Apuntes curso LFS201-Esp de la Linux Fundation

14 de agosto de 2016

Índice

1. Inicio y Apagado del Sistema

1.1. Proceso de Inicio

- La BIOS/UEFI busca y encuentra el gestor de arranque del disco duro
- El gestor de arranque arranca el kernel (o el initrd)
- El kernel (o initrd) arranca el proceso init
- init es el encargado de gestionar el inicio del sistema, utilizando **upstart**, **openrc**, **systemd** o **sysvinit**

La UEFI es un sustituto de la BIOS.

1.2. BIOS

La BIOS contiene todo lo necesario para que el teclado, pantalla y disco funcionen (cómo mínimo, pueden tener más funciones. La BIOS está en un chip de lectura sólo (ROM), para que no dependa nunca del disco y siempre pueda arrancar.

1.3. Gestores de arranque

En GNU/linux hay varios:

- GRUB: El más usado
- LILO: Viejo y abandonado
- efilinux: Carga específicamente UEFI
- Das U-BOOT: Apunta a sistemas embebidos

1.4. Archivos de configuración por defecto

Son los archivos que dan los valores por defecto que usan ciertos programas. En Red Hat están en `/etc/sysconfig`, mientras que en Ubuntu, en `/etc/default`.

1.5. Apagado del sistema

Se hace con **shutdown**. También puede hacerse con **reboot**, **halt** y **poweroff**.

2. GRUB

Características principales:

- Se puede escoger entre distintos sistemas operativos
- Se pueden cargar distintos kernels o ramdisks
- Se pueden cambiar los parámetros de arranque del kernel sin archivos de configuración

2.1. Diferencias entre GRUB y GRUB2

- La primera permite la modificación manual de los archivos creados y la segunda no.
- En la **primera**, las particiones empiezan a contarse desde el **0**, y en la **segunda** desde el **1**:
  - GRUB: sda1 es (hd0,0)
  - GRUB2: sda1 es (hd0,1)

2.2. Selecciones interactivas

Cuando carga el menú, podemos escoger un valor y modificarlo presionando **e** y así entrar en una shell interactiva. En esta podemos modificar el archivo de configuración de esa sección en particular, sin que estos cambios se guarden más allá de la sesión en la que estamos. Por ejemplo, añadiendo la palabra **single**, el sistema arrancará en modo monousuario, normalmente usado para gestiones correctivas. También es posible entrar en una **shell pura**, en la que hay distintas opciones. Generalmente, este se usa cuando hay problemas serios de arranque.

2.3. GRUB2

Ya que es el más usado, se hablará de este. Tiene un archivo y un directorio importantes, `/etc/default/grub` es el archivo importante y el directorio es `/etc/grub.d/`. Los archivos que están en este directorio se ejecutan en orden ascendente. Son todos del tipo: `00_algo`, `20_algo`, `40_algo`.

3. init

Este es el primer proceso que se ejecuta al arrancar el sistema. El más usado es **sysvinit**, pero este tiene ciertas deficiencias (supuestamente), que son que está pensado para un entorno muy diferente al actual. Apuntaban a servidores, con un sólo procesador y en los que el tiempo de inicio y apagado no eran importantes, ya que no solían apagarse y primaba la integridad. Por eso, se ejecuta de manera serial.

3.1. Alternativas a sysvinit

Para luchar con estos problemas, han surgido distintas alternativas. **Upstart** y **systemd** son los más usados, aunque Ubuntu ya ha dicho que abandonará su desarrollo en favor de **systemd**.

3.2. SysVinit

3.2.1. Niveles de ejecución de sysvinit

- Runlevel 0: apagado del sistema
- Runlevel 1: sistema monousuario
- Runlevel 6: reinicio del sistema
- Demás runlevels: depende del sistema

3.2.2. `/etc/inittab`

Este es primer archivo que lee init al ejecutarse. Por cada runlevel, hay un formato similar a este **id\:runlevel(s):action:process**.

3.2.3. Scripts de inicio

El primer script que ejecuta, es **rc.sysvinit**, que está en `/etc/rc.d/` o en `/etc/`. Entonces, se ejecuta el script **rc** de ese directorio con el runlevel deseado cómo parámetro. Entonces, según el runlevel, ejecuta los scripts cuyo nombre es ese runlevel. Por ejemplo, si **rc** tuviese que ejecutar el runlevel 5, ejecutaría los scripts del directorio `rc.d/rc5.d/`.

Apuntes varios:

- Todos los scripts que están en esos directorios son enlaces simbólicos a `/etc/init.d/`
- Los scripts de inicio empiezan por `s`
- Los scripts de detención empiezan por `k`

3.2.4. Comandos

- runlevel: ver el runlevel actual
- telinit: para escoger el runlevel de ejecución por defecto
- chkconfig: ver que scripts se ejecutan en que runlevel

3.3. Upstart

Este maneja eventos en vez de ejecutarse de manera serial. Archivos de configuración de upstart:

- `/etc/init/rcS.conf`
- `/etc/rc-sysinit.conf`
- `/etc/inittab`
- `/etc/init/rc.conf`
- `/etc/rc[0-6].d`

3.3.1. Comandos

- initctl: start, stop, restart, etc

3.4. systemd

El concepto de runlevels está soportado a través de targets, por lo que es compatible con SysVinit.

Características:

- Es compatible con los scripts de SysVinit
- Inicia más rápido que los sistemas anteriores
- Provee altas capacidades de paralelización
- Usa socket y activación D-Bus para iniciar servicios

- Reemplaza scripts de shell scripts con programas
- Ofrece inicio de demonios sobre demanda
- Realiza seguimiento de los procesos usando cgroups
- Soporta la creación de snapshots y restauración del estado del sistema
- Mantiene puntos de montaje y automontaje
- Implementa una elaborada lógica de control de servicio basada en dependencia transaccional
- Puede trabajar como un reemplazo de SysVinit
- No usa scripts en bash, usa archivos `.service`

3.4.1. Archivos de configuración

- `/etc/vconsole.conf`: mapa de teclado por defecto y fuente de consola
- `/etc/sysctl.d/*.conf`: directorio para los parámetros sysctl del kernel
- `/etc/os-release`: archivo de ID de la distribución

3.4.2. Comandos

- systemctl [options] command [name]

3.4.3. Migrar de SysVinit

Wiki de Fedora

4. Árbol del Sistema (FHS)

En GNU/Linux todo son ficheros, y se montan bajo la raíz (/). Hay, por eso, dos grandes diferencias:

- Compartidos y no-compartidos: Los primeros serian los `/etc/` y `/home/`, por ejemplo, y los segundos serian los archivos de bloqueo, cómo por ejemplo `/var/run/keepassx.kdb.lock`.
- Variable y estático: Los archivos estáticos serian los binarios, bibliotecas y documentación (`/bin/`, `/lib/` y `/usr/share/`, por ejemplo) y en general todo lo que no cambie sin ayuda del sysadmin. Los archivos variables son el resto, que se pueden cambiar incluso sin permisos.

El Filesystem Hierarchy Standard se puede descargar en esta web. Aunque en las partes más importantes se sigue, las distribuciones tienden a tener pequeñas diferencias.

Una particularidad es que aunque todo parezca un gran disco duro, en cuanto a que no accedemos a una `C:` o una `D:` cómo en winsux, realmente puede tener distintas particiones del mismo disco duro e incluso otros discos. Un ejemplo típico es tener el `/home/` en otra partición para mayor facilidad de migración. Esta se monta después de que la raíz se haya montado.

4.1. Directorios más importantes

- `/bin/`: Contiene ejecutables a los que puede acceder cualquier usuario, pero sólo los más importantes. Véase, **ls**, **bash** o **echo**. Los que no se consideran lo suficientemente importantes se distribuyen en `/usr/bin/`. Siempre y cuando, recordemos, no se requieran permisos de superusuario para ejecutar estos programas.

- `/boot/`: Contiene los archivos necesarios para arrancar. Los más necesarios son **vmlinuz** (el kernel comprimido) y el **initramfs** (archivo de disco ram (ramdisk)). A estos archivos se les suele añadir cómo sufijo la versión del kernel que se use (**vmlinuz-generic-3.16.0**). Otros archivos o directorios son **config** y **System.map**. El primero sirve cómo referencia, para saber que se ha compilado en el kernel. El segundo es la tabla de depuración del kernel, entrega direcciones hexadecimales de todos los símbolos del kernel.

- `/dev/`: Este directorio contiene dispositivos especiales, es decir, dispositivos externos. Estos podrían ser **USB** o **discos duros**. Estos dispositivos son gestionados mediante **Udev**, que crea nodos en `/dev/` según esté configurado (esto se discute más en profundidad en el capítulo 8).

- `/etc/`: Aquí están los ficheros de configuración.

- `/home/`: Aquí está la raíz de los usuarios, por así decirlo. Todos sus archivos deberían estar aquí.

- `/lib/`: Aquí están las bibliotecas necesarias para ejecutar los binarios de `/bin/` y `/sbin/`. También podemos encontrar los módulos del kernel en `/lib/modules/`, igual que los archivos PAM en `/lib/security/`.

- `/media/`: En este directorio se montan los archivos de medios extraíbles, tales cómo el DVD o el USB. Suelen montarse aquí si van a estar montados de manera habitual. Los que no estarán montados habitualmente...

- `/mnt/`: Aquí se montaran en este directorio. Suele apuntar a usarse con samba o NFS.

- `/opt/`: Se usa para grandes programas cuyo contenido no se quiere que esté disperso por todo el sistema de ficheros. Esto tenia más sentido cuando aún no estaba muy extendido el uso de gestores de paquetes, ahora ya apenas se usa para la mayoría de programas.

- `/proc/`: Aquí se montará un pseudo-sistema de archivos montado en memoria, por lo que una vez apagado en este directorio no hay nada. En los archivos que se montan en este directorio podemos ver mucha información, del tipo las particiones que tenemos, la ram libre o información acerca de procesos que están corriendo.

- `/sys/` : Aquí se monta el pseudo-sistema de archivos **sysfs**. Otra vez, montado en memoria. **sysfs** se utiliza tanto para recopilar información sobre el sistema, como también modificar su comportamiento mientras se ejecuta. Es más moderno y se adhiere a el estándar que dice que los archivos de ese directorio sólo contienen una linea o valor.

- `/root/`: El home del superusuario.
- `/sbin/`: En este directorio están los archivos binarios esenciales para arrancar, restaurar, recuperar y/o reparar los binarios en el directorio `/bin/`, `/usr/` y `/home/`.
- `/tmp/`: Cómo su nombre indica, un directorio para ficheros temporales. Algunas distros borran lo que hay ahí con cada reinicio. Esas son **Debian** y derivadas, por ejemplo. En cambio, **Slackware** no lo hace por defecto. Este directorio se suele usar para compilar o para descomprimir grandes ficheros, por lo que se monta en ram. Por eso hay que evitar que se creen ficheros grandes, ya que sin querer (o queriendo) podemos hacer que el sistema se quede sin ram y se colapse.
- `/usr/`: En este irán los binarios que no se relacionen con el arranque p arreglo del sistema.
- `/var/`: Contiene archivos variables (thanks, Sherlock). Se guardan los logs o las webs, en `/var/log/` y `/var/www/` respectivamente.
- `/run/`: Este no es uno de los que está .ªceptados", pero ya se encuentra en casi todas las distribuciones, ya sea llamado `/run/` o `/var/run/`. Sirve cómo directorio transitorio que contiene información sobre los procesos que se ejecutan, cómo su PID.

5. Kernel

5.1. Función del kernel

- Iniciar el sistema
- Planificar procesos
- Gestión de memoria
- Controlar acceso al hardware
- Entrada/Salida (E/I | I/O) entre programas y dispositivos de almacenamiento
- Implementar sistemas de archivos
- Control de seguridad
- Control de red

5.2. Linea de comandos

Grub les pasa esos comandos. Según si es grub o grub2, las lineas son ligeramente distintas. Ejemplo:

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.18.1 root=UUID=9d6b5801-9c7e-4c17-9068-49923952338e ro rhgb qui
```

5.3. Parámetros de inicio del kernel

- En las fuentes, "Documentation/kernel-parameters.txt"
- Online, Web
- Ejecutando "man bootparam"
- Parámetros básicos
  - ro: monta el directorio raíz en modo sólo-lectura en el inicio.
  - root: sistema de archivos raíz.
  - rd\LVM\LV: activa el sistema de archivos root en el volumen lógico especificado.
  - rd\NO\LUKS: deshabilita la detección de cifrado LUKS.
  - rd\NO\DM: deshabilita la detección de DM RAID.
  - LANG: es el lenguaje del sistema.
  - SYSFONT: es la fuente de consola.
  - KEYTABLE: es el nombre del archivo de configuración del teclado.
  - rhgb: para soporte de booteo gráfico en sistemas Red Hat.
  - quiet: deshabilita la mayoría de los mensajes

5.4. sysctl

Modifica los parámetros en tiempo de ejecución.

5.4.1. Mostrar los valores actuales

```
$ su -c "sysctl -a"
```

5.4.2. Modificar valores

```
$ su -c "sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'"
$ su -c "sysctl net.ipv4.ip_forward=1"
```

5.4.3. Hacer fijos esos valores

Modificar /etc/sysctl.conf. Para aplicar los parámetros de esos archivos:

```
$ sysctl -P
```

5.5. Herramientas

La mitad de este capitulo habla de parámetros y de uso de los siguientes comandos. Cómo me parece una gilipollez transcribir lo que se puede ver con man $comando, paso mucho de hacerlo.

- `lsmod`: Lista los módulos cargados.
- `insmod`: Carga módulos directamente.

- `rmmod`: Remueve módulos directamente.
- `modprobe`: Carga o descarga módulos, utilizando una base de datos preconstruida de los módulos con información de sus dependencias.
- `depmod`: Reconstruye la base de datos de dependencia de módulos; es requerida por modprobe y modinfo.
- `modinfo`: Proporciona información acerca de un módulo.

5.6. Archivos de configuración

Están en `/etc/modprobe.d/` con un formato de un comando por linea lo que se carga y un '#' al principio de la linea que no se carga.

6. Udev

6.1. Características generales

udev crea los nodos de los dispositivos según haga falta en el momento. De esta manera se evita tener grandes cantidades de nodos, un problema que alcanzó su culminación en el kernel 2.6.

Los números **mayor** y **menor** hacen referencia al controlador asociado al dispositivo:

```
# Un disco duro interno, conectado por sata
# El mayor seria el 8, el menor serian el 1, 2 y 5 detrás de sda
$ ls /dev/sda* -l
brw-rw---T 1 root disk 8, 0 dic  9 00:35 /dev/sda
brw-rw---T 1 root disk 8, 1 dic  9 00:36 /dev/sda1
brw-rw---T 1 root disk 8, 2 dic  9 00:35 /dev/sda2
brw-rw---T 1 root disk 8, 5 dic  9 00:35 /dev/sda5

# Un disco duro externo. conectado por USB
# Vemos que sigue mostrando 8 cómo mayor, pero lo identifica cómo floppy
$ ls /dev/sdc* -l
brw-rw---T 1 root floppy 8, 32 dic  9 21:35 /dev/sdc
brw-rw---T 1 root floppy 8, 33 dic  9 21:35 /dev/sdc1
```

Es un demonio (ya sea **udevd** o **systemd-udevd**) y monitoriza un netlink socket. Cuando se enchufan o quitan nuevos dispositivos, uevent (parte del kernel) envía un mensaje a través del socket que udev está leyendo, el cual realiza la función adecuada respecto a ese evento según las reglas.

Los componentes de udev son:

- libudev: que permite el acceso a la información de los dispositivos
- udevd: que gestiona el directorio /dev
- udevadm: para el control y diagnóstico

La manera idónea de utilizar udev es que el directorio /dev esté vació al cargar el kernel y se van añadiendo a medida que hacen falta. Para esto, hay que usar una imagen initramfs, que contendría unos dispositivos preestablecidos por si hiciesen falta para el arranque.

6.2. udev y hotplug

hotplug es el encargado de detectar cambios en los dispositivos en caliente y de notificar a udev.

La información que el último necesita (nombres adecuados, números mayor y menor, permisos, etc.) los obtiene del registro existente en sysfs y unos cuantos archivos de configuración.

El principal archivo de configuración es `/etc/udev/udev.conf`. Las reglas para los nombres de los dispositivos están ubicadas en el directorio `/etc/udev/rules.d`. A través de la lectura de la página man de udev es posible obtener una gran cantidad de información específica acerca de cómo configurar reglas para situaciones comunes.

6.3. El proceso que sigue udev

Cuando udev recibe un mensaje desde el kernel acerca de dispositivos que están siendo añadidos o eliminados, analiza los archivos de configuración en `/etc/udev/rules.d/*.rules` para determinar si hay reglas que apliquen al dispositivo en cuestión.

Entonces udev realiza su trabajo, que entre otros es:

- Asignación de nombre a los nodos de dispositivo.
- Creación de nodos de dispositivo y links simbólicos.
- Ajuste de permisos de archivo y dueño para los nodos de dispositivo.
- Realizar otras acciones para inicializar el dispositivo y ponerlo disponible.

6.4. Archivos de reglas udev

Los nombres están en el directorio `/etc/udev/rules.d/`, y son del tipo `30-usb.rules` y `90-mycustom.rules`. `udev` siempre busca archivos con el sufijo .rules y los lee de forma alfabética ascendente. El nombre estándar corresponde a dos dígitos seguido por un nombre descriptivo de la regla y terminando con el sufijo .rules.

6.5. Creando reglas udev

El formato es simple: <match><op>value [, ...] <assignment><op>value [, ...]

Consta de dos partes:

- La primera parte consiste en uno o más pares indicados por == . Estos tratan de coincidir con los atributos y/o características de un dispositivo a algún valor.

- La segunda parte consiste en una o más asignaciones clave-valor, las que asignan un valor a un nombre, tal como un nombre de archivo, pertenencia a un grupo, incluso permisos de archivo, etc.

Si el nuevo dispositivo no coincide con ninguna regla, se usan los valores por defecto.

```
# El siguiente es un ejemplo para un dispositivo fitbit:
$ cat /etc/udev/conf.d/rules.d/99-fitbit.rules

SUBSYSTEM=="usb", ATTR{idVendor}=="2687", ATTR{idProduct}=="fb01", SYMLINK+="fitbit", MODE

# Otro ejemplo de una regla para las interfaces de red
$ cat /etc/udev/rules.d/70-persistent-net.rules
# This file was automatically generated by the /lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.

# PCI device 0x14e4:/sys/devices/pci0000:00/0000:00:15.0/0000:06:00.0 (tg3)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:59:2d:f5:23:ba", ATTR{d
KERNEL=="eth*", NAME="eth0"

# PCI device 0x14e4:/sys/devices/pci0000:00/0000:00:16.0/0000:07:00.0/ssb0:0 (b43)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="10:f4:11:9a:75:c9", ATTR{d
KERNEL=="wlan*", NAME="wlan0"
```

7. Particionado y Formateo de Discos

Nota del autor: este capitulo es casi una copia literal, no le veo el sentido a escribir esto con mis palabras

7.1. Tipos comunes de discos

Existe una gran variedad de tipos de discos duros; cada uno está caracterizado por el tipo de bus de datos a través del cual se conecta, como también otros factores como velocidad, capacidad (de almacenaje)y cuán bien operan múltiples unidades de forma simultánea.

- IDE y EIDE (Entorno de desarrollo integrado e IDE mejorado): Estos fueron el estándar en notebooks y PCs de escritorio por años. Sin embargo, son pequeños (en capacidad de almacenaje) y lentos en comparación a hardware más moderno, por lo cual actualmente están obsoletos; de hecho, los controladores no están disponibles en máquinas actualizadas.
- SATA (Serial ATA): Este tipo fue diseñado para reemplazar a Parallel ATA (PATA) (el cual fue conocido originalmente como IDE). Tenían una mayor transferencia de datos, cables más pequeños y eran detectados como dispositivos SCSI por el sistema operativo, lo cual simplificó el tema de

escribir controladores de software (entre otras cosas), aún siendo que el hardware no es realmente SCSI.

En comparación a PATA, SATA ofrece un cable de tamaño menor (7 pines), sustitución en caliente (hot swapping) y una transferencia de datos más rápida y eficiente. Los controladores más nuevos pueden manejar 16 GB/s, pero 3 GB/s y 6 GB/s son los valores más comunes en dispositivos del segmento usuario normal.

- SCSI (Interfaz de Sistema para Pequeñas Computadoras): Estos han sido el pilar de los servidores empresariales por décadas. Mientras que pueden tener una capacidad menor que los discos SATA, tienen a ser mucho más rápidos y a trabajar en paralelo mucho mejor, de la forma en la que se requiere para algunas configuraciones en RAID.

Hay varias versiones de SCSI: Fast, Wide, Ultra y UltraWide, lo cual torna un poco confusas las cosas. Además, hay muchos controladores de dispositivo diferentes, dependiendo del hardware específico. Eso no sucede en SATA, ya que en ese caso existen controladores estandarizados que pueden adaptarse a una gran variedad de hardware.

Los discos SCSI van desde un rango pequeño (bus de 8 bits) a uno amplio (bus de 16 bits), con una tasa de transferencia desde 5MB por segundo (un valor bajo, correspondiente a dispositivos SCSI estándar) a cerca de 160MB por segundos (Ultra-Wide SCSI-3).

La mayoría de los PCs usan unidades SCSI de un solo extremo o diferenciales. Desafortunadamente, los dos tipos no son compatibles entre sí. De todas formas, ambos tipos pueden coexistir en el mismo controlador.

Los controladores de una sola terminación soportan hasta 7 dispositivos, con una longitud de cable de cerca de 6 metros. Los controladores diferenciales soportan hasta 15 dispositivos, con una longitud máxima del cable de unos 12 metros.

- SAS: El Serial Attached SCSI es un protocolo serial punto a punto nuevo, que viene a reemplazar a la interfaz SCSI. Las tasas de transferencia son similar a SATA, pero el rendimiento general es mejor.

- USB: Los dispositivos de Bus Universal en Serie incluyen memorias y discos duros externos USB. El sistema operativo los ve como dispositivos SCSI.

En la misma categoría están las unidades SSD modernas (dispositivos de estado sólido), las cuales han bajado de precio, no tienen partes movibles, usan menos energía que las unidades de disco giratorio y tienen velocidades de transferencia más rápidas. Los SSD internos son instalados de forma similar y en los mismos encapsulados que los discos convencionales.

Los SSD todavía cuestan un poco más pero el precio está bajando. Es algo común tener discos SSD y convencionales en una misma máquina, en donde las operaciones de rendimiento crítico y de alto tráfico de datos son llevadas a cabo en las unidades SSD.

7.2. Geometría del disco

Es un concepto algo desfasado, ya que viene de cuando los discos duros eran más manuales, pero aún se aplica en muchos tipos de discos, en los llamados giratorios.

Estos se componen de uno o más **platos**, cada uno de los cuales es leído por uno o más **cabezales**. Los cabezales leen una **pista** a medida que el disco va girando.

Estas pistas están divididas en bloques de datos llamados **sectores**, generalmente de 4kb cada uno. Un **cilindro** es un grupo de sectores que está formado por más de una pista en todos los platos.

Cómo decíamos, este concepto está algo anticuado. No se cumple, por ejemplo, en los discos USB o los discos SSD, ya que ambos son flash.

7.3. Particiones

En términos geométricos, estas son grupos físicamente unidos de sectores o cilindros. Una de las particiones primarias puede ser designada como una partición extendida, la cual puede ser subdividida en particiones lógicas.

SCSI y estándares relacionados como SATA, suportan hasta 15 particiones en el disco. Las particiones 1-4 son primarias o extendidas; las particiones 5-15 son lógicas, y sólo puede haber una extendida, pero esta puede dividirse en las lógicas que hagan falta hasta 15.

7.4. Tabla de particiones

Esta está en el registro de arranque principal (MBR), que es de 512 bytes de longitud y es independiente del sistema operativo. Esta tiene 64 bytes de largo y está ubicada después del byte 446 del registro de arranque. Nota para los curiosos: hay 2 bytes más al final del MBR, lo cual se conoce como el número mágico, firma del MBR, o marca de final de sector, que tiene siempre el valor 0x55A.

Los primeros 446 bytes están reservados para el cargador de arranque (**Grub**, por ejemplo). Cada entrada de la tabla tiene 16 bytes de largo y describe una de las cuatro posibles particiones primarias. La información que contiene es la siguiente:

- Bit activo
- Dirección de inicio en formato cilindro/cabezal/sectores (**CHS**), que por cierto no usa GNU/Linux.
- Código del tipo de partición (**ext4**, **ntfs**, etc)
- Dirección final en CHS (también ignorado en el ñú)
- Sector de inicio, contando linealmente desde 0
- Número de sectores en la partición

7.5. Nombre de los dispositivos

En **SCSI** y **SATA**:

- El primer disco es `/dev/sda`
- El segundo es `/dev/sdb`

Las particiones:

- La primera partición del primer disco es `/dev/sda1`
- La segunda es `/dev/sda2`
- Etc.

En los discos IDE (o PATA) son igual que arriba pero sustituyendo la **s** por **hd**. Por ejemplo, la segunda partición del segundo disco es `/dev/hdb1`

7.6. Utilidades

`blkid` nos da información acerca de todas las particiones, diciendo su **UUID** y tipo de partición (SWAP, ext3, etc). `lsblk` nos dará otro tipo de información y además en forma de árbol. Nos dice, a diferencia de `blkid`, el punto de montaje, entre otras cosas.

