

# Comandos de administración

Linux

# Usuario root

El usuario root, a veces llamado superusuario, es el usuario administrador del sistema. Está identificado con el número de usuario cero (uid=0) y tiene permisos sobre todo el sistema sin ningún tipo de restricción.

El usuario root puede acceder a cualquier archivo, ejecutar, instalar y desinstalar cualquier aplicación, modificar los archivos de configuración del sistema y administrar usuarios. Si dispones de la contraseña de root tendrás control total sobre todo el sistema.

Los comandos se pueden ejecutar con permisos de root usando sudo:

***>sudo shutdown -r 16:00***

# ps → (process status: estado de los procesos)

Nos muestra lo que queramos saber de los procesos que están corriendo en nuestro sistema. Cada proceso está identificado con un número llamado PID. Si hacemos...

**>ps -A**

...nos mostrará un listado de todos los procesos, su PID a la izquierda y su nombre a la derecha.

Si queremos más información:

**>ps aux**

# ¿Cómo buscar el pid de un programa que se está ejecutando?

```
ps -A | grep calcu
```

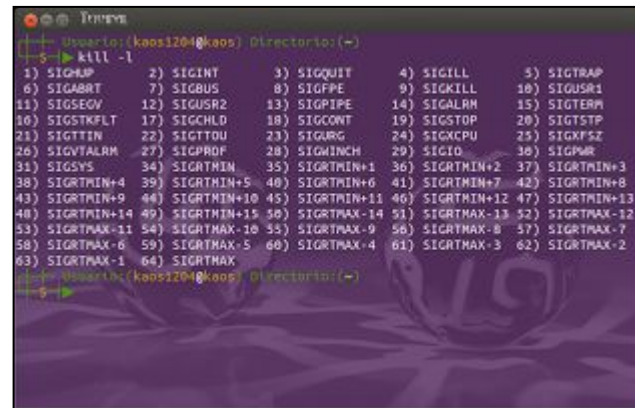
NOTA: usamos calcu porque queremos buscar un programa que se llama calculator

# kill → (kill: matar)

Permite enviar señales a uno o varios procesos del sistema. Las más utilizadas suelen ser la de matar un proceso (9 o SIGKILL), pararlo (TERM) o reiniciarlo (1 o HUP) pero hay muchas más que pueden ser útiles en ocasiones.

El listado completo de señales disponibles puede visualizarse ejecutando:

**>kill -l**



```
Usuario:(kaos1204@kaos) Directorio:(~)
$ kill -l
 1) SIGABRT   2) SIGINT    3) SIGQUIT   4) SIGILL     5) SIGTRAP
 6) SIGABRT   7) SIGBUS    8) SIGFPE    9) SIGKILL   10) SIGUSR1
11) SIGSEGV  12) SIGUSR2  13) SIGPIPE  14) SIGALRM   15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD  18) SIGCONT  19) SIGSTOP   20) SIGTSTP
21) SIGTTIN  22) SIGTTOU 23) SIGURG   24) SIGXCPU   25) SIGXFSZ
26) SIGTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO     30) SIGPWR
31) SIGSYS   34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
Usuario:(kaos1204@kaos) Directorio:(~)
$
```

Como señal podemos utilizar el número correspondiente a la izda del nombre de la señal (SIG...) o escribir directamente el nombre sin el "SIG" que le precede, por ejemplo "STOP"

La sinapsis del comando sería:

**>kill [señal] <pid> [...]**

Por ejemplo para solicitar que termine un proceso cuyo PID es "3760", se utiliza la señal TERM (15), que es la señal por defecto si no se escribe otra. Así que para solicitar el fin de ejecución de uno basta con ejecutar kill seguido del PID correspondiente:

**>kill 3760**

Para forzar que uno o varios procesos terminen de forma inmediata (sin solicitar ni preguntar...) usamos la señal SIGKILL (9). Hay que ser cautos al usar esta señal porque fuerza a los procesos a terminar inmediatamente sin permitirles terminar de forma limpia, es decir, puede que no borre los PID, que no deje terminar las peticiones pendientes, etc:

**>kill -9 3760**

Ángel González M.

