ADMINISTRACIÓN DE SISTEMAS

(((((((RESUMEN)))))))

Contents

Chapter 1

SEGURIDAD BÁSICA

1.1 CONTROL DE ACCESO

1.1.1 *Tipos de control de acceso*

- Discrecional (DAC) : Los objetos tienen propietarios, los cuales controlan sus permisos
- Obligatiorio (MAC) : Cualquier operación se contrasta con las reglas de la política del sistema
- Basado en roles (RBAC) : Combinación de DAC y MAC

1.1.2 *Control de acceso discrecional*

El control de acceso no está centralizado, sino que queda desperdigado en el código: entre llamadas del sistema, sistema de archivos...

Los objetos tienen propietarios (quienes crean el objeto), los cuales gestionan sus permisos de acceso. Quedan identificados por su **ID de usuario** (UID) y **de grupo** (GID). La propiedad de un proceso se divide entre el UID/GID **real** (RUID/RGID, quien lo ejecuta realmente) y **efectivo** (EUID/EGID, los permisos efectivos otorgados).

El **superusuario** (root) actua como propietario de todos los archivos, y realiza ciertas operaciones de administración. Su **UID** es **0**.

Normalmente el EUID y RUID coinciden. pero se puede modificar con los bits **setuid/setgid**. Al activarse, otorgan el EUID/EGID del propietario del objeto en vez del que ejecuta el fichero.

El **sticky bit** previene a los usuarios borrar/renombrar ficheros de los que no sean dueños dentro de un directorio.

Los permisos en el sistema de ficheros tienen el formato XXXX, donde cada X es un vector de 8 bits tal que:

- El primer campo indica bits especiales
- El segundo indica permisos del usuario propietario
- El tercero, del grupo propietario
- El cuarto, del resto de usuarios

Permiso	Símbolo	Valor	Ej. (Textual)	Ej. (Número)
Lectura	r	4	-r———	0400
Escritura	w	2	——w—-	0020
Ejecución	x	1	———x	0001
setuid	s	4	—s———	4000
setgid	s	2	——s—	2000
sticky bit	t	1	———t	1000

Nota: Los bits especiales sustituyen el permiso de ejecución visualmente en notación textual

1.1.3 *Autenticación*

La autentiación se encarga de verificar la identidad de un usuario para poder determinar si puede realizar una operación.

La colección de bibliotecas **PAM** (Pluggable Authentication Module) permite modificar la autentiación de las aplicaciones sin necesidad de recompilar, separando autenticación de escalación de permisos.

Los archivos de configuración se encuentran en:

- /etc/pam.conf
- /etc/pam.d/

Sus módulos se encuentran en:

- /lib/security

Tareas PAM	Descripción
auth	Autenticación
account	Capacidades de cuenta
session	Procedimientos de arranque/apagado
password	Mantenimiento de contraseña

Flags PAM	Descripción
required	Obligatorio, comprueba otros módulos incluso tras fallo
requisite	Obligatorio, regreso en caso de fallo
optional	Usados en caso de que otros módulos no devuelvan fallo o éxito
sufficient	Válido si no ha habido fallos
include	Inclusión de otros archivos de configuración

Módulos PAM	Descripción
pam_unix	Autenticación contra contraseñas hasheadas
pam_limits	Limites de recursos
pam_rootok	Comprueba que usuario es root
pam_cracklib	Prueba seguridad de contraseña, p.e. contra ataques de diccionario
pam_passwdqc	Alternativa a módulo anterior que incluye, p.e. generación de contraseñas
pam_permit	Siempre dice si
pam_deny	Siempre dice no
pam_warn	Aviso a log
pam_motd	Mensaje del día
pam_securetty	Restringe logins a solo terminales seguras
pam_wheel	Verifica que usuario pertenece a wheel para recibir permisos de root
pam_winbind	Autenticación contra dominios Windows
pam_nologin	Previene a otros usuarios acceder al sistema si existe /etc/nologin (salvo root)

Ejemplos:

```
# Formato: tarea flag módulo [parámetros]

# Permitir a todos los usuarios su a root sin contraseña (/etc/pam.d/su)
auth        sufficient      pam_permit.so

# Prohibe usar su a cualquier usuario
auth        requisite       pam_deny.so

# Deshabilitar login directo a root (/etc/pam.d/login)
auth        required        pam_securetty.so

# Forzar contraseñas fuertes
password    requisite       pam_passwdqc.so     min=12,10,10,8,6 retry=3
```

LDAP (Lightweight Directory Access Protocol) es un protocolo a nivel de aplicación de acceso a un servicio de directorio que sustituye las consultas a /etc/passwd y /etc/shadow a consultas a servidores LDAP a través de PAM.

1.2 CUENTAS DE USUARIO

Todos los usuarios UNIX requieren tener una cuenta, compuesta por:

- Nombre de usuario (login/username)
- Contraseña (passwd)
- Grupo(s)
- UID, GID
- Directorio *home*
- Login shell
- Ficheros de inicio

id muestra el UID, GID del grupo principal y GIDs de grupos secundarios de un usuario

1.2.1 *Tipos de usuario*

- Cuentas normales
- Cuentas de administrador (root)
- Cuentas especiales para servicios (nobody, lp, bin...)

1.2.2 *Ficheros de cuentas de usuario*

Fichero	Descripción
/etc/passwd	Contiene cuentas de los usuarios del sistema y contraseñas si no se usa shadow. Legible por todos los usuarios
/etc/shadow	Contiene contraseñas cifradas de los usuarios. Solo accesible por root
/etc/group	Contiene grupos del sistema
/etc/gshadow	Equivalente de shadow para grupos. No se suele usar

Formato:

```
# /etc/passwd
login:passwd:UID:GID:info:home-dir:shell

# login: 32 caracteres, recomendado 8
# passwd: Contraseña cifrada, x si se encuentra en shadow
# UID: Por convención entre 1000-30000
# info: Nombre real e información del usuario

# /etc/group
name:passwd:GID:login-list

# /etc/shadow
login:passwd:a:b:c:d:e:f:g

# a: Fecha (dias a partir de 1/1/70 [epoch]) de último cambio de contraseña.
#       0 fuerza cambio
# b: Dias para poder realizar cambio
# c: Dias antes de expiración de contraseña
# d: Dias antes de expiración para avisar al usuario
# e: Días despues de expiración para desactivar cuenta
# f: Días para inhabilitar cuenta
# g: Reservado
```

1.2.3 *Ficheros de inicio*

Residen en $HOME. Su uso depende de cada programa e intérprete. Suelen comenzar por '.'. Al crear nuevo usuario, se obtienen de **/etc/skel**.

Login shell (usuario entra al sistema)

Fichero	Descripción
/etc/profile	Leido automáticamente por bash al haacer login (normalmente tty). Configura variables generales del sistema
$HOME/.bash_profile	Normalmente solo indica los archivos a leer a continuación
$HOME/.bash_login	Comandos a ejecutarse al iniciar sesión
$HOME/.profile	Equivalente a /etc/profile, especifico para el usuario
$HOME/.bash_logout	Comandos a ejecutar al cerrar sesión

No-login shell (cuando no es necesario usuario/contraseña o sustituye a otro usuario con su)

Fichero	Descripción
/etc/bash.bashrc	/etc/profile para no-login shells
/etc/bashrc	Leido solo por bash tras bash.bashrc
$HOME/,bashrc	Fichero de personalización individual, también para login shells. Prompt, aliases, programas...
$HOME/.bash_aliases	Opcional para separar aliases de .bashrc
$HOME/.bash_history	Historial de comandos

1.2.4 *Operaciones sobre usuarios*

Creación

- Añadir nuevo usuario en /etc/passwd (utliizando **vipw**)
- Añadir x y editar /etc/shadow (utilizando **vipw -s**)
- Añadir grupo/a grupos (**vigr**)
- Crear *home* y copiar ficheros de *skel*
- Fijar nuevo propietario (**chown**), grupo (**chgrp**) y permisos (**chmod**) del directorio
- Fijar contraseña (**passwd**, -e para expirarla y obligar a cambiarla al iniciar sesión)

Comandos de bajo nivel: **useradd**, **usermod**, **groupadd**, **groupmod**

En lotes: **newusers** (formato de *passwd*, contraseñas no encriptadas), **chpasswd** (actualizar contraseñas)

Para el usuario: **passwd**, **chsh**, **chfn**, **newgrp**

De alto nivel: **adduser**, **addgroup**

Eliminación

- Inhabilitar cuenta (**temporal**: shell a /bin/false o /usr/bin/nologin, * en contraseña en /etc/passwd; **definitivo**: borrar entradas en passwd/shadow)
- Backup de *home*
- Eliminar ficheros de usuario

Comandos: **userdel**, **groupdel**

1.2.5 *Cambio de usuario*

su ejecuta otra shell bajo otro usuario (por defecto root), cambiando UID/GID. Pide contraseña del usuario destino

sudo ejecuta un programa bajo la identidad de otro usuario. Se configura su comportamiento en /etc/sudoers

Ejemplos:

```
# /etc/sudoers
# Indica que usuario puede ejecutar que comando en que máquina
user_name   hostname=[ETIQ:]command,...
%group_name hostname=[ETIQ:]command,...

# Root puede ejecutarlo todo
root        ALL=ALL
# Alias para varios programas
Cmnd_Alias  Dump=/usr/etc/dump,/usr/etc/restore
```

```
# Luis lo puede ejecutar todo en CS y los programas de DUMP en FISICAS
luis        CS=ALL, FISICAS=DUMP
# Fernando puede ejecutar tcpdump en CS
fernando    CS=/usr/local/bin/tcpdump
# María puede usar adduser en cualquier máquina sin necesidad de contraseña
maria       ALL=NOPASSWD:/usr/local/bin/adduser
```

1.3 CRIPTOGRAFÍA BÁSICA

La criptografía es la ciencia de escribir en código secreto. Sus principales funciones son:

- Proveer un método de **autenticación**
- Asegurar la **privacidad y confidencialidad** de un mensaje, especialmente si el medio no es seguro
- Afianzar la **integridad** del mensaje, que no haya sido alterado
- **No repudio**, es decir, que el emisor sea el que realmente ha enviado un mensaje

1.3.1 *Criptografía de clave secreta / simétrica*

Utiliza una clave única tanto para cifrar como descifrar. La mayor dificultad se encuentra en que el emisor y receptor conozcan la misma clave sin comprometerla.