```
# Ejemplos reales de un disco duro cifrado con luks
$ su -c "blkid"
/dev/mapper/debian--bittorrent-swap_1: UUID="50275b69-3233-463e-b5cd-fbcc2242e657" TYPE="s
/dev/sda5: UUID="2aa73a39-c258-4762-bea2-5723e514e9b4" TYPE="crypto_LUKS"
/dev/sda1: UUID="02b37faa-1e97-4832-b1b6-593f94969018" TYPE="ext2"
/dev/mapper/sdb5_crypt: UUID="F2nvMK-vyHD-9TPn-Ujdw-LU4A-dgQH-o688m1" TYPE="LVM2_member"
/dev/mapper/debian--bittorrent-root: UUID="59a37eab-93d6-40c3-b626-ed958cafd90a" TYPE="ext
/dev/mapper/debian--bittorrent-home: UUID="8122ab0e-fc6b-456b-89d8-e740b2c7bb62" TYPE="ext
/dev/mapper/media_cifrado: UUID="8a183d31-bed3-4499-82f3-4d200a2d5243" TYPE="ext4"
/dev/sdc1: UUID="e89fdf45-fa39-4edc-8e50-094ed8c98814" TYPE="crypto_LUKS"

$ su -c "lsblk"
NAME                                  MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                     8:0    0 298,1G  0 disk
sda1                                    8:1    0   243M  0 part  /boot
sda2                                    8:2    0     1K  0 part
sda5                                    8:5    0 297,9G  0 part
  sdb5_crypt (dm-0)                   254:0    0 297,9G  0 crypt
    debian--bittorrent-root (dm-1)    254:1    0   9,3G  0 lvm   /
    debian--bittorrent-swap_1 (dm-2)  254:2    0   3,5G  0 lvm   [SWAP]
    debian--bittorrent-home (dm-3)    254:3    0   285G  0 lvm   /home
sdc                                     8:32   0 298,1G  0 disk
sdc1                                    8:33   0 298,1G  0 part
```

7.7. Dimensionado de las particiones

Sólo "necesitan"dos, la partición **raíz** y **swap**. Esto es en la teoría, en la practica es recomendable crear más. En el ámbito de los servidores, se suele separar, al menos, `/tmp`, `/home` y `/var`. Esto permite añadir seguridad y facilitar ciertas tareas, cómo la de hacer backups.

7.8. Respaldar y restaurar la tabla de particiones

Recordar que hacer esto es peligroso.

```
# Respaldar. Esto creará un archivo llamado mbrbackup en el directorio en el que se ejecut
$ su -c "dd if=/dev/sda of=mbrbackup bs=512 count=1"

# Restaurar
$ su -c "dd if=mbrbackup of=/dev/sda bs=512 count=1"
```

Hay varias utilidades que permiten editar la tabla de particiones. Algunas de ellas son:

- fdisk
- gfdisk
- sdisk
- parted
- gparted
- cfdisk (wrapper de fdisk)
- cgdisk (wrapper de gdisk)

8. Cifrado de Discos

La mayoría de distribuciones usan LUKS. Este es un cifrado a nivel de dispositivo de bloque. LUKS se gestiona a través de `cryptsetup`, que además gestiona otros métodos cómo dm-crypt o loop-AES, que es compatible con TrueCrypt (o su nuevo fork VeraCrypt).

8.1. Crear una partición cifrada

```
# Formatear la partición. AVISO: Borra todos los datos existentes
$ su -c "cryptsetup luksFormat /dev/VG/MYSECRET"
# Pedirá una contraseña, que no se puede olvidar bajo ningún concepto

# Se puede concretar el cifrado a usar
$ su -c "cryptsetup luksFormat --cipher aes /dev/VG/MYSECRET"

# Una vez hecho, se puede abrir
$ su -c "cryptsetup --verbose luksOpen /dev/VG/MYSECRET SECRET"
```

```
# Una vez abierto, se le da un sistema de ficheros a la partición
$ su -c "mkfs.ext4  /dev/mapper/SECRET"

# Se monta...
$ su -c "mount /dev/mapper/SECRET /mnt"

# Y ya se puede usar

# Para desmontarla y cerrarla del todo...
$ su -c "umount /mnt"
$ su -c "cryptsetup --verbose luksClose SECRET"
```

Nota del autor: En los apuntes no se menciona, pero antes de cifrar una partición, es recomendable llenarla con datos aleatorios. Esto se puede hacer ejecutando `cat /dev/urandom >/dev/sdx`, siendo x la partición o disco duro a llenar de datos

8.2. Auto montar la partición en el arranque

Para hacerlo hay que poner las líneas adecuadas en los ficheros `/etc/fstab` y `/etc/crypttab`. Ejemplo:

```
# En /etc/fstab. No necesita nada particular, exceptuando que la partición a
# montar tiene que ser la que queda al descifrar el disco
/dev/mapper/SECRET_cifrado /home/alguien/Disco_Duro_Cifrado ext4 defaults 0 2

# En /etc/crypttab
SECRET UUID=2aa73a39-c258-4762-bea2-5723e514e9b4 none luks
```

9. Sistemas de Archivos y el VFS

El motivo por el que GNU/Linux permite tantos tipos de sistemas de archivos es que usa VFS (Virtual FileSystem). Esta es una capa de abstracción por encima del sistema de ficheros.

GNU/Linux implementa el VFS como lo hacen todos los sistemas operativos modernos. Cuando una aplicación necesita acceder a un archivo, ésta interactúa con la capa de abstracción del VFS, el cual traduce todas las llamadas E/S (operaciones de lectura/escritura, etc) en código específico relacionado con el sistema de archivos real en particular.

De este modo, tanto el sistema de archivos usado cómo el medio físico no necesitan ser considerados por los programas. Además, los archivos de red (NFS) pueden ser tratados de forma transparente.

9.1. Sistemas de archivos transaccionales

Los nuevos sistemas de archivo de alto rendimiento incluyen lo que se conoce cómo **journaling**. Estos sistemas de archivos se recuperan más rápido de los crashes o caídas con poca corrupción de datos. Tiene la desventaja de que hace más operaciones.

En un FS (FileSystem) con journaling las operaciones se agrupan en transacciones. Para llegar a modificar un archivo, la transacción debe llevarse a cabo sin ni un sólo error. Se mantiene un registro de estas transacciones, y de haber un error sólo suele hacer falta modificar la última transacción.

FS con journaling:

- ext3
- ext4
- reiserfs
- JFS
- XFS
- brtfs

10. Características de los Sistemas de Archivos

10.1. Inodos

Un inodo es una estructura de datos del disco que describe y almacena los atributos de un archivo. La información que contiene es:

- Permisos
- Usuario y grupo
- Tamaño
- Registros de tiempo
  - Último acceso
  - Última modificación
  - Última modificación del inodo

Nota: el nombre del fichero se guarda en el inodo del directorio en el que está

10.2. Archivos de directorio

Estos se usan para relacionar un inodo a un nombre de archivo. Hay dos métodos, con enlaces duros, que apuntan a un inodo asociado y con enlaces simbólicos, que apuntan a un nombre de archivo.

10.3. Atributos extendidos

Estos se asocian a los metadatos, y no son interpretados por el sistema de archivos. Estos valores se guardan en los inodos de los archivos y pueden ser modificados y configurados sólo por el usuario **root**. Se visualizan con `lsattr` y se modifican con `chattr`.

Hay cuatro espacios de usuario, usuario, confianza, seguridad y sistema.

10.4. Utilidades para modificar el sistema de ficheros

Para formatear, se puede usar `mkfs` directamente o `mkfs.$filesystem`, en el que $filesystem es el FS a escoger.

```
# Ambos hacen lo mismo
$ su -c "mkfs -t ext4 /dev/sda10"
$ su -c "mkfs.ext4 /dev/sda10"
```

Para verificar errores, `fsck`. Tiene el mismo modo de uso que el anterior.

```
$ su -c "fsck -t ext4 /dev/sda10"
$ su -c "fsck.ext4 /dev/sda10"
```

Para montar, se usa `mount` y para desmontar `umount`.

```
$ su -c "mount -t ext4 /dev/sdb4 /home"
$ su -c "umount /home"
```

11. Características del sistema de archivos: intercambio, cuotas y uso

11.1. Swap

El **área de intercambio** o **swap** en inglés es un sistema para añadir memoria de manera virtual. Funciona de dos maneras:

- Algunos programas no usan toda la memoria que podrían usar. Esto es debido a que los hijos de los procesos reciben una copia de las regiones de la RAM de sus padres, que usan el método **COW** (Copy On Write) en el cual los hijos sólo reciben una nueva copia cuando se produce un cambio.
- Cuando hay mucha demanda de la RAM, algunas regiones poco activas se mueven al swap para ser llamadas sólo cuando se requiera.

Habitualmente se ha recomendado un uso de swap de el doble de la RAM que tenga el equipo, pero esto ha quedado ya desfasado, dado el aumento de la RAM en los equipos. Sólo se cumple hasta llegar a 1GB de RAM, siendo el swap de 2GB. A partir de los 2GB de RAM, ahora se recomienda dejarlo en 2GB de swap.

11.1.1. Utilidades

```
$ cat /proc/swaps
$ free -o
$ mkswap # formatea una partición o archivo de intercambio
$ swapon # activa una partición o archivo de intercambio
$ swapoff # desactiva una partición o archivo de intercambio
```

11.2. Cuotas

Las cuotas son límites en el espacio máximo usado regulado según usuarios y/o grupos. De esta manera se evita que los usuarios agoten recursos comunes con otros usuarios del mismo servicio.

```
$ quotacheck # genera y actualiza los archivos que llevan la cuenta de las cuotas
$ quotaon # habilita el sistema de cuotas
$ quotaoff # deshabilita el sistema de cuotas
$ edquota #  se usa para editar cuotas de usuarios o grupos
$ quota # informa acerca del uso y límites de las cuotas
```

Estas utilidades son las genéricas, luego cada sistema de archivos particular puede tener equivalentes o añadidos, cómo por ejemplo $\mathbf{xfs_{quota}}$.

11.2.1. Configuración

Hay que seguir los siguientes pasos:

- Para poder usar esta propiedad, la partición debe estar montada con las características que lo activan. Esto es agregar las opciones **usrquota** y/o **grpquota** a `/etc/fstab`, que recordemos es el archivo que monta las particiones al inicio del sistema.
- Ejecutar **quotacheck** para configurar las cuotas.
- Habilitar las cuotas en el sistema de archivos.
- Configurar las cuotas con el programa **edquota**.

```
# este se ejecutaria sólo la primera vez si no estuviese montado con las propiedades neces
$ su -c "mount -o remount /home"
$ su -c "quotacheck -vu /home" # este sólo se ejecuta en la configuración inicial
$ su -c "quotaon -vu /home" # habilita la cuota
$ su -c "edquota someusername" # asigna la cuota
$ su -c "quotaoff -av"# deshabilita la cuota
$ su -c "quota" # da un reporte sobre las cuotas
```

11.3. Utilidades del sistema de archivos

```
$ df -h # muestra el uso y la capacidad disponible del sistema de archivos
$ dfc # muestra el uso y la capacidad disponible del sistema de archivos pero con colores
$ du -h # se usa para evaluar cuánto espacio está usando un directorio (y sus subdirectori
```

12. Los sistemas de archivos ext{2,3,4}

12.1. Historia de ext4

El sistema **ext2** fue el primero nativo de GNU/Linux y está disponible en todos los sistemas, pero apenas se utiliza hoy en día (exceptuando en la partición `/boot`, que se sigue recomendando). **ext3** fue la primera extensión que incluyó journaling. Esta era la única diferencia con **ext2**.

ext4 e incluyó por primera vez en la versión 2.6.19 del kernel, y su designación como experimental fue quitada en la versión 2.6.28. Incluyó mejoras significativas como el uso de **extents** para archivos largos, en vez de listas de bloques de archivos. Extents es un grupo de bloques continuos, y está pensado para mejorar el rendimiento de grandes archivos y reducir la fragmentación.

12.2. Características de ext4

El tamaño del bloque se selecciona en el momento de creación del sistema de archivos. Este puede ser de 512, 1024, 2048, o 4096 bytes, aunque por defecto usa este último.

El número de inodos en el sistema de archivos también puede ser ajustado, lo cual podría ahorrar espacio en disco. La característica llamada **reservación de inodos** consiste en reservar x cantidad de inodos en el momento de crear un directorio para ser usados en el futuro. Esto mejora el rendimiento por que cuando se crean nuevos archivos estos tienen sus inodos asignados.

Si la ruta de un enlace simbólico es menor que 60 caracteres, se crea un enlace simbólico rápido, el cual se almacena directamente en el inodo, evitando así la necesidad de leer un bloque de datos.

12.3. Diseño de ext4

Todos los campos se escriben usando el orden little-endian. (Extracto de la wikipedia: *Una máquina little endian asigna los bytes menos significativos en el extremo más bajo de la memoria, mientras que una máquina big endian asigna los bytes menos significativos en el extremo más alto.*)

Los bloques, cómo ya se ha mencionado, se dividen en grupos de bloques, cada uno de los cuales contiene inodos y bloques adyacentes, reduciendo de este modo el tiempo de acceso a estos.

El diseño de los bloques estándar es el siguiente: El grupo de bloques **0** no usa los primeros 1024 bytes, para ceder ese espacio a los sectores de `/boot`. Los superbloques empiezan en el primer bloque, exceptuando el primero por el motivo ya mencionado. Siguen los descriptores de grupo y la GDT (Global Descriptor Table).

Los bloques de datos se preasignan a archivos antes de ser usados. De este modo, si este archivo crece puede usar el espacio adyacente, evitando así la fragmentación.

El superbloque (sólo hay uno por sistema de ficheros) contiene unos campos que contienen el estado con el que se cerró la última vez el sistema de ficheros y si necesita verificación. Estos pueden ser **limpio**, **sucio** o **desconocido** (clean, dirty y unknown). En contra de lo que los nombres pueden sugerir, significan: se montó correctamente la última vez, (generalmente) que está montado y que no se desmontó de forma limpia (cómo cuando hay un fallo de la corriente), respectivamente. En los dos últimos casos se usará fsck Otros campos contienen la fecha de la última verificación y el número de veces que se ha montado.

12.4. Grupos de bloques

Después del de arranque, que se ha mencionado en el punto anterior, hay una serie de grupos del mismo tamaño con el mismo diseño:

12.5. Herramientas

Con **dumpe2fs** herramienta podemos ver la información que se ha comentado antes.

```
$ su -c "dumpe2fs /dev/sda1"

Filesystem volume name:  RHEL7
Last mounted on:         /
Filesystem UUID:         9d6b5801-9c7e-4c17-9068-49923952338e
Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)
Filesystem features:     has_journal ext_attr resize_inode dir_index filetype needs_recove
Filesystem flags:        signed_directory_hash
Default mount options:   user_xattr acl
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:             1908736
Block count:             7630592
Reserved block count:    381529
Free blocks:             5353383
Free inodes:             1682479
First block:             0
Block size:              4096
Fragment size:           4096
Group descriptor size:   64
Reserved GDT blocks:     1024
Blocks per group:        32768
Fragments per group:     32768
Inodes per group:        8192
Inode blocks per group:  512
Flex block group size:   16
Filesystem created:      Wed Sep 3 03:52:55 2014
Last mount time:         Fri Oct 24 09:18:58 2014
Last write time:         Fri Oct 24 09:18:58 2014
Mount count:             89
Maximum mount count:     -1
Last checked:            Wed Sep 3 03:52:55 2014
Check interval:          0 (<none>)
Lifetime writes:         103 GB
Reserved blocks uid:     0 (user root)
Reserved blocks gid:     0 (group root)
First inode:             11
Inode size:              256
Required extra isize:    28
Desired extra isize:     28
```

```
Journal inode:           8
First orphan inode:      396118
Default directory hash:  half_md4
Directory Hash Seed:     e488c43e-241c-4014-91d8-6a9d3d6c7784
Journal backup:          inode blocks
Journal features:        journal_incompat_revoke journal_64bit
Journal size:            128M
Journal length:          32768
Journal sequence:        0x00023592
Journal start:           16394


Group 0: (Blocks 0-32767) [ITABLE_ZEROED]
    Checksum 0x2921, unused inodes 7738
    Primary superblock at 0, Group descriptors at 1-4
    Reserved GDT blocks at 5-1028
    Block bitmap at 1029 (+1029), Inode bitmap at 1045 (+1045)
    Inode table at 1061-1572 (+1061)
    22880 free blocks, 8174 free inodes, 2 directories, 7738 unused inodes
    Free blocks: 9381-9672, 10180-32767
    Free inodes: 19-8192
Group 1: (Blocks 32768-65535) [INODE_UNINIT, ITABLE_ZEROED]
    Checksum 0x473e, unused inodes 8192
    Backup superblock at 32768, Group descriptors at 32769-32772
    Reserved GDT blocks at 32773-33796
    Block bitmap at 1030 (bg #0 + 1030), Inode bitmap at 1046 (bg #0 + 1046)
    Inode table at 1573-2084 (bg #0 + 1573)
    14918 free blocks, 8192 free inodes, 0 directories, 8192 unused inodes
    Free blocks: 33797, 33800-33919, 34108-34511, 34521-34559, 34784-34815, 37053-38015, 3
    38529-38911, Free inodes: 8193-16384
.....
Group 196: (Blocks 6422528-6455295) [INODE_UNINIT, ITABLE_ZEROED]
    Checksum 0x946d, unused inodes 8192
    Block bitmap at 6291460 (bg #192 + 4), Inode bitmap at 6291476 (bg #192 + 20)
    Inode table at 6293536-6294047 (bg #192 + 2080)
    32768 free blocks, 8192 free inodes, 0 directories, 8192 unused inodes
    Free blocks: 6422528-6455295
    Free inodes: 1605633-1613824
    ....
Group 232: (Blocks 7602176-7630591) [INODE_UNINIT, ITABLE_ZEROED]
    Checksum 0xa174, unused inodes 8192
    Block bitmap at 7340040 (bg #224 + 8), Inode bitmap at 7340056 (bg #224 + 24)
    Inode table at 7344160-7344671 (bg #224 + 4128)
    28416 free blocks, 8192 free inodes, 0 directories, 8192 unused inodes
    Free blocks: 7602176-7630591
    Free inodes: 1900545-1908736
```

Y con **tune2fs** podemos modificar esos valores:

```
Para cambiar el número máximo de montajes entre verificaciones del sistema de archivos (ma
```

```
$ su -c "tune2fs -c 25 /dev/sda1"

Para cambiar el intervalo de tiempo entre verificaciones (interval-between-checks):

$ su -c "tune2fs -i 10 /dev/sda1"

Para listar el contenido del superbloque incluyendo los valores actuales de los parámetros

$ su -c "tune2fs -l /dev/sda1"
```

12.6. Información del Superbloque

Contiene:

- Cuenta de montajes y cuenta máxima de montajes. La cuenta se reinicia cada vez que se ejecuta **fsck** para verificar el sistema de archivos. Por defecto este se hace cada 180 días (configurable con *tune2fs).
- Tamaño del bloque
- Bloques por grupo
- Cuenta de bloques por sistema
- Cuenta de inodos disponibles
- ID del sistema operativo

12.7. Características de los bloques de datos e inodos

El bloque de datos y el mapa de bits de inodos son bloques cuyos bits contienen 0 para cada inodo libre y 1 para cada uno que es usado. Hay uno de cada por cada bloque. La tabla de inodos tiene tantos grupos consecutivos cómo necesite para cubrir el número de inodos del grupo. Cada inodo requiere 128 bytes; por lo tanto, un bloque de 4 KB puede contener 32 inodos.

Recordemos que el número de inodo no es dependiente del disco, su valor se calcula desde el número de bloque y el offset de la tabla del inodo.

Los sistemas de archivos ext2 y ext3 no han incorporado aún el uso de **extents** para organizar archivos grandes (si lo incorpora ext4, y es uno de los motivos por los que supone una importante mejora respecto a estos). En vez de eso, el arreglo de punteros a bloques de datos $i_{block}[]$, de largo $EXT2_{NBLOCKS}=15$, es descrito por el inodo.

12.8. Mejoras de ext4

Este sistema:

- Es compatible con sus dos predecesores
- El tamaño máximo del sistema de ficheros es de 1 EB (ExaByte) (desde 16 TB), y el máximo de un fichero es de 16 TB (desde 2 TB).

- Número ilimitado de directorios
- Usa **extents**, mejorando así el manejo de ficheros grandes y reduce la fragmentación (casi inexistente)
- Usa asignación multibloque, la cual puede asignar todo el espacio de golpe.
- Puede preasignar el espacio de disco de un archivo, haciendo que este sea contiguo (y, otra vez, reduciendo la fragmentación)
- Usa allocate- on- flush, una técnica de rendimiento la cual retrasa la asignación de bloques hasta que los datos son escritos en el disco
- Usa checksums para el journal, lo cual mejora la confiabilidad
- Usa timestamps mejoradas, medidas en nanosegundos
- Incluye soporte de snapshots

13. Los sistemas de archivos XFS y btrfs

13.1. XFS

Características:

- Hasta 16 EB para el total del sistema de archivos
- Hasta 8 EB para un archivo individual
- Usa E/S de DMA (Direct Memory Access)
- Garantiza una tasa de E/S
- Ajusta tamaño de banda para igualar los dispositivos RAID o LVM subyacentes
- Puede manejar información de las cuotas con journaling (esto reduce el tiempo de recuperación del sistema en caso de fallo)
- Soporta atributos extendidos

13.1.1. Mantenimiento

Una ventaja muy importante es que este se puede realizar en caliente, es decir, estando el sistema de ficheros montado. Puede desfragmentar, ampliar o realizar respaldos o restauraciones.

Para crear respaldos y restaurar se usa `xfsdump` y `xfsrestore`. Una curiosidad es que estos pueden ser pausados en cualquier momento y retomados a conveniencia. Además, son multihilo, por lo que se realizan muy rápido.

Las herramientas típicas para manejo de cuotas pueden ser usadas (véase el capitulo 13) pero si se usa `xfs-quota` se podrán modificar los atributos propios de este sistema de ficheros.

13.2. BRTFS

Este es un sistema de ficheros que apunta a ser usado en el **ámbito empresarial**. Su intención es ofrecer propiedades que ext4 no ofrece, tales cómo resolver la falta de pooling, snapshots, checksums e integral multi-device spanning. Es considerada experimental, por lo que realmente pocos vendedores lo han usado en sus productos.

Una de sus características principales es la habilidad para tomar **snapshots** (imágenes instantáneas) frecuentes del sistema de archivos. Dado que usa la técnica **COW** (Copy On Write) estas imágenes no ocupan mucho espacio. Es posible revertir el sistema de ficheros a un estado anterior o incluso pasarle al kernel un parámetro que cargue un estado anterior. También tiene su propio sistema de gestión de particiones, similar al que tiene **LVM**.

14. Logical Volume Manager (LVM)

Sirve para crear y gestionar particiones virtuales. Tiene muchas ventajas frente al uso de particiones lógicas o primarias, particularmente por su facilidad en el momento de modificar el tamaño de las particiones una vez creadas.

Los grupos de volúmenes (VG) son los que contendrán todo el montaje. Dentro de cada uno, habrá un número **n** de volúmenes lógicos (que seria el equivalente de las particiones en los discos físicos), tantos cómo se requieran. Dentro de estos es dónde se encontraran los sistemas de archivos.

Hay muchas herramientas para gestionar el **LVM**. La mayoría de sistemas cuentan con **system-config-lvm**, que además es gráfica.

Hay un impacto en el rendimiento, ya que se añaden capas y eso siempre cuesta algo. Aún así se puede mejorar si se usa **striping** (dividir los datos en más de un disco).

14.1. LVM y RAID

Al igual que el **RAID**, los volúmenes lógicos no están limitados por los discos físicos. Esto quiere decir que podemos tener un volumen lógico en el disco **n** y otra en el disco **n+1** sin que el sistema operativo se resienta.

Para crear volúmenes lógicos primero debemos crear grupos de volúmenes.

El LVM comparte varias características con LVM, de hecho puede estar montado encima de un RAID. Así se dispondría de la redundancia del RAID junto a la escabilidad de LVM.

14.2. Volúmenes lógicos y sus Grupos

Las particiones se convierten a volúmenes físicos, y estos se agrupan en grupos de volúmenes, cómo ya hemos contado. Hay varias herramientas relacionadas con LVM. Si empiezan por **vg**, quiere decir que manejan los grupos de volúmenes (Volume Group en inglés), si empiezan por **pv** hacen referencia a las particiones físicas (Physical Volume). Por último, si empieza pos **lv**, hace referencia a volúmenes lógicos (Logical Volume):

- vgcreate
- vgextend

- vgreduce
- pvcreate
- pvdisplay
- pvmove
- pvremove
- lvchange
- lvcreate
- lvconvert

Para ver todas las herramientas, se puede ejecutar `man lvm` en la terminal. Realmente no es necesario explicar que hace cada herramienta, ya que el propio nombre lo indica.

14.3. Proceso de creación de un grupo de volúmenes

- Crear partición física
  - `fdisk,cfdisk,gdisk,cgdisk`
- Crear volumen físico
  - `pvcreate`
- Crear grupo de volúmenes
  - `vgcreate`
- Crear volúmenes lógicos
  - `lvcreate`
- Formatear volumen lógico
  - `mkfs`
- Montar volúmenes lógicos
  - `mount`

14.4. Redimensionar volúmenes lógicos

Esta es una de las grandes ventajas de LVM, es muy sencillo redimensionar, tanto acortar cómo alargar particiones. Hay que tener en cuenta que el sistema de archivos que contenga el volumen lógico hay que adaptarlo a la partición y LVM no se encarga de ello. Para esto, se usa `resize2fs` para ext4, por ejemplo. Depende del sistema de archivos, se podrá operar con el si está montado o no, será conveniente revisarlo antes de hacerlo.

Ejemplo de redimensionamiento:

```
# Alargar
$ su -c "lvextend -L +500m /dev/vg/milvm"
$ su -c "resize2fs /dev/vg/milvm"

# Acortar
$ su -c "umount /dev/vg/milvm"
$ su -c "fsck -f /dev/vg/milvm" # comprobamos que no hay errores en el sistema de ficheros
$ su -c "resize2fs /dev/vg/milvm 500M"
$ su -c "lvreduce -L 200M /dev/vg/milvm"
$ su -c "mount /dev/vg/milm /mnt/"

# Acortar volumen físico
$ su -c "pvmove /dev/sdc1"
$ su -c "vgreduce vg /dev/sdc1"
```

14.5. Snapshots de LVM

Estas son una copia exacta de los volúmenes lógicos. Para hacerlas:

```
# Crear el snapshot
$ su -c "lvcreate -l 128 -s -n misnapshot /dev/vg/milvm"
# Montar el snapshot
$ su -c "mount -o ro /dev/vg/misnapshot /mnt"
# Quitarla
$ su -c "lvremove /dev/vg/misnapshot"
```

15. RAID

Un **RAID** reparte la actividad de escritura y lectura en múltiplos discos físicos a la vez en vez de uno sólo. De este modo se mejora la integridad y capacidad de recuperación en caso de fallo de un disco, además de aumentar el rendimiento. Hay muchos tipos de RAID a escoger según las necesidades relativas a la seguridad, rendimiento, complejidad y costo.

Se dice que usando un **RAID** (Redundant Array of Independent Disks) aumenta el rendimiento, pero es sólo usando controladoras (drivers) modernas, tales cómo el **SCSI**. RAID se puede implementar por parte de software o hardware. Si se usa por parte de hardware, el sistema de archivos no es consciente de que está realmente en RAID, por lo que no necesita ninguna configuración. Por ejemplo, dos discos duros de 512GB con hardware de RAID, se verá cómo un disco de 1TB.

Una desventaja del RAID de hardware es que si la controladora de disco falla, debe ser reemplazada por una compatible, lo cual no es siempre sencillo, ya que pueden haber desaparecido del mercado, por poner un ejemplo. Con RAID de software no tendremos ese problema, ya que mientras los discos duros que se usen sean compatibles con GNU/Linux, se podrán usar en RAID.

Tres características del RAID son:

- **mirroring**: escribe los mismos datos en más de un disco
- **striping**: divide o reparte los datos en más de un disco
- **parity**: datos extra se almacenan para detectar y reparar problemas, dando así una mayor tolerancia a fallos

15.1. Tipos de RAID

Hay distintas especificaciones según complejidad:

- **RAID 0**: usa **stripping** solamente, lo que significa que no se redundan, en cada disco hay distintos datos. La única ventaja que aporta es la mejora del rendimiento, ya que se paralelizan las tareas de E/S.
- **RAID 1**: usa **mirroring** solamente, lo que significa que sólo se redundan los datos, sin ninguna mejora de rendimiento. Requisito: dos discos.
- **RAID 5**: usa una banda de **paridad** de rotación, esto significa que si un disco falla, no se producirá pérdida de datos, sólo se reducirá el rendimiento. Requisito: tres discos.
- **RAID 6**: usa discos con **stripping** y **paridad** dual, esto significa que puede soportar la pérdida de dos discos. Requisito: cuatro discos.
- **RAID 10**: usa un set de datos con **mirroring** y **stripping**. Requisito: cuatro discos.

Cómo regla general, cuantos más discos, mejor rendimiento.

15.2. Proceso de configuración RAID de software

- Crear particiones en cada disco
- Crear el dispositivo RAID
- Formatear el dispositivo RAID
- Capturar los detalles del RAID para asegurar su persistencia
- Montar el dispositivo
- Agregar las lineas necesarias al `/etc/fstab` para asegurar que se monte cuando se reinicie

Ejemplo del procedimiento:

```
$ su -c "fdisk /dev/sdb"
$ su -c "fdisk /dev/sdc"

$ su -c "mdadm --create /dev/md0 --level=1 --raid-disks=2 /dev/sdb1 /dev/sdc1"

$ su -c "mkfs.ext4 /dev/md0"

$ su -c "mdadm --detail --scan >> /etc/mdadm.conf"

$ su -c "mount /dev/md0 /mnt"

$ su -c "echo '/dev/md0 /mnt ext4 defaults 0 2' >> /etc/fstab"
```

15.3. Distintos procedimientos de mantenimiento

```
# Para ver el estado del RAID
$ cat /proc/mdstat
$ su -c "mdadm --detail /dev/md0"

# Detener el RAID
$ su -c "mdadm -S /dev/md0"

# Configurar monitor automático y envío de correo
$ su -c "echo 'MAILADDR alguien@algun.sitio' >> /etc/mdadm.conf"
$ su -c "service mdmonitor start"
```

15.4. Configurar redundancia

Esta es una de las características más importantes del RAID, por lo que hay que configurarla:

```
# La opción -x 1 es la que concreta que hay que usar un disco de reserva
$ su -c "mdadm --create /dev/md0 -l 5 -n3 -x 1 /dev/sda8 /dev/sda6 /dev/sdb1 /dev/sdc1"

# Comprobar que la redundancia funciona
$ su -c "mdadm --fail /dev/md0 /dev/sdb1"

# En caso de fallo real, hay que quitar el miembro defectuoso y añadir el disco nuevo
$ su -c "mdadm --remove /dev/md0 /dev/sdb1"
$ su -c "mdadm --add /dev/md0 /dev/sde2"
```

16. Seguridad del sistema local

Cuando se habla de seguridad, se habla tanto de seguridad lógica cómo de seguridad física. Si se se hace todo lo posible en materia de seguridad lógica, pero cualquier persona puede tener acceso físico al ordenador (y por tanto tener otros vectores de ataque que no se pueden cubrir con la seguridad lógica), se en la tarea de proteger el sistema.

Lo primero que debe quedar claro es que la seguridad informática no es un valor seguro. Es decir, por mucho que se haga, siempre habrá una manera de

que los intrusos entren en un ordenador. Lo único que podemos hacer es ponerlo lo más difícil posible (que no es poco).

El mayor problema al aumentar la seguridad, es encontrar un punto de equilibrio entre seguridad y productividad, ya que la seguridad suele acarrear incomodidades que según el tipo que sean, pueden dificultar el trabajo. Si las medidas de seguridad son complicadas, las usuarias no seguirán los protocolos, por lo que habrá un agujero en la seguridad.

Cuando se habla de seguridad, se habla de proteger cuatro áreas, física, local, remota y personal. En esta sección se hablará de los factores locales.

16.1. Crear una política de seguridad

Una política de seguridad debería cumplir lo siguiente:

- Ser simple
- Ser actualizada constantemente
- Estar plasmada en un documento y disponible en línea de ser necesario
- Describir tanto políticas cómo procedimientos
- Especificar acciones restrictivas
- Especificar procedimiento a ejecutar cuando se da una brecha de seguridad

  Deben ser genéricas y fáciles de entender, para facilitar su seguimiento. Deben proteger la información que necesita ser protegida, denegar el acceso a los servicios y proteger la privacidad de los usuarios.

16.1.1. Que incluir en esta

Métodos de protección de la información para evitar que sea leída o no autorizada por personal no autorizado (esto incluye trabajadores que no necesiten esa información) y protección ante alteraciones no autorizadas.

Los aspectos esenciales son:

- Confidencialidad
- Integridad
- Disponibilidad
- Consistencia
- Control
- Auditoría

Hay que recordar que el aspecto más débil de la cadena de seguridad es el factor humano y puede ser aprovechado mediante ingeniería social.

16.1.2. Hacer un análisis de riesgos

Hay que preguntarse lo siguiente: Que quiero proteger? Contra quien lo protejo? Cuánto tiempo, personal y dinero es necesario para conseguir la protección adecuada?

16.1.3. Filosofía de seguridad

Generalizando, hay dos tipos de filosofías. Una es permitir cualquier cosa que no esté denegada explícitamente (blacklist) y la otra prohibir cualquier cosa que no esté permitida explícitamente (whitelist).

La segunda es la más usada ya que es la que más seguridad provee. Es recomendable usar esta, a menos que realmente se esté en un ambiente de confianza.

16.1.4. Guías generales de seguridad

- El factor humano es el más débil: Hay que educar al usuario y mantenerlo contento, ya que la mayor parte de vulneraciones de seguridad son internas y sin intención maliciosa.
- No existe un ordenador invulnerable: El único sistema informático seguro es el no está conectado a nada, en una habitación segura y apagado.
- La paranoia es buena: Hay que sospechar, estar atento y ser perseverante al asegurar un sistema informático.

16.2. Actualizaciones

Hay que prestar atención a las actualizaciones, ya que en sistemas estables sólo proveen de parches de seguridad ante problemas conocidos. Mucha gente usa los agujeros de seguridad encontrados desde que se hace pública la vulnerabilidad hasta que se publican los parches que arreglan el problema. Estos ataques se llaman **ataques del día cero** (o 0day en inglés), pero son más raros. Lo habitual es aprovechar esa vulnerabilidad atacando sistemas que se toman a la ligera la política de actualizaciones.

No es nada raro encontrar administradoras que son reacias a aplicar actualizaciones nada más son liberadas, principalmente por malas experiencias con empresas de software privativo, ya que es cierto que pueden causar más problemas que los que solucionan. Sin embargo, es extremadamente raro que pase esto en el ámbito de GNU/Linux, por lo que no es justificable retrasar la aplicación de parches.

16.3. Acceso físico al hardware

Si se tiene acceso al hardware se pueden aprovechar muchas vulnerabilidades. Algunos ejemplos:

- Key logger: un aparato que captura todo lo que se presiona en el teclado y se almacena y/o envía a un servidor remoto.
- Analizador los paquetes de red
- Arrancar con un live CD y acceder/modificar el contenido del disco (esto se puede mitigar cifrando los discos)

Para asegurar el hardware hay que hacer lo siguiente:

- Asegurar estaciones de trabajo y servidores (principalmente los puertos USB)

- Proteger los enlaces de red de personas no confiables
- Proteger los teclados para que no puedan ser modificados
- Proteger la BIOS para que no se pueda cambiar a acceso por live CD

16.3.1. Protección de la BIOS

La BIOS es el nivel más bajo al que permite configuración un ordenador. Es buena práctica ponerle una contraseña para dificultar el acceso.

Apunte: Esto es lo que dice el curso, de hecho recalca en que es algo necesario pero en realidad es muy fácil resetear la BIOS y hacer que olvide esa contraseña, sólo hay que quitar la pila y esperar aproximadamente un minuto. Por lo tanto, si el acceso al interior del ordenador no está protegido por un candado, le veo más bien poco sentido.

También es recomendable estar atento a las actualizaciones del firmware de la BIOS, y mantenerse actualizado dentro de lo posible. Aún así, muchas actualizaciones no son en materia de seguridad, por lo que hay administradoras que prefieren no aplicarlas, ya que de hacerlo mal o de estar mal hechas, dejaría el ordenador inutilizable.

16.3.2. Protección del cargador de arranque (GRUB)

Se puede modificar GRUB para poder registrarse cómo el usuario sin que pregunte la contraseña, por lo que es una buena práctica ponerle una contraseña. Esto no evitará que se pueda arrancar con un live CD, por lo que es recomendable usarlo en conjunto con la contraseña en la BIOS.

Para poner una contraseña en GRUB 1, se puede usar la herramienta `grub-md5-crypt`, que preguntará por una contraseña y luego la configurará. Después de esto, sólo hay que añadir la linea siguiente a `/boot/grub/grub.conf`, debajo de la línea que contiene la entrada timeout:

```
password --md5 $1$mas.adkaskdaksdkasdoow
```

La cadena que empieza por el símbolo $ es la contraseña que se introdujo anteriormente hasheada.

En la versión 2 de GRUB no se puede modificar el fichero a mano, ya que de eso se encargan las herramientas de grub, por eso se modificaran las configuraciones de `/etc/grub.d/`. Más información al respecto en el foro de ubuntu.

16.3.3. Seguridad del sistema de archivos

Esto hace referencia a las opciones con las que se monta un sistema de ficheros, usualmente mediante `/etc/fstab`. Los valores que hacen referencia a la seguridad son **nodev**, **nosuid**, **noexec** y **ro**, a grandes rasgos.

16.3.4. setuid y setgid

Los programas, a menos que tengan en su fichero de configuración la posibilidad de pasar permisos a otro usuario, corren con los permisos del que lo ejecuta, sin importar realmente de quien sea el programa mientras el usuario que lo ejecuta tenga permisos de ejecución, aunque puede tener capacidades

restringidas. Un ejemplo es `ifconfig`, que permite ser ejecutado por un usuario común para ver la IP de la máquina, pero en cambio no permite cambiar la IP.

Al configurar el bit **setuid** (set user id) de un archivo ejecutable, se puede modificar el comportamiento habitual del programa y otorgar permisos de acceso al dueño en vez de al usuario que lo ejecuta. Además se puede configurar el bit **setgid**, que haría lo mismo que el anterior pero haciendo referencia al grupo del dueño.

Hay que enfatizar que esto es complejo de hacer correctamente, es decir, manteniendo la seguridad del sistema. Es más recomendable hacer correr un demonio con menos privilegios.

Ambos bits se configuran de la siguiente manera:

```
# setuid
$ chmod u+s file.sh
# setgid
$ chmod g+s file.sh
```

17. Módulos de seguridad de GNU/Linux

En esta sección se hablará de los módulos de seguridad de Linux (LSM) (aquí si hace referencia sólo al kernel), concretamente implementados mediante **SELinux**.

17.1. Qué son los LSM?

El fin de estos módulos es implementar **controles de acceso** obligatorios sobre n variedad de peticiones realizadas al kernel. De este modo se minimizarían los cambios en el kernel, su sobrecarga y además permite flexibilidad, pudiendo escoger que módulos se quieren aplicar.

Esto lo hace interceptando las peticiones al kernel e inyectando código que comprueba los permisos, añadir protección contra ataques malintencionados, entre otros.

17.2. Alternativas de LSM

Recordemos que LSM es lo que se implementa, cómo se implementa es cuestión de escoger. Por defecto, la mayoría de distribuciones escogen SELinux (menos Ubuntu, que usa apparmor). Veamos las alternativas que hay:

- SELinux: `http://selinuxproject.org/page/Main_Page`
- AppArmor: `http://apparmor.net`
- Smack: `http://schaufler-ca.com`
- TOMOYO: `http://tomoyo.sourceforge.jp`

Sólo se puede usar un método a la vez, por lo que en este capitulo se hablará principalmente de **SELinux** y de manera secundaria de **AppArmor**.

17.3. Resumen de SELinux

SELinux ha sido desarrollado originalmente por la NSA y lleva formando parte de RHEL durante mucho tiempo, por lo que es el

Apunte: Hay algunas personas que no confían en ello, por lo de que la NSA quiere joder a toda persona con ordenador. Pero, en mi opinión, es un miedo infundado, no por que haya que confiar en la NSA, sino por que SELinux es de código abierto, por lo que si alguien duda (y seguro que mucha gente ha dudado), sólo hay que auditarlo.

Se puede definir SELinux cómo un conjunto de reglas de seguridad que se usan para determinar qué procesos pueden acceder a determinados archivos o puertos del sistema.

Se usa de tres formas conceptuales:

- Por **contexto**: etiquetas a archivos, procesos y puertos. Un ejemplo son los usuarios y grupos.
- Por **reglas**: describen el control de acceso en términos de contexto, procesos, archivos, puertos, usuarios, etc.
- Por **políticas**: Son un conjunto de reglas que describen las decisiones a tomar por SELinux

Los **contextos** son nombres usados por **reglas** para definir que usuarios, procesos y puertos **interactúan** entre ellos.

17.4. Modos de SELinux

Hay tres modos que puede usar:

- **Enforcing**: Es el modo más seguro, se ejecuta todo el código de SELinux y se aplican las políticas. Todas las violaciones quedan registradas y auditadas.
- **Permisive**: Habilita el código de SELinux pero lo deja pasar todo, sólo advierte de que sería denegado si las reglas estuviesen aplicándose mediante el registro.
- **Disabled**: Sin aplicar nada.

Estos modos se seleccionan en un archivo (generalmente `/etc/selinux/config`). En ese mismo archivo están muy bien documentadas todas las copiones posibles.

17.5. Herramientas

Hay distintas herramientas:

```
# Despliega el modo y política actual
$ su -c "sestatus"
# Ver el modo actual
$ su -c "getenforce"
# Cambiar el modo actual
$ su -c "setenforce Permissive"
```

`setenforce` se usa para cambiar los modos al vuelo entre **Enforcing** y **Permissive**, pero no permite deshabilitarlo por completo, su uso es más bien para el de comprobar que las políticas son correctas cuando algo falla. Para deshabilitarlo completamente, se puede añadir la línea `SELINUX=disabled` en el archivo `/etc/selinux/config` o agregar el parámtero `selinux=0` al kernel y reiniciar.

Si no se va a deshabilitar para siempre, no es recomendable usar los métodos anteriores, ya que cuando se habilite de nuevo tendrá que etiquetar todos los archivos de nuevo y consumirá mucho tiempo. De ser este el caso, es usar el modo **Permissive**.

17.6. Políticas de SELinux

Se suele definir en el mismo archivo que se ha comentado anteriormente, `/etc/sysconfig/selinux`. Sólo se permite tener una política activa a la vez, y normalmente requiere de un reinicio y un reetiquedado de los ficheros (lo cual es muy lento) al activarlo. Las políticas más comunes son:

- **targeted**: Es la que suele venir por defecto y restringe procesos específicos. Los procesos de usuario común y del init no están monitorizados. Además, se imponen restricciones de memoria para todos los procesos para evitar o reducir la posibilidad de un desbordamiento del búffer (Buffer Overflow).
- **minimum**: Una variación de **targeted**, sólo los procesos seleccionados por la administradora.
- **MLS**: Significa Multiple Levels Security en inglés, y es la más restrictiva. Todos los procesos se ponen en dominios de seguridad específicos con políticas particulares para ellos.

17.7. Herramientas de contexto

Cómo ya se dijo antes, los contextos son etiquetas que se aplican a archivos, directorios, puertos y procesos. Estas etiquetas describen el control de acceso. Los cuatro tipos de contexto que existen son:

- Usuario
- Rol
- Tipo
- Nivel

Cómo el contexto más usado es el tipo, es en el que focalizaremos. La convención de etiquetas determina que este tipo de etiquetas debe terminar con `\_t`, cómo por ejemplo `kernel\_t`.

17.8. Herramientas extendidas

Hay algunas herramientas que fueron extendidas para ser usadas con SELinux, tal cómo `ps` o `ls`. Se suele usar el mismo parámetro para todas estas herramientas, que suele ser **-Z**. Por ejemplo:

```
$ ls -Z /var/www/html/index.html
-rw-r--r--  username username system_u:object_r:httpd_sys_content_t /var/www/html/index.ht
```

Otras herramientas extendidas son `cp`, `mv`, `mkdir`.

17.9. Herencia de contextos

Los archivos heredan en función del directorio padre en el que estén, pero algo que suele ocasionar problemas es al mover los archivos de directorio. Por ejemplo, un fichero creado bajo `/var/www/` tendrá la etiqueta de nginx/apache, pero si se crea el archivo en `/tmp` y luego se mueve a `/var/www/`, la etiqueta seguirá siendo la de `/tmp`. Un ejemplo:

```
$ touch /var/www/html/index.html && ls -Z /var/www/html/index.html
-rw-r--r--  drymer users  system_u:object_r:httpd_sys_content_t /var/www/html/index.html

$ touch /tmp/example.txt && ls -Z /tmp/example.txt
-rw-r--r--  drymer users  system_u:object_r:user_tmp_t /tmp/example.txt

$ mv /tmp/example.txt /var/www/html/ && ls -Z /var/www/html/
-rw-r--r--  drymer users  system_u:object_r:user_tmp_t /tmp/example.txt
-rw-r--r--  drymer users  system_u:object_r:httpd_sys_content_t /var/www/html/index.html
```

Se tiene que cambiar la etiqueta a mano, ya que sino `example.txt` será inaccesible para SELinux.

17.10. Más herramientas

Para restablecer el contexto, cómo se vio que hace falta en el punto anterior, se puede usar `restorecon`. Lo que hace es reetiquetar los archivos de un directorio en función de la configuración del directorio padre. Para arreglar la etiqueta de `/tmp/example.txt` del punto anterior, se usaría:

```
$ su -c "restorecon -Rv /var/www/http/"
```

Otro problema habitual es crear la configuración por defecto de un contexto. Esto se puede realizar con la herramienta `semanage fcontext`, que es proveída por el paquete `policycoreutils-python` y permite desplegar y cambiar el contexto por defecto de los archivos. Un ejemplo de configuración inicial para `/var/www/http`:

```
$ su -c "semanage fcontext -a -t https_sys_content_t /var/www/html"
$ su -c "restorecon -RFv /var/www/html"
```

Hasta que no se ejecuta `restorecon`, los archivos y el propio directorio padre no son reetiquetados, por lo que es importante no olvidarlo.

17.11. Booleanos en SELinux

Estos se usan para el cambio de políticas de SELinux en caliente. Para verlos, ejecutar lo siguiente:

```
$ su -c "semanage boolean -l"

SELinux boolean                State  Default Description
ftp_home_dir                   (off  ,  off)  Allow ftp to read and write files in the use
smartmon_3ware                 (off  ,  off)  Enable additional permissions needed to supp
xdm_sysadm_login               (off  ,  off)  Allow xdm logins as sysadm
xen_use_nfs                    (off  ,  off)  Allow xen to manage nfs files
mozilla_read_content           (off  ,  off)  Control mozilla content access
ssh_chroot_rw_homedirs         (off  ,  off)  Allow ssh with chroot env to read and write
postgresql_can_rsync           (off  ,  off)  Allow postgresql to use ssh and rsync for po
allow_console_login            (on   ,   on)  Allow direct login to the console device. Re
spamassassin_can_network       (off  ,  off)  Allow user spamassassin clients to use the n
authlogin_shadow               (off  ,  off)  Allow users login programs to access /etc/sh
httpd_can_network_relay        (off  ,  off)  Allow httpd to act as a relay
openvpn_enable_homedirs        (on   ,   on)  Allow openvpn to read home directories
```

Para ver una salida más simple, se puede entubar el anterior comando con `grep` y el valor que se busca o se pude usar `getsebol` y el valor que se busca. Para cambiar el valor de manera no persistente se usa `setsebol` tal que así:

```
$ su -c "setsebol ssh_chroot_rw_homedirs on"
ssh_chroot_rw_homedirs --> on

# Al reiniciar, este valor se habrá perdido. Para hacerlo persistente:
$ su -c "setsebol -P ssh_chroot_rw_homedirs on"
ssh_chroot_rw_homedirs --> on
```

17.12. Herramientas de resolución de Problemas

Siempre se escapará algún archivo del control de la administradora de sistemas, y cuando se vea que algo no funciona se podrá instalar el paquete `setroubleshoot-server`. Suponiendo que se esté monitorizando un servidor web:

