efisico`. ¿Por qué sucede esto?

    g) Trata, finalmente, de crear el enlace simbólico `/home/alumno/esim2` al fichero `/miparticion/efisico`. ¿Puedes? ¿Por qué sucede esto?

8.4. Copias de seguridad

26. Como usuario `alumno` y dentro de tu directorio de inicio, haz una copia de seguridad comprimida con la orden `tar` en un fichero llamado `copseg1.tgz` de los ficheros del directorio `/sbin` que comienzan por la letra «c». Repite la copia en un fichero llamado `copseg2.tgz` pero ahora usa además la opción **h**. Restáuralas en directorios separados y compara el contenido de ambos directorios. ¿Qué diferencia has notado? Razónalo.

27. Extrae solo el fichero `sbin/cupsd` de la copia `copseg2.tgz`.

28. Usando ahora la herramienta `cpio` como superusuario, haz lo siguiente:

    a) Crea una copia de seguridad completa del directorio `/etc` en un fichero llamado `/root/copia-etc-total.cpio`.

b) Crea el directorio `/etc/bin` y copia a él los ejecutables `/usr/bin/ls` y `/usr/bin/cp`. A continuación, haz una copia de seguridad incremental de `/etc`, copiando sólo los ficheros que hayan cambiado desde la anterior copia de seguridad total. Esta copia debe quedar en un fichero llamado `/root/copia-etc-incremental1.cpio`.

c) Modifica el fichero `/etc/motd` y mueve el fichero `/etc/issue` al directorio `/root` para simular que lo has borrado. Haz otra copia de seguridad incremental de `/etc`, copiando sólo los ficheros que hayan cambiado desde la anterior copia de seguridad incremental. La copia debe quedar en el fichero `/root/copia-etc-incremental2.cpio`.

d) Finalmente, usando única y exclusivamente las copias de seguridad hechas, restaura en `/root/etc` una copia de `/etc`. ¿Es `/root/etc` una copia idéntica del actual `/etc`?

29. Vamos a experimentar ahora con la orden `rsync`[4]:

a) Como superusuario, crea un directorio llamado `/respaldo` y haz lo necesario para que el usuario *alumno* pueda usarlo para almacenar una copia de seguridad (ningún otro usuario tendrá acceso al mismo).

b) Como usuario `alumno`, realiza una copia del directorio personal `/home/alumno` en el directorio `/respaldo`. Para ello, simula una copia «remota» usando tu propia máquina (llamada `localhost`) como «servidor».

c) De nuevo como usuario `alumno`, copia el fichero `/etc/default/grub` a `/home/alumno` y borra el fichero `.bash_profile` dentro de `/home/alumno`. Ahora, repite la copia de seguridad anterior, pero haciendo que en el «servidor» los ficheros modificados/borrados se almacenen en un directorio llamado `/respaldo/historico`. Localiza el fichero borrado `/home/alumno/.bash_profile` dentro de `/respaldo/historico`.

[4]Recuerda que para poder realizar estos ejercicios es necesario primero activar el demonio `sshd`. Ejecuta para ello (como superusuario): `systemctl start sshd`

Apéndice

A. Otros mecanismos de protección

Además del mecanismo de seguridad tradicional basado en permisos, usuarios, grupos y programas con bits SETUID y SETGID, Linux dispone también de otros mecanismos de seguridad más sofisticados y flexibles. A continuación describimos dos de los más utilizados, ACLs y SELinux, siendo SELinux el que viene activado por defecto en sistemas Fedora. La descripción es breve pues sólo pretende dar a conocer los mecanismos para que el alumno que quiera pueda indagar más sobre ellos.

A.1. Listas de control de acceso (ACLs)

Linux implementa ACLs como las vistas en teoría, es decir, permite asociar a un fichero una lista de usuarios y grupos junto con los permisos de acceso de cada uno. Para poder usar ACLs, es necesario que el sistema de ficheros tenga soporte y que se haya montado con la opción `acl`. Sistemas de ficheros como Ext2/3/4, ReiserFS y Btrfs implementan ACLs.