Dos esquemas:

- **Cifras de flujo**: Operan sobre un bit/byte en cada paso, junto a un mecanismo de feedback que modifica constantemente la clave, generando cifras diferentes para el mismo mensaje en distintas iteraciones
- **Cifras de bloque**: Se cifra un bloque completo de datos con la misma clave. El resultado es consistente

Ejemplos:

- **Data Encription Standart** (DES, 1970): Cifrado de bloques de 64 bits. Vulnerable y en desuso
- **Advanced Encryption Standard** (AES, 2001): Cifrado en bloques de hasta 256 bits.

1.3.2 *Criptografía de clave pública / asimétrica*

Consiste en dos claves matematicamente relacionadas tal que no se pueda derivar una de la otra (fuciones de una vía). Una es usada para descifrar y otra para cifrar. Además, una es una **clave pública** que todos pueden conocer, y una **clave privada** que no se da a conocer a nadie.

Ejemplo: Bob quiere enviarle un mensaje a Alice:

1. Bob cifra su mensaje con la clave pública de Alice
2. Alice descifra el mensaje usando su clave privada, y es la única que lo puede leer

Ahora: Alice quiere asegurarse que el mensaje que le envía a Bob es suyo:

1. Alice encripta el mensaje con su clave privada
2. Bob descifra el mensaje con la clave pública de Alice, por lo que sólo la ha podido enviar ella

Esto se conoce como un reto de cifrado, que consiste en devolver un número aleatorio (nonce) firmado para autenticar un par de una conexión

Ejemplos:

- **Rivest-Shamir-Adleman** (RSA): Se utiliza un producto grande de dos primos y aritmética modular
- **Digital Signature Algorithm** (DSA): Basado en exponenciación modular y logaritmia discreta
- **Elliptic Curve Cryptography** (ECC): Utiliza álgebra de curvas elípticas

1.3.3 *Funciones hash*

Son funciones de una vía que devuelven un valor de longitud fija indescifrable a partir de los datos originales. Sirven como firmas electrónicas, como por ejemplo las contraseñas de los sistemas operativos.

Ejemplos: - **Message Digest** (MD): Hashes de 128 bits - **Secure Hash Algorithm** (SHA): Hashes de tamaño variable

1.3.4 *¿Cómo se combinan los diferentes métodos de encriptación?*

Queremos establecer una comunicación rápida y segura entre Alice y Bob:

1. Alice genera una **clave de sesión aleatoria** mediante criptografía de clave privada
2. Alice cifra la clave de sesión usando la clave de sesión de Bob y se la envía
3. Bob descifra el mensaje usando su clave privada. Alice sabe que solo la puede recibir Bob, pero él no sabe que es de Alice
4. Alice calcula el valor hash del mensajem lo cifra con su clave privada y envía a Bob
5. Bob desencripta el hash con la clave pública de Alice y obtiene el hash de la clave de sesión que recibió antes
6. Si coinciden, sabe que el mensaje es de Alice (porque el hash coincide y lo desencriptó con su clave pública)

Para el uso en contraseñas, se suelen salar (**salt**) añadiendo una cadena adicional para evitar que la misma cadena tenga el mismo hash. Protege de ataques de diccionario (hashes con asociación conocida) y tablas arcoiris (hashes precomputados)

1.3.5 *Modelos de confianza*

La confianza permite asegurar que el destinatario es el que dice ser y no está siendo suplantado. Algunos modelos son:

- **Kerberos** conforma una tercera parte de confianza que distribuye claves mediante un servidor de autenticación y un servidor de tiquets
- **Pretty Good Privacy** (PGP), donde los usuarios guardan su propio conjunto de claves públicas
- **Certificados**, que permiten a un conjunto de terceras partes autenticarse entre ellas y entre los usuarios

Certificados y autoridades certificadoras

Los **certificados** son documentos digitales que:

- **Establecen la identidad** (clave pública) del propietario
- Realizan una **asignación de autoridad** (el qué y qué no puede hacer el propietario)

Normalmente incluyen:

- Clave pública
- Nombre
- Fecha de caducidad
- Nombre de la autoridad certificadora emisora
- Número de serie
- Firma digital

Las ***autoridades de certifiación*** son repositorios de claves públicas y pueden ser cualquier agencia que emita certificados. Se requiere obtener la clave pública de un destinatario de su CA, la cual mantiene relaciones de confianza con otras CA.

1.3.6 *SSL/TLS*

Secure Socket Layer y **Transport Layer Security** proveen comunicaciones seguras independientes de la aplicación sobre protocolos de Internet, especialmente HTTPS. Se recomienda mínimo TLS 1.2 o superior.

1.4 SSH

Secure Shell (SSH) es un reemplazo a rlogin, rcp y telnet. Utiliza autenticación de clave pública (RSA, DSA y ED25519) y cifra la comunicación. Es un proyecto de código abierto (OpenSSH).

1.4.1 *Modos de autenticación*

- **Clave pública**: El cliente manda un mensaje al servidor con su clave pública, firmado con la clave privada del cliente. El servidor comprueba que la clave es aceptable y la firma correcta
- **Contraseña**: El cliente envía un mensaje con una contraseña en texto plano encriptada solo por TLS
- **Basado en anfitrión**: Un anfitrión para varios clientes provee autenticación para todos ellos. El cliente envía una firma creada con la clave privada del anfitrión al servidor.

1.4.2 *SSHD (servidor)*

Métodos de autenticación de logins:

- (Inaceptable) Si el nombre del host remoto se encuentra en **$HOME/.rhost**, entra sin comprobar contraseña
- Usar clave pública almacenada en **/etc/ssh/ssh_known_hosts** para verificar identidad del cliente. Si el host remoto conoce la clave privada, en **/etc/ssh/ssh_host_key**, puede entrar sin contraseña
- (Mas seguro) Mediante claves asimétricas, el usuario tiene acceso a su clave privada
- Método usuario/contraseña mediante telnet, pero con conexión cifrada

Se puede configurar en /etc/ssh/sshd_config:

```
# Configuración recomendada
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication yes
PermitRootLogin no
```

1.4.3 *Conexión*

La primera vez que un cliente se conecta a un servidor, recibe una clave pública. Si la acepta, se almacena en **$HOME/.ssh/known_hosts**.

SSH además permite ejecutar una única órden remotamente. Nota:

- ” expande las variables en la máquina remota
- “” expande las variables en la máquina local
- Cuidado con cómo estan delimitados los comandos para SSH y los comandos a ejecutar localmente

SSH lee de entrada estandar. Es necesario redirigir stdin a /dev/null (también ssh -n) para su funcionamiento en scripts

scp permite copiar archivos de la maquina local a la remota y viceversa

1.4.4 *Generación de claves*

Se utiliza **ssh-keygen**. Se puede añadir una contraseña tradicional para protegerla localmente (passphrase).

Para poder conectarse de manera remota sin contraseña, se añade la clave pública del cliente a la máquina remota en **$HOME/.ssh/authorized_keys**. Se puede realizar mediante **ssh-copy-id**

$HOME/.ssh requiere que el usuario sea dueño y sea de su grupo, directorio con permisos 700 y ficheros con permisos 600.

1.4.5 *Tunelamiento*

SSH también permite establecer conexiones TCP seguras mediante SSL (**ssh -L localport:remotehost:remoteport**)

Ejemplo:

```
# Establece a través del servidor ssh home una conexión entre el puerto local
# 8080 y el puerto remoto 80 de remote.service.com
ssh -L 8080:remote.service.com:80 joeuser@home
```

Chapter 2

CONFIGURACIÓN BÁSICA

2.1 ARRANQUE Y PARADA DE SO

2.1.1 *Proceso de arranque*

El proceso de arranque se asegura que el sistema está listo para ser usado por los usuarios.

Un arranque tiene dos fases: **arranque hardware** y **arranque del sistema operativo**

Arranque hardware

El programa de arranque se encuentra en una ROM en la placa base, y se activa al recibir una señal eléctrica.

1. El programa de arranquecarga los valores por defecto en los registros
2. El programa conprueba las características y el funcionamiento del hardware
3. El programa lee y almacena en memoria el cargador del SO
4. Pasa el control al cargador del SO

Cargador (bootloader)

El cargador se encuentra en los primeros sectores del disco (**Master/Volume Boot Record**). La ubicación y tamaño del cargador está preacordada para poder iniciar varios SO.