```
$ touch /var/www/http/error.txt
$ tail /var/log/messages
Nov 21 13:00:00 rhel7 setroubleshoot: SELinux is preventing /usr/sbin/httpd from getatr ac
```

17.13. Recursos adicionales

- `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/SELinux_Users_and_Administrators_Guide/`
- `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/`

18. Procesos

Apunte: Los contenidos de este capitulo tienen telita, pero son los que vienen. Este es, hasta el momento, el capítulo más aburrido, por el tema en si

y sobretodo por la manera en la que está escrito. En este he copiado de manera mucho más similar al contenido real que en los anteriores.

Un **proceso** es la esencia de una aplicación. Puede consistir o no de más hilos o hebras (subprocesos). Estos tienen atributos enumerados, cómo pueden ser running (corriendo) o sleeping (durmiendo). También con que permisos se está ejecutando, cómo los de los usuarios o de acceso a la red.

18.1. Procesos, programas y hebras

GNU/Linux siempre se ha caracterizado por la rapidez en crear, modificar, conmutar y destruir procesos. El motivo principal es que los procesos se tratan siempre cómo procesos sin relación a otros, aunque estos tengan padre (procesos multihilos). De este modo se evita complicar la manera de tratarlos, por que se trata del mismo modo el proceso con permisos root de nginx que los subprocesos en los cuales se han rebajado los permisos y se ejecutan con el usuario www-data, por ejemplo.

Además, **GNU/Linux** respeta los estándares **POSIX**, y en este aspecto, todos los subprocesos tienen su propio identificación. Siguiendo con el anterior ejemplo:

```
$ ps aux | grep nginx
root     23369  0.0  0.0  22484  1160 ?        Ss   mar06   0:00 nginx: master process /us
www-data 23370  0.0  0.2  23300  3732 ?        S    mar06   9:26 nginx: worker process
www-data 23371  0.0  0.2  23392  3880 ?        S    mar06   9:18 nginx: worker process
www-data 23372  0.0  0.2  23432  4040 ?        S    mar06   8:52 nginx: worker process
www-data 23373  0.0  0.2  23284  3744 ?        S    mar06   9:00 nginx: worker process
```

18.2. El proceso init

Este es el primer proceso en ejecutarse en el sistema, por lo que tiene le ID 1. Es independiente del sistema de arranque que se use. También será el último proceso que se apague al reiniciar o apagar el ordenador.

18.3. Procesos

Cada proceso tiene ciertos atributos (cómo mínimo): **pid** (Process ID), **ppid** (Parent Process ID), **pgid** (Process Group ID). También pueden tener código de un programa, datos, variables, descriptores de archivos y un ambiente.

Cómo ya hemos dicho, casi todos los procesos penden de init, ya que es el primer proceso en ejecutarse del sistema. Pero no todos penden de el, hay una excepción que son los procesos que abre el kernel, que aparecerán con [] alrededor de estos cuando se ejecuta `ps`.

Si el proceso padre de cualquier proceso que se ejecute muere, este pasa a ser hijo del init. Este tipo de procesos se les llama **zombie**, ya que no han podido leer la salida del que antes era su padre correctamente. De haberlo hecho, este les habría dicho que debian cerrarse. init tiene una función que se encarga de ir verificando a los **zombies** y dejar que mueran de manera correcta. También se le llama **asesino de zombies** o **segador de niños** (esto sale en los apuntes, literalmente).

Por razones históricas, el PID más grande se ha limitado a un número de 16 bits, el cual corresponde a 32768. Este se puede modificar cambiando el valor de `/proc/sys/kernel/pid_max`, aunque generalmente no hay motivo para hacer esto, ya que cuando se llega a este número, se reinicia el contador de procesos.

18.4. Atributos de un proceso

Hay cuatro atributos: el programa, contexto (estado), permisos y recursos asociados.

18.5. Permisos de los procesos y setuid

Cómo ya se ha comentado, cada proceso tiene permisos basados en el usuario que lo ha invocado. Tal cómo se comentó en la sección de seguridad local, los programas marcados con una **s** en el bit de ejecución tienen un identificador de usuario diferente al de su usuario real. Este tipo de programas son conocidos cómo **setuid**. Cuando este dueño es root, es cuando se conoce que este programa tiene potencial para ser muy dañino.

Un ejemplo de este tipo de herramientas setuid, es **passwd**, la herramienta usada para cambiar las contraseñas. Cualquier usuaria puede ejecutarlo y actualizar los archivos restringidos `/etc/passwd` y `/etc/shadow`, que es dónde se almacenan las contraseñas de los usuarios.

18.6. Estados de un proceso

Los estados en los que puede estar un proceso son los siguientes:

- **En ejecución**: cuando un proceso está ejecutándose o en la **cola de ejecución**.
- **Durmiendo**: cuando está esperando que una petición (habitualmente **E/S**) que no puede continuar hasta que se complete. En el momento en el que pueda hacerlo, pasará al primer estado.
- **Detenido**: cuando ha sido suspendido. Esto lo suele hace un programador cuando quiere examinar el contexto del proceso de ejecución, con un depurador o cuando presiona Ctrl-Z.
- **Zombie**: se explicó aquí.

18.7. Modos de ejecución

Hay dos tipos, el modo **usuario** y el modo **sistema** (también llamado modo **kernel**. Estos modos no son a nivel de software, sino de hardware. Estos modos no son un estado del sistema sino más bien una característica de los procesadores de más de un núcleo (todos, a estas alturas de la vida).

18.7.1. Modo usuario

Este modo tiene menos privilegios que el modo **kernel**. Al iniciar el proceso, este es aislado en su propio **userspace**, lo que promueve la seguridad y la estabilidad.

Cada proceso ejecutado de este modo, tiene su propio espacio de memoria racionado, que puede compartir con otros userspaces o no.

Un proceso llamado por el usuario root o cualquier programa con el setuid en root, se ejecuta en modo usuario siempre, excepto si este proceso necesita acceder al hardware, en cuyo caso se le da una capacidad limitada a este.

18.7.2. Modo kernel

En este modo la CPU tiene acceso completo a todos los componentes físicos del ordenador. Cómo ya se ha dicho, todos los procesos se ejecutaran de primeras en el modo usuario y de necesitar acceder a las funciones físicas del ordenador, debe realizar una llamada de sistema y cambiar el contexto del proceso. Este cambio se hará al escribir archivos nuevos, crear un nuevo proceso, etc.

Siempre se ejecutan en este modo las llamadas al sistema, nunca el código de los programas en si. Cuando la llamada termina, se devuelve un valor y se vuelve al modo usuario.

18.8. Demonios

Un **demonio**. es un proceso que está siempre en ejecución en segundo plano. Algunas características:

- Suelen ser más eficientes, ya que sólo se ejecutan en caso de ser necesario.
- Suelen iniciar en el arranque.
- Sus nombres suelen terminar en una **d**, para dejar claro que son demonios, cómo por ejemplo **httpd** o **udevd**.
- Generalmente no tienen una terminal de control, ni dispositivos de entrada o salida.

Si se usa **SysVinit**, se pueden encontrar los scripts de los servicios en el directorio `/dev/init.d/`, algunos de los cuales son demonios.

Si se quiere ver uno de estos scripts de demonios, se puede echar un ojo a `/dev/init.d/ntp`, que suele estar en todos los sistemas.

18.9. Procesos creados por el kernel

Ya se ha comentado que hay unos pocos procesos que son creados directamente por el kernel. En concreto, hay dos tipos de procesos que crea por si mismo. Estos son:

- **Procesos internos**: Se ocupan del mantenimiento, cómo por ejemplo asegurar que la carga se equilibra por igual en las CPU. Este tipo de procesos suele estar durmiendo.
- **Procesos externos**: Son los que inicia el kernel pero están corriendo cómo un proceso normal. Este tipo de procesos son muy raros y tienen una vida muy corta.

Para ver los procesos de ambos tipos, se puede ejecutar `ps -elf` en una terminal. Un ejemplo de los **internos**, seria kworker y un ejemplo de los **externos**, seria syslogd (para ver que es esto, ejecutar `man syslogd`).

18.10. Creación de procesos y forking

Un sistema GNU/Linux está creando procesos de manera continua. A veces se llama **forking**, a la característica de estos sistemas que hace que el proceso padre continúa ejecutándose mientras el proceso hijo se inicia. En otras ocasiones se usa otra llamada al kernel que es **exec**, que se caracteriza por hacer que el proceso hijo coja el identificador del padre y este muera sin más. Algunos sistemas ***nix** antiguos usan otro tipo de llamada al kernel similar a las dos anteriores, que es el **spawn**.

En realidad no hay mucha diferencia en términos de velocidad o optimización que hagan escoger el fork o el exec, ya que lo no consume más tiempo crear un hilo hijo que otro proceso.

18.11. Creación de procesos en una terminal

Cómo se gestiona el tema de la creación de procesos al usar comando en un intérprete cómo **bash**?:

- Se crea un fork del proceso padre, es decir, **bash**.
- Una llamada al sistema pone a dormir el proceso de la shell padre.
- El comando se carga en el userspace del hijo a través de una llamada **exec**. Dicho de otra manera, el comando ejecutado usa el espacio de memoria del proceso padre.
- El comando se completa y el proceso hijo muere con una llamada del sistema de salida.
- El proceso de bash despierta y crea un nuevo intérprete bash. Entonces, se queda a la espera de un nuevo comando, y en el caso de llegar, comenzaría de nuevo el bucle.

Si el comando ejecutado en la shell se ejecuta en segundo plano (añadiendo un símbolo de `&` al final), no se llega a crear un fork del proceso ni todos los pasos siguientes, sino que simplemente el proceso bash vuelve a estar listo para ejecutar otro comando, aún estando ese primer proceso ejecutándose en segundo plano.

Hay algunos comandos que están construidos en la propia shell, por lo que no llegan a necesitar sus propios procesos. Esto quiere decir que no llegan a usar nunca **exec** ni **fork**. Dos de estos son `echo` y `kill`. Cómo se puede comprobar esto? De la siguiente manera:

- Abrimos dos terminales.
- En una ejecutamos `while true; do echo "lol"; done`. Esta orden ejecuta un bucle infinito que imprime la palabra "lol.[a] cada vuelta que da.
- En la segunda terminal, hacemos algo similar: `while true; do ps aux|grep echo | grep -v grep; done`. Esta linea ejecuta otro bucle infinito en la que a cada vuelta, mira los procesos activos con `ps`, los filtra con `grep` buscando algún programa llamado `echo` y filtra fuera los propios procesos del `grep`.

Se verá que no aparece nada de nada en la segunda terminal, lo cual demuestra que `echo` no levanta ningún proceso.

18.12. Bibliotecas estáticas y compartidas

Los programas están construidos usando bibliotecas de código (también llamadas librerías por culpa de la mala traducción del inglés de la palabra *library*). Hay dos tipos, **estáticas** y **compartidas**. Las primeras son inyectadas en el momento de compilación y no es necesario que estas estén en el sistema operativo, las segundas en cambio si, ya que la biblioteca se carga en el programa en tiempo de ejecución.

El uso de las compartidas es más eficiente, ya que no tiene que cargarse una vez por programa que las use, sino que de haber dos programas que usan la misma, usan el mismo recurso compartido. (Aunque dicen que snap es molón por lo contrario, con un par).

Uno de los problemas de las bibliotecas compartidas es que deben ser vigiladas cuidadosamente, específicamente su versionado. Esto es debido a que si estas se actualizan y cambia su funcionamiento las aplicaciones que las usen tardaran más o menos tiempo en adaptarse a estas, pudiendo dejar un sistema roto. Esto no suele ser un problema, ya que la mayoría de distribuciones GNU/Linux tienen su repositorio de programas en el que miran cuidadosamente (o no, según la distribución) que todos los programas funcionen correctamente.

18.12.1. Herramientas

`ldd` es un programa para que sirve para encontrar las bibliotecas compartidas de los demás programas. Por ejemplo:

```
$ ldd ‘which nano‘
linux-gate.so.1 =>  (0xb77a6000)
libncursesw.so.5 => /lib/i386-linux-gnu/libncursesw.so.5 (0xb775f000)
libtinfo.so.5 => /lib/i386-linux-gnu/libtinfo.so.5 (0xb773f000)
libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb75d6000)
libdl.so.2 => /lib/i386-linux-gnu/i686/cmov/libdl.so.2 (0xb75d2000)
/lib/ld-linux.so.2 (0xb77a7000)
```

`ldconfig` es otra herramienta a tener en cuenta. Ser suele ejecutar al inicio del sistema y usa el archivo `/etc/ld.so.conf` para saber en que directorios debe buscar las bibliotecas compartidas. Antes de mirar en ese directorio, lo que hace es mirar en la variable de entorno **LD_LIBRARY_PATH**, por lo que se puede usar si se tiene la biblioteca en una dirección poco habitual y sólo se quiere usar esa de manera puntual. Para que al concretar esa variable no se queda fija en la terminal desde la que se hace, se puede ejecutar de tal modo:

```
LD_LIBRARY_PATH=/home/drymer/Instalados/nano nano
```

18.13. Herramientas para manejar procesos

18.13.1. `ulimit`

Está construido sobre la base de un comando **bash** (aunque por algún estúpido motivo no dicen cual), el cual despliega los límites asociados a los procesos. Se pueden desplegar tal que así:

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 14069
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 99
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 14069
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Una sysadmin podría cambiar algunos de estos valores para restringir las capacidades de algunos procesos, por ejemplo para evitar que un usuario acapare todos los recursos del sistema, cómo puede ser la memoria. También se puede usar para ampliar las capacidades, por ejemplo un servidor de ficheros en un sitio con más de 3000 usuarios podría necesitar que más de 1024 archivos estén abiertos a la vez.

Hay dos tipos de límites, llamados **soft** y **hard**, respectivamente **blando** y **duro** (o estricto). Los primeros son los que sólo puede modificar root, y los segundos los que puede modificar un usuario, con el límite que el administrador le haya puesto.

Por ejemplo, para cambiar el número de archivos abiertos, se ejecutaria lo siguiente:

```
$ ulimit -n 1600
```

Recordemos, esto lo podría ejecutar un usuario siempre que el administrador haya permitido previamente que el usuario suba a ese límite.

Aún así, hay que saber que `ulimit` sólo modifica estos valores para la shell abierta, por lo que si se quieren cambios persistentes hay que modificar el archivo `/etc/security/limits.conf`, que está muy bien documentado, y reiniciar.

18.13.2. `nice` y `renice`

Esta herramienta permite establecer prioridades de los procesos. La idea es que nice baja la prioridad del proceso con el que se ejecuta para dejar espacio a los demás procesos. Cuanto mayor es el número de nice, menor es la prioridad que tendrá y por lo tanto, menor el "derecho.ªl acceso a los recursos que tendrá.

El rango va desde -20 a 19. Por lo tanto, se podría usar del siguiente modo: `nice -n 5 cat &`. Hay que tener en cuenta que un nice con una prioridad baja vaya a ejecutarse más lentamente, ya que es un número relativo. Lo que hay que tener en cuenta para saber el tiempo aproximado que puede tardar la orden, es el uso de los recursos que hay en ese momento. Si todos los recursos están

libres, el comando `cat` del ejemplo anterior se ejecutará cómo al momento y sin espera.

`renice` sirve para cambiar la prioridad de un proceso en ejecución. En condiciones normales, un usuario no puede aumentar la prioridad, para que pueda el administrador debe editar el archivo `/etc/security/limits.conf`. Así que para cambiar la prioridad de un proceso, se ejecutará: `renice +3 13484`, siendo **13484** el proceso cuya prioridad de se quiere cambiar.

19. Señales

Las señales son usadas para emitir notificaciones a los procesos en función a eventos. Pueden ser ejecutados tanto desde fuera cómo de dentro del proceso. Para mandar cualquier tipo de notificación, se usa `kill`, `pkill` y `killall`.

19.1. Que son las señales?

Es uno de los métodos más antiguos usados en GNU/Linux, y cómo ya se dijo antes, se usa para mandar notificaciones a los procesos sobre los eventos de manera **asíncrona**.

Un ejemplo de un evento asíncrono es el de una usuaria usando un proceso para mandar una llamada al kernel para que termine el proceso `nano`. Hay que tener en cuenta que para que funcione, la usuaria debe ser dueña del proceso.

Depende de cómo esté hecho el programa, reaccionará de manera distinta a las señales que se le envíen, exceptuando **SIGKILL** y **SIGSTOP** (se verán a continuación), que no pueden ser manipuladas de ningún modo por el programa y siempre terminarán con su proceso.

19.2. Tipos de señales

Para ver la lista de señales de GNU/Linux, se ejecuta la siguiente orden:

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Para ver los usos de estas con más profundidad, se puede ejecutar `man 7 signal` en la terminal.

19.3. Herramientas

Los procesos nunca podrán enviar señales directamente a otro proceso, siempre tendrán que hacer la petición al **kernel**. Para eso, tenemos la herramienta `kill`.

Si no se le añade ningún parámetro, `kill` manda una señal **SIGTERM**, que es una señal de finalización que el programa no tiene por que obedecer inmediatamente. Muchos programas no se cierran sin más sino que se toman su tiempo para limpiar todos los recursos que ha estado usando. Esta es la manera correcta de terminar los procesos. Pero a veces falla, por lo que entonces hay que ejecutar `kill -9 1999` o `kill -SIGKILL 1999` (siendo 1999 el número del proceso). Ambos hacen lo mismo, que es mandar una señal **SIGKILL** al proceso, que termina de manera inmediata con este.

Después de esta, existe la herramienta `killall`, que cómo su propio nombre indica, mata todos los procesos con un nombre que se le pase cómo argumento. Los argumentos son los mismos que en `kill`. Por ejemplo, para matar todos los procesos llamados de **bash** se haría del siguiente modo:

```
killall bash
killall -9 bash
killall -SIGKILL bash
```

El primero mandaría la señal **SIGTERM**, cómo ya se ha mencionado antes, y los dos siguientes una señal **SIGKILL**.

`pkill` permite ser un poco más selectivo. Por ejemplo, la siguiente orden matará todos los procesos de `bash` que se llamen `tail`:

```
pkill -u bash tail
```

O para hacer que `syslogd` relea su fichero de configuración:

```
pkill -HUP syslogd
```

20. Monitorización del sistema

Una de las tareas de la **sysadmin** es monitorizar el sistema de forma eficaz. Por suerte, GNU/Linux trae muchas herramientas por defecto para ello, sin contar que en los repositorios hay muchas otras. En este capítulo se verán sobretodo las primeras.

La mayoría de estas herramientas hacen uso de los pseudosistemas de archivos `/proc` y `/sys`, por lo que en realidad se pueden usar esas herramientas o mirar en esos directorios. Lo recomendado es lo primero, ya que suelen dar los resultados en un formato más legible.

Veremos con más profundidad algunas de las siguientes herramientas (algunas están repetidas dados sus múltiples usos):

20.1. `/proc` y `/sys`

Se dice de estos que son pseudosistemas de archivos por que están **montados en memoria**. Aunque en su mayoría están pensados para ser leídos, se pueden modificar ciertos archivos (siempre teniendo en cuenta que serán cambios temporales).

Cuadro 1: Monitorización de procesos y carga de estos

Herramienta	Propósito	Paquete
top	Actividad de los procesos actualizada dinámicamente	procps
uptime	Ver cuanto tiempo lleva el sistema encendido	procps
ps	Información sobre los procesos en ejecución	procps
pstree	Árbol de procesos y sus hijos	pstree
mpstat	Uso de los procesadores	systat
sar	Recopila distintos tipos de información y la muestra	systat
iostat	Estadísticas E/S	systat
numastat	Información acerca de **NUMA** (Non-Uniform Memory Architecture)	numactl
strace	Información acerca de las llamadas al sistema de un proceso	strace

Cuadro 2: Monitorización de memoria

Herramienta	Propósito	Paquete
free	Resumen del uso de la memoria	procps
vmstat	Estadísticas del uso de la memoria y los bloques E/S	procps
pmap	Mapa de memoria de un proceso	procps

20.1.1. `/proc`

```
$ ls /proc/
1      14     18111  21624  25238  29679  3158  3792  743        dri          loadavg
10     149    18112  22     25265  3      316   3946  765        driver       locks
10833  15     18113  2225   25266  30092  318   3947  8          execdomains  meminfo
10834  15789  182    2258   25269  3015   3198  4034  8247       fb           misc
10835  15817  183    2263   26     30317  320   4121  9047       filesystems  modules
10836  15819  187    2265   2683   30318  3226  4223  acpi       fs           mounts
10839  15843  188    2272   27     3032   344   4821  asound     interrupts   mtrr
10840  16     19     23     2710   3033   345   4822  buddyinfo  iomem        net
10841  1693   1913   23655  2752   304    3541  4887  bus        ioports      pagetypeinfo
10842  17     1914   23656  2793   30429  3542  4889  cgroups    irq          partitions
113    1758   19661  23657  2841   3047   3734  4890  cmdline    kallsyms     sched_debug
12     1759   2      23662  2882   305    3743  490   consoles   kcore        self
12885  176    20     24     2904   306    3745  5217  cpuinfo    keys         slabinfo
13     178    20615  24163  29675  30655  3788  536   crypto     key-users    softirqs
13240  18     21187  24178  29676  30723  3789  6     devices    kmsg         stat
13777  180    21346  25     29677  3096   3790  7     diskstats  kpagecount   swaps
13795  181    21622  25218  29678  3104   3791  731   dma        kpageflags   sys
```

Algo que puede llamar la atención es que hay muchos directorios con números por nombre. Esto es debido a que cada proceso tiene un directorio en `/proc`,

Cuadro 3: Monitorización de E/S

Herramienta	Propósito	Paquete
sar	Recopila distintos tipos de información y la muestra	systat
iostat	Estadísticas E/S	systat
vmstat	Estadísticas del uso de la memoria y los bloques E/S	procps

Cuadro 4: Monitorización de red

Herramienta	Propósito	Paquete
netstat	Estadísticas de la red	netstat
iptraf	Muestra información de las interfaces de red	iptraf
tcpdump	Análisis detallado de los paquetes de red	tcpdump
wireshark	Análisis detallado del tráfico de la red	wireshark

esté en el estado que esté. Si se mira uno al azar:

```
$ ls -l /proc/18111/
total 0
dr-xr-xr-x 2 root drymer 0 may  1 20:40 attr
-rw-r--r-- 1 root root   0 may  1 20:40 autogroup
-r-------- 1 root root   0 may  1 20:40 auxv
-r--r--r-- 1 root root   0 abr 30 17:45 cgroup
--w------- 1 root root   0 may  1 20:40 clear_refs
-r--r--r-- 1 root root   0 abr 22 21:15 cmdline
-rw-r--r-- 1 root root   0 may  1 20:40 comm
-rw-r--r-- 1 root root   0 may  1 20:40 coredump_filter
-r--r--r-- 1 root root   0 may  1 20:40 cpuset
lrwxrwxrwx 1 root root   0 abr 22 21:39 cwd -> /var/www/gnu-social
-r-------- 1 root root   0 may  1 20:40 environ
lrwxrwxrwx 1 root root   0 abr 22 21:39 exe -> /usr/bin/sudo
dr-x------ 2 root root   0 abr 22 21:39 fd
dr-x------ 2 root root   0 abr 22 21:39 fdinfo
-r-------- 1 root root   0 abr 30 17:45 io
-r--r--r-- 1 root root   0 may  1 20:40 limits
-rw-r--r-- 1 root root   0 may  1 20:40 loginuid
-r--r--r-- 1 root root   0 abr 22 21:39 maps
-rw------- 1 root root   0 may  1 20:40 mem
-r--r--r-- 1 root root   0 may  1 20:40 mountinfo
-r--r--r-- 1 root root   0 may  1 20:40 mounts
-r-------- 1 root root   0 may  1 20:40 mountstats
dr-xr-xr-x 6 root drymer 0 may  1 20:40 net
dr-x--x--x 2 root root   0 may  1 20:40 ns
-rw-r--r-- 1 root root   0 may  1 20:40 oom_adj
-r--r--r-- 1 root root   0 may  1 20:40 oom_score
-rw-r--r-- 1 root root   0 may  1 20:40 oom_score_adj
-r--r--r-- 1 root root   0 may  1 20:40 pagemap
-r--r--r-- 1 root root   0 may  1 20:40 personality
lrwxrwxrwx 1 root root   0 abr 22 21:39 root -> /
-rw-r--r-- 1 root root   0 may  1 20:40 sched
-r--r--r-- 1 root root   0 may  1 20:40 sessionid
-r--r--r-- 1 root root   0 may  1 20:40 smaps
-r--r--r-- 1 root root   0 may  1 20:40 stack
-r--r--r-- 1 root root   0 abr 22 21:14 stat
-r--r--r-- 1 root root   0 abr 30 17:45 statm
-r--r--r-- 1 root root   0 abr 23 08:03 status
-r--r--r-- 1 root root   0 may  1 20:40 syscall
```