Si quisiéramos forzar que todos los procesos con un determinado nombre finalicen inmediatamente usaríamos "killall" en lugar de kill. Por ejemplo para cerrar varios conkys que tengamos en el escritorio:

```
>killall -9 conky
```

Otro ejemplo sería el de suspender un proceso, para ello le enviamos la señal de STOP (19) seguida del proceso. Si no conocemos el ID de la señal podemos hacerlo también a través del nombre. En esta señal el proceso quedaría suspendido, por lo que todavía figuraría en la lista de procesos y podríamos reanudarlo posteriormente (próximo ejemplo):

```
>kill -19 3760
```

```
o
```

```
>kill -STOP 3760
```

Ahora que sabemos suspender procesos, es interesante conocer como reactivarlos, para ello usamos la señal CONT (18). En este ejemplo vamos a “revivir” el proceso anterior:

```
>kill -18 3760
```

```
o
```

```
>kill -SIGCONT 3760
```

*Ángel González M.*

Una de las señales más importantes es HUP (1). Esta señal para y reinicia el proceso indicado, también se puede aplicar con el nombre del proceso además del ID.

**>kill -HUP 3760**

o con el nombre del proceso:

**>killall -HUP script.sh**

En caso de querer utilizarlo para por ejemplo, reiniciar todos los procesos "conky" usaríamos killall en lugar de kill:

**>killall -HUP conky**



# sudo → (super-user do: hacer como superusuario)

Permite a los usuarios ejecutar acciones con los privilegios de seguridad del root, de manera segura.

Por defecto linux trae desactivada la cuenta del "root", por seguridad y para administrar el sistema existe un grupo de usuarios denominado "sudoers users" (administradores o admin), los cuales pueden obtener permisos de root, mediante la utilización de "sudo".

El usuario con el que instalamos Ubuntu, se encuentra incluido en este grupo de administradores. En la terminal se utiliza el comando "sudo", anteponiéndolo a la orden o comando a ejecutar:

**>sudo orden**

*Ángel González M.*

# su → (switch user: cambio de usuario)

Cambiar de usuario sin necesidad de hacer un cierre o cambio de sesión:

su nombreusuario

La contraseña que nos pedirá, es la del usuario al que vamos a cambiar, no la del usuario en el que estamos.

Si omitimos el nombre de usuario en el comando, cambiará a la cuenta del "root" (si está activada):

**>su**

# Programando tareas en Linux – comandos sleep, at y cron / crontab

La diferencia entre AT y Cron es que el primero no es persistente, por lo que si reiniciamos la PC se perderá la tarea que le encomendamos.

# sleep

Este comando espera la cantidad de segundos pasados como parámetro. Podemos utilizarlo cuando deseemos ejecutar un proceso después de una cantidad de tiempo. Un ejemplo simple (pero sin utilidad) sería listar un directorio dentro de 60 segundos:

```
(sleep 60; ls /home/usuario ) &
```

Para cortar la ejecución presionar CTRL + c.

# at: Ejecuta un comando a la hora que quieras

At ejecuta un comando o programa a determinada hora determinado día. Su sintaxis es bastante sencilla.

Por ejemplo, si deseamos descargar determinado archivo a las 09:00, ejecutamos:

```
>echo 'wget -c www.example.com/files.iso' | at 09:00
```

Si deseamos hacerlo a las 21:00, pero de mañana, ejecutamos:

```
>echo 'wget -c www.example.com/files.iso' | at 21:00 tomorrow
```

El comando at soporta diferentes formatos de fecha/hora y también ofrece la posibilidad de leer los comandos a ejecutar de un archivo, para más información: man at

*Ángel González M.*

# at (continuación)

Cuando tenemos varios procesos ejecutados con AT, podemos consultarlos con el comando, nos indicará el número de proceso:

```
> atq
```

Cuando sepamos el proceso que queremos matar, solo tenemos que teclear:

```
atrm nº_proceso
```

Por lo tanto, si quiero matar el proceso nº 3, solo tengo que poner:

```
> atrm 3
```

# Cron

Es un demonio (proceso en segundo plano) que se ejecuta desde el mismo instante en el que arranca el sistema. Comprueba si existe alguna tarea (job) para ser ejecutado de acuerdo a la hora configurada en el sistema. Muy importante que la zona horaria esté bien configurada ya que de lo contrario las ejecuciones puede que no coincidan con los ajustado.

En función de la distribución, se inicia utilizando las carpetas `/etc/rc.d/` o `/etc/init.d` y cada minuto comprueba los ficheros `/etc/crontab` o `/var/spool/cron` es busca de posibles ejecuciones (ahora empieza todo a cuadrar).