5. El cargador transfiere el control al núcleo del SO

El cargador mas común es el **Grand Unified Bootloader** (GRUB). Se instala normalmente en el MBR y permite decidir SO a arrancar. Se configura en **/boot/grub/grub.cfg**. Además de un menú de selección, ofrece un editor de órdenes de arranque y una shell interactiva (del propio GRUB)

Ejemplo:

```
# Opción por defecto
default=0
# Tiempo de espera
timeout=10
# Contraseña
password --md5 $1$4hKKr1$LvSjN89PmeeHXBljr13yq0
# Fondo de pantalla
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
# Partición donde se encuentra el núcleo
root (hd0,0)
# Núcleo y parámetros
kernel /boot/vmlinux-2.6.37 ro root=LABEL=/
# initrd
initrd /boot/initrd-2.6.37.img
# Partición a usar
rootnoverify (hd0,2)
# Transferir control
chainloader +1
```

Núcleo (kernel)

Para evitar núcleos muy grandes, la mayoría de las opciones se compilan como módulos y se cargan cuando necesitan. Estos se encuentran definidos en **initrd_version.img**

1. El núcleo comprueba el hardware del sistema
2. EL núcleo se prepara para ejecutar el sistema: inicializa tablas internas, crea estructuras de datos, etc...
3. El núcleo carga initrd y le transfiere el control
4. Initrd carga módulos del sistema y devuelve el control
5. Crea el proceso Init y le transfiere el control

Init (PID 1)

Init se encarga de terminar el proceso de arranque y dejar el sistema en modo multiusuario mediante una serie de scripts con las acciones a realizar. Los mensajes de arranque quedan grabados en /var/log/messages (accesible mediante **dmesg**).

11. Init chequea sistemas de ficheros: monta los sistemas permanentes y activa areas de intercambio (*swap*)
12. Init activa demonios (*daemons*) y funcionalidades de red
13. Borra archivos temporales
14. Habilita el login a los usuarios

Hay dos programas init generalmente usados: **sysv-init** y **systemd**.

Arranque System V (initd)

Niveles de ejecución

Originalmente eran tres: **parado**, **modo monousuario** y **modo multiusuario**. En total, son:

- 0: Preparar apagado de máquina: si puede apagarse por si solo, que lo haga
- 1: Modo administrador del sistema: tododo los FS montados, pequeño conjunto de procesos del modo monousuario
- 2: Modo multiusuario: No compartir recursos
- 3: Modo multiusuario: Compartición habilitada
- 4: Modo multiusuario: A definir por el usuario
- 5: Parar y rearrancar el sistema
- s/S: Modo monousuario: solo montado FS raíz

Se puede modificar el nivel de ejecución con **/sbin/telinit** y el nivel de ejecución en curso (y previo) con **runlevel**.

inittab

SystemV organiza y generaliza estos y otros estados a través de **/etc/inittab**.

Cada línea es una entrada con cuatro campos:

1. Identificador (dos letras): a cada terminal le corresponde un fichero dispositivo /dev/ttyXX en el que debe iniciarse un proceso getty
2. Nivel de ejecución. Puede tener varios
3. Acciones: Como y cuando ejecutar el proceso
4. Path de acceso al programa

Acción	Descripción
respawn	Reinicia el proceso si acaba
wait	Init esperará a que termine el proceso antes de leer la siguiente entrada
once	Inicia el proceso solo una vez al alcanzar su estado de ejecución
boot	Ignora nivel de ejecución y lo ejecuta en el arranque
bootwait	Ejecutar en arranque, esperar a que acabe
off	Ignorar entrada
initdefault	Especifica nivel por defecto
sysinit	Ejecutar durante arranque, máxima prioridad
powerwait	Ejecutado al recibir señal SIGPWR (problema con alimentación), espera a que acabe
powerfail	powerwait sin esperar
ctrlaltdel	Ejecutado al recibir señal SIGINT (CTRL + ALT + SUPR)

Ejemplo:

```
id:2:initdefault:
si::sysinit:/etc/init.d/boot
~~:S:wait:/sbin/sulogin
10:0:wait:/etc/init.d/rc 0
11:1:wait:/etc/initd.rc 1
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
```

Modo monousuario

Este modo sirve para tareas administrativas que requieran control completo del sistema. Se monta solo el FS raíz, solo estan disponibles órdenes de / y solo estan ejecutandose los servicios imprescindibles.

Init crea el shell por defecto (/bin/sh) como root (ocasionalmente /sbin/sulogin para pedir contraseña).

Se accede manualmente a través de GRUB (al no tener ninguna proteción se recomienda añadir contraseña al cargador) o automaticamente si hay problemas no solucionables en el arranque.

Modo multiusuario

El modo multiusuario:

1. Comprueba el FS raíz con fsck (si no ha sido desmontado correctamente o periodicamente)
2. Monta / en modo lectura-escritura
3. Comprueba el resto de FS y los monta
4. Activa particiones de intercambio (**swapon -a**)
5. Activa cuotas de disco (**quotacheck/quotaon -a**)
6. Lanza *daemons* (crond, atd, cupsd, syslogd, etc. . .)
7. Activa red y lanza *daemons* de red (rcpbind, named, routed, nfsd, etc. . .)
8. Limpia archivos temporales (/tmp)
9. Crea terminales (**mingetty/getty**), y borra el fichero **/etc/nologin** si fuera necesario.

Ficheros de inicialización

Son programas que se ejecutan tras la inicialización del núcleo antes de que los usuarios puedan iniciar sesión.

Normalmente se utiliza las entradas de *inittab* para arrancar estos programas. Este ejecuta **/etc/init.d/rc**, que a su vez ejecuta en orden alfabetico los ficheros en **/etc/rcLEVEL.d**.

Un fichero K mata procesos y detiene subsistemas; viceversa con S. Los ficheros son enlaces simbólicos a programas en **/etc/inid.d** .

Ejemplo:

```
$> ls /etc/rc1.d
K01xdm          K18netbase  K20xfs          K50netatalk
K11cron         K20acct     K20xntp3        K89atd
K12kerneld      K20lpd      K25netstd_nfs   K90sysklogd
K15netstd_init  K20quota    K30netstd_misc  S20single
```

/sbin/chkconfig permite configurar los demonios a lanzar o finalizar a cada nivel.

Arranque systemd

systemd se encarga de:

- Ser el gestor del sistema y servicios
- Ser una plataforma para desarrollar software
- Permitir la interacción entre aplicaciones y *kernel*

Diseñado como reemplazo a System V, ofrece muchas funcionalidades (aunque sea a consta de la *filosofía UNIX*), como por ejemplo:

Servicio	Descripción
consoled	Consola del sistema
journald	Registro de eventos
logind	Gestión de login
networkd	Configuración de red

Servicio	Descripción
timedated	Fecha y hora
udevd	Gestión de dispositivos (/dev), firmware, ...
libudevd	Librería de interacción con udevd

Unidades

Un *unit file* es un fichero de configuración de un servicio. Se cargan desde **/etc/systemd/system** (administrador, enlaces al segundo) y desde **/usr/lib/systemd/system**

Un *unit* puede ser:

Unidad	Notas
.service	Arranque/parada/reinicio, dependencias...
.socket	De red, IPC, buffer FIFO
.device	Dispositivos a gestionar por udev o sysfs
.mount	Punto de montaje a gestionar
.automount	Punto montaje automatizado
.swap	Área de intercambio
.target	Punto de sincronización entre unidades
.path	Para activación por path
.timer	Temporizador para arrancar unidades
.snapshot	Reconstruye estado del sistema tras realizar cambios
.slice	Gestión de recursos
.scope	Información de buese de interfaces

Algunas de las directivas de sección disponibles son:

Directiva	Descripción
Description=	Describe funcionalidad de la unidad
Documentation=	URIs para documentación (paginas man/web)
Requires=	Unidades dependencia obligatorias
Wants=	Dependencias no obligatorias
BindsTo=	Requires donde la unidad actual para si la dependencia para
Before=	Activar antes de ciertas unidades
After=	Activar despues de ciertas unidades
Conflicts=	Detiene unidades que no pueden ejecutarse simultneamente
Condition...=	Prueba condiciones antes de arrancar unidad. Si no se cumplen, se omite unidad
Assert...=	Incumplir condición implica fallo
WantedBy=	La unidad es dependencia opcional de otra
RequiredBy=	La unidad es dependencia oblligatoria de otra
Alias=	Nombre de unidad alternativo
Also=	Habilitación/deshabilitación en conjunto
DefaultInstance=	Unidad por defecto para unidades plantilla
Type=	Define tipo. Ver a continuación

Una unidad puede ser del tipo:

Tipo	Descripción
simple	Por defecto, el proceso principal se especifica en la línea inicial
forking	El servicio es hijo de otro
oneshot	Proceso con vida corta, esperar antes de continuar
dbus	Toma nombre en dbus, continúa a siguiente unidad
notify	Envía una notificación al terminar de arrancar, y systemd prosigue
idle	Espera a que el resto de tareas arranquen

Ejemplo:

```
# /etc/systemd/system/sshd.service -> graphical.target
```

Control de servicios

Se puede controlar los servicios mediante **systemctl**

```
systemctl list-units [--type service --all]
systemctl {start|stop|restart|reload|enable|disable|kill|status|is-active} [name.service]
```

Se pueden ver el árbol de procesos de un servicio con **systemd-cgls** y el estado y eventos de un servicio con **journalctl**.