```
dr-xr-xr-x 3 root drymer 0 abr 30 17:45 task
-r--r--r-- 1 root root   0 may  1 20:40 wchan
```

Se puede ver que es una terminal que ha ejecutado `sudo su` y que se encuentra en un directorio llamado `/var/www/gnu-social`. Para ver el estado de el proceso:

```
$ cat /proc/18111/status
Name:   sudo
State:  S (sleeping)
Tgid:   18111
Pid:    18111
PPid:   21346
TracerPid:      0
Uid:    0       0       0       0
Gid:    1000    1000    1000    1000
FDSize: 256
Groups: 24 25 27 29 30 44 46 104 113 114 116 1000 1003
VmPeak:     5828 kB
VmSize:     5128 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      1500 kB
VmRSS:      1500 kB
VmData:      396 kB
VmStk:       136 kB
VmExe:       112 kB
VmLib:      2208 kB
VmPTE:        24 kB
VmSwap:        0 kB
Threads:        1
SigQ:   2/14069
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000037a07
CapInh: 0000000000000000
CapPrm: ffffffffffffffff
CapEff: ffffffffffffffff
CapBnd: ffffffffffffffff
Cpus_allowed:   3
Cpus_allowed_list:      0-1
Mems_allowed:   1
Mems_allowed_list:      0
voluntary_ctxt_switches:        5
nonvoluntary_ctxt_switches:     5
```

Hay otras entradas que proveen de información más general acerca del sistema. Por ejemplo, para ver las estadísticas de interrupciones de las CPU:

```
$ cat /proc/interrupts
   CPU0       CPU1
  0:  404527272       2401   IO-APIC-edge      timer
  1:       3726          0   IO-APIC-edge      i8042
  7:          1          0   IO-APIC-edge
  8:          1          0   IO-APIC-edge      rtc0
  9:       3169          0   IO-APIC-fasteoi   acpi
 12:      10877          0   IO-APIC-edge      i8042
 19:          1          0   IO-APIC-fasteoi   ehci_hcd:usb2
 20:          0          0   IO-APIC-fasteoi   ohci_hcd:usb4
 21:    4810857          4   IO-APIC-fasteoi   ehci_hcd:usb1
 22:          0          0   IO-APIC-fasteoi   ohci_hcd:usb3
 23:        290          0   IO-APIC-fasteoi   nouveau
 40:   82653702        280   PCI-MSI-edge      ahci
 41:   46531564        236   PCI-MSI-edge      eth0
NMI:     295959     295978   Non-maskable interrupts
LOC:  109224869  274549064   Local timer interrupts
SPU:          0          0   Spurious interrupts
PMI:     295959     295978   Performance monitoring interrupts
IWI:          0          0   IRQ work interrupts
RES:   63908426  114160726   Rescheduling interrupts
CAL:       1142        822   Function call interrupts
TLB:     100021      95202   TLB shootdowns
TRM:          0          0   Thermal event interrupts
THR:          0          0   Threshold APIC interrupts
MCE:          0          0   Machine check exceptions
MCP:      17276      17276   Machine check polls
ERR:          1
MIS:          0
```

Antes se mencionó que algunos parámetros se podían ajustar. Esto se haría en el directorio `/proc/sys`. En este directorio se encuentran los siguientes subdirectorios:

- `abi/`: Contiene información binaria de programas
- `debug/`: Parámetros de depuración
- `dev/`: Parámetros de los dispositivos
- `fs/`: Parámetros de los sistemas de archivos
- `kernel/`: Parámetros del kernel
- `net/`: Parámetros de la red
- `vm/`: Parámetros de la memoria virtual

Dado que los archivos de los anteriores directorios son sencillos, su edición también lo es, si se quiere. Se puede editar simplemente volcando el contenido del comando `echo`. Por ejemplo, para cambiar el número máximo de los hilos del sistema:

```
$ cat /proc/sys/kernel/max-threads
129428
$ su -c "echo 100000 > /proc/sys/kernel/max-threads"
$ cat /proc/sys/kernel/max-threads
100000
```

20.1.2. `/sys`

Este usa el sistema de ficheros virtual sysfs. Aunque lo normal en los kernels modernos es que se monte bajo `/sys`, no es un requisito necesario para su funcionamiento. En general se usa para gestionar los dispositivos hardware, pero incluye información que no está relacionado con ello. Los archivos en **sysfs**, cómo ya se ha comentado, suelen tener archivos de configuración de una linea para su fácil edición sin editores de texto, pero tampoco es un requisito. Dado que según el kernel, hay distinta información en ese directorio, no tiene mucho sentido mostrar lo que hay en el, ya que es variable.

20.2. `sar`

Esta es una herramienta de monitorización genérico. Su nombre significa Systems Activity Reporter (reportador de actividades del sistema). Consta de dos partes, el backend, que es **sadc**, que recopila información del sistema, y la herramienta de la terminal, la propia ejecución de `sar`, que nos muestra esa información formateada. **sadc** guarda toda la información recopilada en `/var/log/sa` de manera diaria, aunque ambas pueden ser modificadas. Se puede forzar la recolección de datos, pero al ser instalada configura `crontab` para que lo haga, por lo que en principio no es necesario.

Un ejemplo de un reporte por defecto:

```
$ su -c "sar 3 3"
Linux 3.2.0-4-686-pae (debian-bittorrent)       04/05/16        _i686_  (2 CPU)

17:02:52        CPU     %user     %nice   %system   %iowait    %steal     %idle
17:02:55        all      9,90      0,00      3,92     39,08      0,00     47,10
17:02:58        all     10,30      0,00      3,89     40,71      0,00     45,10
17:03:01        all      7,37      0,00      4,19     23,62      0,00     64,82
Media:          all      9,18      0,00      4,00     34,42      0,00     52,39
```

Para ver sus opciones con más profundidad, hay que revisar su manual. No tiene sentido copiar el **man** aquí.

Existe una herramienta para generar gráficos de sar, llamado `ksar`. Se puede ver el proyecto aquí. Está hecho en java, así que igual no vale la pena.

21. Monitorización de procesos

Algunas de las herramientas más usadas para este tema son, cómo ya se mencionó en el capítulo anterior, `ps`, `pstree` y `top`. Para revisar estas herramientas, ver la tabla nº 1.

21.1. ps

ps es una de las herramientas que más se usa cómo sysadmin. Es una manera sencilla de que un proceso está ejecutándose.

ps, cómo otras herramientas, ha sido usada en muchos sistemas operativos, tales cómo **unix**, **BSD** y **GNU**. Por este motivo acepta los parámetros de modos distintos, ajustándose a estos.

- **Unix**: Deben ser precedidas por un guión simple
- **BSD**: NO deben ser precedidas por un guión simple y pueden agruparse
- **GNU**: Las opciones largas son precedidas por dos guiones simples.

Por lo tanto, en GNU/Linux, tenemos que `(ps -a -u -x)` == `(ps -aux)` == `(ps aux)`.

Un ejemplo de la ejecución de esta orden es el siguiente:

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2300   652 ?        Ss   mar02   2:07 init [2]
root         2  0.0  0.0      0     0 ?        S    mar02   0:03 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    mar02  20:09 [ksoftirqd/0]
root         6  0.0  0.0      0     0 ?        S    mar02   0:41 [migration/0]
root         7  0.0  0.0      0     0 ?        S    mar02   1:03 [watchdog/0]
root         8  0.0  0.0      0     0 ?        S    mar02   0:43 [migration/1]
root        10  0.0  0.0      0     0 ?        S    mar02  11:19 [ksoftirqd/1]
root        12  0.0  0.0      0     0 ?        S    mar02   0:52 [watchdog/1]
root        13  0.0  0.0      0     0 ?        S<   mar02   0:00 [cpuset]
root        14  0.0  0.0      0     0 ?        S<   mar02   0:00 [khelper]
root        15  0.0  0.0      0     0 ?        S    mar02   0:00 [kdevtmpfs]
root        16  0.0  0.0      0     0 ?        S<   mar02   0:00 [netns]
root        17  0.0  0.0      0     0 ?        S    mar02   0:44 [sync_supers]
root        18  0.0  0.0      0     0 ?        S    mar02   0:00 [bdi-default]
root        19  0.0  0.0      0     0 ?        S<   mar02   0:00 [kintegrityd]
root        20  0.0  0.0      0     0 ?        S<   mar02   0:00 [kblockd]
root        22  0.0  0.0      0     0 ?        S    mar02   0:06 [khungtaskd]
root        23  0.0  0.0      0     0 ?        S    mar02   1:30 [kswapd0]
root        24  0.0  0.0      0     0 ?        SN   mar02   0:00 [ksmd]
root        25  0.0  0.0      0     0 ?        SN   mar02   0:00 [khugepaged]
root        26  0.0  0.0      0     0 ?        S    mar02   0:00 [fsnotify_mark]
root        27  0.0  0.0      0     0 ?        S<   mar02   0:00 [crypto]
root       113  0.0  0.0      0     0 ?        S    mar02   0:00 [khubd]
root       149  0.0  0.0      0     0 ?        S<   mar02   0:00 [ata_sff]
root       176  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_0]
root       178  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_1]
root       180  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_2]
root       181  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_3]
root       182  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_4]
root       183  0.0  0.0      0     0 ?        S    mar02   0:00 [scsi_eh_5]
root       187  0.0  0.0      0     0 ?        S    mar02   0:00 [kworker/u:4]
root       188  0.0  0.0      0     0 ?        S    mar02   0:00 [kworker/u:5]
```

```
root       304  0.0  0.0      0     0 ?        S<   mar02   0:00 [kdmflush]
root       305  0.0  0.0      0     0 ?        S<   mar02   0:00 [kcryptd_io]
root       306  0.0  0.0      0     0 ?        S<   mar02   0:00 [kcryptd]
root       316  0.0  0.0      0     0 ?        S<   mar02   0:00 [kdmflush]
root       318  0.0  0.0      0     0 ?        S<   mar02   0:00 [kdmflush]
root       320  0.0  0.0      0     0 ?        S<   mar02   0:00 [kdmflush]
root       344  0.0  0.0      0     0 ?        S    mar02  62:29 [jbd2/dm-1-8]
root       345  0.0  0.0      0     0 ?        S<   mar02   0:00 [ext4-dio-unwrit]
root       490  0.0  0.0   2896   940 ?        Ss   mar02   0:00 udevd --daemon
116        536  0.5  1.7  36032 32508 ?        S    abr17 239:48 /usr/bin/tor --defaults-t
root       731  0.0  0.0      0     0 ?        S<   mar02   0:00 [kpsmoused]
root       743  0.0  0.0      0     0 ?        S<   mar02   0:00 [cfg80211]
root       765  0.0  0.0      0     0 ?        S<   mar02   0:00 [ttm_swap]
```

- **VSZ**: el tamaño de la memoria virtual en KB
- **RSS**: tamaño de la memoria residente, es decir, la memoria física que está usando una tarea en KB
- **STAT**: el estado de un proceso, que puede ser **S** si duerme o **R** si está en ejecución. También pueden tener un carácter extra:
  - **N**: prioridad baja (Nice)
  - **L**: páginas bloqueadas en memoria
  - **s**: es del dueño de la sesión
  - **l**: es multihilo
  - +: se ejecuta en primer plano

21.2. `pstree`

Esta herramienta muestra lo mismo que `ps` pero en otro formato, el forma de arbol, concretamente. Se ve del siguiente modo:

```
$ pstree -aAp
init,1
  |-bitlbee,26636 -p 6667 -P /var/run/bitlbee.pid -F
  |   `-bitlbee,28591 -p 6667 -P /var/run/bitlbee.pid -F
  |-mcabber,25888 -f .mcabber/daemons
  |-netdata,25087
  |   |-apps.plugin,11499 1
  |   |-charts.d.plugin,11768 /usr/libexec/netdata/plugins.d/charts.d.plugin 1
  |   |   `-sleep,21831 0.2
  |   |-tc-qos-helper.s,13764 /usr/libexec/netdata/plugins.d/tc-qos-helper.sh 1
  |   |   `-sleep,21829 0.980
  |   |-{netdata},25088
  |   |-{netdata},25090
  |   |-{netdata},25091
  |   |-{netdata},25092
  |   |-{netdata},25093
  |   |-{netdata},25094
```

```
|   |-{netdata},25095
|   `-{netdata},25097
|-bash,25410
|   `-pstree,21832 -aAp
|-bash,8128
|   `-ssh,1644 drymer@192.168.1.232
|-netdata,25087
|   |-apps.plugin,11499 1
|   |-charts.d.plugin,11768 /usr/libexec/netdata/plugins.d/charts.d.plugin 1
|   |   `-sleep,21831 0.2
|   |-tc-qos-helper.s,13764 /usr/libexec/netdata/plugins.d/tc-qos-helper.sh 1
|   |   `-sleep,21829 0.980
|   |-{netdata},25088
|   |-{netdata},25090
|   |-{netdata},25091
|   |-{netdata},25092
|   |-{netdata},25093
|   |-{netdata},25094
|   |-{netdata},25095
|   `-{netdata},25097
|-nginx,29675
|   |-nginx,29676
|   |-nginx,29677
|   |-nginx,29678
|   `-nginx,29679
|-php5-fpm,30092
|   |-php5-fpm,6878
|   |-php5-fpm,15684
|   `-php5-fpm,15708
|-sshd,19661
|   |-sshd,4976
|   |   `-sshd,4995
|   |       `-bash,4996
|   |           `-tmux,12965 attach-session
|   `-sshd,29030
```

Cómo siempre, para ver más opciones sobre esta herramienta, RTFM.

21.3. `top`

Esta herramienta muestra la carga que tiene cada proceso, por defecto en orden descendente. Es mejor aún `htop`, que muestra lo mismo que `top` con algunos añadidos y colores. Se ve tal que así:

```
top - 19:37:08 up 77 days, 22:30,  1 user,  load average: 1,12, 1,03, 0,80
Tasks: 142 total,   1 running, 141 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,8 us,  1,5 sy,  0,4 ni, 90,0 id,  4,2 wa,  0,0 hi,  0,1 si,  0,0 st
KiB Mem:   1813160 total,  1670244 used,   142916 free,   207788 buffers
KiB Swap:  3665916 total,    69512 used,  3596404 free,   776988 cached
```

```
  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 4223 drymer    20   0 68368  34m 5324 S  51,5  1,9   2536:54 weechat
27276 drymer    20   0  4532 1192  896 R  17,2  0,1   0:00.04 top
 3946 drymer    20   0 19088  15m 1232 S   5,7  0,9 493:58.46 tmux
11768 netdata   39  19  5400 2328 1280 S   5,7  0,1   0:28.89 charts.d.plugin
    1 root      20   0  2300  652  604 S   0,0  0,0   2:09.63 init
    2 root      20   0     0    0    0 S   0,0  0,0   0:03.79 kthreadd
    3 root      20   0     0    0    0 S   0,0  0,0  21:09.88 ksoftirqd/0
    6 root      rt   0     0    0    0 S   0,0  0,0   0:50.06 migration/0
    7 root      rt   0     0    0    0 S   0,0  0,0   1:04.43 watchdog/0
    8 root      rt   0     0    0    0 S   0,0  0,0   0:52.11 migration/1
   10 root      20   0     0    0    0 S   0,0  0,0  12:03.69 ksoftirqd/1
   12 root      rt   0     0    0    0 S   0,0  0,0   0:53.06 watchdog/1
   13 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 cpuset
   14 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 khelper
   15 root      20   0     0    0    0 S   0,0  0,0   0:00.00 kdevtmpfs
   16 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 netns
   17 root      20   0     0    0    0 S   0,0  0,0   0:45.40 sync_supers
   18 root      20   0     0    0    0 S   0,0  0,0   0:00.98 bdi-default
   19 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 kintegrityd
   20 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 kblockd
   22 root      20   0     0    0    0 S   0,0  0,0   0:06.29 khungtaskd
   23 root      20   0     0    0    0 S   0,0  0,0   1:30.25 kswapd0
   24 root      25   5     0    0    0 S   0,0  0,0   0:00.00 ksmd
   25 root      39  19     0    0    0 S   0,0  0,0   0:00.00 khugepaged
   26 root      20   0     0    0    0 S   0,0  0,0   0:00.00 fsnotify_mark
   27 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 crypto
  113 root      20   0     0    0    0 S   0,0  0,0   0:00.10 khubd
  149 root       0 -20     0    0    0 S   0,0  0,0   0:00.00 ata_sff
  176 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_0
  178 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_1
  180 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_2
  181 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_3
  182 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_4
  183 root      20   0     0    0    0 S   0,0  0,0   0:00.00 scsi_eh_5
  187 root      20   0     0    0    0 S   0,0  0,0   0:00.00 kworker/u:4
```

`top` tiene un menú interactivo, es decir, no se le pasan opciones cómo tal pero en el momento de ejecución. Por ejemplo, al presionar la tecla del número **1**, se muestra la carga de cada CPU por separado. Con la **h** veremos las opciones que se pueden usar. Cómo alternativas, a parte de `htop`, también está `ntop` y `atop`.

22. Monitorización y ajuste de E/S

El E/S es muy importante para el rendimiento del ordenador, ya que constantemente se está escribiendo. Si las escrituras o lecturas son lentas, el sistema se resiente. Se considera que un sistema tiene un cuello de botella en la E/S cuando la CPU está esperando sin hacer nada a que una operación E/S termi-

ne. Hay que ir con cuidado por que a veces lo que parece ser poca memoria es en realidad un problema de E/S.

22.1. Herramientas

22.1.1. `iostat`

Es la más importante en este tema. Puede generar reportes con mucha información, permitiendo controlar exactamente que se muestra. Por defecto se ve así:

```
$ iostat
Linux 3.2.0-4-686-pae (debian-bittorrent)       19/05/16        _i686_  (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
   3,79    0,36    1,61    4,21    0,00   90,03

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              18,05        75,34       103,24  507462194  695404336
dm-0             18,23        75,34       103,24  507454710  695371440
dm-1             16,28        74,98       102,98  505008041  693611596
dm-2              0,00         0,00         0,01      12524      78468
dm-3              0,04         0,36         0,25    2433561    1681376
```

La información que muestra es un breve resumen del uso de las CPU, los **tps** (transacciones E/S por segundo). La información está partida en función de las particiones del disco que se tengan. Unas cuantas opciones útiles que se le pueden pasar son:

- `-k`: muestra el valor en kilobytes en vez de bloques
- `-m`: muestra el valor en megabytes en vez de bloques
- `-N`: por nombre de dispositivo
- `-x`: extendido Un valor a tener en cuenta de esta opción es **%util**, el último. Cuanto más cerca esté del 100 %, más probable será que haya un embotellamiento de E/S.

22.1.2. `iotop`

Es similar a la ya mencionada `top`, en cuanto a que muestra en tiempo real la escritura.

22.1.3. `ionice`

Similar a `nice`, permite ajustar la prioridad E/S de un proceso dado y la planificación general. Algunos parámetros útiles:

- `-p`: Se le puede pasar un pid y modificar su prioridad al vuelo.
- `-c`: Se le pasa la planificación. Los valores que se le pueden pasar son:

- `1`: idle, es decir, no tiene acceso a E/S a menos que no haya ningún proceso que lo necesite.
- `2`: Best Effort, es decir, funciona con el método Round Robin (este va rotando el derecho a E/S según llegan los procesos que lo requieren). Es el que se usa por defecto.
- `3`: Real time, es decir, echa a cualquier proceso que esté usando la E/S.

- `-n`: Este sólo se les puede pasar a los valores **Best Effort** y **Real Time** acompañado de un número del **0** al **7** (siendo **0** el mayor) para concretar su prioridad.

Nota: esta herramienta sólo se puede usar si se usa el planificador **CFQ**, que se menciona en el siguiente capitulo.

23. Planificación de E/S

Esto es importante ya que el sistema depende de una planificación para minimizar el acceso a hardware para evitar un desgaste innecesario de hardware, asegurar la integridad de los datos, garantizar el acceso a las aplicaciones que necesitan E/S y priorizar las tareas importantes.

El planificador es una interfaz entre la capa de los bloques y los controladores del dispositivo físico. Requisitos de un planificador:

- Tiempo de acceso a hardware al mínimo
- Manejar la cola de peticiones
- Evitar la fragmentación en medida de lo posible, es decir, usar bloques contiguos, ya que estos aceleran el acceso
- La cola debe ser manejada con celeridad
- Priorizar las lecturas sobre las escrituras, ya que las primeras necesitan que los procesos terminen y la segunda no
- Repartir el ancho de banda de manera justa,

No hay un planificador mejor que otro en ámbitos generales, cada uno debe ser usado según las circunstancias. No es lo mismo un servidor con una base de datos monstruosa que un ordenador personal.

Uno de los siguientes debe estar compilado en el kernel:

- Completely Fair Queueing (CFQ)
- Deadline scheduling
- noop

Habitualmente se usa alguno de los dos primeros.

23.1. Dispositivos SSD

Estos son relativamente nuevos, y traen muchas diferencias respecto al uso de discos duros físicos. En estos no se prioriza según necesidad sino de manera "justa", ya que al ser discos SSD tienen x lecturas y escrituras limitados. Por ello, usan la **nivelación por desgaste**.

Se puede confirmar si un disco duro es SSD ejecutando `cat /sys/block/sda/queue/rotational` y devolviendo esta **1**.

23.2. Cambios del planificador en caliente

Todos los planificadores tienen parámetros que se pueden modificar para ajustar el comportamiento. Cómo siempre, estos están en sys.

23.3. CFQ

Complete Fair Queue, que viene a ser Colar Completamente Justa. Su intención es la de repartir el ancho de banda de igual manera entre todos los procesos. Se usa el algoritmo Round Robin en todas las colas, las cuales usan **FIFO** (First In, First Out).

23.3.1. Parámetros ajustables

HZ son las unidades de tiempo por segundo (jiffies), que es usada por el kernel cómo medida de tiempo.

- **quantum**: longitud máxima de la cola en una ronda de servicios (por defecto, 4)
- **queued**: solicitud mínima de asignación por cola (por defecto, 0)
- $\textbf{fifo}_{\textbf{expiresync}}$: timeout FIFO para solicitudes síncronas (por defecto, HZ / 2)
- $\textbf{fifo}_{\textbf{expireasync}}$: timeout FIFO para solicitudes asíncronas (por defecto, 5 * HZ)
- $\textbf{fifo}_{\textbf{batchexpire}}$: velocidad a la que expira FIFO (por defecto, HZ / 8)
- $\textbf{back}_{\textbf{seekmax}}$: búsqueda hacia atrás en KB (por defecto, 16kb)
- $\textbf{back}_{\textbf{seekpenalty}}$: penalización por una búsqueda hacia atrás (por defecto, 2)

23.4. Planificador Deadline

Su objetivo es reordenar las solicitudes para mejorar el rendimiento en general y prevenir las latencias. Por cada solicitud se asocia una **fecha límite** (deadline). Además, las peticiones de lectura tienen una mayor prioridad que las de escritura, ya que son más rápidas de realizar. Con este método, se mantienen cinco colas distintas:

- Dos listas ordenadas por bloque de inicio, escritura y lectura.

- Dos listas ordenadas por tiempo **FIFO**, escritura y lectura.
- La cola de envío al controlador del dispositivo.

