Tema 3: Actividades administrativas básicas

Administración de Sistemas e Redes

Tomás Fernández Pena

tf.pena@usc.es

Índice

1. Comandos básicos para la gestión de procesos

En el tema anterior vimos como ejecutar comandos del shell:

- otros comandos ajenos al shell se ejecutan igual

En cada momento se están ejecutando un gran número de procesos:

- procesos de sistema (kernel, *daemons*)
- procesos de usuarios

En esta sección trataremos la gestión de los procesos que se están ejecutando:

- listar procesos en ejecución
- detener y matar procesos
- controlar la prioridad de ejecución

1.1. Ver los procesos en ejecución

Existen varias herramientas para ver los procesos en ejecución, la más importante es el comando `ps`

`ps` (***process status***)

lista los procesos con su PID, datos de usuario, tiempo, identificador del proceso y linea de comandos usada

```
$ ps
  PID TTY          TIME CMD
 6368 pts/0    00:00:00 bash
 7441 pts/0    00:00:00 ps
```

sin opciones, `ps` sólo muestra los procesos lanzados desde el terminal actual y con el mismo `EUID` que el usuario que lo lanzó

Opciones de ps `ps` tiene un gran número de opciones, que se pueden especificar de 3 maneras:

1. opciones UNIX: pueden agruparse y se preceden por un guión: `ps -ef`
2. opciones BSD: pueden agruparse y van sin guión: `ps uxa`
3. opciones largas GNU: precedidas de dos guiones: `ps --user tomas`

Algunas opciones:

- `-e` o `ax`: muestra todos los procesos
- `-u` (o `U` o `--user`) *usuario*: muestra los procesos de un usuario
- `u`: salida en formato usuario
- `j`: salida en formato *job* (muestra PID, PPID, etc.)
- `-f` o `l`: salida en formato largo
- `f`: muestra un árbol con la jerarquía de procesos
- `k` (o `--sort`) *campo*: ordena la salida por algún campo (p.e. `ps uxak rss`)
- `-o` (o `o` o `--format`) *formato*: permite definir el formato de salida `ps -o ruser,pid,comm=Comando`

para más opciones ver la página de manual de `ps`

Ejemplo:

```
$ ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1516   536 ?        S    09:43   0:00 init [2]
root         2  0.0  0.0      0     0 ?        S    09:43   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   09:43   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    09:43   0:00 [migration/1]
.............
tomas     6475  0.1  4.9 140180 50920 ?        Sl   09:51   0:18 /usr/lib/mozilla-thunde
tomas     6528  0.2  3.6 116396 37948 ?        Sl   10:01   0:25 /usr/lib/mozilla-firef
```

en este ejemplo:

- `%CPU` y `%MEM`: porcentajes de uso de CPU y memoria
- `VSZ`: memoria virtual del proceso, en KBytes
- `RSS`: tamaño de la memoria residente (*resident set size*) en KBytes
- `STAT`: estado del proceso; puede ser:

Código	significado
D	*Uninterruptible sleep* (usualmente IO)
R	Ejecutándose(*running*) o en cola de ejecución
S	*Interruptible sleep* (p.e. esperando un evento)
T	Detenido
Z	Proceso *zombie*

cuando se usa formato BSD puede aparecer otro código acompañando al principal:

Código	significado
<	alta prioridad
N	baja prioridad
L	páginas bloqueadas (*locked*) en memoria
s	líder de sesión
l	multi-threaded
+	proceso en foreground

pstree muestra el árbol de procesos (similar a `ps f`)

```
init-+-acpid
     |-atd
     |-bonobo-activati
     |-clock-applet
```

```
|-cron
|-cupsd
|-dbus-daemon-1
|-dcopserver
|-dirmngr
|-2*[esd]
|-events/0-+-aio/0
|          |-ata/0
|          |-ata/1
|          |-kblockd/0
|          |-khelper
|          `-pdflush
|-events/1-+-aio/1
|          |-kacpid
|          |-kblockd/1
|          `-pdflush
|-exim4
|-famd
|-firefox-bin---wvMime---ggv
...
```

top

ps da una versión estática de los procesos

- top nos da una lista actualizada a intervalos

```
top - 17:34:08 up  7:50,  6 users,  load average: 0.12, 0.31, 0.27
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.2% us,  2.0% sy,  0.0% ni, 91.0% id,  0.0% wa,  0.8% hi,  0.0% si
Mem:   1026564k total,   656504k used,   370060k free,    65748k buffers
Swap:  2048248k total,        0k used,  2048248k free,   336608k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6130 root      15   0 63692  48m 9704 S  8.7  4.9   8:03.34 XFree86
 6341 tomas     15   0 14692 8852 6968 S  4.3  0.9   1:55.13 metacity
 6349 tomas     16   0 32792  14m 9232 S  1.3  1.5   0:41.60 gnome-terminal
 6019 tomas     15   0  7084 3184 1896 D  0.3  0.3   0:23.22 famd
 6401 tomas     15   0 16756 8280 6856 S  0.3  0.8   0:02.49 geyes_applet2
 6427 tomas     15   0 18288  10m 8112 S  0.3  1.0   0:09.04 wnck-applet
 7115 tomas     15   0 26312  13m  11m S  0.3  1.4   0:00.61 kio_uiserver
 7390 tomas     15   0 45016  30m  18m S  0.3  3.0   0:38.69 kile
    1 root      16   0  1516  536  472 S  0.0  0.1   0:00.61 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    ......................
```

- en la cabecera nos muestra un resumen del estado del sistema
  - hora actual, tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos
  - número total de tareas y resumen por estado
  - estado de ocupación de la CPU y la memoria
- por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU (los más costosos arriba)
- pulsando `h` mientras se ejecuta `top`, obtenemos una lista de comandos interactivos
- para salir, `q`
- Algunos campos de top
  - VIRT: Tamaño total del proceso (código, datos y librerías compartidas cargadas), VIRT=SWAP+RES
  - SWAP: Memoria que ha sido *swapped out* o que aún no ha sido cargada
  - RES: Memoria residente (RAM ocupada por el proceso)
  - CODE y DATA: Memoria ocupada por el código y datos (datos y pila, pero no librerías compartidas) del proceso
  - SHR: Memoria compartida (memoria que puede ser compartida con otros procesos)
  - P: Última CPU usada (SMP)
  - nFLT: Número de fallos de página para el proceso

strace

Muestra las llamadas al sistema realizadas por un proceso en ejecución

- Ejemplo de un `strace` sobre un `top` en ejecución

```
$ strace top
gettimeofday({1195811866, 763977}, {4294967236, 0}) = 0
open("/proc/meminfo", O_RDONLY)         = 3
fstat64(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7f5
```

```
read(3, "MemTotal:      2066348 kB\nMemFre"..., 1024) = 728
close(3)                                = 0
munmap(0xb7f55000, 4096)                = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY) = 3
fstat64(3, {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
fcntl64(3, F_SETFD, FD_CLOEXEC)         = 0
getdents(3, /* 52 entries */, 1024)     = 1020
getdents(3, /* 64 entries */, 1024)     = 1024
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)          = 4
read(4, "1 (init) S 0 1 1 0 -1 4194560 44"..., 1023) = 185
close(4)
......
```

Ejecución en segundo plano

Por defecto, los comandos corren en primer plano (*foreground*): el shell espera a que termine el comando antes de aceptar uno nuevo

- para ejecutar un comando en segundo plano (*background*) hacerlo con &

  ```
  $ sleep 10
  $ sleep 10 &
  ```

- para terminar un proceso en foreground `Ctrl-C`
- para pausar un comando en foreground usar `Ctrl-Z`
  - `bg` pasa el proceso a background
  - `fg` lo devuelve a foreground
- Ejemplo:

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

- El comando `jobs` permite ver la lista de comandos (*jobs*) en background lanzados desde el shell, así como su estado (`fg` y `bg` pueden referirse a uno de los jobs)

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

1.2. Señalización de procesos

El comando básico para enviar señales a un proceso es `kill`

- Ctrl-C y Ctrl-Z son atajos para enviar señales SIGINT (2) y SIGTSTP (20)
- `kill -l` lista el conjunto de señales

```
$ 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     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      ....
```

- para ver su significado, ver `man 7 signal`

Sintaxis de `kill`

`kill` *`[señal]`* `PID`

- *señal* puede indicarse mediante el número o el código:

    - `kill -9` y `kill -KILL` son equivalentes
- las señales más comunes son:
    - `SIGHUP` (1): cuelgue del terminal o muerte del proceso controlador
    - `SIGTERM` (15): mata el proceso permitiéndole terminar correctamente
    - `SIGKILL` (9): mata el proceso sin permitirle terminar
    - `SIGSTOP` (19): para el proceso
    - `SIGCONT` (18): continúa si parado
    - `SIGINT` (2): interrupción de teclado (Ctrl-C)
    - `SIGTSTP` (20): stop de teclado (Ctrl-Z)
    - `SIGQUIT` (3): salida de teclado (Ctrl-\)

Algunas características de las señales:

- La señal que se envía por defecto es `TERM` (15)
    - los procesos pueden ignorar esta señal y no terminar
    - las señales `KILL` (9) y `STOP` (19) no pueden ignorarse
- En bash, cuando enviamos una señal `SIGHUP` a un shell, este se lo reenvía a todos sus hijos
- Cuando cerramos un terminal en un entorno gráfico, o abandonamos una sesión, se envía un `SIGHUP` a todos sus hijos
- La mayoría de los demonios (*daemons*) responden a la señal `SIGHUP` volviendo a leer sus ficheros de configuración:
    - en vez de matar y reiniciar un demonio podemos hacer un `kill -HUP` para reiniciarlo

Ejemplos

```
$ yes >/dev/null &
[1] 9848
$ yes >/dev/null &
[2] 9849
$ ps
 PID TTY      TIME  CMD
9834 pts/7 00:00:00 bash
```

```
9848 pts/7 00:00:02 yes
9849 pts/7 00:00:01 yes
9850 pts/7 00:00:00 ps
$ kill -STOP 9849
[2]+ Stopped          yes >/dev/null
$ jobs
[1]- Running          yes >/dev/null &
[2]+ Stopped          yes >/dev/null
$ kill -CONT 9849
$ jobs
[1]- Running          yes >/dev/null &
[2]+ Running          yes >/dev/null &
$ kill -KILL 9848
$ kill -1 9849
[1]- Matado          yes >/dev/null
[2]+ Colgar          yes >/dev/null
```

Otros comandos

pgrep busca en la lista de procesos para localizar el PID a partir del nombre (similar a `ps | grep`)

- Ejemplo:

```
$ pgrep -u root sshd # PID del proceso sshd de root
```

pkill permite enviar señales a los procesos indicándolos por nombre en vez de por PID

- Ejemplo:

```
$ pkill -9 proceso
```

- si hay varios procesos con el mismo nombre los mata a todos
- en vez de un nombre admite un patrón (p.e. pkill ´l.*´)
  - tener cuidado con su uso (es fácil matar procesos de forma errónea)

killall similar a pkill, pero no admite patrones en el nombre, y tiene otras opciones

nohup normalmente, cuando salimos de un login shell (`logout`) o cerramos una un terminal, se envía una señal `SIGHUP` a todos los procesos hijos[1]:

- si lanzamos un proceso en background y salimos de la sesión el proceso se muere al morir el shell desde el que lo iniciamos

El comando `nohup` permite que un lanzar un comando ignorando las señales `SIGHUP`

`nohup` *`comando`*

- la salida del comando se redirige al fichero `nohup.out`

exec `exec` ejecuta un comando reemplazando al shell desde el que se lanza Ejemplos:

```
$ yes > /dev/null &
[1] 14724
$ yes > /dev/null &
[2] 14725
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14724 pts/3 00:00:02 yes
14725 pts/3 00:00:02 yes
14726 pts/3 00:00:00 ps
$ pgrep yes
14724
14725
$ pkill -9 yes
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14730 pts/3 00:00:00 ps
[1]- Matado         yes > /dev/null
[2]+ Matado         yes > /dev/null
```

Más ejemplos:

[1] en teoría, en bash, podemos fijar el comportamiento de la shell modificando la opción `huponexit` con `shopt`

```
$ nohup yes > /dev/null &
[1] 9620
$ kill -HUP 9620
$ ps
 PID TTY      TIME  CMD
8293 pts/5 00:00:00 bash
9620 pts/5 00:00:13 yes
9621 pts/5 00:00:00 ps
$ kill 9620
[1]+ Terminado         nohup yes > /dev/null
```

1.3. Manejo de la prioridad y recursos de un proceso

Cuando un proceso se ejecuta, lo hace con una cierta *prioridad*

- las prioridades van desde -20 (prioridad más alta) a 19 (prioridad más baja)
- por defecto, los procesos se ejecutan con prioridad 0
  - un usuario normal solo puede asignar prioridades más bajas (números positivos)
  - `root` puede asignar prioridades más altas (números negativos)
- los comandos para manejo de prioridades son `nice` y `renice`

`nice`

Permite lanzar un comando con una cierta prioridad

- Sintaxis

  `nice -n` *`ajuste comando`*

  la prioridad por defecto se modifica por *`ajuste`*

- Ejemplo:

```
$ nice -n 10 abiword &     # disminuye la prioridad en 10
$ ps -o pid,pri,ni,stat,cmd
  PID PRI  NI STAT CMD
 7133  24   0 Ss   bash
 7431  14  10 SN   abiword
 7552  23   0 R+   ps -o pid,pri,ni,stat,cmd
```

```
$ nice -n -1 abiword
nice: no se puede establecer la prioridad: Permiso denegado
```

`renice`

Permite cambiar la prioridad de un proceso que está en ejecución

- Sintaxis:

  `renice` *`pri`* `[-p` *`pid`*`] [-g` *`pgrp`*`] [-u` *`user`*`]`

- las opciones son:
  - `-p` *`pid`* cambia la prioridad para el proceso especificado
  - `-g` *`pgrp`* cambia la prioridad para los procesos ejecutados por los usuarios que pertenecen al grupo con ID=*`pgrp`*
  - `-u` *`user`* cambia la prioridad para los procesos del usuario especificado
- Ejemplo:

```
$ abiword &
[1] 7681
$ renice 10 -p 7681
7681: old priority 0, new priority 10
$ renice 3 -u tomas
503: old priority 0, new priority 3
```

Control de los recursos de un proceso

El comando interno de bash `ulimit` permite controlar los recursos de los que dispone un proceso arrancado por el shell

- Sintaxis

  `ulimit [`*`opciones`*`] [`*`limite`*`]`

- Algunas opciones:
  - `-a` muestra los límites actuales
  - `-f` máximo tamaño de los ficheros creados por el shell (opción por defecto)
  - `-n` máximo número de ficheros abiertos

- `-s` máximo tamaño de la pila
- `-t` máximo tiempo de cpu
- `-S`/`-H` usa los límites *soft* y *hard*
  - el usuario puede incrementar su límite *blando*, pero sin superar el límite duro
  - estos límites pueden ser fijados en el `/etc/profile`, `/etc/bash.bashrc`

- Para más información `help ulimit`
- Ejemplo: limitar el tamaño de los ficheros creados a 1 KByte

```
$ ulimit -f 1
```

1.4. Análisis básico del rendimiento del sistema

Además de `ps` y `top` existen comandos básicos que nos pueden mostrar el estado del sistema en cuanto a uso de CPU y consumo de memoria[2]

`uptime`

Muestra la hora actual, el tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos (lo mismo que en la primera línea de `top`)

- Ejemplo:

```
$ uptime
 20:25:03  up 25 days, 11:12, 13 users,  load average: 3.00, 3.07, 3.08
```

`w`

Además de la información dada por `uptime`, el comando `w` muestra información sobre los usuarios y sus procesos

- Ejemplo:

```
$ w
 20:24:52  up 25 days, 11:11, 13 users,  load average: 3.10, 3.09, 3.08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
paula    pts/1    godello.dec.usc. 12:08pm  8:15m  8.39s  8.36s  ssh -p 1301 -X paula@p
```

[2] Veremos otros comandos de monitorización más complejos (como `vmstat` o `sar`) en la asignatura ASR II

```
paula    pts/2    godello.dec.usc. 12:09pm  7:30   0.11s  0.11s  bash
pichel   pts/4    -                11:08am  7.00s  0.33s  0.33s  -bin/tcsh
pichel   pts/5    -                 7:12pm 56:56   0.26s  0.16s  ssh usceljpc@sc.cesga.
pichel   pts/6    -                 4:35pm 21:15  16.61s  0.02s  /bin/sh reordena.sh ma
tomas    pts/8    jumilla.dec.usc.  8:24pm  0.00s  0.05s  0.02s  w
```

- Definiciones:
  - `LOGIN@` la hora a la que se conectó el usuario
  - `IDLE` tiempo que lleva ocioso el terminal
  - `JCPU` el tiempo de CPU consumido por los procesos que se ejecutan en el TTY
  - `PCPU` tiempo consumido por el proceso actual (el que aparece en la columna `WHAT`)

`free`

Muestra la cantidad de memoria libre y usada en el sistema, tanto para la memoria física como para el swap, así como los buffers usados por el kernel (similar a lo mostrado en la cabecera de `top`)