Archivos de configuración

Fichero	Descripción
/etc/hostname	Nombre de host del sistema
/etc/vconsole.conf	Mapa de teclado y fuente de la consola
/etc/locale.conf	Idioma y *locale* del sistema
/etc/modules-load.d/*.conf	Carga de módulos estática en el arranque
/etc/sysctl.d/*.conf	Parámetros sysctl extra para **/etc/sysctl.conf**
/etc/tmpfiles.d/*.conf	Archivos de configuración creados/eliminados durante arranque/ejecución
/etc/binfmt.d/*.conf	Registro de formatos binarios adicionales
/etc/os-release	Estandarización de archivos de ID de distribución
/etc/machine-id	ID de máquina (reemplazando archivo ID de máquina de D-Bus)
/etc/machine-info	Información de metadatos sobre el host para systemd-hostnamed

2.1.2 *Proceso de parada*

El proceso de parada deja el sistema en un estado consistente. Normalmente (**shutdown**):

1. Cinco minutos antes, se crea **/etc/nologin** para prevenir inicios de sesión
2. Se notifica a los usuarios, se previene la ejecución de **getty**. Se almacena la hora de parada en **/var/log/wtmp**
3. Envia SIGTERM a procesos de ejecución
4. Se paran los *daemons*
5. Se expulsa a usuarios conectados
6. Envia SIGKILL a procesos en ejecución
7. Actualizaciones al FS pendientes con **sync**
8. Según tipo de apagado, se apaga, reinicia, o cambia a modo monousuario

2.1.3 *Caidas del sistema y problemas en el arranque*

Algunas razones de caidas del sistema son

- Fallos y errores irrecuperables de hardware
- Fallos de luz
- Temperatura
- Problemas entrada/salida
- Problemas con sistema de ficheros

Y de problemas de arranque:

- Fallo de hardware
- Incapacidad de leer el FS
- Areas fuera del FS del disco dañadas (tabla de particiones)
- Incopatibilidad de hardware
- Errores de configuración del sistema

2.2 CONFIGURACIÓN BÁSICA DE RED

2.2.1 *Conceptos básicos*

Los estandares están documentados por los **RFC** (Request for Comments)

Capa de enlace

Opera en el enlace al que está conectado fisicamente el host. Diferentes estandares (Ethernet, WiFi, Token Ring...) y disposiciones (malla, anillo, bus, estrella...)

Capa de red

El protocolo de Internet (**IP**) identifica los nodos en la red a los que encaminar datagramas.

IPv4 consiste en una dirección de 4 bytes (xxx.xxx.xxx.xxx). Se organizó originalmente en 5 clases, pero ahora todas son sin clase (**CIDR**) y se utiliza máscaras de red para separar el identificador de red de un nodo (Ej. 155.210.154.15/16, la red es 155.210.0.0 y los hosts se identifican con los últimos 16 bits).

Para **encaminar** (determinar la ruta de una red IP) se puede utilizar información estática (tablas) o dinámica (demonios). Normalmente se utiliza **OSPF** (Open Shortest Path First) dentro de una red y **BGP** (Border Gateway Protocol) entre redes.

Capa de transporte

Se encarga de asegurar la entraea de datos a las aplicaciones. Son fundamentales los protocolos **TCP** (Transmission Control Protocol, *streams*) y **UDP** (User Datagram Protocol, datagramas).

TCP asegura la recepción en orden para mantener una conexión, mientras que UDP no asegura recepción para ser más ligero.

Capa de aplicación

Especifican los protocolos e interfaces que las máquinas utilizan para comunicarse. Por ejemplo, el **DNS** (Domain Name System) asigna correspondencias entre IP y hostnames.

2.2.2 *Orgnaización de red en Linux*

Servicios y puertos

Un **servicio** es una aplicación basada en un **protocolo** de la capa de aplicación (http, tbp, ftp...).. Un **puerto** es un número identificador de una conexión lógica por red. Los primeros 1024 puertos estan reservados para servicios bien conocidos (en **/etc/services**). Un servicio queda identificado por la tupla protocolo-puerto.

Daemons

Un ***daemon*** (demonio) gestionan servicios. Ejemplos son sshd, dhcpd, sendmail, httpd...

inetd es un superdemonio que escucha en múltiples puertos y lanza otros demonios cuando son requeridos. Se encuentra su configuración en **/etc/inetd.conf**

Servicios DNS

named/BIND es un servidor estandar para publicar registros y resolver consultas DNS.

dnsmasq actua como una cache de DNS para un nodo local, ignorando el acceso al servidor DNS. Responde a peticiones DHCP relativas al mapeo de aliias a direcciones IP.

Ambos acceden a **/etc/hosts** para sus consultas.

Soporte de red

Linux soporta multidud de protocolos y **tarjetas de red** / **NIC** (Network Interface Card). A cada tipo de NIC le corresponde un dispositivo de comunicacion distinto, ethX (Ethernet), trX (Token Ring), pppX (PPP), slX (SLIP).

Ademas existe el dispositivo loopback (lo, 127.0.0.1) al propio dispositivo.

Los dispositivos se pueden consultar en **/sys/class/net**. Para interactuar, quedan mapeados en **/dev**.

2.2.3 *Instalación de un nuevo nodo a una red*

1. Instalar fisicamente la tarjeta de red, configurar y conectar a la red existente
2. Instalar el software de red, recompilar núcleo/módulo
3. Obtener información de red necesaria
   - Dirección IP y nombre del nodo
   - Dominio de red
   - Dirección IP de los nodos servidores DNS

    - Máscara de subred
    - Dirección de red y *broadcast*
4. Editar los ficheros de configuración y de arranque (dos alternativas)
    - Comandos **ifup**, **ifdown**, **ifquery** y fichero **/etc/network/interfaces**
    - NetworkManager
5. Notificar al exterior
6. Probar la configuración de red

2.2.4 *Configuración con ifupdown y /etc/network/interfaces*

```
# Loopback
auto lo
iface lo inet loopback

# Ejemplo de configuración dinámica
auto eth0
iface eth0 inet dhcp
   hostname turza

# Ejemplo de configuración estática
iface eth0 inet static
   addeess 192.168.0.3
   netmask 255.255.255.0
   broadcast 192.168.0.255
   network 192.168.0.0
   gateway 192.168.0.1

# Para soporte WPA/WPA2 es necesario el paquete <<wpasupplicant>>
allow-hotplug ath0
iface ath0 inet dhcp
   wpa-ssid wifi_casa
   wpa-psk 000102030405060708090
      a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
```

net-tools (obsoleto, no persistente)

- **ifconfig** : Configuración de interfaz de red (sustituido por **ip**)
- **route**: Configuración de enrutamiento, pudiendo añadir, eliminar y modificar rutas
- **netstat**: Información de la red
- **ifup/ifdown**: Activa/desactiva una interfaz de **/etc/network/interfaces**
- **iwconfig**: Configura una red inalámbrica

Activación de encaminamiento

ip_foward controla si el núcleo puede encaminar paquetes entre interfaces. Se puede consultar el estado con el comando **cat /proc/sys/net/ipv4/ip_foward**. Se puede habilitar temporalmente (**sysctl -w net.ipv4.ip_foward=1**) o permanente (añadir a **/etc/sysctl.conf net.ipv4.ip_foward=1** y ejecutar **sysctl -p /etc/sysctl.conf**).

Rutas permanentes

```
auto eth0
iface eth0 inet static
   address 192.168.1.2
   netmask 255.255.255.0
   up route add -net 172.16.0.0 netmask 255.255.0.0 gw 192.168.1.1
```

2.2.5 *NetworkManager*

Es un demonio que trata se simplificar y automatizar la configuración de interfaces de red. Se puede configurar en **/etc/NetworkManager/NetworkManager.conf**:

```
# keyfile es el plugin genérico válido para todo tipo de conexiones
# Se puede definir interfaces a no gestinar con unmanaged-devices
```

```
unmanaged-devices=interface-name:eth0
unmanaged-devices=mac:00:22:68:1c:59:b1
```

Se interactua con sus *front-ends*:

- **nm-connection-editor**: Aplicación gráfica para entornos de ventanas
- **nmtui**: Aplicación de terminal con menús
- **nmcli**: Aplicación para línea de comandos

```
# Muestra estado general de NetworkManager
nmcli general status

# Todas las conexiones
nmcli connection show

# Visualizar dispositivo
nmcli dev status

# Activar una conexión Ethernet
# Dinámica
nmcli connection add type ethernet con-name nombre ifname interfaz

# Estática
nmcli conncetion add type ethernet con-name nombre ifname interfaz ip4 dir gw4 dir_gw
nmcli connection up nombre

# Añadir DNS
nmcli con mod nombre ipv4-dns "155.210.12.33"

# Parar una conexión Wi-Fi
nmcli dev wifi list
nmcli connection add type wifi ssid eduroam con-name nombre ifname interfaz
nmcli con modify nombre wifi-sec.key-mgmt wpa-psl
nmcli con modify nombre wifi-sec.psk unizar
nmcli radio wifi on

# Añadir ruta estática
nmcli con add type ethernet con-name micon0 ifname eth0 ip4 10.0.0.1/16 \
   gw4 10.0.0.254
nmcli con modify micon0 +ipv4.routes "172.31.0.16/16 10.0.0.254"
```

2.2.6 *iproute2*

```
# Ver enlaces
ip link list

# Tablas ARP
ip neigh show

# Direcciones IP
ip add show (dev ...)

# Rutas
ip route show

# Reglas de rutas
ip rule list

# Añadir ruta estática
ip route add 172.31.0.0/16 via 10.0.254 dev eth0
```

2.2.7 *DHCP*

Es un protocolo que permite a un servidor prestar parámetros de red a nodos. Usa puertos 67 y 68 sobre UDP. Un servidor almacena la información, y los clientes se conectan periodicamente al servidor por broadcast y cargan los datos recibidos.