23.4.1. Parámetros ajustables

- $\textbf{read}_{\textbf{expire}}$: el tiempo en el que se garantiza una solicitud de lectura (por defecto, HZ / 2)
- $\textbf{write}_{\textbf{expire}}$: el tiempo en el que se garantiza una solicitud de lectura (por defecto, 5 * HZ)
- $\textbf{writes}_{\textbf{starved}}$: a cuántas solicitudes de lectura se debería dar preferencia por sobre las de escritura (por defecto, 2)
- $\textbf{fifo}_{\textbf{batch}}$: cuántas solicitudes deberían moverse desde la cola de envío cuando han expirado los plazos (por defecto, 16)
- $\textbf{front}_{\textbf{merges}}$: si se habilita, la unión se hace únicamente hacia delante, lo cual puede acelerar las colas si se sabe que no se necesitará unir hacia atrás

24. Memoria: monitorización y ajustes

Ajustar el uso de la memoria es un proceso complejo, ya que está íntimamente relacionado con el rendimiento E/S. La RAM se suele usar para guardar en caché el contenido de los archivos en disco. Dada su complejidad, es recomendable modificar valores de uno en uno usando los archivos de `/proc/sys/vm/` analizando el comportamiento con cada cambio y una vez hecho, ya se podrán dejar fijos los valores. Las tareas suelen ser:

- **Controlar los parámetros del flujo**: cuántas páginas sucias se permiten en el swap y cada cuando se vacía
- **Controlar el comportamiento del swap**: cuantas páginas que reflejen el contenido de un archivo pueden quedarse en la memoria y cuantas deben ser escritas por que no hay copias de este
- **Controlar cuánto desborde de memoria se permite (buffer overflow)**: hay programas que no necesitan la memoria que piden, por lo que sirve para controlar errores

24.1. `/proc/sys/vm`

Este directorio contiene distintos archivos para controlar la **Memoria Virtual**. No se puede concretar que archivos habrá exactamente ya que varia según la versión del kernel. Hay que recordar que estos valores sólo deben modificarse para hacer pruebas, ya que los cambios no permanecen al reiniciar. Para que lo hagan, hay que modificar el archivo `/etc/sysctl.conf`. Los valores que se puede modificar se pueden encontrar en su propia documentación, en el archivo llamado `Documentation/sysctl/vm.txt` que se encuentra en la raíz del código del kernel. **Apunte:** si vuestra distribución compila los kernels o lo habéis compilado vosotros a mano, con hacer un `locate vm.txt` lo encontraréis fácilmente.

24.2. Herramientas

Hay tres herramientas cómo tal que se suelen usar para controlar la memoria. Estas son `free`, `vmstat` y `pmap`. Las tres están en el paquete llamado `procps`.

24.2.1. `vmstat`

Muestra información acerca de la memoria, paginación, E/S, actividad del procesador y procesos. Se ve tal que así:

```
vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 1  1  67616 154120 311392 678792    0    0     3     2    2    0  5  2 88  4
```

Esto muestra los promedios desde el último reinicio. Los campos son los siguientes:

Campo	Subcampo	Significado
Procesos (procs)	r	Nº de procesos a la espera de ingresar en el planificador
Procesos (procs)	b	Nº de procesos durmiendo
Memoria (memory)	swp	Memoria usada (KB)
Memoria (memory)	free	Memoria libre (KB)
Memoria (memory)	buff	Memoria usada como buffer (KB)
Memoria (memory)	cache	Memoria en caché (KB)
Área de intercambio (swap)	si	Swap desde el disco (KB)
Área de intercambio (swap)	so	Swap hacia el disco (KB)
E/S (I/O)	bi	Bloques recibidos desde el dispositivo (bloques/segundo)
E/S (I/O)	bo	Bloques enviados hacia el dispositivo (bloques/segundo)
Sistema (system)	in	Interrupciones/segundo
Sistema (system)	cs	Cambios de contexto/segundo
CPU	us	Tiempo de CPU ejecutando código de usuario (porcentaje)
CPU	sy	Tiempo de CPU ejecutando código del kernel (porcentaje)
CPU	id	Tiempo ocioso de la CPU de (porcentaje)
CPU	wa	Tiempo de espera por E/S (porcentaje)

Si se pasa el valor `-S m` se usaran MB en vez de KB. Para más información, RTFM. Para que quede constancia, en `/proc/meminfo` se puede ver un resumen largo de las estadísticas de memoria.

24.3. OOM Killer

Este es un error común, se produce cuando la memoria se agota. OOM significa **Out Of Memory**. El OOM-killer decide que procesos deben matarse para liberar memoria más eficientemente. Este no es un proceso sencillo, aunque pueda parecerlo. Alguien explicó por que no lo es usando un símil con los aviones y su combustible, que puede verse aquí. Sin entrar en más detalles, el algoritmo para decidir de que manera matar procesos cuando se produce el OOM debe ser heurístico (según la wikipedia, en este caso la heurística es un método basado en la experiencia que puede utilizarse como ayuda para resolver problemas de diseño). De este modo, se calcula un valor llamado **maldad**, que se puede leer en

el fichero `/proc/$PID/oom_score`. Cuando se produzca un OOM, OOM-killer escogerá que proceso matar según ese valor, cuánto más alto más posibilidades de que sea ese el elegido.

25. Sistemas de gestión de paquetes de bajo nivel

Estos permiten a la administradora de sistemas de sistemas automatizar la instalación, actualización, configuración y eliminación de programas. Lo que se intenta conseguir estos sistemas es:

- Reunir todos los archivos asociados al software concreto en un sólo archivo.
- Permitir que sea más fácil instalar
- Verificar la integridad del archivo mediante una base de datos interna
- Autenticar el origen del archivo
- Facilitar actualizaciones
- Agrupar los paquetes por características lógicas
- Administrar dependencias de los paquetes

En los paquetes se suele incluir además distintos metadatos, cómo los checksums, números de versión, dependencias, mantenedor del paquete, etc. Todo esto almacenado en una BBDD interna que se puede consultar en cualquier momento.

25.1. Tipos de paquetes

Pueden ser distintos:

- Paquetes binarios con archivos listos para ser implementados. Dependen de la arquitectura. Debian y derivados usan este método.
- Paquetes de código fuente que se compilan en la máquina en la que se instalan. No depende de la arquitectura (aunque debe soportar en la que se instala). Slackware y derivados usan este método.
- Paquetes independientes de la arquitectura que se ejecutan bajo intérpretes. No depende del sistema operativo. Con python y lisp se pueden crear paquetes de este tipo.

En los sistemas de 64 bits se puede usar lo que se llama **multilib**, que permite tener tanto binarios de 64 bits cómo de 32 bits.

Gestores de paquetes:

- `RPM`: Red Hat y derivados
- `dpkg`: Debian y derivados
- `portage/emerge`: Gentoo y derivados
- `pacman`: Arch y derivados
- `slackpkg`: Slackware

En este curso se verá sobretodo `dpkg` y `RPM`.

25.2. Niveles de los gestores

Hay dos, lógicos, a bajo nivel y a alto nivel. En el primer tipo entran `RPM` y `dpkg`, en el segundo entran `yum` y `apt/apt-get/aptitude`. Las de bajo nivel simplemente instalan o desinstalan, no sirven para gestionar dependencias. Las segundas, en cambio, si permiten la gestión de dependencias automática.

25.3. Fuentes de los paquetes

Generalmente, las distribuciones se encargan de proporcionar los repositorios de paquetes. Si se habla de distribuciones grandes, cómo debian, RHEL, slackware o suse, se tiene la seguridad de que estos están bien securizados contra ataques y que tienen una calidad mínima. En el momento en el que se empiezan a usar repositorios de terceros, se corre el riesgo de que estos no sean de tanta calidad. Son un buen método para tener programas que de otra forma no tendríamos, pero en el ámbito de la administración de sistemas profesional, está desaconsejado si estos no tienen un mínimo de credibilidad.

25.4. Creación de paquetes de software

Construir tu propio paquete puede ser necesario si el programa que quieres usar no está en tu distribución. En general es más fácil compilar e instalar el programa sin más, pero hay escenarios en el que crear un `.deb` o un `.rpm` puede facilitar la vida, cómo el supuesto de tener que instalar el programa en muchos ordenadores, o en pocos si el tiempo de compilación del programa es muy largo.

La creación de paquetes, además, permite automatizar ciertas tareas. Algunas de estas son la creación de enlaces simbólicos, creación de directorios y configuración de permisos, por ejemplo.

25.5. Sistemas de control de versiones

Los proyectos de software libre se pueden ir volviendo complejos con el tiempo. Es una manera muy sencilla de ver los cambios que han habido con el tiempo, tirar hacia atrás un cambio que haya hecho la desarrolladora que ha hecho que el programa no funcione y muchas otras cosas. Los sistemas más usados de VCS son `mercurial`, `subversion` y sobretodo `git` en los últimos años.

`git` lo creó Linus Torvalds, el creador del **kernel** linux, debido precisamente a que necesitaba una manera eficaz de rastrear los cambios en el proyecto.

25.5.1. Cómo funciona `git`

Su estructura es la siguiente:

- Base de datos de objetos
  - Blobs: trozos de datos binarios
  - Trees: conjuntos de blobs con los nombres de archivos y sus atributos
  - Commits: cambios en el código
- Caché de directorios

25.6. `rpm`

`rpm` quiere decir **RedHat Packet Manager** y fue desarrollado por Red Hat (no shit Sherlock). `rpm`, a menos que se concrete, no buscará nunca los paquetes en internet. Por lo general, seremos las administradoras quien nos encargaremos de crear o descargar el paquete y luego instalarlo con la herramienta.

Los archivos `rpm` suelen depender de la distribución, de hecho instalar un paquete de una distribución diferente de la cual ha sido pensada es casi imposible (**Apunte:** esto es un poco exagerado, yo he usado paquetes de fedora, centos o rosa en slackware sin ningún problema).

25.6.1. Ventajas de `rpm`

- Ver a que paquete permanece un archivo del sistema
- Ver la versión de un paquete
- Instalar y desinstalar limpiamente
- Verificar la correcta instalación de un paquete
- Distinguir los archivos de documentación y elegir no instalarlo si se quiere
- Usar `ftp` o `http` para instalar desde internet

25.6.2. Nombres de los archivos

Siguen un estándar, que se puede consultar en su web. El formato de un paquete binario seria:

- `<nombre>-<version>-<lanzamiento>.<distro>.<arquitectura>.rpm`

Un ejemplo: `sed-4.2.1-10.el6.x84_64.rpm`.
El formato de un paquete de código fuente:

- `<nombre>-<version>-<lanzamiento>.<distro>.src.rpm`

Un ejemplo: `sed-4.2.1-10.el6.src.rpm`.

25.6.3. Directorio de la base de datos

Por defecto es `/var/lib/rpm/`. Está en formato **Berkeley DB**. Nunca debería modificarse a mano, sólo con `rpm`. Un par de parámetros útiles son `--dbpath`, que permite concretar un directorio distinto de la BBDD. Esto podría usarse para examinar la BBDD de otro sistema. También es útil `--rebuilddb`, que reconstruye la base de datos en función de los paquetes instalados, por si hemos quitado algo a mano y se ha roto algo.

25.6.4. Archivo rc

Muchos programas en **GNU/Linux** tienen archivos de configuración que tiene un nombre del tipo `<nombreDelProgramarc>` y `rpm` es uno de ellos. Siguiendo el estándar xdg, se puede poner tanto en `/usr/lib/rpm/rpmrc`, `/etc/rpmrc` y `~/.rmprc`.

25.6.5. Consultas

Las consultas se realizan usando el parámetro `-q` junto a otros:

- `rpm -q bash`: versión de `bash` instalada.
- `rpm -qf bash`: nombre del paquete que instaló `bash`.
- `rpm -ql bash`: archivos que instaló `bash`.
- `rpm -qi bash`: muestra información de `bash`.
- `rpm -qa`: lista todos los paquetes instalados.
- `rpm -qp --requires foo-1.0-1x86.rpm`: muestra lista de dependencias.
- `rpm -Va foo-1.0-1x86.rpm`: consultar la consistencia de un paquete respecto a la BBDD.

25.6.6. Acciones típicas

1. Instalar Mediante `-i`. Con `-h` se muestra el progreso.

   ```
   su -c "rpm -ivh foo-1.0-1x86.rpm"
   ```

2. Desinstalar Mediante `-e`.

   ```
   su -c "rpm -e foo-1.0-1x86.rpm"
   ```

3. Actualizar un paquete Mediante `-U`. Hay que tener en cuenta que si el paquete no estuviese instalado, se instalará.

   ```
   su -c "rpm -Uvh foo-1.0-1x86.rpm"
   ```

4. Refrescar paquetes Se le hace un **checu checu** mediante `-F`.

   ```
   su -c "rpm -Fvh *.rpm"
   ```

25.6.7. `rpm2cpio`

Esta herramienta sirve para ver el contenido de un paquete `rpm` sin que haya que instalarlo.

```
rpm2cpio foobar.rpm | cpio --extract --make-directories
```

25.7. `dpkg`

Esta herramienta la usan todas las distribuciones derivadas de Debian. Con `rpm`, no está diseñada para conectarse a Internet para descargar paquetes, sino que sólo instala y desinstala. Esta herramienta gestiona los programas formateados en `.deb`. Si el paquete que se quiere instalar necesita una dependencia no instalada, devolverá un error.

25.7.1. Nombres de los archivos

Siguen un estándar. El formato de un paquete binario seria:

- `<nombre>_<version>-<número de revisión>_<arquitectura>.deb`

Un ejemplo: `sed_4.2.1-1_amd64.rpm`.

25.7.2. Paquetes de código fuente

Consiste de tres partes, el código fuente cómo tal en un archivo `.tar.gz`, un archivo de descripción (`.dsc`) con el nombre del paquete y metadatos, y por último un segundo archivo `.tar.gz` con los parches que han creado los que mantienen el paquete para adaptarlos a la distribución.

```
apt-get source weechat
ls -ld weechat*
drwxr-xr-x 9 drymer drymer    4096 jun  5 13:00 weechat-0.3.8
-rw-r--r-- 1 drymer drymer   16331 dic 17  2012 weechat_0.3.8-1+deb7u1.debian.tar.gz
-rw-r--r-- 1 drymer drymer    2424 jun  5 13:00 weechat_0.3.8-1+deb7u1.dsc
-rw-r--r-- 1 drymer drymer 2488165 jun  3  2012 weechat_0.3.8.orig.tar.bz2
```

25.7.3. Consultas típicas

1. Listar todos los paquetes instalados

   ```
   dpkg -l
   ```

2. Listar los archivos instalados por un paquete

   ```
   dpkg -L wget
   ```

3. Mostrar información de un paquete instalado

   ```
   dpkg -p sshfs
   ```

4. Mostrar información de un paquete

   ```
   dpkg -I sshfs_2.4-1_i386.deb
   ```

5. Listar archivos contenidos de un paquete

   ```
   dpkg -c sshfs_2.4-1_i386.deb
   ```

6. Mostrar a que paquete pertenece un archivo

   ```
   dpkg -S /etc/nginx/nginx.conf
   ```

7. Verificar la integridad de un paquete instalado

   ```
   dpkg -V wget
   ```

8. Instalar

```
dpkg -i sshfs_2.4-1_i386.deb
```

9. Desinstalar

```
dpkg -r sshfs
```

10. Desinstalar y purgar

```
dpkg -P sshfs
```

26. Sistemas de gestión de paquetes de alto nivel

Los sistemas de gestores de paquetes de alto nivel trabajan con bases de datos de programas disponibles y gestionan las dependencias, a diferencia de sus homólogos de bajo nivel. Permiten lo siguiente:

- Usar repositorios tanto locales cómo remotos
- Automatizar la instalación o desinstalación de paquetes
- Resolver dependencias automaticametente
- Ahorrar tiempo al no tener que buscar nombres de paquetes ni sus dependencias

Los repositorios suelen ser proporcionados por las propias distribuciones, aunque también hay proveedores de software independientes.

En este capitulo hablaremos de tres instaladores de paquetes de alto nivel, `apt`, `zypper` y `yum`.

26.1. `yum`

Esta es una herramienta de alto nivel, a diferencia de su homólogo `rpm`. Realmente es una interfaz para este, provee un frontend.

26.1.1. Archivos de repositorio

Están en `/etc/yum.repos.d/` con una extensión `.repo` y tienen un formato cómo el siguiente:

```
[nombre-del-repo]
name=Descripción
baseurl=http://ejemplo.com
enabled=1
gpgcheck=1
```

26.1.2. Consultas típicas

1. Buscar un paquete con alguna palabra clave

   ```
   yum search ssh
   ```

2. Mostrar información sobre un paquete

   ```
   yum search openssh-server
   ```

3. Mostrar los paquetes instalados, por actualizar o por instalar

   ```
   yum list [installed | updates | available]
   ```

4. Mostrar paquetes que proveen de cierto archivo

   ```
   yum provides /etc/nginx/nginx.conf
   ```

5. Verificación de paquetes Primero de todo, se instala el paquete `yum-plugin-verify`. Es una extensión, nunca se ejecutará directamente. Para verificar un sólo paquete:

   ```
   yum verify openssh-server
   ```

   Para verificar el paquete y sus dependencias:

   ```
   yum verify-all openssh-server
   ```

6. Instalar

   ```
   yum install openssh-server
   ```

7. Instalar `rpm` local

   ```
   yum localinstall paquete.rpm
   ```

8. Actualizar

   ```
   yum update
   ```

9. Ver los plugins

26.2. `zypper`

Igual que `yum`, `zypper` es un frontend de `rpm`. En general, hace lo mismo pero de un modo distinto. Esta herramienta se usa en SUSE Linux y openSUSE.

26.2.1. Consultas típicas

1. Mostrar lista de actualizaciones disponibles

   ```
   zypper list-updates
   ```

2. Mostrar los repositorios disponibles

   ```
   zypper repos
   ```

3. Buscar una coincidencia del string pasado en algún paquete

   ```
   zypper search ssh
   ```

4. Desplegar información de un paquete

   ```
   zypper info openssh-server
   ```

5. Buscar que paquete provee un archivo

   ```
   zypper search --provides /etc/nginx/nginx.conf
   ```

6. Instalar un paquete

   ```
   zypper install openssh-server
   ```

7. Actualizar los paquetes instalados

   ```
   zypper update
   ```

8. Desinstalar un paquete

   ```
   zypper remove openssh-server
   ```

9. Usar la shell

   ```
   zypper shell
   > install bash
   ...
   > exit
   ```

10. Agregar repositorio

    ```
    zypper addrepo url alias
    ```

11. Quitar repositorio

    ```
    zypper removerepo alias
    ```

26.3. APT

Apunte: Según parece, este capitulo se hizo antes de que existiese la herramienta llamada `apt`, sustituto de `apt-get`. Por lo tanto, hace referencia a varias herramientas (Advanced Packaging Tool), y no al sustituto. Para evitar confusiones, cuando se hable de APT en mayúsculas, se hablará del conjunto al que se refiere el curso, que son `apt-get` y `apt-cache`. En minúsculas (si se menciona), será el instalador.

Cómo ya se ha dicho, se hará referencia a `apt-get` y `apt-cache`. También hay otras interfaces, cómo `aptitude` y el Centro de Software de Ubuntu, pero no se tocarán aquí.

`apt-get` es la herramienta principal, con esta se instalaran, borraran y actualizaran paquetes.

26.3.1. Consultas típicas

1. Buscar un paquete

   ```
   apt-cache search nginx
   ```

2. Mostrar información básica

   ```
   apt-cache show nginx
   ```

3. Mostrar información detallada

   ```
   apt-cache showpkg nginx
   ```

4. Mostrar dependencias

   ```
   apt-cache depends nginx
   ```

5. Mostrar paquete que contiene un archivo

   ```
   apt-file search nginx.conf
   ```

6. Mostrar todos los archivos de un paquete

   ```
   apt-file list nginx
   ```

7. Sincronizar repositorios

   ```
   apt-get update
   ```

8. Instalar paquete

   ```
   apt-get install nginx
   ```

9. Quitar paquete

   ```
   apt-get remove nginx
   ```

10. Quitar paquete y purgar todos sus archivos

    ```
    apt-get --purge remove nginx
    ```

11. Actualizar paquetes

    ```
    apt-get upgrade
    ```

12. Actualizar distribución Hay que usarlo con cuidado.

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

13. Borrar paquetes innecesarios

    ```
    apt-get autoremove
    ```

14. Borrar paquetes en caché

    ```
    apt-get clean
    ```

27. Gestión de cuentas de usuario

El sistema GNU/Linux es multiusuario, por lo que la gestión de estos recae sobre la administradora de sistemas. Cada uno tiene su propio espacio de directorios y archivos, generalmente bajo `/home/`. Es importante el uso de usuarios distintos, aunque estos no los usen personas, para separa privilegios. Por ejemplo, nginx usa la usuaria y grupo `www-data`. Si en una aplicación web o el propio nginx hubiese alguna vulnerabilidad que permitiese el acceso al sistema de archivos, al estar los permisos separados, sólo podría leer y modificar aquellos para los que tenga permiso.

La única cuenta que debería tener acceso a todo el sistema es la de root. **Apunte**: Por ello mismo, el uso de `sudo` está desaconsejado, ya que permite a un usuario común poder acceder a todo el sistema.

27.1. Atributos de una cuenta de usuario

Cada usuaria existente tiene una linea en `/etc/passwd`, que describe sus atributos. Por ejemplo:

```
bitlbee:x:130:129::/var/lib/bitlbee/:/bin/false
```

Aquí se describe el nombre de usuario, contraseña, número de identificación de usuario (**UID**), número de identificación de grupo (**GID**), comentario, directorio `/home/` y la shell de inicio.

27.2. Creación de una cuenta

Hay dos órdenes posibles a usar, `adduser` y `useradd`, que es la que veremos:

```
su -c "useradd drymer"
```

Por defecto, usará los algoritmos de asignación automática de UID, GID y shell. En el siguiente orden, el anterior comando ejecuta lo siguiente:

- El siguiente UID máximo después de UID_MIN, definido en `/etc/login.defs`, es asignado
- Se crea un grupo con el nombre de usuario con GID=UID
- Se crea su directorio home en `/home/drymer/`
- Se le asigna `/bin/bash` cómo shell
- El contenido de `/etc/skel/` (skel de skeleton, esqueleto) se copian a `/home/drymer/` (por defecto archivos relacionados con `bash` y con el sistema X Window)
- Se asigna un símbolo de exclamación o dos (! o !!) en `/etc/shadow`, lo que indica a la administradora de sistemas que debe asignar una contraseña

Estos valores se pueden modificar fácilmente usando los siguientes parámetros:

```
su -c "useradd -s /bin/csh -m -k /etc/skel -c 'Soy drymer, wea' drymer"
```

27.3. Modificación y eliminación de una cuenta

Se elimina fácilmente, usando `userdel`. Por defecto no borrará el `/home/` de la usuaria (para ello, hay que pasar el parámetro `-r`), sólo se borrará toda referencia del usuario en `/etc/passwd`, `/etc/group` y `/etc/shadow`.

```
userdel drymer
```

Para modifica una cuenta, se usa `usermod`:

```
usermod --help
Modo de uso: usermod [opciones] USUARIO

Opciones:
  -c, --comment COMENTARIO      nuevo valor del campo GECOS
  -d, --home DIR_PERSONAL       nuevo directorio personal del nuevo usuario
  -e, --expiredate FECHA_EXPIR  establece la fecha de caducidad de la
cuenta a FECHA_EXPIR
  -f, --inactive INACTIVO       establece el tiempo de inactividad después
de que caduque la cuenta a INACTIVO
  -g, --gid GRUPO               fuerza el uso de GRUPO para la nueva cuenta
de usuario
  -G, --groups GRUPOS           lista de grupos suplementarios
  -a, --append                  append the user to the supplemental GROUPS
```

```
mentioned by the -G option without removing
him/her from other groups
  -h, --help                    muestra este mensaje de ayuda y termina
  -l, --login NOMBRE            nuevo nombre para el usuario
  -L, --lock                    bloquea la cuenta de usuario
  -m, --move-home               mueve los contenidos del directorio
personal al directorio nuevo (usar sólo
junto con -d)
  -o, --non-unique              permite usar UID duplicados (no únicos)
  -p, --password CONTRASEÑA     usar la contraseña cifrada para la nueva cuenta
  -R, --root CHROOT_DIR         directory to chroot into
  -s, --shell CONSOLA           nueva consola de acceso para la cuenta del
usuario
  -u, --uid UID                 fuerza el uso del UID para la nueva cuenta
de usuario
  -U, --unlock                  desbloquea la cuenta de usuario
  -v, --add-subuids FIRST-LAST  add range of subordinate uids
  -V, --del-subuids FIRST-LAST  remove range of subordinate uids
  -w, --add-subgids FIRST-LAST  add range of subordinate gids
  -W, --del-subgids FIRST-LAST  remove range of subordinate gids
```

27.4. Cuentas bloqueadas

Estas son las cuentas de usuaria que vienen por defecto con permiso de ejecución de programas pero sin permiso de inicio de sesión, ya que no tienen una contraseña asociada. Puede parecer que no sirven para nada, pero su fin es ese, el de ejecutar programas con otros usuarios para separar permisos de manera más sencilla. Unos ejemplos:

```
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
```

Si se intenta iniciar sesión con alguno de estos usuarios, se devuelve el texto que haya en `/etc/nologin.txt` si existe.

Se puede bloquear una cuenta existente usando el parámetro `-L`, y se desbloquea con `-U`. Otra forma de bloquearla automáticamente es añadirle una fecha de expiración mediante la herramienta `chage`, que se explica más adelante:

```
chage -E 2016-09-11 drymer
```

Si la fecha que se pone está en el pasado, se bloqueará automáticamente.