La orden `ls -l` indica qué ficheros tienen ACLs asociadas mostrando un signo «+» a la derecha de los permisos. Por ejemplo:

```
[root@localhost media]# ls -l fichero
-rw-r-xr--+ 1 root root 0 jul 25 00:03 fichero
```

Podemos obtener la ACL asociada a un fichero con la orden `getfacl`:

```
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
user:internet:r-x
group::r--
mask::r-x
other::r--
```

y modificarla con la orden `setfacl`:

```
[root@localhost media]# setfacl -m "u:internet:r-x" fichero
[root@localhost media]# setfacl -m "g:bin:rw-" fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
user:internet:r-x
group::r--
group:bin:rw-
mask::rwx
other::r--
```

También podemos usar la orden `setfacl` para borrar una entrada particular de un usuario o grupo, o borrar toda la lista, como en los dos siguientes ejemplos:

```
[root@localhost media]# setfacl -x "u:internet" fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
```

```
# group: root

user::rw-
group::r--
group:bin:rw-
mask::rw-
other::r--

[root@localhost media]# setfacl -b fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

A.2. SELinux

Security-Enhanced Linux (SELinux) es un módulo de seguridad del núcleo de Linux que proporciona soporte para políticas de seguridad basadas en *control de acceso obligatorio* o MAC (*Mandatory Accesss Control*). Con SELinux, los programas de usuario y los servidores (demonios) del sistema se confinan para que sólo puedan acceder a ciertos ficheros y a ciertos recursos de red. Al limitar los privilegios al mínimo necesario para que algo funcione, se reduce o elimina la capacidad de estos programas y demonios para producir cualquier tipo de daño en el sistema si su seguridad se ve afectada (por ejemplo, por un fallo de programación o una mala configuración).

SELinux es independiente del mecanismo de control de acceso tradicional de Linux y no posee el concepto de superusuario. Esto último hace que incluso programas ejecutados por el superusuario tengan ciertas restricciones, a diferencia de lo que ocurre en un sistema sin SELinux, donde un proceso de superusuario no tiene ninguna restricción y, por tanto, puede comprometer todo el sistema si presenta fallos de seguridad. Al ser mecanismos independientes, lo que un proceso pueda hacer con cierto recurso dependerá tanto de SELinux como de los permisos que haya establecidos en el recurso y el usuario/grupo al que pertenezca el proceso.

SELinux asigna a cada proceso un *contexto* formado por un *papel* (o role), un *nombre de usuario* y un *dominio* (o tipo). El nombre de usuario no tiene relación con ningún nombre de usuario definido en el sistema, aunque a veces coinciden. Podemos usar la opción `-Z` de `ps` para ver el contexto que tiene cada proceso.

Los ficheros, puertos de red y ciertos elementos hardware también tienen un contexto formado por un nombre, un papel (o role, que rara vez se usa) y un tipo. También aquí podemos usar la opción `-Z` de `ls` para ver estos contextos. La orden `ls -l` nos indicará que un fichero tiene un contexto asociado mostrándonos un punto a la derecha de los permisos.

Teniendo en cuenta todo lo anterior, lo que un proceso pueda hacer con un recurso dependerá del contexto del proceso, del contexto del recurso y de lo que las reglas de política de SELinux establezcan.

B. Diseño de un plan de copias de seguridad

El requisito general de un plan de copias de seguridad es que se pueda restaurar el sistema completamente en un tiempo aceptable, teniendo en cuenta a la vez el tiempo que tarda en hacerse la copia de seguridad y la facilidad de restaurar únicamente unos pocos ficheros.

Los factores más importantes a considerar en el diseño de un plan de copias de seguridad son:

- QUÉ es lo más importante del sistema, de cara a seleccionar qué ficheros se van a copiar y dónde están esos ficheros.

- QUIÉN realizará las copias: los diferentes propietarios de los ficheros o el administrador. Realmente ambas decisiones son combatibles pudiendo conllevar planes de copia de seguridad que actúen en paralelo, de manera que cada usuario puede establecer, si lo estima oportuno, su plan de copias individual orientado a los datos que él maneja, mientras que el administrador puede plantear un plan de copias general teniendo en cuenta el sistema completo.