Se puede configurar un servidor en **/etc/dhcpd.conf**

```
# Nombre de Dominio
option domain-name "midominio.com";
# Servidores de nombres
option domain-name-servers 10.0.2.3, 193.14.7.9;
# Tiempo por defecto que dura una asignación
default-lease-time 600;
# Duración máxima de una asignación
max-lease-time 7200;
# Máscara de red
option subnet-mask 255.255.255.0;

subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.10 192.168.0.20;
   option broadcast-address 192.168.0.255;
   option routers 192.168.0.1;
}

host marte {
   hardware ethernet 52:54:00:12:34:70;
   fixed-address marte.mired.com;
}
```

En **/etc/default/dhcp** se encuentra la interfazz que sirve las DNS, y en **/var/lib/dhcp/dhcp.leases** estan las IP asignadas. Se puede consultar un DNS con **dig**.

2.2.8 *Transición de IPv4 a IPv6*

Aunque IPv6 supone una mejora significativa, aún no ha sido adoptada. Para mitigar la falta de direcciones IPv4 se utiliza **NAT** (Network Address Translation).

La NAT trata de remapear un espacio de direcciones IP en otros mediante dispositivos de encaminamiento. Así, varias máquinas en una red privada pueden acceder a máquinas externas.

Otros métodos son *Teredo Tunneling* y *Dual IP Layer Operation*.

2.3 FIREWALLS

Consisten en un filtro de paquetes, tal que solo puedan acceder los paqueres deseados a la red. Es la herramienta de seguridad de red más básica.

Dos tipos:

- De capa de red
  - Sin estado: No guardan estado. Sencillos, rápidos pero con políticas sencillas
  - Con estado: Mantiene el estado de las conexiones, conoce si un paquete es una conexión en curso o es un a nueva conexión, y de cual
- De aplicación (procesos en vez de interfaces/puertos)

En Linux se implementa el filtrado de paquetes con **iptables** a nivel de usuario y **Netfilter** a nivel de *kernel*.

2.3.1 *Firewall con iptables*

iptables aplica cadenas de reglas ordenadas a los paquetes. La tabla por defecto se llama **filter**, que contiene tres cadenas por defecto: **FOWARD** (se aplican a paquetes que llegan a una interfaz y deben salir por otro), **INPUT, OUTPUT** (paquetes que entran/salen del host).

Cada regla tiene un objetivo que determina qué hacer con los paquetes emparejados con una regla. Los objetivos son **ACCEPT, DROP, REJECT, LOG, QUEUE** (pasa paquetes a un programa local de usuatio a través de un módulo del kernel), **RETURN** (termina las cadenas definidas por el usuario), **ULOG**, u otra cadena.

```
iptables -A chain-name -i interface -j target [[!] -p|-s|-d|-i|-o|-f|--sports|--dports ...]
iptables -P chain-name target
iptables -F chain-name

# Descarta todos los paquetes que sale hacia 192.168.1.1
iptables -A OUTPUT -d 192.168.1.1 -j DROP

# Inicializar la tabla filter
# NOTA: SE RECHAZA POR DEFECTO TODO FOWARD E INPUT
iptables -D
iptables -P INPUT DROP
iptables -P FOWARD DRO

# Ej: eth1 (128.138.101.4) está conectado a internet, y eth0 (10.1.1.1) a una red interna (10.1.1.2)

# Permitir todas las conexiones que se originan dentro de una red interna
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT

# Dejar pasar conexiones a servicios de red (SSG, HTTPS(S))
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j ACCEPT

# Permitir conexiones SSH al firewall desde la rd interna
iptables -A INPUT -i eth0 -d 10.1.1.1 -p tcp --dport 22 -j ACCEPT

# loopback y pings
iptables -A INPUT -i lo -d 127.0.0.1 -p ANY -j ACCEPT
iptables -A INPUT -i eth0 -d 10.1.1.1 -p icmp --icmp-type 8 -j ACCEPT

# Se hace logging de conexiones no permitidas (/var/log/[kern.log | messages])
iptables -A INPUT -i eth1 -j LOG
iptables -A FORWARD -i eth1 -j LOG
```

2.3.2 *Firewall con estado*

Se puede hacer seguimiento de conexiones mediante la extensión **-m state** –**state**. Se clasifica cada paquete en:

- NEW: Intentando establecer una conexión nueva
- ESTABLISHED: Parte de una conexión ya existente
- RELATED: Relacionada pero no parte de una conexión existente
- INVALID: No es parte de una conexión existente e incapaz de crear una nueva conexión

```
iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

2.3.3 *Soporte NAT*

A través de la tabla **nat**, con las cadenas **PREROUTING, POSTROUTING, INPUT, OUTPUT**, y objetivos:

- DNAT: Cambia la dirección de destino del paquete antes del enrutamiento (*Port fowarding*)
- SNAT: Cambia la dirección de orignn del paquete despues del enrutamiento
- MASQUERADE: SNAT para asignación de IP dinámica

```
# Estática
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 194.236.50.155
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 \
   -j DNAT --to 10.1.1.2:22
#Dinámica
```

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 \
   -j DNAT --to 10.1.1.2:22
```

2.3.4 *Utilidades adicionales*

Las reglas definidas no son persistentes. Para ello, se utiliza:

- iptables-save: Vuelca a stdout las reglas activas
- iptables-restore: Carga desde stdin nuevas reglas
- iptables-apply: restore pero que no se activa si el nuevo conjunto corta la conexión

2.3.5 *nftables*

Es el futuro reemplazo a **iptables**.

```
nft add/delete table ip filter

nft add chain [<family>] <table-name> <chain-name>
   { type <type> hook <hook> priority <value> \; }
   nft add chain ip foo input
   { type filter hook input priority 0 \; policy drop\;}
```

Chapter 3

ALMACENAMIENTO

3.1 FICHEROS

3.1.1 Tipos de ficheros

Ficheros regulares

Estan formados por secuencias de bytes. Normalmente el sistema de ficheros no impone su estructura. Soportan acceso secuencial y aleatorio.

- Símbolo: -
- Creación: varios
- Borrado: rm

Directorios

Almacenan referencias a otros ficheros y directorios. Permiten realizar colecciones de elementos. Tiene dos entradas especiales: padre (..) y actual (.). Estos definen un sistema de ficheros jerárquico.

- Símbolo: d
- Creación: mkdir
- Borrado: rmdir, rm -r

Enlaces duros (hard links)

Referencia que asocia un nombre a un fichero. El FS guarda el número de referencias a cada fichero y sólo lo elimina cuando se borra su último enlace. No puede haber enlaces duros entre sistemas de ficheros.

- Símbolo: -
- Creación: ln
- Borrado: rm

Enlaces simbólico (symlinks)

Apuntan a otro fichero por su nombre. Permiten crear bucles y apuntar a ficheros no existentes. Se pueden establecer entre sistemas de ficheros.

- Símbolo: l
- Creación: ln -
- Borrado: rm

Ficheros de dispositivos de carácteres/bloque

Mecanismo para abstaaer hardware y permitir que los usuarios se comuniquen con el hardware y periféricos.

- De caracter (c): E/S byte a byte (/dev/null, /dev/urandom...). No tienen buffering,
- De bloques (b): E/S en bloques de datos. Soportan siempre acceso aleatorio, se accede mediante cache.
  - sd = dispositivo SCSI/SATA (/dev/sda1)
  - sr = CD/DVD (/dev/sr0)
- Creación: mknod
- Borrado: rm

Local Domain Sockets

Mecanismo de comunicación entre procesos. Se emplea un fichero en vez de un par IP-puerto. Bidireccionales. Permiten la identificación del proceso origen.

- Símbolo: s
- Creación: socket()
- Borrado: rm

Tuberias con nombre

Mecanismo para comunicación entre procesos mediante colas FIFO. Unidireccionales. Permite mezclar mensajes. Más cómodo en scripts que sockets.

- Símbolo: p
- Creación: mknod / mkfifo
- Borrado: rm

3.1.2 *Comandos de gestión de ficheros*

chmod cambia los permisos de los ficheros y directorios

umask permite cambiar la mascara de modo de creación de fichero, que limpia los bits que se le indican.

/usr/include/linux/limits.h contiene la lista de caracteres permitidos para nombrar un archivo. Su nombre puede contener hasta 256 caracteres y su ruta puede tener 4096.

/usr/share/misc/magic.mgc guarda las secuencias *magic number*, que contienen al principio una firma en forma de cadena de bits que representa su tipo.

file intenta clasificar el tipo de fichero de cada argumento. Utiliza tres tests: FS, *magic number* y lenguaje. Muestra el primer emparejamiento exitoso.

find busca archivos dados unos criterios de búsqueda.

which muestra la ruta de cualquier ejecutable en la PATH. No sigue enlaces simbólicos.

whereis localiza el binario, fuentes y paginas man para un comando.

locate lee una base de datos de **updatedb** para encontrar un fichero mas rápido.