27.5. ID de usuaria en `/etc/passwd`

La convención es que los ID menores a 1000 pertenecen es una cuenta especial y pertenece al sistema; las cuentas de usuario normales empiezan a partir de 1000:

```
grep drymer /etc/passwd
drymer:x:1000:100:,,,:/home/drymer:/bin/bash
```

El valor de ID mínimo se saca de la variable **UID** del archivo `/etc/login.defs`. Al crear un usuario usando `usermod`, este buscará la variable mencionada e irá subiendo hasta encontrar una que no esté ocupada.

Se considera una mala practica modificar los archivos `/etc/{passwd, group, shadow}`, ya que para eso están las herramientas usermod. El editarlo a mano podría llevar, en casos extremos, a la corrupción del archivo.

27.6. `/etc/shadow`

Sólo contiene una linea por usuario, cómo:

```
grep drymer /etc/shadow
drymer:$5$8PQkp/HC/0TC.$IOWiP1yOE5.bVAeho65Dc2Kw4awZ05AOnDhBM.QCAY9:16832:0:99999:7:::
```

Apunte: Evidentemente esta no es mi contraseña, de haberla puesto alguien podría usar una herramienta de crackeo de contraseñas, tal cómo hashcat, y sacarla.

Este archivo tiene los campos separados por dos puntos (**:**). Estos campos corresponden, en orden:

- Nombre de usuario
- Contraseña
- Días desde el último cambio (formato de tiempo POSIX)
- Cantidad mínima de días a esperar que la contraseña pueda ser cambiada
- Cantidad máxima de días a esperar que la contraseña pueda ser cambiada
- Número de días que se lleva avisando a la usuaria de que la contraseña expira
- Número de días desde que la contraseña ha expirado
- Fecha de expiración
- Campo reservado

El hash usado en la contraseña sigue un formato único, que es 6 + ocho caracteres de salt + **$** + un hash de la contraseña usando sha512.

27.6.1. Por que usar `/etc/shadow`?

Apunte: Esta pregunta podía tener sentido hace años, pero en la actualidad todos las distribuciones lo usan.

Hay que usarlo por varios motivos:

- Habilita la caducidad de las contraseñas
- Los permisos son más estrictos que en `/etc/passwd`

27.7. Gestión de contraseñas

Se hace con la herramienta `passwd`:

```
passwd --help
Modo de uso: passwd [opciones] [USUARIO]

Opciones:
  -a, --all                     informa del estado de las contraseñas de
todas las cuentas
  -d, --delete                  borra la contraseña para la cuenta indicada
  -e, --expire                  fuerza a que la contraseña de la cuenta
caduque
  -h, --help                    muestra este mensaje de ayuda y termina
  -k, --keep-tokens             cambia la contraseña sólo si ha caducado
  -i, --inactive INACTIVO       establece la contraseña inactiva después de
caducar a INACTIVO
  -l, --lock                    bloquea la contraseña de la cuenta indicada
  -n, --mindays DÍAS_MIN        establece el número mínimo de días antes
de que se cambie la contraseña a DÍAS_MIN
  -q, --quiet                   modo silencioso
  -r, --repository REP          cambia la contraseña en el repositorio REP
  -R, --root CHROOT_DIR         directory to chroot into
  -S, --status                  informa del estado de la contraseña la cuenta
indicada
  -u, --unlock                  desbloquea la contraseña de la cuenta indicada
  -w, --warndays DÍAS_AVISO     establece el aviso de caducidad a DÍAS_AVISO
  -x, --maxdays DÍAS_MAX        establece el número máximo de días antes de
cambiar la contraseña a DÍAS_MAX
```

Toda contraseña escogida, si se tiene `pam` activo, es examinada por `pam_cracklib`, que se encargará de decir si la contraseña es buena o no.

Sin permiso de root, sólo se puede cambiar la contraseña propia, lógicamente.

27.8. `chage`: envejecimiento de contraseñas

Una buena practica de seguridad es ir cambiando las contraseñas, ya que de este modo si alguien consigue acceso a una máquina, puede perderlo al cambiarla, además de que limita el tiempo necesario para crackearla. Es una política complicada de aplicar, ya que los usuarios pueden encontrarla molesta, y de hecho puede llevar a otras malas prácticas, cómo usar contraseñas sencillas o apuntarlas en papeles delante del ordenador. Por lo tanto, hay que usarla con cuidado.

Los parámetros que se pueden usar son los siguientes:

```
chage -h
Modo de uso: chage [opciones] USUARIO

Opciones:
  -d, --lastday ÚLTIMO_DÍA      establece el día del último cambio de la
contraseña a ÚLTIMO_DÍA
```

```
  -E, --expiredate FECHA_CAD    establece la fecha de caducidad a FECHA_CAD
  -h, --help                    muestra este mensaje de ayuda y termina
  -I, --inactive INACTIVA       deshabilita la cuenta después de INACTIVA
días de la fecha de caducidad
  -l, --list                    muestra la información de la edad de la cuenta
  -m, --mindays DÍAS_MIN        establece el número mínimo de días antes de
cambiar la contraseña a DÍAS_MIN
  -M, --maxdays DÍAS_MAX        establece el número máximo de días antes de
cambiar la contraseña a DÍAS_MAX
  -R, --root CHROOT_DIR         directory to chroot into
  -W, --warndays DÍAS_AVISO     establece los días de aviso de expiración a
DÍAS_AVISO
```

Apunte: Un parámetro útil es `-d`, que se puede usar con el valor 0 para forzar el cambio de contraseña en el siguiente inicio de sesión. Por lo tanto, si se cree que se ha sufrido una incursión en la máquina, se podría usar el siguiente snippet para forzar el cambio de contraseñas de todos los usuarios en el siguiente inicio de sesión:

```
for user in ‘cat /etc/passwd|grep "^[a-z|A-Z]*:x:100.:" | cut -d':' -f1‘
do
    chage -d 0 $user
done
```

Lo que hace es parsear los usuarios de `/etc/passwd` cuyo ID es mayor que 1000 y forzar el reinicio. Esto es una muestra, que por cierto no se menciona en el curso, de por que es muy conveniente saber un poco de programación en `bash`. Incluso saber un poco de `python` puede venir bien. Ambos lenguajes ayudan mucho en el momento de automatizar ciertas acciones.

Se dice mucho que cualquier sysadmin debería saber que hace cualquier herramienta que tenga tres caracteres o menos, y aunque se puede colar alguna herramienta que no entre dentro de esto, es generalmente cierto. Para encontrar estas herramientas:

```
ls {/bin,/usr/bin} | grep "^..$\|^...$"
```

27.9. Shell restringida

Se puede invocar con `bash -r` o según la distribución con `rbash`. Tiene las siguientes características:

- Impide salirse del `$HOME`
- No se puede redefinir las variables `$SHELL`, `$ENV` y `$PATH`.
- No permite usar rutas absolutas en ejecutables
- No permite redireccionar entrada o salida

Estas son unas pocas, para verlas todas, se puede mirar en el manual de bash con `man bash`.

27.10. Cuentas restringidas

Estas son las que se quiere dar un acceso limitado. Para ello, en `/etc/passwd` se debe poner la shell `rbash`. De no existir, debería crearse algún modo para que pueda usarla, ya que `/bin/bash -r` no se puede usar en ese archivo. **Apunte:** en el curso se dice que se puede solucionar ejecutando `cd /bin/; ln bash rbash`, pero no le veo sentido ya que se ejecuta del mismo modo que bash. De modo que yo haría lo siguiente.

```
nano /bin/rbash

# y dentro de este
cd $HOME
/bin/bash -r
```

27.11. La cuenta root

Apuntes: Si a estas alturas aún no sabemos que es, mal. Tampoco se va a decir nada nuevo, pero en fin, hay que ser minucioso.

Esta cuenta sólo debería usarse con propósitos administrativos, sólo cuando sea absolutamente necesario. Se recomienda el acceso a esta cuenta mediante `su` o `sudo` (**Apunte:** yo sólo recomiendo `su`, por motivos que ya se mencionaron antes). Mediante **PAM**, se puede restringir el acceso a su a algunas cuentas en concreto. También puede ser útil configurar `audit` para que registre todos los accesos a root.

27.12. `ssh`

Este es una herramienta de administración remota. Generalmente de terminal, aunque es posible arrancar programas gráficos si se configura correctamente. **Apunte:** Si, si es habitual , por mucho que diga la Policía.

Esta es la herramienta más segura para conectarse a máquinas remotas y también de pasar archivos (mediante `scp`), por algo las siglas `ssh` corresponden a **Secure Shell**.

Algunos ejemplos de las posibilidades de ambas herramientas:

```
# conectar por ssh a mi servidor vps
ssh drymer@daemons.cf
# esto crea un puerto que está a la escucha, el 8080, que permite ser usado cómo proxy y s
ssh -D 8080 drymer@daemons.cf
# enviar el archivo.txt al home del servidor remoto
scp archivo.txt drymer@daemons.cf:/home/drymer/
# copiar el archivo.txt del servidor remoto en el home de la máquina local
scp drymer@daemons.cf:/home/drymer/archivo.txt /home/drymer/
# copiar todo el home remoto en el directorio en el que estoy
scp -r drymer@daemons.cf:/home/drymer/ .
```

27.12.1. Archivos de configuración

En el $HOME hay un directorio muy importante, que es el `.ssh`. Ahí se guardaran varios archivos importantes:

```
ls .ssh/
authorized_hosts  id_rsa      known_hosts
config            id_rsa.pub
```

Estos archivos tienen la función siguiente:

- `authorized_hosts`: archivo con todas las claves públicas que pueden conectarse al servidor.
- `config`: un archivo de configuración.
- `id_rsa`: la clave privada de ssh.
- `id_rsa.pub`: la clave pública de ssh. Si se quiere conectar a los servidores remotos sin usar la contraseña, habrá que agregar esta clave al archivo `authorized_hosts` de la máquina remota.
- `known_hosts`: este archivo se crea sólo y guarda la clave pública de los servidores a los que nos conectamos. Si en la siguiente conexión detecta que la clave ha cambiado, no dejará conectar ya que considera que hemos sufrido un **MITM**. Si hemos sido nosotros que hemos cambiado la clave, habrá que quitar la linea guardada en ese archivo.

Antes del primer uso de `ssh` cómo cliente, hay que crear las claves mediante la orden `ssh-keygen`:

```
ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/drymer/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/drymer/.ssh/id_rsa.
Your public key has been saved in /home/drymer/.ssh/id_rsa.pub.
The key fingerprint is:
a5:3b:dd:12:39:f3:11:a9:b9:18:fb:1d:38:b3:9a:a5 drymer@debian-bittorrent
The key's randomart image is:
+--[ RSA 4096]----+
|                 |
|              .  |
|           . o   |
|          o + .  |
|        S B .    |
|         * O .   |
|        = O +    |
|         * * .   |
|        E.o .    |
+-----------------+
```

Apunte: De este modo se creará una clave `rsa` de 4096 bits. Las de 2048 bits de momento se siguen considerando seguras, pero cómo no cuesta nada ser un poco más paranoico, por que no serlo?

Al ejecutar la orden anterior veremos que pregunta por una contraseña. Podemos dejarla en blanco si queremos hacer conexiones automáticas al servidor. Siempre será un poco más inseguro, pero será más cómodo. Una vez más, hay que escoger entre comodidad y seguridad.

28. Gestión de grupos

El uso de los grupos es el de extender los permisos de las usuarias y extenderles las áreas de trabajo. Los grupos están definidos en `/etc/group`, que cumple el mismo objetivo que `/etc/passwd` con los usuarios. Un ejemplo de este archivo es el siguiente:

```
cat /etc/group
root:x:0:root
bin:x:1:root,bin
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
```

28.1. Herramientas

Las herramientas para manejar los grupos son `groupadd`, `groupdel`, `groupmod` y `usermod`. Se puede editar a mano el fichero mencionado, pero igual que antes no es recomendable dado la posibilidad de corrupción del archivo. De querer hacerlo, habría que hacerlo con `vigr`, que es un enlace simbólico a `vipw`.

Sólo cabe mencionar una cosa acerca de las herramientas mencionadas, y es que `usermod`, al añadir grupos, debe usarse siempre con el parámetro `-a`, ya que sino borrará los grupos existentes y se quedará sólo con el que se use en ese momento.

28.2. Grupos privados (UPG)

Antes se mencionó que al crear un usuario, siempre se crea un grupo con el mismo nombre. Esto es importante por que el `umask` se especifica en el `/etc/profile` y se establece en **002** en los usuarios con GID = UID. Este esquema es de permisos 644 (rw-rw-r--) y los directorios con 755 (rwxrwxr-x). Se hablará de =umask en el próximo capítulo.

29. Permisos de archivos y propietarias

Cada archivo en GNU/Linux, y recordemos que una de sus características es que todo son archivos, tiene especificado que propietaria, grupo y otros pueden acceder a el. Se puede visualizar con `ls -l`:

```
ls -l
total 68
drwx------ 398 drymer users 20480 feb  6 01:29 Biblioteca
drwx------  28 drymer users 20480 jun 13 12:28 Descargas
lrwxrwxrwx   1 drymer users    26 feb 23 12:32 Documentos -> .owncloud/Casa/Documentos/
drwx------   8 drymer users  4096 nov 25  2015 ISO
lrwxrwxrwx   1 drymer users    24 feb 23 12:33 Imagenes -> .owncloud/Casa/Imagenes/
drwxr-----  33 drymer users  4096 jun  1 09:11 Instalados
lrwxrwxrwx   1 drymer users    17 feb 23 18:25 Musica -> .owncloud/Musica/
lrwxrwxrwx   1 drymer users    23 feb 23 12:33 Scripts -> .owncloud/Casa/Scripts/
```

```
drwxr-----  17 drymer users  4096 may 19 13:32 Videos
drwx------   3 drymer users  4096 jun 13 10:08 tor-browser_en-US
```

Aquí se puede ver que el **usuario** es drymer y el grupo es **users**. En mi caso, cómo soy un pelín paranoico, siempre pongo los permisos sólo con permiso de lectura y escritura para mi usuario, a los demás grupos ni agua. Un apunte, es que el directorio `Documentos` y los otros con similar escritura, son enlaces simbólicos, por eso tienen unos permisos raros y una **l** al principio de todo.

29.1. Derecho de acceso

Cada archivo o directorio tiene un grupo de elementos, cómo se puede ver. El primero de todos, una **d**, sólo se usa en el caso de los directorios. Los tres siguientes, **rwx**, son permisos de escritura (r), lectura (w) y ejecución (x). El permiso de ejecución en el caso de los directorios lo que hace es permitir o denegar la entrada en este. Cuando aparece un guión (-), significa que no se tiene permiso para hacer lo que corresponda.

También existen otros permisos ya mencionados, el **setuid** y el **setgid**.

29.2. Modificación de permisos

Esta se realiza con `chmod`. A menos que se use la usuaria **root**, sólo se podrá modificar los archivos propios. Un ejemplo de su uso:

```
touch hola
ls -l hola
-rw-rw---- 1 drymer users 0 jun 13 12:50 hola
chmod 777 hola
ls -l hola
-rwxrwxrwx 1 drymer users 0 jun 13 12:50 hola
```

Se pueden usar dos notaciones para dar permisos. La primera es cómo he mostrado, en notación octal, y la segunda con representaciones simbólicas, que son del tipo `chmod u+rwx`. La notación octal permite usar un número para concretar los permisos de los tres bits que lo tocan. El funcionamiento es el siguiente:

- **4** si se quiere dar permiso de lectura
- **2** si se quiere dar permiso de escritura
- **1** si se quiere dar permiso de ejecución

Así, **7** seria `rwx`, **6** seria `rw-` y **5** `r-x`.

29.3. Modificación de usuario y grupo propietario

Esto se realiza con `chown` y `chgrp`. En este caso sólo root puede cambiar los propietarios de los archivos. Estas herramientas no tienen mucho misterio, sólo hay que saber que el parámetro `-R` en `chown` hace que se comporta de manera recursiva y que se le puede pasar el grupo que se quiere que sea suyo del siguiente modo: `chown -R drymer.users /home/drymer/`.

29.4. `umask`

Como ya se ha mencionado antes, los permisos por defectos vienen dados por `umask`. Depende de la distribución estará en un número o en otro, pero en general suele estar en `022`, que hace que los archivos creados sólo se puedan leer por miembros del grupo por defecto y los demás no puedan. Se puede cambiar el `umask` en cualquier momento ejecutando `umask 0022`, pero los cambios no son permanentes, si se quiere esto debería cambiarse en `/etc/profile`, ya que el contenido de este archivo entra en la terminal que se usa cada vez que se abre una nueva. Hay que recordar que cambiarlo no cambiará los permisos actuales, sólo los creados en el futuro.

29.5. Access Control List (ACL)

Esta es la lista de control de acceso, que todo GNU/Linux lleva implementado por defecto. Esta es una extensión del modo de establecer permisos que hemos visto ya, y va incorporado en el propio kernel.

29.5.1. Herramientas

Para ver la acl de un directorio o fichero, se usa `getfacl`:

```
getfacl /home/
getfacl: Eliminando '/' inicial en nombres de ruta absolutos
# file: home/
# owner: root
# group: root
user::rwx
group::r-x
```

Para configurar estas, se usa `setfacl`:

```
ls -l hola
-rw-rw---- 1 drymer users 0 jun 13 13:18 hola
setfacl -m u:drymer:rwx hola
ls -l hola
-rw-rwx---+ 1 drymer users 0 jun 13 13:18 hola
```

Hay que tener en cuenta que si usa con un directorio, todos los archivos que se creen en este heredaran esta **acl**. Para borrarla:

```
ls -l hola
-rw-rwx---+ 1 drymer users 0 jun 13 13:18 hola
setfacl -x u:drymer hola
ls -l hola
-rw-rw----+ 1 drymer users 0 jun 13 13:18 hola
```

Se puede ver que el símbolo + del final sigue apareciendo, ya que la acl sigue existiendo aunque los permisos del archivo sean igual que antes.

30. Pluggable Authentication Modules (PAM)

Este sistema provee un mecanismo unificado para asegurar la autenticación e identificación de usuarias y programas. De este modo, `su`, `ssh` y `login`, entre otras, antes usaban cada una su propio sistema de autenticación. Ahora estas usan **PAM**, con la librería **libpam**. Estas librerias se encuentran, según el sistema operativo y su arquitectura, en `/lib/security/*`, `/lib64/security/*` o `/lib/x86_64/-linux-gnu`. Cada aplicación que usa esta librería, tiene archivos de configuración en `/etc/pam.d/`.

El proceso que se sigue es el siguiente:

- Se llama a la aplicación que usa **PAM**.
- La aplicación llama a **libpam**.
- La biblioteca comprueba la existencia de archivos de configuración en `/etc/pam.d/`, en los que se concreta que módulos se usaran.
- Se ejecuta cada módulo referenciado.

30.1. Archivos de configuración

Un ejemplo de los programas que hay en `/etc/pam.d/`:

```
ls /etc/pam.d/
atd       chsh            common-password               cron     login     other     samb
chfn      common-account  common-session                cups     monit     passwd    sesm
chpasswd  common-auth     common-session-noninteractive  dovecot  newusers  polkit-1  smtp
```

Un ejemplo del contenido del archivo de configuración de `ssh`:

```
cat /etc/pam.d/sshd
# PAM configuration for the Secure Shell service

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
auth       required     pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
auth       required     pam_env.so envfile=/etc/default/locale

# Standard Un*x authentication.
@include common-auth

# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
# account  required     pam_access.so

# Standard Un*x authorization.
```

```
@include common-account

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional     pam_motd.so  motd=/run/motd.dynamic noupdate
session    optional     pam_motd.so # [1]

# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Set up SELinux capabilities (need modified pam)
# session  required     pam_selinux.so multiple

# Standard Un*x password updating.
@include common-password
```

A continuación desgranaremos partes comunas del anterior archivo con los archivos de otros servicios. El tipo **type** (valga la redundancia) acepta los siguientes parámetros:

- **auth**: Indica que hay que pedir la identificación del usuario.
- **account**: Comprueba que los permisos de la cuenta sean correctos, tales como que la contraseña no ha caducado.
- **password**: Actualiza el id de la contraseña.
- **session**: Provee de funciones antes y/o después de que se establezca sesión, cómo por ejemplo la comprobación de nuevo correo para el usuario y su envío a este.

El parámetro **control** establece cuán necesario es un módulo:

- **required**: Si no existe se ejecutará el resto de módulos, pero la aplicación fallará y no se informará a la usuaria cual ha fallado.
- **requisite**: Igual a **required** exceptuando que cuando no se encuentra un módulo, termina el conjunto de módulos y devuelve su estado.
- **optional**: No es necesario, pero si es el único módulo fallará.
- **sufficient**: Si este módulo carga correctamente, se terminará el conjunto de módulos sin cargar ninguno más. Sino, se cargarán los siguientes módulos, a menos que sea el único, en cuyo caso devolverá el fallo.

El tipo **module-path** da la ruta del módulo, y **module-arguments** sus argumentos. Hay más tipos que los mencionados, cómo por ejemplo **include** y **substack**. Para verlos en mayor profundidad, **RTFM** con `man pam.d`. absorber

30.2. Autenticación LDAP

LDAP son las siglas de **Protocolo Ligero de Acceso a Directorios**, es un protocolo pensado para centralizar la autenticación de los servicios de una red. Se usa TLS, lo que hace la conexión segura.

LDAP usa **PAM**, **system-config-authentication**, **authconfig-tui**, **DN** (dominio), **TLS**, **openldap-clients**, **pam$_{\text{ldap}}$** y **nss-pam-ldapd**. Hay cinco archivos de configuración claves para la configuración de **LDAP**, que son:

- `/etc/openldap/ldap.conf`
- `/etc/pam_ldap.conf`
- `/etc/nscld.conf`
- `/etc/sssd/sssd.conf`
- `/etc/nsswitch.conf`

31. Métodos de respaldos y recuperación de la información

Este es uno de los trabajos principales de una sysadmin, encargarse de que nunca, bajo ninguna circunstancia, se pierda información. Por ello, un sistema sin una política bien definida de **copias de seguridad**, está mal administrado.

Ahora, el problema es decidir que se respalda. Por que, aunque cada vez los GB en discos duros son más baratos, las empresas consumen también cada vez más, por lo que a cuanto más respaldo, más dinero costará hacerlo. Por ello hay que encontrar un término medio entre lo que es necesario y lo que estaría bien respaldar. Un ejemplo, en orden de prioridad:

- El negocio depende de ello:
  - Datos relacionados con el negocio
  - Archivos de configuración del sistema
  - Archivos de usuarias
- El negocio no depende de ello pero no sobra:
  - Directorios de cola (imprimir, email, etc)
  - Logs (`/var/log/`)
- No hay que hacer backup
  - Pseudo sistema de archivos cómo `/proc/`
  - Swap

31.1. Unidades de cinta

Estas ya no se suelen usar tanto, ya que son relativamente lentas y sólo se puede acceder a ellas secuencialmente. Ahora ya sólo se suele usar cómo respaldo secundario y más por tener las máquinas antiguas que por haberlas comprado nuevas para ello.

31.2. Tipos de backups

Hay distintos tipos de respaldos, cada uno con su propia utilidad y momento de uso. Lo habitual, de hecho, es usar varios de ellos.

- **Completo**: Respalda todo lo que contiene una máquina
- **Incremental**: Respalda todos los cambios desde la última copia, ya sea incremental completo.
- **Diferencial**: Respalda todos los cambios desde el último backup completo.
- **Nivel incremental múltiple**: Respalda todos los cambios desde el último backup en el mismo nivel.

31.3. Herramientas de respaldo

De bajo nivel, hay varias:

- `cpio`: crea un contenedor
- `tar`: crea un contenedor
- `gzip`, `bzip` y `xz`: comprime un archivo o un contenedor
- `dd`: copia sectores en bruto
- `rsync`: sincroniza arboles de directorios, ya sea en la misma máquina o en una remota
- `dump` y `restore`: hacen backups y los restauran, respectivamente, además de que lo hacen mirando en el sistema de ficheros directamente lo cual es más eficaz
- `mt`: sirve para hacer y restaurar backups de cintas

31.3.1. `cpio`

Significa CoPy In and Out, y es de las primeras herramientas de compresión que existen desde los tiempos de **Unix**. En un principio estaba pensado para realizar copias en cintas. Aunque no sólo sirve para eso, `tar` surgió cómo alternativa y se usa bastante más, aunque hay algunas excepciones. Una es la herramienta que ya vimos, `rpm2cpio`, otra es el kernel de Linux, que usa `cpio` para la creación de `initramfs` e `initrd`.

31.3.2. `tar`

Esta herramienta se usa mucho para hacer copias de seguridad, muchas veces después se comprime el `tar` creado con una herramienta cómo `gunzip`, por ejemplo.

1. Ejemplos de uso Para hacer un backup de `/root/`, por ejemplo:

```
tar cvf backup.tar /root/
```

Para restaurarlo en el directorio en el que estemos:

```
tar xvpf backup.tar /root/
```

Después de esto, se puede jugar con otras opciones, cómo restaurar directamente en una cinta de copias.