- Ejemplo:

```
$ free
             total       used       free     shared    buffers     cached
Mem:       3098556    2969388     129168          0     419300    1674492
-/+ buffers/cache:     875596    2222960
Swap:      6144736    3129376    3015360
```

- la columna `shared` no significa nada (obsoleta)
- Opciones:
  - `-b,-k,-m,-g` memoria en bytes/KBytes/MBytes/GBytes
  - `-t` muestra una línea con el total de memoria (física + swap)
  - `-s` *`delay`* muestra la memoria de forma continua, cada *`delay`* segundos

1.5. Herramientas gráficas

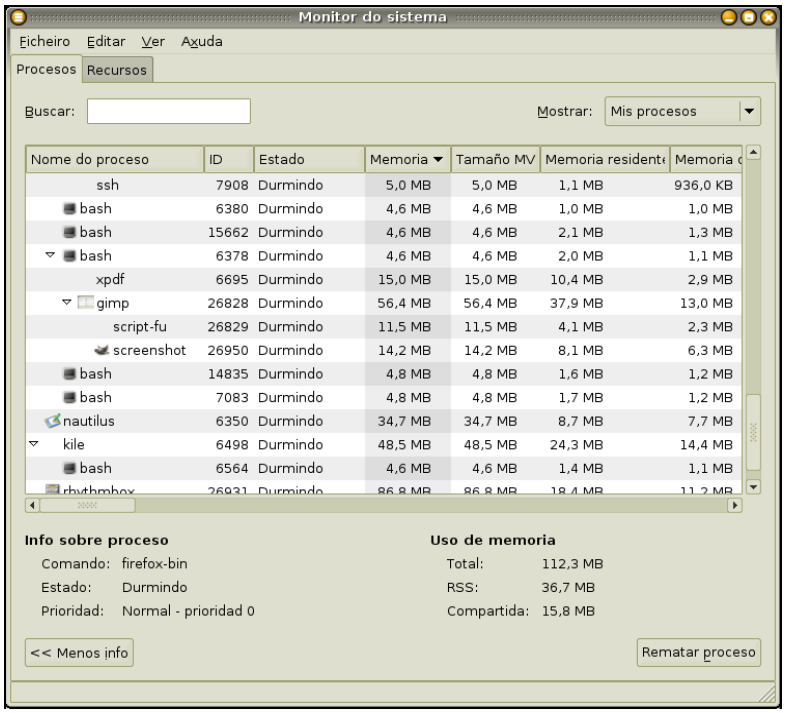
Además de los comandos comentados, si disponemos de un entorno X Windows podemos usar alguna herramienta gráfica en vez de `top` y `ps`:

- proporcionan una visión más clara y son fáciles de usar

Algunas herramientas son

- `gnome-system-monitor`: visor de procesos y monitorizador de recursos del sistema de GNOME
- KDE System Guard (`ksysguard`): gestor de tareas y monitor de rendimiento de KDE

Ejemplo: `gnome-system-monitor`



1.6. El directorio `/proc`

Pseudofilesystem que guarda información sobre el sistema y los procesos

- se inicializa durante el arranque
- está implementado en memoria y no se guarda en disco

- la estructura del directorio `/proc` depende de la versión del kernel
- los comandos vistos (`ps`, `top`, etc.) obtienen la información sobre los procesos de este directorio

Algunos ficheros y directorios son:

- `cpuinfo`: información estática de la CPU
- `meminfo`: información de uso de la memoria
- `partitions`: información sobre las particiones
- `filesystems`: sistemas de ficheros soportados por el kernel
- `version`: versión y fecha del kernel
- `bus/`: directorio con información de los buses PCI y USB
- `cmdline`: línea de arranque del kernel
- `devices`: dispositivos del sistema de caracteres o bloques
- `modules`: módulos del kernel
- `net/`: directorio con información de red
- `interrupts`: muestra el número de interrupciones por IRQ
- `ioports`: lista los puertos de entrada salida usados en el sistema

Además, existen un directorio por cada proceso, que se identifica con el PID del proceso, en el que se puede encontrar información sobre cada proceso, incluidos:

- el directorio desde que se invoco el proceso (enlace `cwd`)
- nombre del ejecutable (enlace `exe`) y la línea de comandos con la que fue invocado (fichero `cmdline`)
- entorno en que se ejecuta el proceso (fichero `environ`)
- estado del proceso (fichero `status`)
- descriptores de ficheros abiertos y archivos o procesos relacionados (directorio `fd`)
- mapa de memoria (fichero `maps`)

Nota: en el kernel 2.6 de Linux aparece un nuevo pseudofilesystem (`/sys`) que reemplaza al `/proc` en lo que se refiere a visualización y ajuste de dispositivos

2. Gestión del sistema de ficheros

UNIX tiene múltiples comandos para trabajar con ficheros y directorios: `ls`, `rm`, `cp`, `mv`, `mkdir`, `rmdir`, `touch`, `etc.`

- estos comandos tienen opciones que es importante conocer
- ver las páginas de manual para las distintas opciones

En esta sección trataremos:

- los diferentes tipos de ficheros y sus atributos
- los permisos de acceso para ficheros y directorios
- la creación de enlaces
- la localización de ficheros
- la creación de particiones y sistemas de ficheros

2.1. Tipos de ficheros y atributos

La mayoría de los sistemas de ficheros definen 7 tipos de ficheros:

Ficheros normales son los usuales; se crean con distintos programas (`vi`, `cp`, `touch`, `etc.`) y se borran con `rm`

Directorios contiene referencias a otros ficheros y directorios; se crean con `mkdir` y se borran con `rmdir` o `rm -r`

Ficheros de dispositivos de caracteres o bloques permiten la comunicación con el hardware y los periféricos; se crean con `mknod` y se borran con `rm`

- caracteres: entrada/salida byte a byte
- bloques: entrada salida en bloques de datos

Tuberías con nombre (*named pipes*) también llamados ficheros FIFO, permiten la comunicación entre procesos; se crean con `mknod` y se borran con `rm`

Sockets comunican procesos en la red; se crean con `socket()` y se borran con `rm` o `unlink()`

Enlaces simbólicos también llamados enlaces *blandos*: apuntador a otro fichero; se crean con `ln -s` y se borran con `rm`

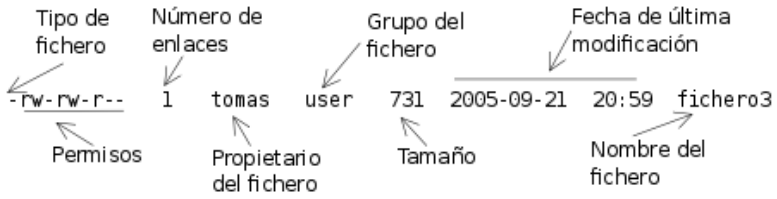
El comando `file` nos permite determinar el tipo de un fichero:

- para ficheros normales, distingue según contenido (fichero de imagen, pdf, ASCII, etc)
- Ejemplo:

```
$ file /dev/xconsole
/dev/xconsole: fifo (named pipe)
$ file fichero1
fichero1: PDF document, version 1.2
$ file fichero2
fichero2: Microsoft Office Document
$ file fichero3
fichero3: PNG image data, 750 x 686, 8-bit/color RGB, non-interlaced
```

Atributos de un fichero

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



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

Carácter	Tipo
-	fichero normal
d	directorio
l	enlace simbólico
c	fichero de dispositivo de caracteres
b	fichero de dispositivo de bloques
p	tubería
s	socket

Número de enlaces indica el número de nombres (enlaces duros) del fichero

- en el caso de un directorio, esto corresponde con el número de subdirectorios (incluidos `.` y `..`)

Tamaño es el tamaño en bytes

- con `ls -lh` se ve el tamaño de forma más legible
- el tamaño máximo de un fichero depende del filesystem usado

Fecha especifica la fecha de última modificación del fichero

- podemos actualizarla con el comando `touch`

Nombre la longitud máxima del nombre es de 255 caracteres

- evitar el uso de espacios y caracteres especiales como `*,$, ?, ´, ", /, \`

Permisos de ficheros y directorios

UNIX proporciona tres operaciones básicas para realizar sobre un fichero o directorio: lectura (`r`), escritura (`w`) y ejecución (`x`)

- Efecto sobre un fichero:
  1. lectura (`r`): permite abrir y leer el fichero
  2. escritura (`w`): permite modificar o truncar el fichero (para borrarlo, basta con que el directorio tenga permiso de escritura)
  3. ejecución (`x`): permite ejecutar el fichero (binario o script)
- Efecto sobre directorios:
  - ejecución (`x`): permite entrar en el directorio (pero no listar su contenido, ni crear ficheros o directorios)
  - lectura y ejecución (`rx`): permite listar el contenido del directorio (pero no crear ficheros o directorios)
  - escritura y ejecución (`wx`): permite crear, borrar o renombrar ficheros (pero no listar su contenido)
  - acceso total (`rwx`)

Los permisos se aplican en tres categorías:

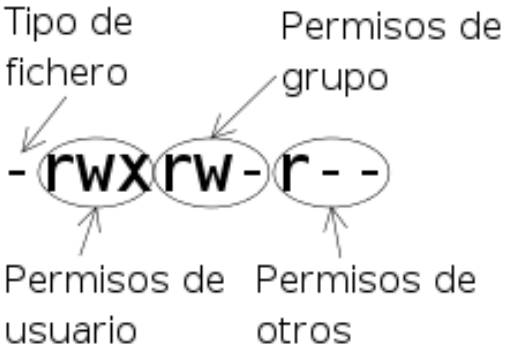
- Permisos de usuario (`u`): propietario del fichero (por defecto, el usuario que lo creó)

- Permisos de grupo (`g`): grupo del fichero (por defecto, grupo principal del usuario que lo creó)
- Permisos de otros (`o`): resto de usuarios

Cada usuario cae en uno solo de estas categorías:

- p.e. al propietario se le aplican los permisos de usuario, aunque sean más restrictivos que los de grupo

Los permisos se identifican con 9 caracteres:



Cambio de permisos

El comando para modificar los permisos es `chmod`

- Formato

  `chmod [-R]` *`operación ficheros`*

- `-R` indica acceso recursivo
- solo el propietario del fichero (o `root`) puede cambiar los permisos

`operación` indica como cambiar los permisos, y puede especificarse mediante símbolos o números:

- Permisos simbólicos: formato *`quien op permisos`*

  quien especificado por `u`, `g`, `o` o `a` para todos

  op puede ser `+` para añadir permisos, `-` para quitar o `=` para establecer

  permisos especificados por `r`, `w`, `x`

  Ejemplos:

- `chmod u+x temp.dat`
  añade permisos de ejecución para el usuario (manteniendo los permisos existentes)
- `chmod ug=rw,o=r temp.dat`
  lectura y escritura para usuario y grupo y sólo lectura para el resto
- `chmod -R =r *`
  pon, de forma recursiva, permisos sólo lectura para todos (ugo)
- `chmod a-x *.bak`
  quita el permiso de ejecución para todos
- `chmod g=u temp.dat`
  pon los permisos de grupo igual a los del usuario
- `chmod a= *`
  quita los permisos a todos

- Permisos numéricos:
  - *operación* se representa por un número **octal** de tres dígitos, para `u`, `g`, y `o` respectivamente
  - cada dígito vale:
    - 4 para `r`, 2 para `w` y 1 para `x`
    - para combinaciones, se suman:
      p.e. `rw` es 6, `rx` es 5 y `rwx` es 7

  Ejemplos:
  - `chmod 750 temp.dat`
    permisos `rwx` para usuario, `rx` para grupo y ninguno para otros
  - `chmod 043 temp.dat`
    ninguno para usuario, `r` para grupo y `wx` para otros

Permisos especiales

Además de `rwx` existen los permisos `setuid`/`setgid` (`s`) y `sticky bit` (`t`)

`setuid y setgid` están relacionados con los atributos de los procesos:

- cuando un proceso se crea se le asigna un UID/GID real y un UID/GID efectivo

UID/GID real identificadores de usuario y grupo del usuario que lanzó el proceso (y que puede matarlo)

UID/GID efectivos determinan las operaciones que el proceso puede hacer sobre los objetos del sistema

- por ejemplo, un proceso con UID efectivo 0 (`root`) puede manipular todos los ficheros del sistema

- lo normal es que los UID/GID normal y efectivo de un proceso coincidan

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

- `ps axo ruid,rgid,euid,egid,cmd` para ver números
- `ps axo ruser,euser,rgroup,egroup,cmd` para nombres

Los permisos `setuid`/`setgid` permiten que un proceso lanzado por un usuario se ejecute con EUID/EGID de otro usuario

- Ejemplo: el programa `passwd`

  ```
  -rwsr-xr-x 1 root root 25872 2005-07-25 23:15 /usr/bin/passwd
  ```

  cuando un usuario ejecuta `passwd` este proceso puede modificar el fichero `/etc/shadow` propiedad de root

Fijar `setuid`/`setgid`

- Forma simbólica

  `chmod u+s` fija `setuid`
  `chmod g+s` fija `setgid`

- Forma numérica: 4000 `setuid`, 2000 `setgid`

Ejemplo:

```
$ ls -l temp
-rw-r-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+s temp; ls -l temp
-rwSr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+x temp; ls -l temp
-rwsr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod 6740 temp; ls -l temp
-rwsr-S---  1 tomas gac 0 2005-09-22 18:07 temp
```

IMPORTANTE: es peligroso poder ejecutar procesos con permisos de otro usuario

- debería evitarse el uso de ficheros `setuid`/`setgid`

sticky bit solo se usa en directorios

- permite crear ficheros en el directorio (si tiene permiso de escritura), pero solo los puede borrar:
  - el propietario del fichero
  - el propietario del directorio
  - el superusuario

Ejemplo:

```
$ ls -ld /tmp
drwxrwxrwt 15 root root 3072 2005-09-22 19:09 /tmp/
```

Para activar el `sticky bit`

- `chmod +t` *`dir`*
- `chmod 1`*`xxx dir`*

Permisos por defecto

Cuando se crea un fichero se cambian los permisos por defecto

- estos permisos pueden modificarse con `umask`

  `umask [`*`opciones`*`]` *`valor`*

  donde *`valor`* son tres dígitos que especifican los permisos para `u`, `g`, `o` según la tabla

Octal	Permisos	Octal	Permisos
0	`rwx`	4	`-wx`
1	`rw-`	5	`-w-`
2	`r-x`	6	`--x`
3	`r--`	7	`---`

- Opciones (dependen de la versión de shell):
  - `-S` muestra los permisos por defecto
- Ejemplo[3]

  ```
  $ umask 027
  $ umask -S
  u=rwx,g=rx,o=
  ```

[3] Comandos que crean ficheros como `touch` o `vi` no ponen permiso de ejecución aunque lo diga `umask`

Cambio de usuario/grupo

Los comandos `chown` y `chgrp` permiten cambiar el propietario y grupo de un fichero

- sólo `root` puede cambiar el propietario
- el grupo puede cambiarse a otro al que pertenezcamos

Formato:

```
chown [opciones] propietario ficheros
chgrp [opciones] grupo ficheros
chown [opciones] propietario:grupo ficheros
```

Algunas opciones

- `-R` recorre recursivamente los subdirectorios
- `-v` (*verbose*) indica las operaciones que realiza

2.2. Enlaces

Permiten referirse a un fichero con otro nombre

Dos tipos:

- Enlaces duros: asignan otro nombre al fichero
  - crean una entrada en el directorio apuntando al mismo nodo-i que el fichero original
  - el fichero no se borra hasta que se borran todos sus enlaces duros
  - no se puede enlazar con ficheros de otra partición
- Enlaces blandos: un fichero que apunta al original
  - si el fichero se borra, el enlace permanece sin apuntar a nada
  - no tienen problema con las particiones

Comando `ln`

Permite crear enlaces

- Formato
  `ln [-s] [`*`opciones`*`] `*`destino`*` [`*`enlace`*`]`
  `ln [-s] [`*`opciones`*`] `*`destino1 destino2 ... [directorio`*`]`
- con `-s` se crean enlaces blandos
- si no se pone nombre del enlace se usa el del *`destino`*
- Ejemplos:

```
pepe@jumilla:~$ ln /home/luis/fich1 fichhard
pepe@jumilla:~$ ln /home/luis/fich2 /home/luis/fich3 .
pepe@jumilla:~$ ls -il /home/luis/fich* ./fich*
293993 -rw-r--r-- 2 luis luis 12 Sep 22 20:19 ./fich2
294011 -rw-r--r-- 2 luis luis 12 Sep 22 21:18 ./fich3
293838 -rw-r--r-- 2 luis luis 13 Sep 22 20:17 ./fichhard
293838 -rw-r--r-- 2 luis luis 13 Sep 22 20:17 /home/luis/fich1
293993 -rw-r--r-- 2 luis luis 12 Sep 22 20:19 /home/luis/fich2
294011 -rw-r--r-- 2 luis luis 12 Sep 22 21:18 /home/luis/fich3
pepe@jumilla:~$ ln -s /home/luis/fich4 blando
pepe@jumilla:~$ ls -il /home/luis/fich4 blando
294012 -rw-r--r-- 1 luis luis 12 Sep 22 21:22 /home/luis/fich4
277445 lrwxrwxrwx 1 pepe pepe 17 Sep 22 21:22 blando -> /home/luis/fich4
```