3.2 JERARQUÍA DEL SISTEMA DE FICHEROS

El sistema de ficheros define la manera de almacenar, recuperar, actualizar e identificar datos en un soporte físico.

3.2.1 *Taxonomía*

El **Filesystem Hierarchy System** (FHS) define los directorios donde deben organizarse los ficheros y los nombra. Se puede consultar con el comando **hier**.

Se puede clasificar según si

- Es compartible entre máquinas diferentes (*shareable/unshareable*)
- Un usuario no administrador puede modificar un fichero (*variable/static*)

Historicamente el sistema de ficheros quedaba dividido en tres ramas:

- **/** : Lo mínimo necesario para el funcionamineto del SO. Esático y no compartible
- **/usr**: Programas, librerias, fuentes... del usuario. Estático y compartible. Historicamente en un disco diferente de /
- **/var**: Datos variables

Desde *systemd* se decidió volver directorios antes separados de /usr a symlinks a sus respectivos directorios en este, volviendolo más uniforme respecto a otros sistemas UNIX.

Directorio	Descripción
/root	Directorio home del administrador
/home	Directorios home de usuarios estandar

Directorio	Descripción
/users	Usuarios **NIS** (Network Information System), tal que conjuntos de máquinas compartan mismos ficheros de configuración
/export/home	(Solo en Solaris) Usuarios NIS
/bin	Comandos esenciales disponibles antes de montar cualquier otro sistema de ficheros
/usr/bin	Comandos ejecutables por el sistema
/sbin	Comandos del administrador, complementarios a comandos de /bin
/usr/sbin	Comandos a utilizar una vez cargado /usr
/usr/local/sbin	Comandos instalados localmente por administrador
/usr/local	Utilizado para instalar software local. Replica estructura de raíz
/opt	Utilizado para la instalación de paquetes de software. Subdivididos por proveedor
/etc	Guarda ficheros de configuración
/dev	Ficheros de dispositivo
/tmp	Ficheros temporales. Es borrado al arrancar
/var	Ficheros que varían con frecuencia
/boot	Contiene todo lo necesario antes de que el kernel pueda ejecutar programas en modo usuario
/mnt	Directorio donde montar FS temporales
/media	Cabecera para puntos de montaje de elementos extraibles
/proc	Contiene sistemas de ficheros virtuales

3.2.2 *Identificación de dispositivos de bloque persistente (/dev)*

La identificación de dispositivos para un mismo controlador de disco no es determinista. Para evitar ambiguedades cada partición contiene un identificador único (UUID), un número de 128 bits. **udev** (*userspace /dev*) se encarga de gestionar las correspondencias, que se encuentran en **/dev/disk/by-uuid/** (**lsblk -f**).

Las reglas de udev al recibir eventos de dispotitivo se encuentran, de mayor a menor prioridad y en orden lexicográfico, en **/etc/udev/rules.d**, **/run/udev/rules.d** y **/lib/udev/rules.d**.

Ejemplo:

```
# 40-usb.media-players.rules
ACTION!="add|change", GOTO="media_player_end"
# catch MTP devices
ENV{DEVTYPE}=="usb_device", GOTO="media_player_start"
# catch UMS devices
SUBSYSTEM!="block", GOTO="media_player_end"
SUBSYSTEMS=="usb", GOTO="media_player_start"
GOTO="media_player_end"
LABEL="media_player_start"
ATTRS{product}=="Rockbox media player", ATTRS{manufacturer}=="Rockbox.org",
ENV{ID_MEDIA_PLAYER}="rockbox"
ATTRS{model}=="*[iI][tT][uU][nN][eE][sS]*", ATTRS{idVendor}=="22b8",
ATTRS{idProduct}=="4810", ENV{ID_MEDIA_PLAYER}="motorola_itunes-phone"
# Series 60 phones
ATTRS{model}=="S60", ATTRS{idVendor}=="0421", ATTRS{idProduct}=="*",
ENV{ID_MEDIA_PLAYER}="nokia_series-60-phones"
ATTRS{model}=="*Motorola Phone (V3i)*", ATTRS{idVendor}=="22b8",
ATTRS{idProduct}=="4810", ENV{ID_MEDIA_PLAYER}="motorola_v3i"
ATTRS{vendor}=="*Apple*", ATTRS{model}=="*iPod*", ENV{ID_MEDIA_PLAYER}="apple_ipod"
ATTRS{vendor}=="[sS][oO][nN][yY]*", ATTRS{model}=="*PSP*",
ENV{ID_MEDIA_PLAYER}="sony_psp"
LABEL="media_player_end"
```

3.2.3 *Sistemas de ficheros virtuales (/proc)*

Contiene entradas virtuales que cambian dinámicamente en vez de ficheros reales.

Procfs contiene información sobre procesos y el estado del kernel. Cada subdirectorio (**/proc/PID**) contiene información del proceso que se está ejecutando.

Sysfs es un sistema de ficheros en memoria que permite compartir información a procesos de uuario.

3.2.4 *Montaje/desmontaje*

El montaje/desmontaje consiste en unir/liberar un sistema de ficheros al árbol de directorios.

La mayoría de FS utilizan particiones o volúmenes lógicos. Se añaden con **mount**. Se montan casi siempre sobre directorios vacíos.

```
# Montaje
sudo mount -t ext3 /dev/sda4 /home
# Liberación
sudo umount /home
# Buscar referencias
fuser -c /home
# Listar montajes
mount | cat /etc/mtab
# Listar dispositivos de bloque
lsblk
```

Para montar volumenes en red **NFS** (*Network File System*) montar añadiendolos en **/etc/fstab** o utilizar autofs (uno de los dos).

En caso de fallo, se puede usar **fuser** o **lsof** para identificar el proceso que está utilizando el sistema de ficheros.

3.2.5 */etc/fstab*

Es el fichero de configuración que contiene FS montados de forma habitual

```
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=afd3b54e-44af-48bc-8ddf-a401670770bf / ext4 errors=remount-ro 0 1s
```

mount -a muestra todas las entradas de /etc/fstab. Se suele usar en scripts de arranque

El campo <dump> si está a 1 indica que requiere una copia de seguridad. El campo <pass> utiliza fsck para determinar si comprobar el sistema de ficheros al arrancar. 0 significa no escanear, 1 y 2 escanear (raíz y el resto, respectivamente).

3.2.6 */etc/mtab*

Contiene todos los sistemas de ficheros montados actualmente, es decir, los montados por fstab y los montados por el usuaio.

```
/dev/sda1 / ext4 rw,errors=remount-ro 0 0
proc /proc proc rw,noexec,nosuid,nodev 0 0
sysfs /sys sysfs rw,noexec,nosuid,nodev 0 0
udev /dev devtmpfs rw,mode=0755 0 0
devpts /dev/pts devpts rw,noexec,nosuid,gid=5,mode=0620 0 0
```

3.3 PARTICIONES

Una **partición** es una división lógica de tamaño conocido de un disco.

fdisk y **parted** son herramientas de particionado. El primero solo soporta particiones MBR (hasta 2 TiB).

3.3.1 *Tablas de particiones*

Una **tabla de partición** es una estructura que almacena la información de sus particiones. Estas son:

- **Master Boot Record** (MBR): Contiene la tabla y el cargador primario del sistema operativo, con *volume records* en cada partición
- **Extended Boot Record** (EBR): MBR que permite encadenar particiones extendidas (una partición primaria subdividida en mas particiones)
- **GUID Partition Table** (GPT): Estandar moderno UEFI. Utiliza *Logical Block Addressing* en vez de *cylinder-head-sector* para identificar las particiones

3.3.2 *Comandos*

mkfs permite crear sistemas de ficheros. También se puede utilizar **mkfs.{ ext3 | ext4 | BeFS | msdos | vfat ... }**..

mkswap crea una partición de intercambio. Para generar un fichero de intercambio se utiliza **fallocate** o **dd**. Suele ser recomendado el doble de la RAM. **vn.swappiness** controla la agresividad del *swapping*.

blkid muestra el dispositivo, atributos y tipo de sistema de ficheros.

dumpe2fs muestra información de extX.

tune2fs permite ajustar parámetros de extX.

df muestra el espacio libre en particiones.

du muestra el espacio de disco usado por ficheros y subdirectorios.

3.3.3 *Consistencia de sistemas de ficheros*

Para garantizar la consistencia de los datos en disco, se utiliza varios mecanismos.

El **journaling** mantiene un registro de los comandos aún no consolidados en disco. Tienen prioridad las transacciones sobre los datos.

Copy-on-write evita las doblesescrituras al disco de journaling escribiendo siempre en bloques nuevos. Cuando la escritura es correcta actualiza metadatos y libera el bloque antiguo.

fsck comprueba y trata de reparar sistemas de ficheros. Algunos errores son:

- Varios ficheros utilizando el mismo bloque
- Bloques ocupados marcados como libres
- Números de enlaces incorrectos
- i-nodos con información pero sin pertenecer a la entrada del directorio
- Entradas de directorio apuntando a i-nodos ilegales o vacíos

3.4 VOLUMENES DINÁMICOS

Los sistemas de ficheros en Linux se implementan mediante los Virtual File System con el objetivo de mantener una API consistente y abstraer conceptos de bajo nivel.