También se puede hacer respaldos incrementales con `tar`. La clave es el parámetro `-N` o `--newer`, por lo que se usaría del siguiente modo:

```
tar xvpfN '2016-06-30' backup.tar
```

De este modo, sólo se copiaran los archivos más nuevos que esa fecha.

31.3.3. `gzip`, `bzip` y `xz`

Es algo habitual comprimir los archivos que no se consultan habitualmente para ahorrar espacio o si se tienen que transmitir por la red, por lo que aunque comprimir y descomprimir consume tiempo, suele ser menos que el de transmitir los datos sin comprimir.

Las distintas herramientas usan distintos esquemas de compresión:

- `gzip`: Usa el algoritmo Lempel-Ziv y genera archivos `.gz`.
- `bzip2`: Usa el algoritmo Burrows-Wheeler y de codificación de Huffman y genera archivos `.bz2`.
- `xz`: Genera archivos `.xz` y `.lzma`.

`gzip` se suele usar para tamaños relativamente pequeños, ya que es bastante rápido, pero para almacenar archivos o directorios muy grandes se usan los demás. El kernel Linux, por ejemplo, ya no se ofrece en formato `.gz`.

Es muy sencillo integrar estas herramientas con el uso normal de `tar`:

1. Ejemplos de uso

```
tar xvzf backup.tar.gz /root/ # gzip
tar xvjf backup.tar.bz2 /root/ # bzip2
tar xvJf backup.tar.xz /root/ # xz
```

31.3.4. `dd`

Es una herramienta original de **Unix**. Puede replicar discos enteros byte a byte a imágenes u otros discos, entre otras. También puede cambiar el orden de estos, controlar los offsets, tamaño de los bloques, etc.

Además se puede usar para leer dispositivos especiales, cómo `/dev/random`.

1. Ejemplos de uso Un ejemplo de copia de un disco a una imagen, para poder restaurar en caso de problema:

```
dd if=/dev/sda of=sda.img bs=2048
```

El parámetro `if` define el origen, el `of` el destino y `bs` la cantidad de bytes que lee cada vez (incrementa la velocidad).

Crear un archivo de 10 MB lleno de ceros:

```
dd if=/dev/zero of=file bs=1M count=10
```

Copiar el primer disco al segundo:

```
dd if=/dev/sda of=/dev/sdb
```

31.3.5. `rsync`

Se usa para transferir archivos, ya sea en el propio equipo o en la red. Permite copias diferenciales o incrementales, es decir, que primero verifica los cambios en los archivos de origen y sólo los cambios que han habido, lo cual lo hace especialmente útil para la transmisión por red.

1. Ejemplos de uso

   Copiar un directorio local a una máquina remota:

   ```
   rsync ~/.bashrc drymer@192.168.1.92:/home/drymer/
   ```

   No hace falta usar ningún tipo de parámetro, sólo usar la sintaxis correcta. Para coger el `.bashrc` de la máquina remota y copiarla a la local:

   ```
   rsync -Pr drymer@192.168.1.92:/home/drymer/ ~/.bashrc
   ```

   El parámetro `-P` muestra el progreso. Otros parámetros usados para hacer backups son:

   ```
   rsync -avzP /home/drymer/ /var/backups/
   ```

   Muestra el progreso y sólo copia de manera eficiente, es decir, los cambios que han habido. Pero sólo los cambios, no borrará nunca los archivos de `/var/backups/`. Para que lo haga, hay que usar el parámetro `--delete`.

31.3.6. `dump` y `restore`

Estas herramientas no fueron diseñadas para acceder a dispositivos, pero acceden directamente al sistema de archivos, por lo que es muy eficiente y los respaldos creados no tienen las marcas de tiempo modificadas. Sus características:

- Pueden realizar respaldos completos o incrementales
- Son eficientes al hacer un backup completo, reducen el movimiento del cabezal (en los discos IDE)
- Se puede especificar el tamaño de salida y la densidad de la cinta, tamaño de bloque y cuenta (esto no es muy útil que digamos)

- Por defecto usa `/dev/tape/`, pero puede usar cualquier archivo válido.

Sus desventajas:

- Se requieren múltiples pasadas
- Sólo funciona con `ext`
- No se puede ejecutar de forma segura en sistemas de archivos montados

31.3.7. `mt`

Se usa para controlar cintas magnéticas. **Apunte**: No tiene mucho sentido aprender esto, no creo que nadie lo use.

31.4. Software de respaldo

Muchas de las herramientas con una configuración más sencilla, es decir, una gui, usan las herramientas de bajo nivel que hemos comentado. Algunas de ellas son:

- `amanda`
- `bacula`
- `clonezilla`
- `uniscan`

32. Direcciones de red

Las direcciones IP se usan para identificar tanto global cómo locamente de una manera única a los dispositivos conectados a estas. Hay dos formatos de estas, **IPV4**, que son las más usadas actualmente aunque están condenadas a dejar de usarse (de forma global), ya que ya se han terminado todas. La segunda es **IPV6**, que aún está en, aunque deberíamos aprenderla todas.

El formato de:

- **IPV4**: es una dirección de 32 bits, compuesta de 4 octetos (un octeto son 8 bits o un byte) en notación decimal:

148.114.252.10

- **IPV6**: es una dirección de 128 bits, compuesta de 16 octetos en notación hexadecimal:

2003:0db5:6123:0000:1f4f:5529:fe23

32.1. Tipo de direcciones IPV4

- Unicast: asociada a un host especifico. Un ejemplo es 192.168.1.92.
- Red: asociada a la red, el host siempre está a cero (este puede ser de uno a tres octetos, según el tipo de red cómo se verá más adelante). Un ejemplo es 192.168.1.0.
- Broadcast: todos los miembros de una red escuchará esta dirección, se usa para mandar un mensaje a toda la red entera. La sección del host siempre será el número máximo de la red. Por ejemplo: 192.168.1.255
- Multicast: se configura por host, a diferencia de los anteriores, por lo que no todos tendrán. Un ejemplo es 192.168.1.3

32.2. Direcciones especiales

Hay ciertos rangos o direcciones reservados:

- 127.x.x.x: normalmente siempre será 127.0.0.1, pero cualquiera que empiece por 127 vale. Siempre hará referencia al sistema local.
- 0.0.0.0: se usa por sistemas que no saben su propia dirección (o les da igual para el caso de uso). DHCP, por ejemplo, lo usa.
- 255.255.255.255: es una dirección privada genérica de broadcast, para uso interno. Normalmente no sirven de demasiado.
- 10.0.0.0:10.255.255.255 | 172.16.0.0:172.31.255.255 | 192.169.0.0:192.168.255.255: estas siempre se usan para redes internas.

En la wikipedia se puede ver más IP reservadas.

32.3. Tipo de direcciones IPV6

Algunas son similares a las de tipo IPV4:

- Unicast: se entrega un paquete a una sola interfaz, ya sea un enlace local o global.
- Multicast: se entrega a múltiples interfaces.
- Anycast: se entrega a la interfaz más cercana del destino (en términos de distancia de enrutamiento).
- IPv4-managed: una dirección IPV4 mapeada a IPV6, cómo por ejemplo **::FFF:a.b.c/96**

32.4. Clases de red IPV4

En la tabla se pueden ver los distintos tipos.

Clase de red	Rango de octeto de orden más alto	Notas
A	1-126	128 redes, de 16.772.214 hosts por red
B	128-191	16.384 redes, 65.534 hosts por red
C	192-223	2.097.152 redes, 254 por red
D	224-239	Direcciones multicast
E	240-254	Direcciones reservadas

32.5. Máscara de red

Se usa para determinar que porción de la dirección se usa para la red y cual para el host. Por tipo de red, vistas en el anterior punto, las máscaras posibles son las siguientes:

Clase de red	Decimal	Hexadecimal	Binario
A	255.0.0.0	ff:00:00:00	11111111 00000000 00000000 00000000
B	255.255.0.0	ff:ff:00:00	11111111 11111111 00000000 00000000
C	255.255.255.0	ff:ff:ff:00	11111111 11111111 11111111 00000000

Por ejemplo, viendo la máscara de red **255.255.255.0** de la IP **192.168.1.92**, sabemos que es una red de clase C, por lo tanto el host es el cuarto bit únicamente, **92**.

32.6. Hostname

El nombre del host (No shit sherlock). Se usa para identificar un dispositivo de la red sin usar una IP. Se puede hacer mediante los DNS, editando el archivo `/etc/hosts`, entre otras.

Para verlo, se ejecuta `hostname` en la terminal. En mi caso, cómo me gustan los nombres lógicos, se verá así:

```
hostname
netbook
```

Para cambiarlo, se puede pasar cómo parámetro el nombre deseado y ejecutar `hostname` con permiso de root:

```
su -c "hostname darkstar"
darkstar
```

Este cambio sólo es temporal y se irá al reiniciar, si se quiere hacer permanente, hay que editar uno de los archivos siguientes, que cambia según las distribución: `/etc/hostname`, `/etc/HOSTNAME`, `/etc/sysconfig/network`.

33. Configuración de dispositivos de red

A diferencia de otros tipos de dispositivos, los de red no tienen un archivo en `/dev/` asociados a su nombre. En general:

- **ethx**: siendo x un número mínimo de 0, son para dispositivos ethernet

- **wlanx**: siendo x un número mínimo de 0, para dispositivos inalámbricos
- **brx**: siendo x un número mínimo de 0, para puentes de red (bridge)
- **vmnetx**: siendo x un número mínimo de 0, para dispositivos de vmware

A veces los dispositivos virtuales están asociados con los físicos, esto hace que la misma tarjeta de red tenga distintas direcciones IP.

33.1. Problemas asociados a los nombres de dispositivos de red

Estos surgen cuando hay más de una interfaz del mismo tipo. La solución obvia es llamar **eth0** a la primera que encuentre y **eth1**, por ejemplo. El problema es el método del sondeo de dispositivos no es determinístico en los sistemas modernos (sobretodo si metemos a systemd en medio con su versión de udev). Aunque el número de dispositivos no cambie, un cambio de kernel podría hacerlo.

La solución más simple es asociar los nombres a las direcciones MAC, que a menos que las cambiemos nosotras con un programa tipo `macchanger`, no cambiaran solas.

33.2. Tipos de nombres predecibles

De esto se encarga **udev** y **systemd**, en el caso de estar instalado. Por lo tanto, tenemos las siguientes posibilidades:

- **eno1**: incorpora números de índices proporcionados por el firmware o la BIOS, normalmente se usa para dispositivos embebidos.
- **ens1**: incorpora el número de índices de slots hotplug de PCI Express proporcionado por el firmware o la BIOS.
- **enp2s0**: incorpora ubicación física y/o geográfica de la conexión de hardware. (es el que usa systemd ahora)
- **enx7837s324383**: incorpora la MAC.
- **eth0**: el método clásico.

33.3. Archivos de configuración de una tarjeta de red

Dependiendo de la distribución estará en un sitio o en otro:

- **RHEL 6**: `/etc/sysconfig/network` y `/etc/sysconfig/network-scripts/ifcfg-eth0`
- **Opensuse 13.1**: `/etc/sysconfig/network/ifcfg-eno16777736` y `/etc/sysconfig/network/ifcfg-`
- **Ubuntu 14.04**: `/etc/network/interfaces`.

Las tarjeta de red se pueden desactivar y activar con `ifdown` y `ifup` respectivamente, siendo el cambio persistente. Con `ifconfig` e `ip` no lo es.

33.4. Resolución de nombres

No tiene mucho misterio. Es el acto de traducir el nombre de dominio `gnu.org` por la IP `208.118.235.148`. Normalmente de esto se encarga el DNS, mediante la resolución dinámica. Pero también se puede hacer de manera estática editando el fichero `/etc/hosts`.

33.5. `/etc/hosts`

Mantiene una "base de datos"local de hostnames y direcciones IP. Un ejemplo es el siguiente.

```
127.0.0.1       localhost
192.168.1.93    banana
192.168,1.92    bittorrent
```

banana y **bittorrent** son una Banana Pi y un chustaservidor que tengo. Y cómo tengo esto en mi archivo de `hosts`, puedo hacer lo siguiente:

```
ping -c1 banana
PING banana (192.168.1.93) 56(84) bytes of data.
64 bytes from banana (192.168.1.93): icmp_seq=1 ttl=64 time=0.288 ms

--- banana ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.288/0.288/0.288/0.000 ms
```

33.6. Herramientas

33.6.1. `ifconfig`

Muestra información de todas las interfaces:

```
ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 00:e0:27:5f:0c:f8  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 11

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1  (Local Loopback)
RX packets 580  bytes 43960 (42.9 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 580  bytes 43960 (42.9 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Muestra información de una tarjeta de red:

```
ifconfig eth0
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 00:e0:27:5f:0c:f8  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 11
```

Cambia la IP de forma no persistente:

```
su -c "ifconfig eth0 192.168.1.90"
ifconfig eth0
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 192.168.1.90  netmask 255.255.255.0  broadcast 192.168.1.255
ether 00:e0:27:5f:0c:f8  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 11
```

Cambia la MTU (Unidad de Transferencia Máxima):

```
su -c "ifconfig eth0 mtu 1480"
```

33.6.2. `ip`

Esta herramienta es más nueva, más versátil y más eficiente que `ifconfig`. Es más eficiente por que en vez de usar llamadas **ioctl** usa sockets **netlinks**.

Esta herramienta puede usarse desde para desplegar información hasta controlar el enrutamiento y hacer túneles. A continuación, algunos ejemplos.

Mostrar información de todas las redes:

```
ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group def
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
```

Mostrar información de una sola interfaz:

```
ip -s link show eth0
3: eth0: <BROADCAST,MULTICAST> mtu 1480 qdisc pfifo_fast state DOWN mode DEFAULT group def
    link/ether 00:e0:27:5f:0c:f8 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
```

Configurar la dirección de una interfaz:

```
su -c "ip addr add 192.168.1.7 dev eth0"
```

Cambia la MTU (Unidad de Transferencia Máxima):

```
su -c "ip link set eth0 mtu 1480"
```

Configurar la ruta de red:

```
su -c "ip route add 172.16.1.0/24 via 192.168.1.5"
```

33.6.3. `route`

En enrutamiento es el proceso de seleccionar vías en una red a través de la cual enviar el tráfico de red. La tabla de enrutamiento muestra todas las rutas que se usan en el sistema. Se gestiona mediante la herramienta `route` (aunque ya hemos visto que con `ip` también se puede).

Siempre habrá una ruta por defecto, las demás serán para las que cumplan ciertas características. Esta puede obtenerse dinámicamente mediante DHCP o podemos configurarla manualmente. En debian y derivadas, esto se hace editando el parámetro `gateway` del archivo `/etc/network/interfaces`. En rhel, este archivo es `/etc/sysconfig/network`.

Visualizar la tabla de enrutamiento.

```
route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    202    0        0 eth1
loopback        *               255.0.0.0       U     0      0        0 lo
```

Añadir una ruta.

```
su -c "route add default gw 192.168.1.10 eth0"
route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.10    0.0.0.0         UG    202    0        0 eth0
default         192.168.1.1     0.0.0.0         UG    202    0        0 eth1
loopback        *               255.0.0.0       U     0      0        0 lo
192.168.1.0     *               255.255.255.0   U     202    0        0 eth1
```

Esto podría dejar sin conexión la red entera. Para arreglarlo, sólo hay que reiniciar el servicio de red.

33.6.4. Diagnostico de red

Se usan distintas herramientas para controlar que todo vaya correctamente o para averiguar que es lo que falla cuando algo falla. Se suelen usar las siguientes:

1. `ping` Envía paquetes al host que le diga, y si lo encuentra envía un reporte de vuelta con el tiempo de todo el proceso. También se incluye cuantos paquetes se han perdido, entre otras.

2. `traceroute` Sirve para mostrar la ruta de red a un destino, con los saltos que da hasta llegar ahí.

3. `mtr` Combina las herramientas `ping` y `traceroute` mostrando la salida siempre actualizada, cómo `top`.

4. `dig` Sirve para comprobar la funcionalidad de un DNS, ya sea el que usa el sistema u otro. Puede mostrar los registros que se quieran.

34. Firewall

El firewall o cortafuegos en castellano es el sistema encargado de monitorizar y controlar todo el tráfico de red. Estos pueden estar basados en hardware o en software. Pueden estar en ordenadores individuales, aunque es habitual que estén en los propios routers. Hay tres tipos de FW, cada uno más avanzado que el otro:

- Filtrado de paquetes: es de finales de los 80 y solo hacia eso. No tenia en cuenta el estado de la conexión o de donde salia el paquete.
- Filtrado de estado: formó parte de la siguiente generación, y analiza el estado de los paquetes, además de lo anterior. Es vulnerable a los DoS.
- Filtrado de capa de aplicación: tiene en cuenta el tipo de aplicación que está filtrando y lo que debería o no hacer. Permite bloquear comportamiento anómalo y dejar pasar el resto.

Existen dos grandes tipos de aplicaciones de firewall, las que tienen interfaz gráfica y las que no. Se verán las segundas, cómo son `iptables`, `ufw` o `firewall-cmd`.

34.1. `firewalld`

Es el Administrador Dinámico de Cortafuegos. Se permite usar niveles de confianza para las interfaces de red o conexiones. Este es un sustituto más moderno que las `iptables`, por lo tanto no deberían usarse juntas.

Este es un servicio, por lo que con se puede habilitar o deshabilitar cómo cualquier servicio. Si se tiene más de una interfaz de red, hay que activar el **ip forwarding**, es decir, el reenvió de paquetes. Esto se hace, de forma persistente, editando el archivo `/etc/sysctl.conf` y añadiendo `net.ipv4.ip_forward=1`. Luego hay que ejecutar `sudo sysctl -p` para evitar reiniciar y que se active la configuración.

Este programa trabaja con zonas, cada una de las cuales tiene un nivel de seguridad y un comportamiento predefinido para los paquetes. Estas zonas son:

- `drop`: Todos los paquetes entrantes son eliminados sin una respuesta.
- `block`: todas las conexiones entrantes son rechazadas, sólo se permiten las relacionadas con el sistema.
- `public`: no confía en ningún ordenador de la red, solo ciertas conexiones entrantes.
- `external`: se utiliza cuando la mascara de red está en uso, como en los routers. Es similar a `public`.

- `work`: Confía pero no mucho, solo se permiten algunas conexiones entrantes.
- `home`: confía en los demás nodos de la red, pero sigue sin permitir todas las conexiones entrantes.
- `internal`: similar a `work`.
- `trusted`: todas las conexiones son permitidas.

35. Resolución básica de problemas

Esta es una habilidad básica en la sysadmin, ya que muchas veces nos encontraremos problemas. Una parte importante de nuestro trabajo será analizar ese problema, diagnosticar de dónde viene (mala configuración, software, hardware, etc) y arreglarlo.

35.1. Técnicas básicas

Hay una serie de pasos a seguir, algunos de ellos deben repetirse de forma iterativa hasta solucionarlo. Un ejemplo seria el siguiente:

- Identificar y describir el problema
- Reproducir el problema
- Siempre intentar las cosas fáciles primero
- Descartar todas las causas posibles de una en una
- Cambiar una sola cosa a la vez, si eso no resuelve el problema, volverlo a su estado original
- Verificar logs del sistema

35.2. Problemas de red

Cuando hay un problema de red, hay que verificar lo siguiente:

- Configuración de la IP: Hay que ver si la interfaz está arriba y bien configurada.
- Conectividad: Comprobar que hay conexión a internet mediante `ping`. Es mejor apuntar a una IP en vez de un dominio primero, por si acaso hay un problema con las DNS.
- Ruta por defecto: Si no hubiese conexión, comprobar el enrutamiento con `route`.
- DNS: Comprobar con un `ping` a un dominio si está funcionando.

Si todo esto falla, seria recomendable comprobar, por ejemplo, que la controladora esté funcionando. Incluso, habría que comprobar que el cable de red esté conectado correctamente. A veces los problemas más tontos vienen por un cable desenganchado.

35.3. Integridad de archivos

Hay muchas formas de comprobar que los archivos de configuración y los binarios no están corruptos. En general esto depende de la distribución y de su gestor de paquetes, cómo ya se vio en el capitulo de gestión de paquetes.

Hay una herramienta genérica que detecta intrusiones, pero que también sirve para verificar la integridad de los archivos. Esta es `aide`, y se hace mediante `aide check`.

35.4. Problemas en el proceso de arranque

Saber por que etapas pasa el arranque del sistema es necesario si se quiere diagnosticar el problema cuando este falla. Suponiendo que pase la BIOS, algunos de los posibles estados de error son los siguientes:

- No aparece la pantalla del cargador de arranque: Este suele ser GRUB. Hay que mirar si la configuración de esta es correcta (si no es probablemente toque usar un live cd). Si no es algo del GRUB en si, lo más probable es que sea el disco duro lo que falla.
- El kernel no carga: El tan conocido **kernel panic**. Cuando esto pasa, puede que el kernel esté mal configurado o dañado, que no se le pasen los parámetros correctos o que el GRUB no carga el kernel correcto, entre otras.
- El kernel carga pero no monta la raíz: Esto suele deberse a que el GRUB está mal configurado, `/etc/fstab` está mal configurado o no hay soporte para el sistema de archivos de la raíz. Esto último, si se usa `initramfs`, puede deberse a que simplemente no está el módulo metido en este.
- Falla al arrancar `init`: Dado que este se encarga de muchas cosas, es difícil dar un posible diagnostico por que las posibilidades son demasiadas. Lo que hay que hacer es mirar los mensajes que se despliegan antes de que el inicio se detenga.

35.5. Corrupción y recuperación de sistemas de archivos

Habitualmente, el kernel hace comprobaciones de los sistemas de archivos montados al inicio del arranque, por lo tanto si detecta algún problema, lo intenta arreglar usando `fsck`, cómo ya se vio anteriormente. A veces simplemente se ha montado en modo lectura al encontrar un error, por lo que habría que remontarlo con permisos de escritura.

36. Rescate del sistema

Los medios de arranque de emergencia son útiles sobretodo cuando hay problemas en el arranque (cómo se vio en el anterior capitulo), ya que en ese momento estamos limitadas. Casi todas las distribuciones tienen un Live CD, es decir, una imagen que se puede grabar en un pendrive o CD y usar en cualquier ordenador sin instalarlo, pudiendo, además, usar las herramientas que estas traigan. Con cualquier imagen, en general, se puede salir del paso, ya que aunque

no traigan todas las herramientas que se necesiten, siempre se pueden instalar. Pero siempre es mejor usar directamente una imagen que ya las traiga, como puede ser **Knoppix**.

Cómo pasa al estar instalado, las Live CD son todas diferentes, por lo que no se puede decir mucho más que hay que arrancarlo por BIOS. Algunas de ellas arrancaran sin más, y otras tal vez hagan algunas preguntas o den distintas opciones a elegir.

Suponiendo que estemos dentro, el procedimiento es montar los discos duros en `/mnt/` y luego usar `chroot`.

36.1. Dispositivos USB de rescate

Apunte: No hay mucho que contar, aunque se empeñen en rellenar este capitulo. Hay demasiadas distribuciones distintas entre ellas cómo para decir algo genérico.

Las imágenes se "quemanüsando la herramienta `dd`, de la siguiente forma:

```
dd if=imagen.iso of=/dev/sdx
```

Siendo **x** el número asignado al USB. Luego hay que configurar la BIOS para que arranque el USB (hay que tener en cuenta que no todas lo soportan, sobretodo si son viejas) y arrancar.

36.2. Modo de emergencia

Este modo arranca el sistema de archivos como lectura solamente, no se ejecutan los scripts de `init` y casi nada está configurado.

Las ventajas enfrente el arranque **monousuario**, que se verá en el próximo apartado, es que al no arrancar el `init`, si el problema está en este, podremos analizarlo, además de que nos aseguramos de no perder información al montar la raíz cómo lectura.

Para entrar en este modo, se debe seleccionar la entrada desde el menú de arranque de GRUB y presionar la **e** para editarlo. Se agrega la palabra **emergency** sin más a la linea de comandos del kernel y luego se indica al sistema que arranque.

36.3. Modo monousuario

Es similar al anterior modo. Sus características son:

- **init** es iniciado
- Los servicios no se inician
- La red no se activa
- Se montan todos los sistemas de archivos que sea posible
- Se concede acceso root sin solicitar contraseña
- Se presenta una shell de manutención del sistema

Para usar este modo, hay que seguir el procedimiento del Modo de emergencia, pero cambiando la palabra **emergency** por **single**.

37. Licencia

En la medida que lo permita la ley, drymer ha renunciado a todos los derechos de autor y derechos afines o próximos a este trabajo.

Licencia