2.3. Localización de ficheros

Una tarea común de administración es la búsqueda de ficheros que verifiquen ciertas propiedades

- buscar ficheros muy grandes
- buscar ficheros de un determinado usuario
- mostrar los ficheros que se hayan modificado en los últimos 2 días
- buscar ficheros `setuid`/`setgid`

El comando básico para hacer esto es `find`

Comando `find`

Busca a través de la jerarquía de directorios ficheros que cumplan determinado criterio

- Formato

  `find [`*`directorio_de_búsqueda`*`] [`*`expresión`*`]`

- Ejemplo:

  - Busca desde `/etc` los ficheros de tipo socket

    ```
    find /etc -type s
    ```

  - Busca desde `/etc` y `/usr/share` los ficheros que se llamen `magic` o `passwd`

    ```
    find /etc /usr/share -name magic -o -name passwd
    ```

  - Muestra desde el directorio actual todos los ficheros de forma recursiva

    ```
    find
    ```

La *`expresión`* tiene los siguientes componentes:

- opciones: modifican la forma de operación de `find`
- criterio de búsqueda
- acciones: especifica que hacer con los ficheros que encuentra
- operadores: permiten agrupar expresiones

Opciones de `find` normalmente se colocan al principio de la expresión

Opción	Efecto
`-maxdepth` *n*	desciende como máximo *n* directorios
`-mindepth` *n*	empieza a buscar a partir del nivel *n*
`-depth`	procesa el contenido del directorio antes que el propio directorio
`-daystart`	para medidas con tiempo, empieza desde el principio del día actual
`-mount` o `-xdev`	no pasa a otras particiones

Criterios de busqueda

Criterio	Efecto
`-name` *patrón*	busca ficheros que coincidan con el patrón (pueden usarse comodines, escapados)
`-wholename`	permite incluir nombres con el `path`
`-iname`	igual que name pero no distingue mayúsculas/minúsculas
`-regex`	igual pero usa REGEXPR
`-type` *tipo*	busca por tipo de fichero (`b, c, d, p, l, s, f`)
`-size [+/-]n[bck]`	busca por tamaño (tamaño igual, mayor o menor que *n* con `b`=bloques, `c`=bytes y `k`=KB)
`-perm [+/-]`*permisos*	busca por permisos (sin nada, permisos exactos, `-` todos los permisos y `+` alguno de los permisos)
`-user` *nombre*	busca por propietario
`-uid` *n*, `-gid` *n*	busca por UID/GID
`-nouser`, `-nogroup`	busca ficheros con prop./grupo no válidos

Busqueda por atributos temporales

Criterio	Efecto
`-atime [+/-]`*n*	busca ficheros cuya fecha de acceso para lectura coincide con, es anterior a (`+`) o es posterior a (`-`) *n* días
`-mtime [+/-]`*n*	lo mismo, pero con la fecha de última modificación del fichero
`-ctime [+/-]`*n*	lo mismo, pero con la fecha en que se cambió el estado del fichero
`-amin/-mmin/` `-cmin [+/-]`*n*	lo mismo, pero ahora *n* representa minutos
`-newer` *file*	busca ficheros modificados más recientemente que *file*
`-anewer` *file*	ficheros con último acceso más reciente que la modificación de *file*
`-cnewer` *file*	ficheros con cambio de estado más reciente que la modificación de *file*

Acciones de `find` `find` permite realizar distintas acciones con los ficheros que encuentra

- mostrar su nombre (acción por defecto)
- mostrar otra información del fichero
- ejecutar un comando sobre el fichero

Acción	Descripción
`-print`	imprime el nombre de los ficheros que encuentra (acción por defecto)
`-ls`	imprime el nombre de los ficheros con formato de listado largo
`-exec` *`comando`* `\{\} \;`	ejecuta *`comando`* sobre los ficheros encontrados
`-ok` *`comando`* `\{\} \;`	igual que `-exec` pero pregunta antes de ejecutar *`comando`*
`-prune`	si directorio no desciende por el (permite ignorar directorios)

los caracteres `{}` se refieren al fichero que `find` acaba de encontrar y `;` indica el fin del comando

Operadores de `find` permiten agrupar expresiones

Operador	Descripción
`expr1` `-a` *`expr2`*	AND (*`expr2`* no se evalúa si *`expr1`* es falsa)
`expr1` *`expr2`*	igual que `-a`
`expr1` `-o` *`expr2`*	OR (*`expr2`* no se evalúa si *`expr1`* es cierta)
`!` *`expr1`*	NOT (cierto si *`expr`* falsa)
`(` *`expr1`* `)`	agrupan expresiones (hay que escapar los paréntesis)

Ejemplos con `find`

- `find . -maxdepth 1 -user david`
  busca ficheros, sólo en el directorio actual, propiedad de `david`
- `find / -name \*.html -ls`
  busca, en todo el sistema de ficheros, ficheros terminados en `.html` y muestra un listado largo

- `find /home/httpd/html ! -name \*.html`
  busca, desde `/home/httpd/html`, los ficheros que no acaben en `.html`
- `find /home -size +2500k -mtime -7`
  busca, desde `/home`, ficheros más grandes de 2500KB que hayan sido modificados en los últimos 7 días
- `find /home -iname \*.bak -ok rm \{\} \;`
  busca ficheros terminados en `.bak` (sin distinguir mayúsculas/minúsculas) y pregunta si se quiere borrar
- `find /home -iname \*.bak -exec mv \{\} /BAK \;`
  busca ficheros terminados en `.bak` y muevelos a `/BAK`
- `find / -wholename ´/home´ -prune -o -name \*.bak -ls`
  busca excluyendo el directorio `/home`
- `find . -perm 022`
  encuentra ficheros con permisos `-----w--w-`
- `find . -perm +022`
  encuentra ficheros escribibles por `grupo` **O** `otros`
- `find . -perm -022`
  encuentra ficheros escribibles por `grupo` **Y** `otros`
- `find . -perm -g=w,o=w`
  idéntico al anterior
- `find /home/httpd/html -name \*.html \`
  `> -exec grep -l expired \{\} \;`
  lista los nombres de los `.html` que tengan la palabra `expired`

Este último ejemplo funciona, pero es muy ineficiente (¿por qué?)

- otra forma de hacer lo mismo

  ```
  grep -l expired $(find /home/httpd/html -name \*.html)
  ```

si el número de ficheros `.html` es muy grande `grep` puede tener problemas

- podemos usar `xargs`

  ```
  find /home/httpd/html -name \*.html |\
  > xargs grep -l expired
  ```

Otros comandos para localizar ficheros

Existen otros comandos para la localización de ficheros: `which`, `whereis`, `locate`

Comando `which` muestra la localización de comandos

- Formato:

  `which [-a]` *`comando`*

- Opciones:
  - `-a` muestra todas las localizaciones del comando
- Ejemplo:

```
$ which ls
/bin/ls
```

Comando `whereis` muestra la localización del binario, fuente y página de manual de un comando

- Formato:

  `whereis [`*`opciones`*`]` *`comando`*

- Opciones:
  - `-b/-m/-s` muestra sólo el binario/manual/fuente
- Ejemplo:

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

- Para más opciones ver la página de manual

Comando `locate` localiza ficheros rápidamente

- utiliza una base de datos donde guarda la localización de los ficheros (`/var/cache/locate/locatedb`)
- esa base de datos la crea y actualiza el administrador con el comando `updatedb`

- Ejemplo:

```
$ locate \*.bak
/root/.mozilla/firefox/2gwif.default/bookmarks.bak
/var/backups/group.bak
/var/backups/inetd.conf.bak
```

- Ver página de manual para opciones

2.4. Particiones y sistemas de ficheros

Vimos en el tema 2 como crear particiones y sistemas de ficheros en el momento de la instalación

- si añadimos un nuevo disco al sistema ya instalado deberemos crear las particiones y los sistemas de ficheros
- esta operación implica los siguientes pasos:
  1. creación de particiones (comando `fdisk`)
  2. creación de los sistemas de ficheros (comando `mkfs`)
  3. montado de los sistemas de ficheros (comando `mount`)

Creación de particiones

El comando para crear particiones es `fdisk`

- Formato:

  `fdisk [`*`opciones`*`] `*`dispositivo`*

  donde *`dispositivo`* es el dispositivo del disco (`/dev/hd`*`x`* en IDE, `/dev/sd`*`x`* para SCSI o SATA)

- Debemos tener permiso de administrador para usarlo
- Opciones:
  - `-l` muestra la tabla de particiones del dispositivo

`fdisk` se usa mediante un menú:

```
# fdisk /dev/hdb
The number of cylinders for this disk is set to 20805.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Para crear una partición primaria de 5 GB usamos `n` (*new*):

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-20805, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-20805, default 20805): +5G

Command (m for help): p

Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  83  Linux
```

Por defecto, crea la partición tipo Linux (Id 83)

- con `l` (*list*) vemos el tipo de particiones soportadas
- para cambiar el tipo de partición se usa `t` (type)

```
Command (m for help): t 1
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  82  Linux swap / Solaris
```

Para que se guarden los cambios debemos usar `w` (*write*)

Otras herramientas `fdisk` escribe la tabla de particiones $\longrightarrow$ el contenido de los discos se pierde

Existen otras herramientas para modificar las particiones:

cfdisk interfaz para el `fdisk` (también escribe la tabla de particiones)

parted programa de GNU que permite crear, destruir, cambiar el tamaño, chequear y copiar particiones

qtparted programa Linux para manejar particiones, con interfaz gráfico (basado en `libparted`)

Creación de los sistemas de ficheros

Sobre cada partición debemos crear sistemas de ficheros con el comando `mkfs`

- Formato:

  `mkfs [-V] [-t` *`filesystem`*`] `*`dispositivo`*` [`*`n_bloques`*`]`

- Opciones:
  - `-V` verbose
  - `-t` *`filesystem`* tipo de sistema de ficheros a crear (ext2/3/4, xfs, etc.)
    - si no se especifica se crea el por defecto del sistema (en Linux ext2)
  - *`n_bloques`* número de bloques usados para el sistema de ficheros (si no se pone, se usa toda la partición)

`mkfs` es un *front-end* a distintos comandos, que permiten crear particiones de los tipos específicos:

- `mkfs.ext2` o `mke2fs` crea sistemas ext2
- `mkfs.ext3` crea sistemas ext3, equivalente a `mkfs.ext2 -j`
- `mkfs.jfs`, `mkfs.reiserfs`, `mkfs.xfs` crean sistemas JFS, ReiserFS y XFS, respectivamente
- `mkfs.msdos`, `mkfs.vfat` crea sistemas MS-DOS
- `mkswap` crea un sistema de ficheros de tipo Linux swap

Cada uno de estos comandos pueden tener distintas opciones

- ver las páginas de manual para más detalles

Comandos relacionados

- `dumpe2fs` muestra información de sistemas de ficheros ext2/3/4
  - información sobre inodos, bloques y grupos
- `tune2fs` permite ajustar parámetros en sistemas ext2/3/4
  - p.e. define el intervalo entre chequeos automáticos, convierte ext2 en ext3, etc.

- `e2label` cambia la etiqueta de un sistema ext2/3/4
- existen comandos similares para otros tipos de sistemas de ficheros, p.e. `reiserfstune`, `jfs_tune`, etc.

Partición de intercambio

- Si lo que creamos es una partición de intercambio, la debemos inicializar con `mkswap`

```
# fdisk -l /dev/hdb
Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  83  Linux
/dev/hdb2            9690       20805     5602464   82  Linux swap / Solaris
# mkswap /dev/hdb2
Setting up swapspace version 1, size = 5736919 kB
no label, UUID=a6c2849b-c33e-478e-8a8d-fecfe3f18f6d
```

- Una vez creada debemos activarla con `swapon`

```
# swapon /dev/hdb2
# swapon -s
Filename                        Type            Size    Used    Priority
/dev/hda7                       partition       377488  0       -1
/dev/hdb2                       partition       5602456 0       -2
```

- Finalmente, para que se active en el arranque, debe incluirse la entrada correspondiente en el fichero `/etc/fstab`

```
/dev/hda2   none   swap    sw       0       0
```

Montado de los sistemas de ficheros

Para poder acceder a los sistemas de ficheros debemos *montarlos*

- los comandos para montar y desmontar son: `mount` y `umount`

Comando `mount` Permite asociar (*montar*) directorios a sistemas de ficheros

- Ejemplo

  ```
  $ mount -t ext3 /dev/hdb1 /home2
  ```

- Formato

  `mount [`*`opciones`*`] [-t `*`tipo`*`] [`*`disp.`*`] [`*`dir.`*`]`

- Algunas opciones:

  - *`tipo`* tipo de sistema de ficheros (ext2/3/4, reiserfs, vfat, etc.); si se pone `auto` intenta determinar de forma automágica
  - `-a` monta los filesystems listados en `/etc/fstab`
  - `-r`/`-w` monta los sistemas de sólo lectura/escritura
  - `-f` simulación; usado con `-v` (verbose) para chequear
  - `-n` monta sin añadir la entrada a `/etc/mtab`; se usa cuando `/etc` es sólo lectura
  - `-o` *`opciones`* opciones de montado; formato igual al usado en el fichero `fstab`

Comando `umount` Desmonta los sistemas de ficheros

- Formato

  `umount [`*`opciones] directorio`*

- Ejemplo:

  ```
  $ umount /home2
  ```

- Algunas opciones

  - `-a` desmonta los filesystems listados en `/etc/mtab`
  - `-r` si falla, intenta remontar sólo lectura
  - `-f` fuerza el desmontado

- Si hay algún proceso bloqueando el filesystem, este no se puede desmontar:

  - usar el comando `fuser -c` *`directorio`* para ver el PID del proceso

Fichero `/etc/fstab` Al iniciar el sistema se montan los filesystems listados en `/etc/fstab`

- cada línea del fichero tiene las siguientes columnas

  ```
  (file system) (mount point)   (tipo)  (opciones)       (dump)  (pass)
  ```

- Ejemplo:

  ```
  /dev/hda9       /home           ext3    defaults           0       2
  ```

- Alguna de las opciones son:

  - `rw` monta tipo lectura/escritura
  - `ro` sólo lectura
  - `auto`/`noauto` monta/no monta con `mount -a` (monta/no monta al inicio)
  - `exec`/`noexec` Permite/no permite la ejecución de ficheros binarios en la partición
  - `suid`/`nosuid` permite/no permite que los bits setuid y setgid tengan efecto
  - `dev`/`nodev` interpreta/no interpreta dispositivos de bloques o caracteres en el filesystem
  - `async` toda la I/O se realiza de forma asíncrona
  - `user` puede montarlo un usuario (y sólo el que lo monta puede desmontarlo); implica las opciones `noexec`, `nosuid` y `nodev`, a menos que se fuercen (p.e. `user,exec,suid,dev`)
  - `users` puede montarlo/desmontarlo un usuario y el que desmonta no tiene que ser el que lo montó; implica las mismas opciones que `user`
  - `defaults` selecciona opciones por defecto (`rw`, `suid`, `dev`, `exec`, `auto`, `nouser` y `async`)

- Filesystems específicos pueden tener opciones específicas:

  - ver el manual de `mount` para más detalles

- Si un directorio aparece listado en el `fstab` puede montarse sin especificar el dispositivo:

  ```
  $ mount /home
  ```

- Campos `dump` y `pass`

  dump lo usa el comando `dump` para determinar de que filesystems hacer copias de seguridad

  - valor 1 o 0 según si la partición va a tener un backup controlado por `dump` o no (normalmente no se usa)

  pass lo usa el comando `fsck` para determinar el orden en que se chequean los filesystems al iniciar el sistema

  - si 0, el filesystem no se chequea
  - si > 0, los filesystems se chequean en el orden indicado por los números
    - si varios tienen el mismo número, se chequean en paralelo (si es posible)
    - normalmente `/` tendrá 1 y el resto 2

Fichero `/etc/mtab` Contiene una lista de los filesystem que están montados en el sistema

- Ejemplo de fichero `/etc/mtab`

```
$ cat /etc/mtab
/dev/hda1 / ext3 rw,errors=remount-ro 0 0
proc /proc proc rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev/hda9 /home ext3 rw 0 0
/dev/hda8 /tmp ext3 rw 0 0
/dev/hda5 /usr ext3 rw 0 0
/dev/hda6 /var ext3 rw 0 0
usbfs /proc/bus/usb usbfs rw 0 0
/dev/hdb1 /home2 ext2 rw,nodev 0 0
```

Autofs Sistema que permite montar los filesystems "bajo demanda"

- cuando se accede al directorio, este se monta
- se desmonta automáticamente después de un tiempo de inactividad (por defecto, 5 minutos)
- suele usarse para montar sistemas remotos con NFS

- Ficheros de configuración:
  - `/etc/auto.master` define los puntos de montado
    - por cada uno de los puntos definidos, se inicia un proceso `automount` usando los parámetros indicados
    - Ejemplo de `auto.master`:

```
/home      /etc/auto.home
/misc      /etc/auto.misc   --timeout 60
```

  - Los ficheros le indican al `automount` los filesystems a montar
  - Ejemplo de `auto.misc`

```
cdrom      -fstype=iso9660,ro  :/dev/cdrom
windoz     -fstype=vfat        :/dev/hda1 home     -rw,hard,intr
server:/export/home
```

  - esto monta el cdrom y la partición `/dev/hda1` en los directorios `/misc/cdrom` y `/misc/windoz`, y el directorio remoto `/export/home` del sistema `server` en `/misc/home`
- Para más información ver el manual de `autofs` y `automount`, el Autofs Automounter HOWTO o el Automount mini-Howto

Chequeo del sistema de ficheros

Periódicamente es necesario chequear los sistemas de ficheros

- el comando básico para chequeo y reparación es `fsck`

Al igual que `mkfs`, `fsck` es un front-end a comandos específicos para cada filesystem:

- `e2fsck`, `fsck.ext2` o `fsck.ext3` chequean sistemas ext2/ext3
- `fsck.jfs`, `fsck.reiserfs`, `fsck.xfs` para JFS, ReiserFS y XFS
- `fsck.msdos`, `fsck.vfat` para sistemas MS-DOS

Alguno de los errores que pueden aparecer se deben a:

- Varios ficheros que usan el mismo bloque
- Bloques marcados libres y ocupados simultáneamente
- Número de enlaces erróneo

- Nodos-i conteniendo información pero que no están en la entrada del directorio (la información se recupera en el directorio `lost+found` con el número de nodo-i)
- Entradas del directorio que apuntan a nodos-i ilegales o vacíos
- etc.