Normalmente existe una relación directa entre sistemas de ficheros y partición. Esto da lugar a algunas limitaciones (capacidad fija, no se puede repartir datos entre discos, etc...).

Los gestores de volumenes lógicos (**LVM**) permiten:

- Redimensionar dinamicamente el tamaño del volumen
- Realizar *snapshots* para copias de seguridad
- Mover volumenes lógicos entre dispositivos físicos
- Remplazo de discos en caliente
- *Mirroring* (copias en espejo) y *striping* (repartir datos entre discos)

Volumen físico

Es el elemento de almacenamiento de un LVM. Pueden ser particiones o discos duros completos. Recomendable una partición por disco. Se crean con **pvcreate**.

Grupo de volumenes

Es un conjunto de PV. El espacio disponible dentro de un grupo se divide en unidades de tamaño fijo (*extent*). Se crean con **vgcreate** y modifican con **vgchange**.

Volumen lógico

Es el elemento que proporciona la virtualización del almacenamiento. Hay varios tipos de volumenes lógicos. Se recomienda utilizar etiquetas en vez de UUID. Se crean con **lvcreate** y se mapean en **/dev/nombre_vg/nombre_lv**. Es necesario darle un tipo de sistema de ficheros y montarlo como una partición normal.

3.5 CUOTAS

Permiten limitar el espacio disponible para usuarios y grupos en sistemas de ficheros. Pueden ser limitaciones **duras** (no deben ser excedidas) o **blandas** (pueden ser temporalmente excedidas durante un periodo de gracia).

Se aplican de manera independiente sobre **bloques** (datos reales) e **inodos** (información de ficheros). Los nodos suelen usar utilizar al menos un bloque de disco y no se tienen en cuenta en la limitación de bloques. La limitación sobre bloques afecta al tamaño de los ficheros y la de i-nodos a su número.

edquota establece una plantilla de cuota a usar. **edquota -t** permite establecer un periodo de gracia durante el que se puede exceder un límite blando. También se puede utilizar **setquota**.

Es necesario indicar la presencia de cuotas en fstab y remontar el sistema de ficheros. Se puede crear una base de datos de cuotas y tablas de utilización de disco con **quotacheck**.

Las cuotas se activan con **quotaon**.

3.6 VOLUMENES DINÁMICOS CON RAID

Redundant Array of Independent Disks (RAID) es un mecanismo para crear dispositicos lógicos virtuales utilizando 2 o más dispositivos de bloque reales. Tiene las siguientes ventajas:

- Rendimiento: Permite aumentar el ancho de banda mediante *striping*
- Capacidad: Permite alcanzar tamaños mayores a los de los discos individuales
- Fiabilidad: Mediante redundancia y almacenamiento de paridad, permite ser tolerable a algunos fallos

Se puede implementar en hardware (controlador RAID) y en software (a través del SO).

3.6.1 *Niveles RAID*

Nivel	Descripción
LINEAR/JBOD	Concatena un disco tras otro. No utilizado.
0	*Striping*. Reparte datos entre discos sin redundancia
1	*Mirroring*. Escribe datos idénticos en todos los discos a costa de capacidad
5	Distribución de datos y paridad, mas eficiente que RAID 1 pero con tolerancia de un solo disco perdido
6	RAID 5 con 2 discos de paridad
10	RAID 1 + 0. Dos conjuntos de RAID 0 en espejo

3.6.2 *RAID software*

Linux da soporte RAID mediante el módulo **md_mod** y la herramienta **mdadm**, agrupando varios discos en volúmenes lógicos. El resumen de volumenes en RAID se almacena en **/proc/mdstat**. Almacena su configuración crítica en varios **superbloques** de 256 bytes repartidos en disco. Las particiones deben ser del tipo *fd, linux auto*.

Modos de funcionamiento de **mdadm**:

Modo	Descripción
Create	Crea y activa un nuevo array con metadatos
Assemble	Ensambla los componentes de un array creado anteriormente dentro de un array activo
Build	Construye un array que no tenga metadatos. No puede diferenciar así entre la creación inicial y el ensamblado
Follow/Monitor	Monitoriza cambios en el estado del array (salvo RAID 0)
Grow	Cambiar el tamaño, modo, nº de dispositivos ... de un array
Incremental Assembly	Añade un único dispositivo a un array
Manage	Gestiona tareas concretas de un array como eliminar dispositivos con errores o añadir dispositivos extra
Misc	Resto de actividades
Auto-detect	Solicia al kernel que active la autodetección de arrays

Ejemplo:

```
# RAID 6
mdadm --create /dev/md/name --level=6 --raid-devices=4 \
   /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
mdadm --detail /dev/md/name
mkfs.ext4 /dev/md/name
mount /dev/md/name /mnt/mount_point

# RAID 1 que ya contiene datos
mdadm --create /dev/md/name --force --level=1 --raid-devices=1 /dev/sda4
mdadm /dev/md/name -a /dev/sdc3
mdadm -grow /dev/md/name -n 2
```

Su configuración se encuentra en **/etc/mdadm/mdadm.conf**. Puede funcionar sin este fichero pero es habitual crearlo al crear o cambiar arrays de discos.

3.7 COPIAS DE SEGURIDAD

Principios:

- Combertura: ¿Qué proteger?
- Periodicidad: ¿Cuándo hacer las copias?
- Separación: ¿Dónde guardar las copias?
- Historia: ¿Cuánto tiempo almacenarlas?
- Testeo: Protocolos para verificar las copias
- Seguridad: Garantizar la seguridad física y lógica de las copias
- Integridad: ¿Son válidos los datos a copiar?

Estrategias:

- Automatización y sencillez
- Eficiencia y rapidez
- Complejidad de restauración
- Verificación
- Tolerancia a fallos
- Portabilidad

3.7.1 *Planificador*

El planificadoor establece dos politicas: qué datos origen copiar y cuando. Se suele hacer con **cron** o con *timer units* de *systemd*.

Los datos de origen son la información que se quiere guardar. Se puede seguir los criterios de **FHS**.

Hay dos tipos de *backups* en primera copia: **completo** (todo el sistema, largo y costoso) o **parcial** (se guarda la información más valiosa).

En copias subsiguientes, los dos tipos son **incremental** (guarda los ficheros modificados desde el *backup* anterior, rápido pero requiere el resto de los *backups*) y **diferencial** (guarda los cambios desde el estado actual al primer *backup*, mas lento pero solo requiere el original y el diferencial).

Hay varias estrategias posibles como compromiso entre coste de almacenamiento y separación temporal:

- FIFO: Los últimos datos sobreescriben datos mas viejos. Dan problemas con copias diferenciales
- Generacional: Tres conjuntos de *backups* con diferentes frecuencias de actualización
- Torres de Hanoi: El ciclo de actualización de un disco se corresponde a los de sus movimientos en el juego
- Jerárquica: Se utilizan discos independientes para cada periodo de copia

3.7.2 *Comandos*

dump realiza copias de seguridad de FS extX. Puede partir la copia en múltiples volumenes y realizar copias incrementales. Almacena también metadatos de los ficheros.

restore permite recuperar backups completos y sus copias incrementales realizadas por **dump**.

dd copia y convierte ficheros, además de poder leer dispositivos.

tar gestiona archivos .tar (*tarballs*). Opera con ficheros y directorios en vez de bloques, denominados miembros. Permite tambien compresión.

rsync está orientada a sincronizar ficheros entre máquinas. Permite backups completos y diferenciales, y puede operar como un demonio con protocolo propio.

Otras alternativas son Bacula (para sistemas heterogéneos) y Amanda (sistema cliente/servidor).

Chapter 4

GESTIÓN DE PROCESOS

4.1 CONTROL DE PROCESOS

Un proceso es un conjunto de páginas de memoria (código y bibliotecas, datos, pila, *heap*, ...) y algunas estructuras dentro del *kernel* (mapas de direcciones del proceso, estado, prioridad, ..., utilización de recursos, ficheros abiertos, red, señales, dueño, ...).

4.1.1 *Monitorización de procesos*

ps lista los procesos activos (por defecto los que tengan el mismo EUID que el invocador y que esten asociados a la misma terminal) con su PID, usuario, tiempo de ejecución, estado del proceso, ...

Algunas opciones son:

Flag	Descripción
-e	Visualizar todos los procesos
-u	Visualizar procesos de un usuario (efectivo)
-U	Visualizar procesos de un usuario (real)
f	Salida extendida
-o	Especificaa continuación el formato de salida
O	Ordena la salida
-u	Salida orientada al usuario

Columnas:

Código	Descripción
%CPU	Uso de CPU
%MEM	Uso de memoria
VSZ	Tamaño en KiB de memoria virtual
RSS	Tamaño en KiB de memoria residente
STAT	Estado del proceso

Códigos de estado:

Código	Descripción
D	*Sleep* no interrumpible
R	Ejecutandose o ejecutable
S	*Sleep* interrumpible
T	Parado por señal de control
t	Parado por depurador al trazar
W	Paginando (obsoleto)
X	Muerto (en principio no sale)

Código	Descripción
Z	Zombi (el proceso ha terminado pero su padre no ha esperado)
<	Alta prioridad
N	Baja prioridad
L	Paginas encerradas en memoria
s	Líder de sesión
l	Multihilo
+	Proceso de primer plano

top es una versión en tiempo real de ps que se actualiza automaticamente, que además incluye una cabecera. Se le puede otorgar máxima prioridad con **sudo top -q**. También muestra la media de carga de los ultimos 1, 5 y 15 minutos.