- CUÁNDO realizar las copias: En general, es conveniente que las copias de seguridad se realicen de manera periódica y, preferentemente, cuando no haya usuarios trabajando. Por estas razones, lo habitual es tener una sistema automático de copias de seguridad que se puede programar con alguna de las herramientas de planificación de tareas ya vistas en el anterior boletín. Además, es importante tener en cuenta que cuando se va a realizar la copia de seguridad de un sistema de ficheros, es recomendable que este sistema no esté montado, o solamente montado en modo lectura, de cara a asegurar la coherencia de todos los datos en el soporte de almacenamiento usado por dicho sistema de ficheros.
- CADA CUÁNTO es necesario realizar las copias: Esto vendrá determinado principalmente por la frecuencia con la que cambien los datos de los ficheros y la importancia de éstos. Teniendo en cuenta este factor se pueden diseñar diferentes estrategias de copias de seguridad.
- DÓNDE se realizarán las copias y las futuribles restauraciones: Es decir, hay que disponer de dispositivos físicos para el almacenaje local o remoto de las copias y, de igual forma, tener previsto un sistema donde restaurarlas tras una posible pérdida de los datos originales.
- CÓMO vamos a proteger las copias de seguridad: Además de una protección básica contra escritura de las copias de seguridad, es necesario, disponer un sistema de protección física del soporte de las copias, en base a elegir un correcto lugar de almacenamiento, con las condiciones de seguridad apropiadas ante intrusos, condiciones ambientales acordes al dispositivo físico utilizado para las copias, etc.

C. Soportes para copias de seguridad

Los soportes más habituales para guardar copias de seguridad son:

- Cintas magnéticas.
- Soportes ópticos: CD-ROMs o DVDs.
- Discos duros.
- Jukeboxes y similares.

Las cintas magnéticas de almacenamiento de datos son dispositivos de acceso secuencial que hay que rebobinar o avanzar para situarse en el punto de lectura/escritura. Por esta razón, para recuperar ficheros individuales pueden ser lentos. La principal diferencia entre el almacenamiento en cintas y en discos es que éste último es un medio de acceso aleatorio.

Los soportes ópticos (DVDs, CD-ROMs) tienen un mayor grado de fiabilidad que los magnéticos (discos duros, cintas,..), así como una esperanza de vida mayor. Por contra, los soportes ópticos presentan tiempos de L/E mayores que los magnéticos, además de ocupar un espacio físico mayor cuando nos referimos al almacenamiento de una gran cantidad de información.

Un jukebox es un dispositivo de almacenamiento que permite administrar una cierta cantidad de medios ópticos. Consiste en una serie de slots donde estos se alojan, una o más unidades de lectura (drives) y un brazo robot que tiene por misión tomar los discos de los correspondientes slots y colocarlos en la unidad lectora. Estos dispositivos tienen normalmente dimensiones máximas de entre 10 y 1000 slots y 1 a 20 unidades lectoras, alcanzando una capacidad máxima de 124 GB, 665 GB, 1.2 TB, y 5 TB, para almacenamiento.

D. Restauración de un sistema

En general, a la hora de restaurar un sistema, podemos encontrarnos con dos posibles situaciones como punto de partida:

- Si no disponemos de una copia de seguridad completa de todo nuestro sistema, sino que únicamente tenemos copias de algunos de nuestras sistemas de ficheros, lo primero que tenemos que hacer es reinstalar el S.O. y, en el caso de que tengamos copias de seguridad de los ficheros de configuración de este S.O. que recojan los cambios/actualizaciones que fuimos realizando desde su última instalación hasta el momento de la caída del sistema, restaurar estos ficheros. Finalmente habría que restaurar aquellos sistemas de ficheros de los que tenemos copia. Para recuperar un sistema de ficheros, los pasos a llevar a cabo serían:

  1. Crear y montar un sistema de ficheros limpio y vacío.
  2. Entrar en el punto de montaje.
  3. Restaurar la copia de seguridad según la herramienta usada.

- Si se tiene una copia completa de todo el sistema, con todos sus sistemas de ficheros e incluyendo todo el S.O., se puede realizar una restauración completa sin tener que reinstalar dicho S.O. De manera que los pasos a llevar a cabo serían:

  1. Arrancar desde un dispositivo distinto (p.e. un DVD).
  2. Si es necesario, crear los ficheros especiales de dispositivos para los discos (`/dev/sda1`, etc.).
  3. Preparar cada disco duro, creando sus particiones (`fdisk`).
  4. Crear el sistema de ficheros en la partición donde se restaurarán los datos y montarlo en un directorio.
  5. Restaurar la copia de seguridad sobre ese sistema de ficheros.
  6. Desmontar el sistema de ficheros restaurado.
  7. Volver al paso 4, para restaurar otros sistemas de ficheros adicionales.