Algunas de las opciones de `fsck` son:

- `-t` *`filesystem`* tipo de filesystem a chequear
- `-A` chequea los filesystems listados en `/etc/fstab`
- `-N` no ejecuta; simplemente indica lo que haría
- `-R` usado con `-A` no chequea el filesystem raíz

Otras opciones dependen del filesystem particular

- ver las páginas de manual para cada caso

Otras utilidades

- `du`: muestra el espacio de disco usado por los ficheros y subdirectorios de un directorio
  - Formato:

    `du [`*`opciones`*`] [`*`directorio`*`]`
  - Algunas opciones:
    - `-a` muestra valores para ficheros y directorios (por defecto, solo muestra directorios)
    - `-b`, `-k` tamaños en bytes/KBytes
    - `-h` salida más legible
    - `-s` muestra sólo la ocupación total
  - Ejemplo:

```
$ du -sh /home /usr
1,2G    /home
2,3G    /usr
```

- `df`: muestra el espacio de disco usado y disponible de los sistemas de ficheros montados

- Formato:

  `df [`*`opciones`*`]`

- Algunas opciones:
  - `-a` muestra todos los filesystems (incluso los de tamaño 0)
  - `-h` salida más legible
  - `-i` da información sobre los inodos
  - `-l` sólo muestra filesystems locales
  - `-T` muestra el tipo de sistema de ficheros
- Ejemplo:

```
$ df -h
Filesystem          Tamaño Usado  Disp Uso% Montado en
/dev/hda1              67M   50M   13M  80% /
tmpfs                  63M     0   63M   0% /dev/shm
/dev/hda9             272M  8,1M  250M   4% /home
/dev/hda8              23M  1,1M   20M   5% /tmp
/dev/hda5             464M   90M  350M  21% /usr
/dev/hda6              74M   44M   27M  63% /var
```

2.5. Sistemas de ficheros con LVM

En el tema 2 vimos como crear un sistema LVM; algunas de sus ventajas son:

- LVM proporciona mucha más flexibilidad a la hora de distribuir el espacio disponible
- LVM permite mover y cambiar de tamaño los volúmenes creados bajo su control
- Existen varios beneficios inmediatos por usar LVM:
  - Es posible aumentar y decrecer los volúmenes en caliente: esto permite redistribuir el espacio en las particiones según nos sea necesario; también se puede dejar espacio sin asignar e ir asignándolo según vaya siendo necesario
  - Es posible añadir espacio de almacenamiento al sistema de volúmenes: si se añade un nuevo disco a la máquina se puede añadir este espacio a LVM y hacer crecer los volúmenes que contiene para aprovechar el nuevo espacio de almacenamiento

En este apartado veremos comandos para manejar sistemas LVM (nota: todos estos comandos tienen distintas opciones, ver páginas de manual)

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

```
# vgdisplay GrupoVolumen
  --- Volume group ---
  VG Name               GrupoVolumen
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                6
  Open LV               6
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               14,84 GB
  PE Size               32,00 MB
  Total PE              475
  Alloc PE / Size       473 / 14,78 GB
  Free  PE / Size       2 / 64,00 MB
  VG UUID               N2NjFx-7ISe-J7hH-EX03-231w-XfbS-eCYfv0
```

- Información acerca de un volumen lógico: `lvdisplay` o `lvs`

```
# lvdisplay /dev/GrupoVolumen/homelv
  --- Logical volume ---
  LV Name                /dev/GrupoVolumen/homelv
  VG Name                GrupoVolumen
  LV UUID                dI6BqF-LAeG-3fO9-jXcr-vNde-f7iF-y9Oa2l
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                1,00 GB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:1
```

- Información acerca de un volumen físico: `pvdisplay` o `pvs`

```
# pvdisplay /dev/hda2
  --- Physical volume ---
  PV Name               /dev/hda2
  VG Name               GrupoVolumen
  PV Size               9,88 GB / not usable 0
  Allocatable           yes (but full)
  PE Size (KByte)       32768
  Total PE              316
  Free PE               0
  Allocated PE          316
  PV UUID               U6rMMw-5Z9U-fhBH-4R6G-reeJ-ZVha-K4xyHs
```

Manejar volúmenes físicos y grupos de volúmenes

- Creación de un volumen físico (PV), sobre una partición tipo Linux LVM (8e)

  ```
  # pvcreate /dev/hdc1
  ```

- Crear un grupo de volúmenes (VG), de nombre NuevoGrupo a partir de dos PVs

  ```
  # vgcreate NuevoGrupo /dev/hda2 /dev/hdc1
  ```

- Activar un grupo de volúmenes: es necesario hacer esto antes de usarlo, pero normalmente el sistema lo hace por nosotros en el arranque

  ```
  # vgchange -a y NuevoGrupo
  ```

- Borrar un VG (es necesario desactivarlo antes)

  ```
  # vgchange -a n NuevoGrupo
  # vgremove NuevoGrupo
  ```

- Añadir el PV `/dev/hdc1` a un VG ya creado

  ```
  # vgextend GrupoVolumen /dev/hdc1
  ```

- Quitar PVs de un VG

  ```
  # vgreduce NuevoGrupo /dev/hda2
  ```

Trabajar con volúmenes lógicos

- Crear un volumen lógico (LV) de nombre testlv en el VG NuevoGrupo con un tamaño de 4.20 GB

```
# lvcreate -L4.20G -n testlv NuevoGrupo
```

- Crear un volumen lógico con nombre otrotestlv, con 2 stripes (número de volúmenes físicos entre los que se reparte el LV)

```
# lvcreate -i2 -L1G -n otrotestlv vg
```

- Destruir un volumen lógico (hay que desmontarlo antes)

```
# umount /dev/NuevoGrupo/otrotestlv
# lvremove /dev/NuevoGrupo/otrotestlv
```

- Agrandar un LV; se puede especificar el nuevo tamaño en bytes (`-L`) o LEs (`-l`), o la diferencia

```
# lvextend -L12G /dev/GrupoVolumen/homelv
# lvextend -L+1G /dev/GrupoVolumen/tmplv
# lvextend -l+200 /dev/GrupoVolumen/tmplv
```

- Reducir un LV: `lvreduce` funciona igual que el `lvextend`

Una vez creados los volúmenes lógicos sólo resta crear los sistemas de ficheros y montarlos

- se hace de la forma habitual (`mkfs`, `mount`)

```
# mkfs.ext3 /dev/GrupoVolumen/homelv
# mount -t ext3 /dev/GrupoVolumen/homelv /home
```

Si hemos agrandado un LV debemos agrandar el filesystem: comando `fsadm`

- Chequea y redimensiona un sistema de ficheros
- Front-end para los comandos específicos de diferentes filesystems (ext2/3/4, ReiserFS, XFS)

Ejemplo: aumenta a 2G el tamaño del filesystem

```
# fsadm resize /dev/mapper/volgr-usr 2048M
```

Si no se especifica tamaño, aumenta al máximo posible
Otros comandos dependen de cada filesystem particular:

ext2/3/4 comando `resize2fs` (en Debian, paquete `e2fsprogs`): permite alargar o encoger un filesystem ext2/3/etx4[4]

[4]en kernel 2.6 y ext3/4 no es necesario desmontar el filesystem

```
# resize2fs /dev/GrupoVolumen/homelv
```

ReiserFS comando `resize_reiserfs` o `resizefs.reiserfs` (en Debian, paquete `reiserfsprogs`)

XFS el filesystem debe montarse que sea extensible; se usa el comando `xfs_growfs` (en Debian, paquete `xfsprogs`)

JFS como XFS el filesystem debe montarse que sea extensible; se especifica el tamaño en el proceso de remontado:

```
# mount -o remount,resize=1048576 /home
```

2.6. Manejo de discos cifrados

El cifrado que se usa en el proceso de instalación es un cifrado de disco completo (*Full disk encryption*, FDE)

- Se cifran todos los bits del disco o de la partición
- Es diferente del cifrado a nivel de sistema de ficheros (*Filesystem-level encryption*, FLE) en el que se cifra el contenido de los ficheros, no los metadatos (nombre del fichero, fechas de modificación, etc.).

El subsistema de cifrado FDE en Linux 2.6 es *dm-crypt*

- Parte de la infraestructura *device mapper*, utilizada también en LVM2 o RAID software, y puede colocarse por encima de estos
  - Permite cifrar discos completos, particiones, volúmenes lógicos o volúmenes RAI software
- Comando básico: `cryptsetup`
- Permite utilizar el estándar LUKS (*Linux Unified Key Setup*)
- Fichero `/etc/crypttab`: indica en el arranque como descifrar los discos

LUKS

Estándar para cifrado de disco en Linux

- Facilita la compatibilidad entre distribuciones
- Incluye soporte para múltiples claves
- Revocación de contraseña efectiva
- Uso mediante el comando *cryptsetup*, con *dm-crypt* como backend

cryptsetup

Comando básico para el cifrado de disco

- Puede cifrar con o sin LUKS
- LUKS no permite cifrado con clave aleatoria, generada a partir de `/dev/random` o `/dev/urandom`[5]

Ejemplos de uso:

1. Cifrar y activar una partición usando formato LUKS y una contraseña como clave (todos los datos se pierden y debemos reinicir el filesystem)

```
# Para mayor seguridad, desmonta y sobreescribe
# la partición (puede ser muy lento)
umount /dev/hda8
dd if=/dev/urandom of=/dev/hda8
# Formatea la partición, cifrando con LUKS
cryptsetup luksFormat /dev/hda8
# Activa la partición, en el
# dispositivo /dev/mapper/hda8_crypt
cryptsetup luksOpen /dev/hda8 hda8_crypt
# Reinicia el sistema de ficheros
mkfs -t fstipo /dev/mapper/hda8_crypt
# Obtiene el UUID luks
crypsetup luksUUID /dev/hda8
```

2. Cifrar una partición de swap usando `/dev/urandom` como clave aleatoria (sin usar LUKS)

```
cryptsetup create hda7_crypt /dev/hda7 --key-file /dev/urandom
```

3. Usar un fichero de clave para una partición cifrada inicialmente con contraseña, eliminando la contraseña inicial

```
 # Crea un fichero aleatorio para usar como clave
dd if=/dev/urandom of=/etc/keys/hda6.luks bs=1 count=4096
# Cambia los permisos del fichero (debe ser de root)
chown root.root /etc/keys/hda6.luks
```

[5] `/dev/random`: genera números aleatorios basándose en la entropía (ruido) del sistema; `/dev/urandom`: genera números pseudoaleatorios usando la entropía como semilla. `/dev/random` genera números de mayor calidad, pero es lento y puede bloquearse si la entropía del sistema es baja; `/dev/urandom` es un poco menos seguro, pero puede ser adecuado

```
chmod 700 /etc/keys
chmod 400 /etc/keys/hda6.luks
# Añade el fichero hda6.luks como clave
cryptsetup luksAddKey /dev/hda6 /etc/keys/hda6.luks
# Comprueba que existen dos claves (slots)
cryptsetup luksDump /dev/hda6
# Borra el slot 0 con la clave original
# (impide usar esa clave, hacerlo solo después de comprobar
que el fichero luks funciona como clave)
cryptsetup luksKillSlot /dev/hda6 0 --key-file /etc/keys/hda6.luks
# Comprueba que el slot se ha borrado
cryptsetup luksDump /dev/mapper/hda6_crypt
# Por último, modifica el fichero crypttab para indicar
que se use el fichero luks
```

4. Extiende el dispositivo cifrado `GV-homelv_crypt`, después de haber extendido el volumen lógico sobre el que está definido

```
# cryptsetup resize /dev/mapper/GV-homelv_crypt
```

Fichero /etc/crypttab

Especifica en el arranque como se deben descifrar los discos
Ejemplo:

```
#<cifrado>  <original> <fichero clave>      <opciones>
hda7_crypt  /dev/hda7  /dev/urandom         cipher=aes-cbc-essiv:sha256,size=256,swap
hda8_crypt  /dev/hda8  none                 luks
hda6_crypt  /dev/hda6  /etc/keys/hda6.luks  luks
```

Línea 1 área de swap con cifrado aleatorio

Línea 2 partición con cifrado LUKS; `none` indica que se pide una contraseña en el arranque

Línea 3 partición con cifrado LUKS y clave obtenida desde fichero

Para evitar problemas (posibles cambios en el nombre de las particiones), es preferible substituir el nombre del dispositivo original por su UUID obtenido usando `crypsetup luksUUID`

```
# cryptsetup luksUUID /dev/hda8
0e44dcc3-2b1f-4349-9fb3-db82b547acd1
# cat /etc/crypttab
.....
hda8_crypt UUID=0e44dcc3-2b1f-4349-9fb3-db82b547acd1 none luks
.....
```

3. Gestión de usuarios

Todo usuario de un sistema UNIX debe tener una cuenta para poder acceder

- Cuenta UNIX: colección de características lógicas que especifican quien es el usuario y lo que puede hacer en el sistema
- Estas características incluyen:
  - el nombre de usuario (*login* o *user name*)
  - la contraseña (*passwd*)
  - grupo o grupos a los que pertenece
  - un identificador numérico (UID)
  - un identificador numérico del grupo por defecto (GID)
  - un directorio *home*
  - un *login shell*
  - una colección de ficheros de inicio

Dentro de las cuentas asociadas a usuarios podemos encontrar diferentes tipos.

- cuentas normales de usuario
- cuenta del administrador (*root*)
- cuentas especiales de los servicios (*nobody*, *lp*, *bin*, etc.):
  - usadas por servicios internos del sistema
  - aumentan la seguridad, al permitir que servicios del sistema no se ejecuten como *root*

3.1. Ficheros de información de los usuarios

La información de los usuarios y grupos está incluida en los siguientes archivos:

- `/etc/passwd` mantiene la información principal de cada cuenta: nombre de usuario, UID, GID, *login shell*, directorio *home*, contraseña (en sistemas antiguos), ...
- `/etc/shadow` en sistemas actuales, fichero sin permiso de lectura que guarda las contraseñas encriptadas

- `/etc/group` información sobre los grupos definidos en el sistema. nombre del grupo, GID y miembros del mismo
- `/etc/gshadow` contraseñas para grupos (no suele usarse)

Fichero `/etc/passwd`

Ejemplo de líneas de `/etc/passwd`:

```
root:x:0:0:root:/root:/bin/bash
pepe:x:1002:1002:Pepe Pótamo,123,981234321,:/home/pepe:/bin/bash
```

donde se indican (si aparecen `::` seguidos, el campo está vacío):

- `pepe`: identificación de usuario en el sistema, que deberían tener las siguientes características
  - únicos en toda la organización (no sólo en la máquina local)
  - preferiblemente corto, en minúsculas y sin caracteres acentuados (para evitar problemas)
  - fácil de recordar
  - de formato fijo para todos los usuarios (p.e. nombre+apellido)
- `x`: contraseña encriptada
  - si aparece una `x` la contraseña está en el fichero `/etc/shadow`
- `1002`: UID número identificador del usuario
  - para usuarios normales, número entre 1000 y 32767 (o 65535 en sistemas actuales)
  - números por debajo de 1000 para usuarios especiales del sistema (`root` usualmente número 0)
  - el UID para un usuario debería ser único, y el mismo para todas las máquinas
  - se debe evitar reutilizar un UID, para evitar problemas de pertenencia de archivos
- `1002`: GID código del grupo principal al que pertenece el usuario
- `Pepe Pótamo,123,...`: información GECOS
  - cualquier cosa, usualmente el nombre completo del usuario y información adicional (n. de despacho, teléfono, etc.)