La media de carga representa la carga de un procesador respecto a los procesos que puede procesar. Por ejemplo, una carga de 0.5 significaría que está atediendo la mitad de los procesos que podría atender, 1 es que esta atendiendo a todos los procesos y 1.5 que está a la capacidad máxima y que hay otros procesos esperando a ser atendidos. Para procesadores multinúcleo, se suman la carga de cada núcleo para obtener la carga del sistema.

pgrep busca procesos de acuerdo a su nombre y otras características.

4.1.2 *Interacción con procesos vía señales*

Se puede interacturar con procesos enviandoles señales mediante **kill**. **pkill** envía una señal a todos los procesos que puede emparejar con opciones. **killall** envía una señal a todos los procesos que cumplan unas condiciones. También se pueden utilizar atajos de teclado.

Nº	Nombre	Descripción	Defecto	Capturable	Bloqueable	Vuelca núcleo	Atajo
1	HUP	Colgar (limpieza)	Termina	Si	Si	No	
2	INT	Interrumpir	Termina	Si	Si	No	Ctrl+c
3	QUIT	Salir	Termina	Si	Si	Si	Ctrl+\
9	KILL	Matar	Termina	No	No	No	
10	BUS	Error de bus	Termina	Si	Si	Si	
11	SEGV	Error de segmentación	Termina	Si	Si	Si	
15	TERM	Terminar	Termina	Si	Si	No	
17	STOP	Parar	Para	No	No	No	
18	TSTP	Parar (teclado)	Para	Si	Si	Si	Ctrl+Z
19	CONT	Continuar tras parar	Ignora	Si	No	No	
28	WINCH	Cambio de ventana	Ignora	Si	Si	No	
30	USR1	Definido por usuario	Termina	Si	Si	No	
31	USR2	Definido por usuario	Termina	Si	Si	No	

HUP reenvía la señal a todos los procesos hijos. **nohup** fuerza a ejecutar programaas que ignoren HUP, pero deben ser redirigidos a un fichero.

4.1.3 *Prioridades y recursos de proceso*

El *niceness* es la prioridad de ejcución de un proceso. No suele ser requerido cambiarlo. El rango va de -20 a +19. El hijo hereda la prioridad del padre. El dueño del proceso puede reducir la prioridad, *root* puede también aumentarla.

nice permide ajustar la prioridad de un proceso. Algunas shells lo incluyen como un comando propio que no se comporta iguak. **ionice** ajusta la prioridad de la entrada/salida. **renice** permite ajustar la prioridad de un proceso en ejecución.

ulimit permite controlar los recursos empleados por procesos lanzados por el shell.

4.1.4 *Monitorización de actividad con el SO*

strace muestra en tiempo de ejecución las llamadas al sistema que se han realizado y las señales que un proceso ha recibido.

4.2 AUTOMATIZACIÓN DE PROCESOS

4.2.1 *Ejecución programada*

at permite indicar el momento en el que se quiere ejecutar un trabajo. Pasa a un nuevo prompt donde escribir la lista de comandos a ejecutar hasta mandar la señal de fin de transmisión (Ctrl+D). Al terminar, se envía un correo al usuario. El trabajo no se para al salir de la sesión.

Otros comandos relacionados son **atq** (lista de trabajos pendientes), **atrm** (borrar trabajos) y **batch** (ejecuta trabajos cuando la carga es baja [<1.5, según **/proc/loadavg**]).

Los ficheros de configuración son **/etc/at.allow**, **/etc/at.deny** (gestiona quien puede o no puede planificar tareas).

4.2.2 *Ejecución periódica*

cron es el *daemon* que permite crear tareas periódicas. Se definen mediante el comando **crontab**.

Los permisos de uso se encuentran en **/etc/cron.allow**, **/etc/cron.deny**. Los trabajos se especifican en **/var/spool/cron/crontabs** con el siguiente formato:

```
SHELL=/bin/bash
# Borra /tmp todos los días laborales a las 4:30 de la mañana
30 4 * * 1-5 rm -rf /tmp/*
# Escribe la hora cada 15 minutos durante la noche
0,15,30,45 0-8,20-23 * * * echo Hora:$(date) >> /tmp/horas
```

Se puede modificar o reemplazar el fichero con **crontab**.

cron busca en **/var/spool/cron** ficheros para ejecutarlos a la hora indicada. Además ejecuta acciones según **/etc/crontab** y **/etc/cron.d** (normalmente de mantenimiento). Los scripts que se ejecuten de forma periodica se almacenan en **/etc/cron.{hourly|daily|weekly|monthly}**.

```
# /etc/crontab en Debian

# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user command
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron \
   || ( cd / && run-parts --report /etc/cron.daily ) \
47 6 * * 7 root test -x /usr/sbin/anacron
   || ( cd / && run-parts --report /etc/cron.weekly ) \
52 6 1 * * root test -x /usr/sbin/anacron
   || ( cd / && run-parts --report /etc/cron.monthly ) \
#
```

anacron se encarga de ejecutar comandos periodicamente con una frecuencia de días. No asume que la máquina está encendida permanentemente.

Se puede configurar en **/etc/anacrontab**:

```
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
HOME=/root
LOGNAME=root
# These replace cron's entries
1 5 cron.daily run-parts --report /etc/cron.daily
```

```
7 10 cron.weekly run-parts --report /etc/cron.weekly
@monthly 15 cron.monthly run-parts --report /etc/cron.monthly
```

4.2.3 *Timers de systemd*

Son archivos de unidades terminados en *.timer*. Hay temporizadores en tiempo reál (evento del calendario) y monotónicos (intervalo de tiempo, se detienen tras apagarse el sistema).

Para cada timer existe un *.service* que queda activado por el timer.

Se puede listar los timers activos con **systemctl list-timers** [**-all**].

```
# /etc/systemd/system/foo.timer
[Unit]
Description=Ejecuta foo semanalmente y tras cada boot
[Timer]
OnBootSec=15min
OnUnitActiveSec=1w
[Install]
WantedBy=timers.target

# /etc/systemd/system/bar.timer
[Unit]
Description=Ejecuta bar semanalmente
[Timer]
OnCalendar=weekly
Persistent=true
[Install]
WantedBy=timers.target
```

Chapter 5

MONITORIZACIÓN DEL SISTEMA

La monitorización del sistema sirve para:

- Detectar problemas y daños
- Mejorar prestaciones
- Control general de carga
- Auditoría
- Sintonización

Cuanto más detalle en la monitorización, mayor tamaño de los datos registrados, y mayor complejidad para analizarlo.

Algunos comandos útiles son:

- Sistemas de ficheros: df, du, lsof
- Sesiones de usuarios: last, ac
- Procesos: ps, lastcomm, free, vmstat, iostat, sar, /proc, uptime
- Ped: tcpdump, tshark, ping, traceroute, tracepath, tcpflow, netstatm, nmap
- Servicios remotos: SSH, DNS, NFS, SMB, LDAP, Kerberos, SMTP, HTTP
- Seguridad

Sesiones de usuario

Registro de sesiones (usuarios, terminales, entrada y salida de sesión): **/var/run/utmp**, **/var/log/wtmp**. Conexiones desde último wtmp con **last**

Procesos

Registro de procesos (tº ejecución, memoria, recursos E/S, nombre, usuario, terminación): **/var/log/acct**. Se activa mediante **accton**. Se visualiza con **lastcomm**.

Red

Descubrimiento y/o verificación de conectividad (**ping**), servicios de red (**nmap**), y encaminadores (**traceroute**, **tracepath**), actividad en tarjetas de red y servicios (**netstat**), detalle de tráfico (completo: **tcpdump**, resumido: **tcpflow**).

Registro de datos

Registros en **/var/log**. Propietario root o servicio.

sylog es el logger de eventos del sistema. Clasifica mensajes por fuente e importancia. Tres partes:

- **syslogd** (*daemon*)
- **openlog** (bibliotecas)
- **logger** (comando)

Para leer cambios de configuración al arrancar:

```
sudo kill -HUP $(/bin/cat /var/run/syslogd.pid)
sudo systemctl restart rsyslog.service
```

Se puede configutrar en **/etc/syslog.conf**

Ejemplo:

```
# /etc/rsyslog.conf (variante utilizada por Debian)

#
# First some standard log files. Log by facility.
#
auth,authpriv.*            /var/log/auth.log
*.*;auth,authpriv.none     -/var/log/syslog
#cron.*                    /var/log/cron.log
daemon.*                   -/var/log/daemon.log
kern.*                     -/var/log/kern.log
lpr.*                      -/var/log/lpr.log
mail.*                     -/var/log/mail.log
user.*                     -/var/log/user.log
#
# Logging for the mail system. Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                  -/var/log/mail.info
mail.warn                  -/var/log/mail.warn
mail.err                   /var/log/mail.err
```

logrotate implementa políticas de gestión de logs. Se ejecuta mediante cron.

Se puede extraer información relevante de los logs con herramientas como **cacti**, **nagios**, **swatch**, **logcheck**, **sec**, expresiones regulares...