- `/home/pepe`: directorio personal del usuario
- `/bin/bash`: shell interactivo que utilizará el usuario

Fichero `/etc/shadow`

Fichero de acceso restringido que almacena las contraseñas encriptadas:

```
pepe:$1$.QKDPc5E$SWlkjRWexrXYgc98F.:12825:0:90:5:30:13096:
```

Contiene para cada usuario la contraseña encriptada y otros campos separados por :

- día, contado como número de días desde el 1/1/1970 (también conocido como *epoch*), en que la contraseña se cambió por última vez
  - si vale 0 se fuerza a que el usuario cambia su contraseña la primera vez que se conecta
- número de días antes de que pueda ser cambiada
- número de días de validez de la contraseña
- días en que se avisa al usuario de que la contraseña va a caducar
- días, una vez expirada, en que se deshabilitará la cuenta
- día, desde el 1/1/1970, en que la cuenta se inhabilitará
  - si no aparece nada, la cuenta no se inhabilita nunca
- un campo reservado

Fichero `/etc/group`

Información sobre los grupos de usuarios

```
users:x:100:pepe,elena
```

donde tenemos

- nombre del grupo
- contraseña del grupo (no suele usarse)
  - si `x`, se guarda en el fichero `/etc/gshadow`

`grupo:contraseña:administradores:miembros`

- los administradores pueden cambiar la contraseña, añadir usuarios al grupo, etc.

- la contraseña puede fijarse/cambiarse con el comando `gpasswd`

- GID identificador numérico del grupo

- lista de usuarios que pertenecen al grupo

Cambio de grupo

- un usuario puede cambiar de grupo con `newgrp`

  - si el grupo no tiene contraseña y no está en `gshadow` sólo pueden cambiar los miembros del grupo
  - si el grupo tiene contraseña, el usuario debe especificar la contraseña
  - si el grupo aparece en `gshadow`, la lista de miembros en este fichero pueden cambiar sin contraseña

Otros ficheros

Cuando se crea un nuevo usuario, los ficheros de inicio se copia del directorio `/etc/skel`

- el administrador debe crear unos ficheros adecuados para los usuarios, especificando los paths necesarios de ejecución, inicialización de variables del sistema, etc.

- también pueden usarse los ficheros `/etc/profile` o `/etc/bash.bashrc` (ver Tema 3, *Ficheros de inicialización de Bash*)

3.2. Creación manual de una cuenta

Implica los siguientes pasos.

1. Editar el fichero `/etc/passwd` y añadir una línea para la nueva cuenta

   - para evitar corrupción del fichero usar el comando `vipw`

2. Si usamos `shadow`, poner `x` en el campo de contraseña y editar el fichero `/etc/shadow`

   - para evitar corrupción del fichero usar el comando `vipw -s`

3. Editar `/etc/group` para añadir un nuevo grupo, si es necesario, o para añadir el usuario a los grupos que deseemos
    - si es necesario, editar `/etc/gshadow`
4. Crear el directorio del usuario
5. Copiar los ficheros de `/etc/skel` al directorio del usuario
6. Usar `chown`, `chgrp` y `chmod` para fijar el usuario, grupo y permisos del directorio
7. Fijar la contraseña con `passwd`
    - el usuario debería cambiar la contraseña tan pronto como fuera posible
    - puede forzarse modificando el fichero `shadow` o con la opción `-e`

Comando `passwd`

Permite fijar o cambiar la contraseña de un usuario. Opciones:

- `-e`, `--expire` fuerza a que la contraseña de la cuenta caduque; el usuario debe cambiarla en el siguiente login
- `-d`, `--delete` borra la contraseña
- `-l`/`-u`, `--lock`/`--unlock` bloquea/desbloquea la cuenta
- `-m`, `--mindays MIN_DAYS` número mínimo de días entre cambios de contraseña
- `-x`, `--maxdays DÍAS_MAX` número de días de validez de la contraseña
- `-w`, `--warndays DÍAS_AVISO` número de días de aviso de caducidad
- `-i`, `--inactive INACTIVO` días en que se deshabilitará la cuenta una vez expirada la contraseña
- `-S`, `--status` indica el estado de la contraseña (bloqueada `L`, sin contraseña `NP` o con contraseña válida `P`)

Eliminación manual de una cuenta

Implica los siguientes pasos

1. Inhabilitar la cuenta impidiendo el acceso
2. Hacer un backup de los ficheros de usuario
3. Eliminar los ficheros de usuario

La inhabilitación de la cuenta podría ser temporal o definitiva

- Temporal: puede hacerse de diversas formas
  - cambiar el login shell a `/bin/false`, o `/usr/sbin/nologin` (si disponible)
  - cambiar el campo contraseña de `/etc/passwd` a `*` (volviendo a poner `x` se habilita la cuenta),
  - poner una `!` al principio del campo contraseña de `/etc/shadow`,
  - usar `passwd -l`
- definitiva: borrar las entradas del usuario de `/etc/passwd` y `/etc/shadow`

3.3. Comandos para gestión de cuentas

Crear cuentas manualmente es un proceso tedioso:

- existen comandos que nos ayudan en la tarea

Comandos simples de manejo de cuentas

- `useradd` añade un nuevo usuario al sistema; ejemplo.

  ```
  useradd -c .Aitor Tilla.aitor
  ```

  - por defecto, sólo modifica los ficheros `passwd` y `shadow`, no crea el directorio home ni le pone contraseña (cuenta inhabilitada)
  - varias opciones:
    - `-m` crea el directorio home, si no existe (y copia los ficheros de /etc/skel)
    - `-g` *`grupo`* especifica el grupo principal
    - `-s` *`shell`* especifica la shell a utilizar

    - `-e` *`fecha`* fecha de expiración de la cuenta (formato YYYY-MM-DD)
  - Ejemplo:

    ```
    useradd -c .ªitor Tilla.ªitor -m -e 2006-11-02 -s
    /bin/bash -g staff
    ```

- `userdel` borra un usuario del sistema
- `usermod` modifica las cuentas de usuario
- `groupadd`, `groupdel`, `groupmod` lo mismo, para grupos
- `newusers` permite crear varias cuentas a partir de un fichero con nombres de usuario y contraseñas
  - las líneas del fichero deben tener el mismo formato que las del fichero `/etc/passwd`, con la contraseña sin encriptar
- `chpasswd` similar al anterior, permite actualizar las contraseñas de usuarios existentes:

  ```
  echo "pepe:pepepassword"| chpasswd
  ```

- `mkpasswd` obtiene la versión cifrada de una cadena para usar como contraseña:

  ```
  mkpasswd -m sha-512 mypassword
  ```

- `chsh` cambia el shell por defecto del usuario
- `chfn` cambia la información del campo GECOS

Comandos de alto nivel para el manejo de cuentas

- Comandos `adduser`, `addgroup`:
  - hacen de front-end de los de bajo nivel `useradd`, `groupadd` y `usermod`
  - crean los usuarios/grupos en función de la configuración especificada en el fichero `/etc/adduser.conf`
- Herramientas gráficas de gestión de usuarios y grupos:
  - `kuser` (KDE), `user-admin` (GNOME), etc.

Otros comandos relacionados

- `passwd`: permite cambiar la contraseña (ya comentado)
- `chage`: muestra y cambia la información de expiración de la contraseña
  - Formato:

    `chage [`*`opciones`*`] [`*`username`*`]`
  - Algunas opciones:
    - `-l` muestra información de expiración
- `su`: permite cambiar de usuario o pasar a ser administrador
  - Formato:

    `su [`*`opciones`*`] [-]` *`username`*
  - Si no se especifica el *`username`* pasa a administrador
  - Algunas opciones:
    - `-` inicia un login shell
    - `-m`, `-p` o `--preserve-environment` mantiene el entorno (no ejecuta el `.bashrc` del nuevo usuario)
    - `-s`, `--shell=`*`nueva_shell`* usa la shell especificada
    - `-c`, `--commmand=`*`comando`* ejecuta el comando con la identidad del nuevo usuario:

      ```
      su -c ´cat /etc/shadow´
      ```

3.4. Cuotas de disco

Algunos filesystems permiten limitar el uso del disco a los usuarios y grupos: **cuotas**

- Evitan que los usuarios monopolicen el disco
- Pueden causar problemas a los usuarios:
  - preferible instalar más disco o avisar a los usuarios que consuman demasiado

Límites de cuotas:

- **Límite débil**: si la cuenta del usuario o del grupo supera el límite débil, se impondrá un *período de gracia* en el que el usuario podrá reducir la ocupación

- **Límite duro**: se deniega cualquier intento de escribir datos después de este límite
- **Período de gracia**: tras superar el límite débil, si el usuario no resuelve el problema borrando archivos, la cuenta se bloquea

Cuotas de usuario y de grupos

- **Usuario**: fija un máximo al espacio de todos los ficheros del usuario
- **Grupo**: fija un máximo al espacio de todos los ficheros del grupo
  - puede incluir ficheros de varios usuarios

Instalación de cuotas de disco en Debian

Los pasos a seguir son:

1. Instalar el paquete `quota`
2. Modificar `/etc/fstab` para marcar los filesystems que tendrán cuotas:

   ```
   /dev/hda9  /home  ext4 defaults,usrjquota=aquota.user,grpjquota=aquota.group
   ```

3. Remontar el filesystem que hemos modificado

   ```
   mount -vo remount /home
   ```

4. Crear los índices de las cuotas (ficheros aquota)

   ```
   quotacheck -vguma
   ```

5. Activar las quotas:

   ```
   quotaon -va
   ```

6. Usar el comando `edquota` para editar las cuotas de usuarios y grupos

Comando `edquota`

Permite crear, manipular y eliminar cuotas basadas en usuarios o grupos

- Sintaxis:

  `edquota [`*opciones*`] [`*usuario*`|`*grupo*`]`

- Opciones:

- `-u` *`usuario`* configura las cuotas del usuario
- `-g` *`grupo`* configura las cuotas para un grupo
- `-f` *`filesystem`* realiza las operaciones obre un filesystem concreto (por defecto, lo hace sobre todos los filesystems que admitan cuotas)
- `-t` configura el período de gracia
- `-p` *`user1 usuarios`* copia la configuración de cuotas de *`user1`* a los usuarios indicados

- Al ejecutar `edquota` se abre el editor indicado en la variable `EDITOR` (por defecto, `vi`) para modificar las cuotas:
  - se muestran los bloques de 1K en uso, así como los límites *soft* y *hard* (también para i-nodos o ficheros)
    - si un límite está a 0 no se aplica
  - esta información se guarda en los ficheros `aquota.user` y `aquota.group` en el directorio base del filesystem

Otros comandos

Existen otros comandos para la gestión de las cuotas:

- `quotacheck` verifica la integridad de las bases de datos de las cuotas
  - se ejecuta en el script de inicio del sistema de cuotas
  - debe ejecutarse con las cuotas desactivadas
- `quotaon`/`quotaoff` activa/desactiva el sistema de cuotas
- `repquota` genera un informa del uso de las cuotas

```
# repquota /home
*** Report for user quotas on device /dev/hda9
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   34920       0       0              6     0     0
tarabelo  --     728       0       0             31     0     0
tomas     *-     108     100     200  7days       8     0     0
```

- `quota` permite al usuario ver el estado de sus cuotas
  - Algunas opciones:
    - `-g` muestra información sobre las cuotas del grupo del usuario
    - `-v` imprime información incluso para los filesystem sin límite en la cuota
    - `-q` imprime un mensaje si se ha superado la cuota
  - Ejemplo

```
$ quota
Disk quotas for user tomas (uid 1001):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hda9     108*    100     200   6days       9       0       0
      /dev/hda8       1      10      20               1       0       0
```

- Para más información sobre la configuración de las cuotas ver Quota mini-HOWTO

4. Instalación y configuración básica de redes de área local

Linux soporta múltiples protocolos y hardware de red:

- Protocolos como TCP/IP y TCP/IP versión 6, IPX/SPX, PPP, SLIP, X.25, Frame Relay, etc.
- Soporta hardware para redes Ethernet, Token-Ring, etc
- Diferentes NICs (*Network Interface Cards*) implican diferentes dispositivos de comunicación:
  - `eth`*x* para Ethernet, `tr`*x* para Token-Ring, `pppx` para PPP, `sl`*x* para SLIP,
  - Además, existe el dispositivo de *loopback* `lo`
    - Funciona como un circuito cerrado en el que cualquier datagrama que se le pase como parámetro es inmediatamente devuelto a la capa de red del sistema
    - Se utiliza para realizar pruebas, y para un par de aplicaciones de red

- En muchos UNIX estos dispositivos aparecen en `/dev`
  - En Linux se crean dinámicamente por software y no requieren los ficheros de dispositivos
- En Linux puede ser necesario incluir los módulos adecuados para cada dispositivo

En esta sección trataremos la configuración de TCP/IP en redes Ethernet; para más información ver:

- Administración de red en Linux: Linux Network Administrators Guide 2 ed., Olaf Kirch y Terry Dawson
- Linux Networking-HOWTO
- Dispositivos de red soportados en Linux: Linux Hardware Compatibility HOWTO - Network adapters

4.1. Comandos de configuración de red

Los comandos más importantes para configurar la red son:

- `ifconfig`: configuración del interfaz de red
- `route`: configuración del routing
- `netstat`: información de la red

Comando `ifconfig`

Muestra y configura una interfaz de red:

```
$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:12:43:A6:05:5C
          inet addr:193.144.84.77  Bcast:193.144.84.255  Mask:255.255.255.0
          inet6 addr: fe80::211:43ff:fea6:55c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1035446 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1053062 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:196973192 (187.8 MiB)  TX bytes:270128587 (257.6 MiB)
          Interrupt:169
```

- Sintaxis:

```
ifconfig [opciones] [interfaz]
ifconfig interfaz [configuración] [up|down]
```

- Opciones de visualización:
  - `-a` muestra todas las interfaces , incluso las inactivas
  - `-s` muestra información resumida (igual que `netstat -i`)
- En las opciones de configuración se indica entre otras cosas la IP, máscara de red y dirección de broadcast:

```
# ifconfig eth0 193.144.84.77 netmask 255.255.255.0 broadcast 193.144.84.255 up
```

- `ifconfig` permite también configurar el estado del interfaz, por ejemplo, cambiar el MTU, poner modo promiscuo, activar/desactivar ARP, cambiar su dirección hardware (si el dispositivo lo permite), etc.

```
# ifconfig eth0 mtu 500
# ifconfig eth0 -noarp
# ifconfig eth0 hw ether 52:54:00:12:34:56
```

- ver el manual de `ifconfig` para más información

Otros comandos relacionados

Otros comandos de configuración de interfaz son:

- `ifup/ifdown` activan/desactivan un interfaz de red

```
# ifdown eth0
```

- `iwconfig` configura un interfaz wireless

```
# iwconfig eth1 essid "Mi Red"
```

- `ip` muestra y modifica dispositivos y rutas
  - Alternativa a `ifconfig` y `route`
  - Más potente y complejo

Comando `route`

Permite modificar la tabla de routing, mostrando, añadiendo o borrando rutas

- muestra las rutas definidas
- permite añadir/borrar rutas estáticas
- permite definir un gateway de salida por defecto para conectarnos al exterior
- permite configurar el sistema para que actúe como un `router`

Mostrar una tabla de routing

Se usa `route [-n -e -ee]` (equivale a `netstat -r`)

```
$ /sbin/route -n -ee
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface    MSS   W
193.144.84.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0     0     0
0.0.0.0         193.144.84.1    0.0.0.0         UG    0      0        0 eth0     0     0
```

- Opciones:
  - `-n` usa direcciones IP en vez de nombres
  - `-e` emplea el mismo formato que `netstat -r`
  - `-ee` salida larga
- Los flags indican el estado de la ruta
  - `U` la interfaz está activa (Up)
  - `H` el destino es una estación (Host)
  - `G` la ruta usa una pasarela (Gateway)
  - `D` ruta creada dinámicamente por un demonio de encaminamiento o un mensaje ICMP de redirección
  - `M` ruta modificada dinámicamente
  - `R` ruta rehabilitada
  - `!` ruta rechazada
- De las siguientes columnas, algunas no se usan

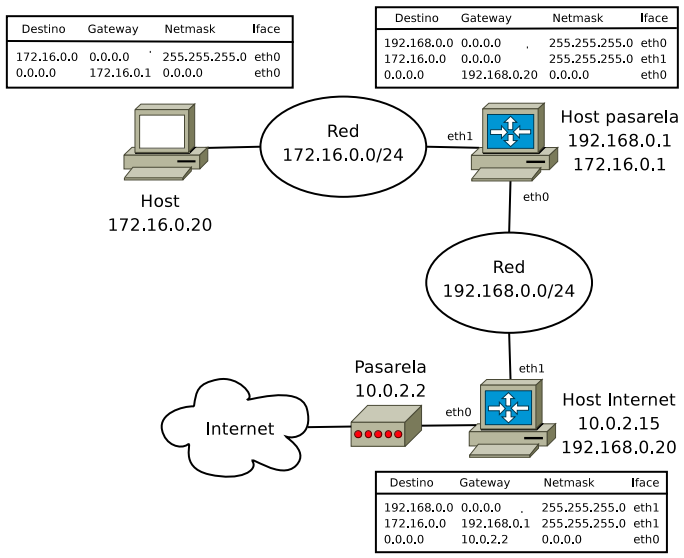
- `Metric` distancia (normalmente en saltos) al destino
- `Ref` número de referencias a la ruta (no usado en linux)
- `Use` número de consultas para la ruta
- `MSS` (*Maximum Segment Size*) tamaño máximo del segmento para las conexiones TCP en esa ruta
- `Window` Tamaño predeterminado de la ventana para las conexiones TCP en esa ruta
- `irtt` (*Initial Round Trip Time*) valor inicial del temporizador TCP

Añadir/borrar rutas estáticas

Se usa

```
route [add|del] [default] [-net|-host] target [netmask
Nm] [gw Gw] [opciones] [[dev] If]
```

Ejemplo: suponer que tenemos la configuración del dibujo, y queremos crear la tabla de rutas para el host Internet



- Añadir la ruta para la red 192.168.0.0/24 en `eth1`

```
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth1
```

- Añadir la ruta por defecto

```
route add default gw 10.0.2.2
```

- Añadir una ruta para la red 172.16.0.0/24, usando como pasarela en host con IP 192.168.0.1

```
route add -net 172.16.0.0 netmask 255.255.255.0 gw 192.168.0.1
```

- El host pasarela tiene que permitir routing entre sus interfaces; pasa eso debemos activar el *ip_forward*:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Otras opciones de configuración

Linux permite otras opciones para configurar la red, como definir alias de IP o configurar opciones sobre el tráfico

Alias de IP

- Permite configurar múltiples direcciones IP a un único dispositivo de red
  - podemos soportar varias subredes IP en una misma Ethernet
  - los alias se indican como *dispositivo:número*
- Ejemplo:

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Opciones del IP

Linux permite configurar diversas opciones sobre el tráfico IP

- los cambios pueden hacerse mediante el comando `sysctl`, escribiendo en los archivos del directorio `/proc/sys/net/ipv4` o de forma permanente en el fichero `/etc/sysctl.conf`
- algunos de estos estos archivos tienen un 0 (opción desactivada) o un 1 (opción activada)
- otros pueden tener un valor

- algunas de las opciones son:
  - `ip_forward` si 1 permite routing entre interfaces (por defecto 0)
  - `ip_default_ttl` el tiempo de vida por defecto de los paquetes (por defecto 64 ms)

Información de la red: comando `netstat`

`netstat` muestra las conexiones de red, tablas de routing y estadísticas de interfaz

- Formato:

  `netstat [`*tipo de información*`] [`*opciones*`]`

- Algunos tipos de información:
  - (nada) muestra la lista de sockets abiertos

```
$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp     0      0    jumilla.dec.usc.e:58946 aiff.usc.es:telnet      ESTABLISHED
tcp     0      0    jumilla.dec.usc.e:43658 ulla.dec.usc.es:1301    ESTABLISHED
tcp     0      0    jumilla.dec.usc.e:35346 sd.cesga.es:ssh         ESTABLISHED
tcp     0      0    jumilla.dec.usc.es:ssh  ulla.dec.usc.es:1688    ESTABLISHED
tcp     0      0    jumilla.dec.usc.es:ssh  teneguia.dec.usc.:35161 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type   State    I-Node Path
unix  8      [ ]         DGRAM           15368    /dev/log
unix  2      [ ]         DGRAM           194110   @/org/kernel/udev/udevd
unix  2      [ ]         DGRAM           15671    @/var/run/hal/hotplug_socket
```

  - `--route`, `-r` muestra las tablas de rutas (igual que `route` admite los flags -n, -e y -ee)
  - `--interface`, `-i` muestra un resumen del estado de las interfaces de red (igual que `ifconfig -s`)

```
$ netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0   5110292      0      0      0  5011769      0      0      0 BMR
```

    - MTU (*Maximum Transmission Unit*): tamaño máximo del datagrama

    - Met: valor de la métrica para esa interfaz
    - RX/TX paquetes recibidos/transmitidos
    - OK/ERR/DRP/OVR paquetes transmitidos correctamente (OK), erróneos (ERR), descartados por falta de recursos (DRP, *drop*) y perdidos por desbordamiento (OVR, *overrun*)
    - Las banderas (*flags*) indican el estado del interfaz:
        - `B`: dirección de difusión activa
        - `L`: la interfaz es un dispositivo de bucle local (*loopback*)
        - `P`: se reciben todos los paquetes (modo promiscuo)
        - `O`: ARP desactivado para este interfaz
        - `M`: el interfaz permite multicast
        - `A`: el interfaz recibe todos los paquetes multicast en la red (*allmulti*)
        - `R`: la interfaz funciona (*running*)
        - `U`: la interfaz está activa (*up*)
    - Este estado puede cambiarse con `ifconfig`:

```
# ifconfig eth0 promisc # Modo promiscuo
# ifconfig eth0 -arp # Desactiva ARP
```

- `--statistics`, `-s` muestra estadísticas para cada protocolo de red

```
$ netstat -s
Ip:
  5001746 total packets received
  7479 forwarded
  0 incoming packets discarded
  4894721 incoming packets delivered
  ......
```

- Cada uno de los modos anteriores tienen diferentes opciones
- Algunas opciones válidas para varios tipos son:
    - `--numeric` o `-n` muestra información numérica en vez de nombres para IPs, puertos, etc.
    - `--continuous` o `-c` imprime la información solicitada de forma continua
    - `--extend` o `-e` muestra información extendida (con `-ee` aún más información)
- Para más información ver la página del manual

Otros comandos de red

Comando `arp`

- `arp` manipula la cache de ARP:
  - muestra la tabla ARP
  - borra entradas
  - añade entradas manualmente

Ejemplo:

```
# arp
Address                  HWtype  HWaddress           Flags Mask   Iface
almansa.dec.usc.es       ether   00:0D:56:6F:E6:90   C            eth0
193.144.84.1             ether   00:E0:63:93:26:E5   C            eth0
teneguia.dec.usc.es      ether   00:C0:4F:A1:5D:89   C            eth0
```

- Flag: `C` dirección completa, `M` dirección añadida manualmente

Algunas opciones:

- `-i` *`interfaz`* muestra las entradas para el interfaz indicado
- `-a` *`hostname`* muestra las entradas para el host especificado
- `-d` *`hostname`* borra las entradas para el host especificado
- `-s` *`hostname hw_addr`* añade manualmente una entrada para el host especificado con la dirección hardware indicada
- `-n` *`interfaz`* no hace traducción de IPs a nombres

Comando `ping`

- Muestra la disponibilidad de conexión y la velocidad de transmisión con un host remoto:

```
$ ping 193.144.84.1
PING 193.144.84.1 (193.144.84.1) 56(84) bytes of data.
64 bytes from 193.144.84.1: icmp_seq=1 ttl=255 time=0.420 ms
64 bytes from 193.144.84.1: icmp_seq=2 ttl=255 time=0.396 ms
64 bytes from 193.144.84.1: icmp_seq=3 ttl=255 time=0.368 ms
```

- `ping` envía paquetes ICMP (*ECHO_REQUEST*) al destino y espera respuesta, midiendo el RTT
- muchos firewalls bloquean el tráfico ICMP por lo que el `ping` no funciona

Algunas opciones:

- `-b` permite `ping` a una dirección de broadcast
- `-c` ***COUNT*** para después de enviar ***COUNT*** paquetes ECHO_REQUEST
- `s` ***packetsize*** especifica el número de bytes a enviar (por defecto 56)

Comando `traceroute`

- Muestra la ruta que sigue un paquete hasta llegar a destino

```
$ traceroute www.elpais.es
traceroute to a1749.g.akamai.net (130.206.192.32), 30 hops max, 40 byte packets
 1  rutfis (193.144.64.1)  1.070 ms  0.688 ms  0.927 ms
 2  * * *
 3  10.56.5.1 (10.56.5.1)  57.463 ms  2.021 ms  1.923 ms
 4  193.144.79.72 (193.144.79.72)  2.507 ms  16.280 ms  2.080 ms
 5  GE2-0-0.EB-Santiago0.red.rediris.es (130.206.204.21)  25.681 ms  2.068 ms  1.965 ms
 6  GAL.SO2-0-0.EB-IRIS4.red.rediris.es (130.206.240.33)  10.959 ms  10.665 ms  10.710 m
 7  130.206.220.59 (130.206.220.59)  20.277 ms  10.781 ms  10.470 ms
 8  a130-206-192-32.deploy.akamaitechnologies.com (130.206.192.32)  11.011 ms  23.482 ms
```

- `traceroute` utiliza el campo TTL de la cabecera IP para obtener respuestas ICMP TIME_EXCEEDED de los host por los que pasa el paquete (envía paquetes UDP)
- los sistemas pueden no enviar mensajes de tiempo excedido: aparecen *
- si los firewalls bloquean el tráfico ICMP no veremos nada
- otros programas similares:
  - `traceproto`: permite especificar el protocolo a usar (TCP, UDP, ICMP) y el puerto a tracear (por defecto 80)
  - `tcptraceroute`: envía paquetes TCP SYN para evitar problemas con firewalls

Comandos `host`, `dig`, `nslookup`

- Permiten obtener la dirección IP de un sistema a partir del nombre o viceversa:

```
$ host www.elpais.es
www.elpais.es is an alias for elpais.es.edgesuite.net.
elpais.es.edgesuite.net is an alias for a1749.g.akamai.net.
a1749.g.akamai.net has address 130.206.192.38
a1749.g.akamai.net has address 130.206.192.32
```

- `nslookup` está desaprobado (*deprecated*) y no se recomienda su uso

Comando `mii-tool`

- Permite ver y/o configurar el estado de la unidad MMI (*Media Independent Interface*) de la tarjeta de red
  - Ethernet usa MII para autonegociar la velocidad de enlace y el modo duplex

```
# mii-tool -v eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 16 rev 0
  basic mode:   autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-contro
  link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-contro
# mii-tool --force=100baseTx-HD eth0
# mii-tool  eth0
eth0: 100 Mbit, half duplex, link ok
```

- Comando parecido (más complejo): `ethtool`

4.2. Ficheros de configuración de red

La configuración mediante `ifconfig` y `route` no se mantiene al apagar el sistema:

- durante el proceso de arranque la red se inicia mediante la ejecución de scripts del `init.d`

- `/etc/init.d/networking` en Debian
- `/etc/init.d/network` en RedHat

- Estos scripts leen los ficheros de configuración de la red
- Fichero `/etc/network/interfaces` en Debian

```
auto eth0
iface eth0 inet static
   address 193.144.84.77
   netmask 255.255.255.0
   network 193.144.84.0
   broadcast 193.144.84.255
   gateway 193.144.84.1
   # dns-* options are implemented by the resolvconf package, if installed
   dns-nameservers 193.144.75.9
   dns-search dec.usc.es
   name Tarjeta de red Ethernet
```

- Fichero `/etc/sysconfig/network-scripts/ifcfg-eth`x en RedHat

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=193.144.84.186
NETMASK=255.255.255.0
GATEWAY=193.144.84.1
TYPE=Ethernet
```

Otros ficheros de configuración

Fichero `/etc/resolv.conf` especifica el dominio y los servidores DNS

- Ejemplo:

```
domain dec.usc.es
search dec.usc.es usc.es
nameserver 193.144.75.9
nameserver 193.144.75.12
```

- si buscamos por un hostname (sin dominio) le añade `dec.usc.es` y si no aparece busca por `usc.es`
- pueden añadirse hasta tres servidores de DNS

Fichero `/etc/hosts` fichero que asocia nombres de hosts con direcciones IP

- permite consultar una IP sin acceder al DNS
- Ejemplo de `/etc/hosts`:

```
127.0.0.1     localhost.localdomain localhost
193.144.84.77 jumilla.dec.usc.es jumilla
```

- la consulta es más rápida que acceder al DNS
  - si las IPs cambian la dirección es incorrecta
- sólo debería aparecer el nodo local y la interfaz de loopback
  - esto permite fijar el nombre y el dominio del sistema
  - en algunas distribuciones (Debian) el nombre también debe ponerse en el fichero `/etc/hostname`
  - el nombre y el dominio pueden obtenerse mediante los comandos `hostname` y `dnsdomainname`

Fichero `/etc/networks` fichero de texto que asocia nombres a redes

- No es imprescindible
- Ejemplo de `/etc/networks`

```
red1 172.16.1.0
red2 172.16.2.0
```

Fichero `/etc/host.conf` configura el comportamiento del *name resolver*

- indica donde se resuelven primero la dirección o el nombre de un nodo
- Ejemplo de `/etc/host.conf`:

```
order hosts,bind
multi on
```

- indica que primero se verifiquen las tablas locales (`/etc/hosts`) y después el DNS
- `multi on` indica que se retornen todas las direcciones válidas que se encuentren en `/etc/hosts`

Fichero `/etc/nsswitch.conf` fichero de configuración del *Name Service Switch*

- centraliza la información de diferentes servicios para la resolución de nombres
  - indica las acciones a realizar para acceder a las diferentes bases de datos del sistema: hosts, contraseñas, servicios, etc.
  - reemplaza la funcionalidad del archivo `host.conf`
  - introducido en la versión 2 de la biblioteca GNU
- Ejemplo de `nsswitch.conf`

```
hosts:          dns files
networks:       files
```

  - indica que un host se busque primero en el DNS y después en el fichero `/etc/hosts`, mientras que una red se busca sólo en `/etc/networks`
- Es posible controlar el comportamiento por medio de acciones, por ejemplo:

```
hosts:          dns [!UNAVAIL=return] files
networks:       files
```

- si el estado de salida del DNS es diferente de *no disponible* no consulta a los ficheros:
  - sólo accede a `/etc/hosts` si el DNS no está disponible
- Los valores de estado disponibles son:
  - `success` la petición se encontró sin errores (acción por defecto `return`)
  - `notfound` no error, pero no se encontró el nodo o la red (acción por defecto `continue`)
  - `unavail` servicio solicitado no disponible (acción por defecto `continue`)
  - `tryagain` servicio no disponible temporalmente (acción por defecto `continue`)

Fichero `/etc/protocols` lista los protocolos que reconoce el sistema operativo

- Ejemplo de `/etc/protocols`

```
ip      0       IP    # internet protocol, pseudo protocol number
icmp    1       ICMP  # internet control message protocol
tcp     6       TCP   # transmission control protocol
udp     17      UDP   # user datagram protocol
.....
```

Fichero `/etc/services` relaciona las aplicaciones con sus correspondientes puertos y protocolos básicos

- Un trozo de `/etc/services`

```
ftp-data        20/tcp   # Datos de ftp
ftp             21/tcp   # Control de ftp
ssh             22/tcp   # SSH por TCP
ssh             22/udp   # SSH por UDP
telnet          23/tcp   # Telnet
smtp            25/tcp   # Correo electrónico
....
```

4.3. Configuración del DHCP

DHCP (*Dynamic Host Configuration Protocol*) permite configurar automáticamente la red de los sistemas a partir de un servidor DHCP

- La información de IPs, DNS, etc. se mantiene centralizada en el servidor
- Al iniciarse, los clientes se conectan al servidor (por broadcast) y cargan su configuración

Configuración del servidor

Se encuentra en el fichero `/etc/dhcp/dhcpd.conf`

- Ejemplo sencillo de configuración

```
option domain-name "midominio.com";  # Nombre de Dominio
option domain-name-servers 10.0.2.3, 193.14.75.9; # Servidores de Nombres
default-lease-time 600; # Tiempo por defecto que dura una asignación
```

```
max-lease-time 7200;    # Duración máxima de una asignación
option subnet-mask 255.255.255.0; # Máscara de red
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; # Dirección de Broadcast
  option routers 192.168.0.1;             # Gateway de la red
}
host marte {
  hardware ethernet 52:54:00:12:34:70;
  fixed-address marte.mired.com;
}
```

- si utilizamos nombres (como `marte.mired.com`) la IP debe ser accesible (por DNS o `/etc/hosts`)
- en el fichero `/etc/default/isc-dhcp-server` debemos especificar el interfaz por el que servimos DHCP
- en `/var/lib/dhcp/dhcpd.leases` están las IPs asignadas
- para más información ver la página de manual de `dhcpd`

Configuración del cliente

Para que el cliente obtenga los datos de DHCP usar:

```
# dhclient eth0
```

- Un comando similar es `pump`

Para que el cliente se configure en el inicio debemos modificar el fichero de configuración de red

- En Debian, fichero `/etc/network/interfaces`:

```
auto eth0
iface eth0 inet dhcp
```

5. Automatización de tareas

En esta sección veremos la utilización de comandos que permiten automatizar tareas repetitivas

1. Tareas que se deben ejecutar en momentos concretos o de forma periódica:
   - `at`, `batch` permiten ejecutar trabajos a una hora específica o bajo determinadas condiciones
   - `cron` permite correr trabajos a intervalos regulares
2. Herramientas para automatizar la configuración de servidores

5.1. Tareas periódicas

Comando `at`

Permite indicar el momento en que se quiere ejecutar un trabajo

- Sintaxis:

  `at [`*opciones*`]` *TIME*

- Al ejecutar `at` pasamos a un nuevo prompt, que nos permite introducir comandos que se ejecutarán a la hora indicada
  - para salvar el trabajo y salir `CTRL-D`
  - al terminar, la salida estándar se envía como un mail al usuario (si está instalado un agente de correo)
  - el trabajo se ejecuta mientras el sistema esté encendido a la hora indicada
- Ejemplo:

```
$ at 11:45
warning: commands will be executed using /bin/sh
at> ls /tmp > lista
at> env DISPLAY=:0 zenity -info -text="Hola"
at> <EOT>
job 4 at Wed Nov 16 11:45:00 2005
```

- Algunas opciones:
  - `-f` *FILE* especifica un fichero conteniendo las acciones a realizar en vez de la entrada estándar
  - `-c` *jobnumber* muestra el trabajo a ejecutar
  - `-m` envía un mail al usuario, incluso aunque no haya salida

- `-v` muestra la hora a la que se ejecutará el trabajo

- *TIME* puede especificarse de varias formas:
  - `HH:MM` por ejemplo `12:54`
  - `HH:MMAM/PM`, por ejemplo `1:35PM`
  - `HH:MM MMDDYY`, por ejemplo `1:35PM 122505`
  - `now + numero unidades`, donde *unidades* puede ser `minutes`, `hours`, `days`, o `weeks`

    ```
    $ at now+2hours
    ```
  - `today`, `tomorrow`, por ejemplo `12:44tomorrow`
  - `midnight` (00:00), `noon` (12:00), `teatime` (16:00)

Comandos relacionados

- `atq` o `at -l` lista los trabajos pendientes del usuario
  - si es el superusuario, lista los trabajos de todos los usuarios
- `atrm` o `at -d` borra trabajos identificados por su número de trabajo
- `batch` ejecuta trabajos cuando la carga del sistema es baja
  - el trabajo empieza en cuanto la carga caiga por debajo de 1.5
  - la carga se obtiene del fichero `/proc/loadavg`

Ficheros de configuración

- El administrador puede controlar la utilización de `at`
  - Ficheros `/etc/at.allow` y `/etc/at.deny`
    - `at.allow` lista los usuarios que pueden usar `at`
    - `at.deny` lista los usuarios que NO pueden usar `at`
- Primero se chequea `/etc/at.allow`
  - si está el usuario, puede usar `at`
  - si no está o el fichero no existe, se chequea `/etc/at.deny`
  - si el usuario no está en `at.deny` puede usar `at`
- Si no existe ninguno de los dos ficheros, solo root puede ejecutar `at`
- Para dar permiso para todos los usuarios crear sólo el fichero `at.deny` vacío

Procesos periódicos

Para crear trabajos que se ejecuten periódicamente se utilizan el demonio `cron` y el comando `crontab`

- `crontab` permite configurar los procesos periódicos
- `cron` se encarga de su ejecución

La utilización de `cron` se gestiona a través de los ficheros `/etc/cron.allow` y `/etc/cron.deny`

- el comportamiento si no existen los ficheros depende de la configuración del sistema
- en Debian, si no existen, todos los usuarios pueden usar `crontab`

Los trabajos se especifican en un fichero de `crontab`, que se guarda en `/var/spool/cron/crontabs`, y que puede tener tres tipos de líneas:

- Comentarios, que empiezan por `#`
- Definición de variables, de tipo *`nombre = valor`*

```
# shell usada para ejecutar los comandos
SHELL=/bin/bash
# Usuario al que se envía (por mail) la salida
# de los comandos (por defecto, se envían
# al propietario del fichero crontab)
MAILTO=pepe
```

- Especificación del trabajo y de la hora de ejecución, de la siguiente forma:

  `minuto hora día mes día_semana comando`

  - el día de la semana de 0 a 7 (0 ó 7 domingo)
  - `*` indica cualquier valor
  - se pueden indicar rangos, listas o repeticiones:
    - `1-5` para indicar de lunes a viernes
    - `0,15,30,45` para indicar cada 15 minutos
    - `0-23/2` en el campo hora, indica realizar cada dos horas (0, 2, 4, etc.)

- Ejemplos:
  - Borra el `/tmp` todos los días laborables a las 4:30 am

    ```
    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
```

Comando `crontab`: crear y editar los trabajos periódicos

- Sintaxis:

  `crontab [-u` *`usuario`*`] {-l|-e|-r}`
  `crontab [-u` *`usuario`*`]` *`fichero`*

- en la segunda forma instala un nuevo `crontab` desde un fichero
- Opciones:
  - `-u` *`usuario`* crea o maneja el crontab de un usuario específico (sólo root)
  - `-e` edita el fichero `crontab`
  - `-l` muestra el fichero `crontab`
  - `-r` borra el fichero `crontab`

El demonio `cron` busca ficheros en `/var/spool/cron` para ejecutarlos a la hora indicada

- además también ejecuta las acciones indicadas en los ficheros `/etc/crontab` y en el directorio `/etc/cron.d/`
- estos ficheros suelen ser de mantenimiento del sistema

De esta forma, el administrador puede crear scripts que se ejecuten con periodicidad horaria, diaria, semanal y mensual

- sólo tiene que colocar esos scripts en los directorios `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` o `/etc/cron.monthly`
- la fecha y hora de ejecución de estos scripts se controla en el fichero `/etc/crontab`

Cron está pensado para sistemas funcionando 24/7

- Si el sistema está apagado a la hora de una acción cron, esa iteración no se realiza
- Problema en sistemas de sobremesa y/o domésticos

Solución complementaria: **Anacron**

Anacron

Ejecuta asíncronamente tareas periódicas programadas

- Al iniciarse el sistema comprueba si hay tareas periodicas pendientes (que no se realizaron por estar el sistema apagado)
- De ser así, espera un cierto retardo y las ejecuta

Fichero de configuración: `/etc/anacrontab`

`período retardo id_del_trabajo comando`

- El período se especifica en días (o como `@monthly`) y el retardo en minutos:

```
1       5   cron.daily   run-parts /etc/cron.daily
7       10  cron.weekly  run-parts /etc/cron.weekly
@monthly 15  cron.monthly run-parts /etc/cron.monthly
```

- En este ejemplo, cada día, los scripts en `cron.daily` se ejecutan 5 minutos después de iniciar el anacron, cada 7 días (10 minutos de espera), se ejecutan los scripts en `cron.weekly` y cada mes (15 minutos) los de `cron.monthly`

Limitaciones de anacron

- Solo para uso del administrador
- Solo permite períodos de días (no vale para tareas que se deban ejecutar varias veces al día)

5.2. Automatización de la configuración

Programas que permiten automatizar la gestión de la configuración de servidores en redes complejas

- La configuración se indica de forma centralizada
- Los diferentes servidores se configuran según lo indicado
- Liberan al administrador de tener que configurar manualmente los diferentes sistemas

Ejemplos más populares (open-source):

- Puppet: utiliza un lenguaje declarativo para describir la configuración de los sistemas
- Chef: Utiliza un Ruby para escribir "recetas" con la configuración de los sistemas

Una comparativa en Wikipedia

Puppet

Puppet usa un paradigma cliente-servidor

- En el servidor se describe la configuración de los sistemas (servicios, usuarios, ficheros,...) a través de ficheros de manifiesto
  - Se usa un lenguaje declarativo propio
  - Los clientes se actualizan periódicamente (30 minutos, por defecto)
- Compatible con múltiples SOs
  - La información concreta de los sistemas la obtiene mediante la utilidad *Facter*
  - Los manifiestos se compilan en un catálogo que contiene los recursos y sus dependencias

Más información: docs.puppetlabs.com
Ejemplo de manifiesto (fichero `site.pp`)

```
class ntp {
    package { "ntp":
        ensure => installed,
    }
    service { "ntp":
        ensure => running,
    }
}

node "cliente.midominio.com" {
    include ntp
    user { "pepe":
        ensure => present,
        uid => "1001", gid => "admin",
```

```
        shell => "/bin/bash", home => "/home/pepe",
        managehome => true,
    }
}
```

6. Copias de seguridad

Realizar copias de seguridad es una de las tareas más importantes del administrador del sistema

- Es casi inevitable que se produzcan pérdidas de información, debido a, entre otras causas:
  - deterioro o borrado accidental por parte de un usuario autorizado
  - ataque intencionado por parte de personas no autorizadas
  - fallo del software o el hardware
  - incendio, robos, y desastres naturales, etc.
- es imprescindible poder recuperar la información perdida

En esta sección veremos los comandos básicos para realizar copias de seguridad en UNIX/Linux; para más información:

- Backup & Recovery, W. Curtis Preston, O'Reilly, 2007
- Linux System Administrators Guide: Capítulo 12, Backups

6.1. Estrategias para las copias de seguridad

Una buena estrategia para copias de seguridad debe tener las siguientes características:

- Ser fácil de usar, preferiblemente si totalmente automática
- Eficiencia y rapidez:
  - compromiso entre el tiempo de backup y el tiempo de recuperación
- Facilidad de restauración
- Capacidad de verificar las copias
  - difícil si el sistema está siendo usado continuamente

- Tolerancia a fallos en los medios de almacenamiento (cintas, etc.)
  - necesidad de mantener al menos dos copias de los backups completos del sistema
  - al menos una de las copias debe almacenarse en otro sitio
- Portabilidad
  - posibilidad de recuperar la información en diferentes sistemas

Componentes de las copias de seguridad

Hay básicamente tres componentes que intervienen en una copia de seguridad:

- Los medios de almacenamiento: cintas, discos, etc.
- El programa de copia: los comandos que mueven los datos de los discos a los medios
- El planificador: decide que información se copia y cuando

Medios de almacenamiento: Dispositivos donde se guarda la información; los más populares son:

- Cintas, desde cintas DDS/DAT de 8 o 3,8mm con capacidades hasta unos 160 GB, hasta cartuchos LTO de varios TBs (p.e. 6 TB en LTO-7 o 8.5 TB en StorageTek T10000D)
  - Dos dispositivos: cinta con no-rebobinado y con rebobinado (en Linux para cintas SCSI `/dev/nst`*X* y `/dev/st`*X* respectivamente, en Solaris `/dev/rmt/`*X* y `/dev/rmt/`*X*)
  - Silos robotizados para grandes infraestructuras
- Discos duros externos, pocos TB por unidad
- Discos ópticos (DVDs, Blue-Ray), hasta 128 GB por disco con BDXL (write once)
- Backup en la nube: capacidad "ilimitada", problemas de tiempo de acceso y problemas legales

Programa de copia: Se encarga de copiar los ficheros seleccionados en el medio de almacenamiento; dos mecanismos básicos

- Basado en imagen: accede al disco a bajo nivel
  - normalmente copias más rápidas, pero mas lento restaurar ficheros individuales
  - programas específicos para diferentes filesystems
  - comandos de este tipo son `dump` y `dd`
- Fichero a fichero
  - acceden a los ficheros a través de llamadas al SO
  - copias más lentas, pero restauración de ficheros individuales más simple
  - comandos de este tipo son `tar`, `cpio` o `afio`

El planificador: Decide cuándo realizar el backup (mediante cron o similar) y cuánta información copiar
Tipos de backup:

Completo se salva toda la información del sistema

Parcial sólo se salva la información más importante y difícil de recuperar (ficheros de usuario, de configuración, directorios de correo, web, etc.)

Incremental sólo se salvan los ficheros modificados desde el último backup completo o incremental

- la copia de seguridad necesita menos tiempo y espacio
- para restaurar los datos necesitaremos el último backup completo y todos los incrementales

Diferencial se salvan los ficheros modificados desde el último backup completo

- los backups son más grandes que en el caso incremental
- para restaurar sólo necesitamos el backup completo y el último diferencial

6.2. Comandos básicos

Veremos los comandos básicos para hacer backups en UNIX: `tar`, `cpio` y `dump`

Comandos `dump` y `restore`

Comandos más comunes para copias de seguridad

- comandos originales de BSD UNIX
- dependen del tipo de *filesystem*

Comando `dump`: Hace copias de un sistema de archivos entero, con las siguientes características:

- Pueden ser copias multivolumen
- Puede salvar ficheros de cualquier tipo (incluido ficheros de dispositivos)
- Los permisos, propietarios y fechas de modificación son preservados
- Puede realizar copias incrementales
- También puede usarse para salvar ficheros individuales (no es lo usual)

El formato y los argumentos de `dump` dependen de la versión utilizada, pero en general es:

`dump [-nivel] [`*`opciones`*`] [`*`ficheros_a_salvar`*`]`

- Nivel de `dump`: entero entro 0-9:
  - 0 implica backup completo
  - mayor que 0 implica copiar sólo los ficheros nuevos o modificados desde el último backup de nivel inferior
  - `dump` guarda información sobre los backups realizados en el fichero `/etc/dumpdates` o `/var/lib/dumpdates`
- Algunas opciones:
  - `-f` especifica el dispositivo o fichero donde salvar la copia
  - `-u` actualiza el fichero `dumpdates` después de una copia correcta

- `-a` determina automáticamente el fin de la cinta (opción por defecto)
- `-j`, `-z` usa compresión con bzlib o zlib (sólo en algunas versiones)

- Ejemplo: backup de nivel 0 de la partición `/home`

```
# dump -0u -f /dev/st0 /home
```

- Ejemplo: backup en una máquina remota usando `ssh` como transporte

```
# export RSH=ssh
# dump -0u -f sistema_remoto:/dev/st0 /home
```

Comando `restore`: Restaura ficheros salvados por `dump`

- Formato:

```
restore acción [opciones] [ficheros_a_recuperar]
```

- Acciones principales:
  - `r` restaura la copia completa
  - `t` muestra los contenidos de la copia
  - `x` extrae sólo los ficheros indicados
  - `i` modo interactivo
    - permite ver los ficheros de la copia
    - con `add` indicamos los ficheros a extraer y con `extract` los extraemos
    - usar `?` para ayuda

- Algunas opciones:
  - `-f` especifica el dispositivo o fichero de la copia
  - `-a` no pregunta de que volumen extraer los ficheros (lee todos los volúmenes empezando en 1)

- Ejemplo: restaurar el backup de `/dev/st0`

```
# restore -rf /dev/st0
```

- Ejemplo: restaurar el backup desde un sistema remoto

```
# export RSH=ssh
# restore -rf sistema_remoto:/dev/st0
```

- Ejemplo: restaurar sólo un fichero

```
# restore -xaf /dev/st0 fichero
```

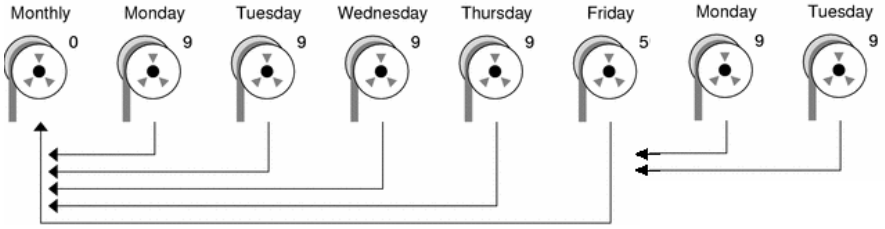
Archivo `restoresymtable`: Se crea cuando se restaura un filesystem completo, en el directorio donde se restaura

- Contiene información sobre el sistema restaurado
- Puede eliminarse una vez finalizada la restauración

Planificación de los backups

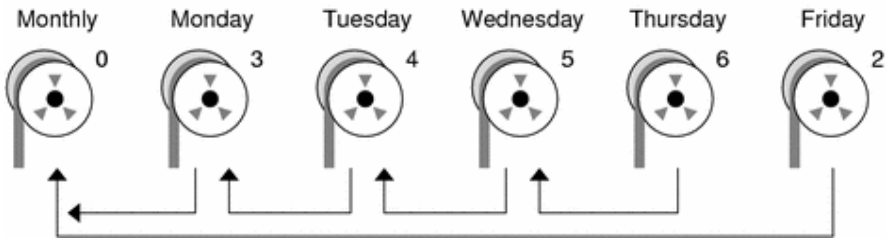
Podemos seguir diferentes estrategias a la hora de planificar los backups

- Ejemplo 1: copia de nivel 0 mensual, de nivel 9 diaria y de nivel 5 semanal



  - necesita 6 o 9 cintas: una para el 0, 4 para los niveles 5 y 1 o 4 para los niveles 9
  - para restaurar necesitamos restaurar en orden:
    1. la copia de nivel 0
    2. la última copia de nivel 5, y
    3. la última de nivel 9, después de la de nivel 5

- Ejemplo 2: copia de nivel 0 mensual, de nivel 2 semanal y de niveles 3, 4, 5 y 6 cada día



- necesita al menos 9 cintas
- para restaurar necesitamos restaurar en orden:
  1. la del nivel 0
  2. la del último viernes (nivel 2)
  3. las diarias desde el último viernes de forma consecutiva

Comando `tar` (Tape ARchiver)

Permite almacenar varios ficheros en uno sólo, manteniendo la estructura de directorios:

- Sintaxis:

  `tar [-]`*`función`*`[`*`modificador`*`] `*`fichero`*` [`*`directorio`*`]`

- Ejemplo: crea un fichero tar conteniendo los ficheros del directorio `/etc`

  ```
  tar cvf copia.tar /etc
  ```

- Puede indicarse un fichero o un dispositivo (p.e. `/dev/fd0`)
- `tar` conserva las propiedades de los ficheros: permisos, usuario/grupo, fechas, etc.
- Funciones principales:
  - `c` crea un nuevo archivo `tar`
  - `x` extrae los ficheros del archivo
  - `t` lista los ficheros del archivo
  - `r` añade nuevos ficheros al final del archivo `tar`
  - `u` almacena sólo los ficheros nuevos o modificados respecto a los del archivo `tar`
  - `A` añade un fichero `tar` a otro
  - `d` obtiene las diferencias entre los ficheros de la copia y los del disco
  - `--delete` borra un fichero del archivo `tar`
- Algunas opciones:
  - `v` verbose, muestra lo que está haciendo

- `f` para indicar el nombre del fichero `tar`; por defecto toma `-` que representa la entrada/salida estándar
- `z` comprime la copia con `gzip`
- `--bzip2` o `j` comprime la copia con `bzip2`
- `l` almacena sólo los ficheros locales (útil con NFS)
- `k` no sobreescribe los ficheros existentes al extraer
- `T` o `--files-from F` obtiene la lista de ficheros a guardar del fichero `F`
- `X` o `--exclude-from=F` excluye los ficheros que concuerdan con los patrones listados en el fichero `F`
- `N` o `--newer DATE` sólo guarda los ficheros más nuevos que `DATE`
- `M` o `--multi-volume` permite crear copias multivolumen (por ejemplo, varios disquetes)

- para más opciones ver la página de info (`info tar`)
- Ejemplos:
  - Extrae todos los ficheros de `copia.tar`

    ```
    tar xvf copia.tar
    ```
  - Extrae el el fichero `passwd` de `copia.tar`

    ```
    tar xvf copia.tar etc/passwd
    ```
  - Copia el contenido de `/tmp` directamente a un disquete

    ```
    tar cvf /dev/fd0 /tmp
    ```
  - Copia un directorio completo

    ```
    (cd dir1 && tar cf - .) | (cd dir2 && tar xvf -)
    ```
  - Copia los ficheros más nuevos que un fichero `control`

    ```
    find dir -newer control ! -type d -print | tar cvfT
    f.tar -
    ```
- Problemas con `tar`
  - Algunas versiones no admiten opciones como la compresión o copia multivolumen
  - Algunas versiones tienen problemas con paths muy largos (más de 100 caracteres)

Comando `cpio`

El comando `cpio` es similar a `tar` en funcionalidad

- crea y extrae archivos, o copia ficheros de un lugar a otro
- maneja archivos en formato `cpio` y formato `tar`

Tres funciones primarias:

1. *Copy-out*: copia ficheros a un archivo, con la opción `-o`
   - Ejemplo: copia todos los directorios desde el actual en el fichero `tree.cpio`

     ```
     $ find . | cpio -ov > tree.cpio
     ```

   - para usar un dispositivo en lugar de un fichero, sustituir `tree.cpio` por `/dev/`*`dispositivo`*
2. *Copy-in*: extrae los ficheros de un archivo, con la opción `-i`

   ```
   $ cpio -idv < tree.cpio
   ```

   - la opción `d` crea los directorios al ir extrayendo
3. *Copy-pass*: usado para copiar ficheros de un árbol de directorios a otro, con la opción `-p`
   - Ejemplo: copia los ficheros del directorio actual y subdirectorios a un nuevo directorio `new-dir`

     ```
     $ find . -depth -print0 | cpio --null -pvd new-dir
     ```

   - la opción `-depth` procesa primero el contenido del directorio y después el directorio (mejor para restaurar)
   - las opciones `-print0` y `--null` evitan problemas con nombres de ficheros que contengan un carácter de *newline*
     - `-print0` termina los nombres de los ficheros con un '\0' en vez de '\n'
     - `--null` o `-0` lee una lista de ficheros terminados por un '\0'

- Para más opciones y uso de `cpio` ver la página de información: `info cpio`

Comando `afio`

Variación de `cpio`, con varias mejoras:

- Permite hacer copias multivolumen
- Permite archivar los ficheros comprimiendolos de uno en uno
  - No comprime los ficheros que no interesa comprimir por que ya lo están (reconoce por extensión)
- Permite verificar la copia con el original (opción `-r`)

Modos de funcionamiento similares a `cpio`

- `-o` guarda a archivo, `-i` extrae de un archivo y `-p` copia directorios
- otras opciones: `-r` verifica el archivo con el filesystem; `-t` muestra el contenido del archivo
- Ejemplos:
  - Salvar a disquete multivolumen comprimido

    ```
    $ find . | afio -ov -s 1440k -F -Z /dev/fd0
    ```

  - Comprobar con el original una copia comprimida en varios disquetes

    ```
    $ afio -rv -s 1440k -F -Z /dev/fd0
    ```

  - Muestra el contenido del archivo:

    ```
    $ afio -tv -s 1440k -F -Z /dev/fd0
    ```

  - Extrae el contenido del archivo

    ```
    $ afio -iv -s 1440k -F -Z /dev/fd0
    ```

  - Copia los ficheros del directorio actual y subdirectorios a un nuevo directorio `new-dir`

    ```
    $ find . -depth -print0 | afio -p0xa directorio_nuevo
    ```

- Para opciones ver la página de manual

Comando `dd`

Comando de copia y conversión de ficheros

- Sintaxis.

  `dd [if=`*`fichero_entrada`*`] [of=`*`fichero_salida`*`] [opciones]`

- Por defecto, copia de la entrada estándar a la salida estándar
- Algunas opciones:
  - `ibs=`*`b`* lee *`b`* bytes de cada vez (tamaño de bloque, por defecto 512)
  - `obs=`*`b`* escribe *`b`* bytes de cada vez
  - `bs=`*`b`* lee y escribe *`b`* bytes de cada vez
  - `cbs=`*`b`* especifica el tamaño del bloque de conversión
  - `skip=`*`n`* salta *`n`* bloques del fichero de entrada antes de la copia
  - `seek=`*`n`* salta *`n`* bloques del fichero de salida antes de la copia
  - `count=`*`n`* copia sólo *`n`* bloques del fichero de entrada
  - `conv=`*`conversión`* convierte el formato del fichero de entrada según el valor de *`conversión`*:
    - `ascii` Convierte EBCDIC a ASCII
    - `ebcdic` Convierte ASCII a EBCDIC
    - `swab` Intercambia cada par de bytes de la entrada
    - `lcase` Cambia las letras mayúsculas a minúsculas
    - `ucase` Cambia las letras minúsculas a mayúsculas
    - `noerror` Continúa después de producirse errores de lectura
- Ejemplo: imagen de floppy de 3.5, con 18 sectores por pista, dos cabezas y 80 cilindros:

  ```
  $ dd bs=2x80x18b if=/dev/fd0 of=/tmp/floppy.image
  ```

  - la `b` representan bloques de 512 bytes (en total 1474560 bytes)
  - la copia se realiza de una sola vez
- Ejemplo: extrae los datos de una cinta con error

  ```
  $ dd conv=noerror if=/dev/st0 of=/tmp/bad.tape.image
  ```

- Ejemplo: `tar` del directorio actual y copia en cinta en el sistema remoto

  `$ tar cjf - . | ssh `*`remoto`*` dd of=/dev/st0`

Comando `mt`

Permite la manipulación directa de la unidad de cinta

- Sintaxis.

  `mt [-f` *`unidad_de_cinta`*`] `*`operación`*` [`*`número`*`]`

- con `-f` indicamos la unidad de cinta a utilizar (si se omite se toma la definida en la variable `TAPE`)

- Algunas operaciones:

  - `stat(us)` muestra el estado de la unidad de cinta
  - `rew(ind)` rebobina la cinta hasta el principio
  - `ret(ension)` alisa y da tensión a la cinta (rebobina hasta el principio, luego hasta el final y nuevamente al principio)
  - `erase` borra la cinta entera
  - `fsf`/`bsf` se avanza/retrocede el número de archivos especificado por *`número`*
  - `eom` salta hasta el final de parte grabada

6.3. Otras aplicaciones

Existen otras aplicaciones y/o comandos que permiten hacer backups y sincronizar ficheros

- Backups: Bacula, Amanda, Flexbackup, rdiff-backup, DAR, BackupPC, UrBackup, Backupninja, Burp, duplicity, safekeep, etc. (más ejemplos de software de backup)

- Sincronización de ficheros y clonado: `rdist`, `rsync`, `pdsh`, `unison`, Partimage, Clonezilla, SystemImager, rsnapshot, etc. (más ejemplos de sincronización y clonado)

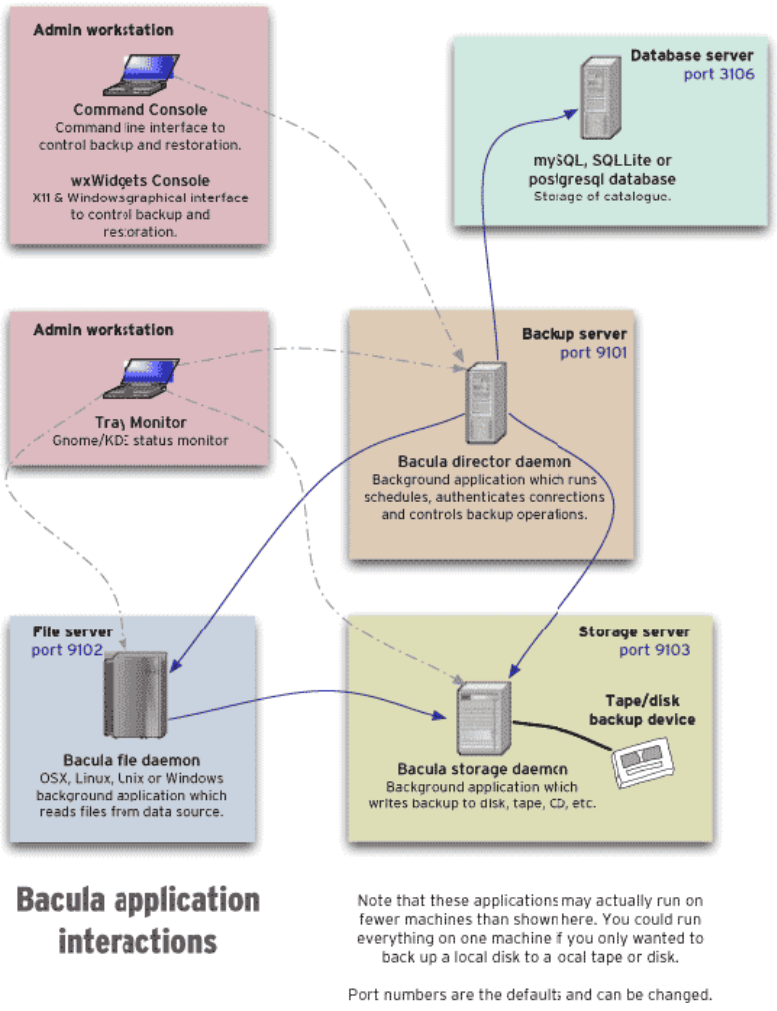
Bacula

- Sofisticado sistema de backup en red con diseño modular

- Permite hacer copias de seguridad de todas las máquinas de una LAN a diferentes medios de backups (cinta, disco,...)

- Soporta MySQL, PostgreSQL o SQLlite para el catálogo

- Hace backups de sistemas UNIX, Linux y Windows
- Para más información, ver blog.bacula.org/documentation o la sección 10.8 del libro *UNIX and Linux System Administration Handbook*, Evi Nemeth et al, 4ª ed.



Amanda

`Amanda`: *Advanced Maryland Automatic Network Disk Archiver*

- Sofisticado sistema de backup en red
- Permite hacer copias de seguridad de todas las máquinas de una LAN a una unidad de cinta en un servidor

- Está disponible en la mayoría de los UNIX y soporta muchos tipos de medios de backup
- Puede hacer uso de SAMBA para copias de sistemas Windows NT
- Se basa en `dump` y `tar`
- Para más información, ver wiki.zmanda.com

Flexbackup

Flexbackup Herramienta de backup flexible para instalaciones de pequeño y medio tamaño

- Más simple de configurar y utilizar que `Amanda` para sitios con un número no muy alto de sistemas
- Usa distintos formatos de archivo: `dump`, `afio`, `GNU tar`, `cpio`, `zip`, etc.
- Permite backups completos e incrementales, como `dump`
- Permite backups remotos a través de `rsh` o `ssh`
- Para más información, ver flexbackup.sourceforge.net

rdiff-backup

rdiff-backup copia un directorio en otro, permitiendo copias remotas

- Hace una copia exacta de los directorios (*mirror*), guardando las propiedades de los ficheros (propietario, permisos, etc.)
- Guarda las diferencias entre copias de los ficheros para poder recuperar un fichero antiguo (*incremental*)
- Sólo transmite las diferencias de los ficheros (similar a `rsync`)
- Para más información ver www.nongnu.org/rdiff-backup

DAR

DAR *Disk ARchiver* comando para hacer backups de árboles de directorios y ficheros

- Permite copiar un filesystem entero a un archivo
- Permite hacer backups completos y diferenciales
- Permite hacer copias multivolumen:
  - divide en archivo en varios ficheros (*slices*) parando antes de crear cada nuevo *slice*
  - interesante para hacer copias en floppy, CD o DVD
- Más información en: dar.linux.free.fr

BackupPC

BackupPC solución de alto rendimiento para backups de sistemas GNU/Linux, WinXX y MacOSX PCs a un servidor o NAS

- No necesita software en el cliente
- Obtiene los backups mediante SAMBA, tar sobre ssh/rsh/nfs o rsync
- Compresión opcional
- Interfaz web para el administrador
- Más información en: backuppc.sourceforge.net/info.html

UrBackup

UrBackup sistema cliente/servidor para GNU/Linux y/o Windows

- Backups completos o incrementales
- Salva particiones completas o directorios
- Configurable desde el servidor o los clientes
- Interfaz web para el administrador
- Más información en: http://www.urbackup.org/documentation.html

Comando `rdist`

Permite distribuir ficheros desde un servidor central a varias máquinas

- sólo copia los ficheros modificados, preservando el propietario, grupo, modo y fechas de modificación
- las versiones actuales pueden funcionar sobre `ssh` (las antiguas funcionaban sobre `rlogin`, con problemas de seguridad)
- utiliza un fichero `distfile` que especifica las acciones a realizar
- Ejemplo de `distfile`

```
SYS_FILES = (/etc/passwd /etc/group /etc/mail/aliases)
HOME_DIRS = (/home/tomas /home/al*)
GET_ALL = (maquina1 maquina5 maquina6)
GET_SOME = (maquina2 maquina8)

all: ${SYS_FILES} -> ${GET_ALL}
notify tomas@localhost;
special /etc/mail/aliases ¨/usr/bin/newaliases¨;

some: ${SYS_FILES} -> ${GET_SOME}
except /etc/mail/aliases;
```

- ejemplo de uso:

```
# rdist -f distfile
```

Comando `rsync`

Similar a `rdist` aunque funciona de forma diferente

- más eficiente que `rdist`, sólo transmite las diferencias entre ficheros
- no usa fichero de configuración: funciona de forma similar a `rcp`
- ejemplo:

```
# rsync -av /home/tomas maquina1:/tmp
```

- ver la página de manual de `rsync` para más detalles

Comando `pdsh`

Permite mandar comandos a un grupo de hosts en paralelo

- usa un algoritmo paralelo de "ventana deslizante" para reducir el número de sockets abiertos en origen
- permite copias en paralelo con los comandos `pdcp` (copia de uno a muchos) y `rpdcp` (copia de muchos a uno)
- ejemplo, copia /etc/hosts a los hosts foo0, foo4 y foo5:

```
# pdcp -w foo[0-5] -x foo[1-3] /etc/hosts /etc
```

- ver la página de manual de `pdsh` y `pdcp` para más detalles

Unison

Aplicación para sincronizar ficheros y directorios entre sistemas

- puede sincronizar entre sistemas Windows y UNIX
- no requiere permisos de root
- permite sincronización en los dos sentidos
- las transferencias se optimizan usando una versión de `rsync`
- tiene un interfaz gráfico sencillo
- para ver un tutorial de uso, hacer:

```
$ unison -doc tutorial
```

Imágenes del sistema

Herramientas que nos permiten obtener imágenes completas del sistema para sincronización de ficheros o réplicas (clones)

Partimage salva particiones completas a un fichero de imagen

- permite recuperar la partición completa en caso de errores
- permite realizar clones de un PC

Clonezilla aplicación opensource para hacer clones masivos

- permite hacer clones de múltiples PCs (40 o más) simultáneamente

- puede usar multicast para distribuir las imágenes
- basado en DRBL (*Diskless Remote Boot in Linux*) y Partclone

SystemImager herramienta para automatizar la instalación de Linux y la distribución de software en una red de PCs (usado en clusters, granjas de servidores o redes en general)