# crontab

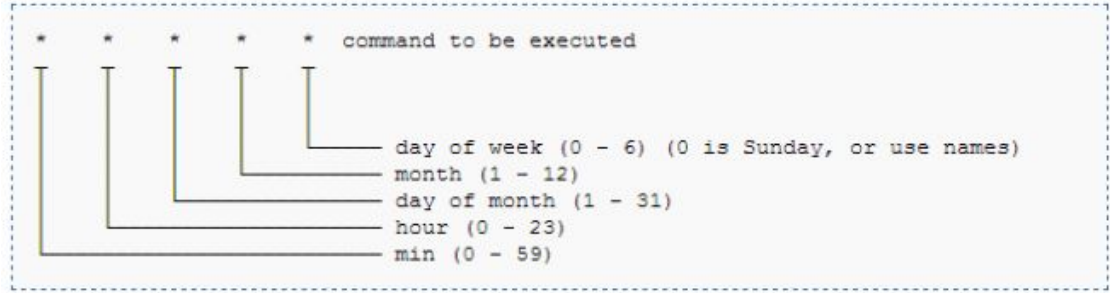
Un archivo de texto. Por simple que parezca la definición es así. Aunque es verdad que se trata de un archivo con contenido especial. Posee una lista con todos los scripts a ejecutar. Generalmente cada usuario del sistema posee su propio fichero Crontab. Se considera que el ubicado en la carpeta etc pertenece al usuario root.

Para generar el archivo propio, cada usuario deberá hacer uso del comando crontab (sí, es el mismo nombre).



Para agregar una tarea o modificar alguna, editamos el archivo crontab con el comando:

`crontab -e`



5 asteriscos y el comando a ejecutar. Cada uno de los 5 asteriscos significa:

m: minuto

h: hora

dom: día de la semana

mon: mes

dow: día del mes

Hay que decir que en Linux existen algunas cadenas de texto reservadas para ejecutar procesos durante determinados periodos:

@reboot: Ejecuta una vez y nada más iniciarse el equipo.

@yearly: ejecuta sólo una vez al año: 0 0 1 1 \*

@monthly: ejecuta una vez al mes y el primer día: 0 0 1 \* \*

@weekly: Todas las semanas, el primer minuto de la primer hora de la semana: 0 0 \* \* 0.

@daily: todos los días a las 12 de la noche: 0 0 \* \* \*

@midnight: Tiene el mismo efecto que el anterior.

@hourly: todas las horas durante su primer minuto: 0 \* \* \* \*

Aunque os hemos puesto cuál sería el formato, al hacer uso de crontab -e se pueden utilizar estos términos para definir el periodo.

*Ángel González M.*

# Ejemplos de crontab

Por ejemplo, deseamos ejecutar un script todos los días a las 23:30Hrs

**30 23 \* \* \* /home/usuario/mi\_script.sh**

Cada \* significa “cualquiera”, es decir, se ejecutará a las 23:30 de cualquier día, cualquier mes y cualquier día de la semana.

Si deseamos ejecutar un script el día de navidad cada año:

**00 00 25 12 \* /home/usuario/saludo\_navidad.sh**

El 25 indica el día del mes y el 12 el mes del año, lo que daría 25 de Diciembre.

Si es que deseamos ejecutar un script todos los martes 13 a la 1am:

**00 01 13 \* 2 /home/usuario/script-martes-13.sh**

El 13 indica el decimotercer día del mes, el \* que le sigue indica que es cualquier mes y el 2 siguiente indica que se ejecute sólo los días martes (los días de la semana se numeran empezando con 1 para Lunes y terminando con 7 para domingo), así, si el 13 de enero cae jueves, no se ejecutará el script, pero si el 13 de abril cae martes, lanzará el script

Si es que deseamos tener Amarok como un despertador, empezando a tocar nuestra lista de reproducción de lunes a viernes a las 7am:

# Ejemplos de crontab

Ejecutar todos los días a las 7 de la tarde

```
00 19 * * * usuario /ubicacion/del/script/consulta.sh
```

Ejecutar todos los domingos a las 7 de la tarde:

```
00 19 * * 0 usuario /ubicacion/del/script/consulta.sh
```

Ejecutar el script todos los 4 de febrero a las 7 de la tarde:

```
00 19 4 2 * usuario /ubicacion/del/script/consulta.sh
```

# Administración de los jobs del Cron

Para reemplazar el archivo existente por otro que defina el usuario se debe utilizar el siguiente comando:

```
>crontab archivo
```

Para editar el archivo existente en la actualidad se utiliza el comando que ya hemos visto

```
>crontab -e
```

Listar todas las tareas existentes en el crontab del usuario:

```
>crontab -l
```

Borrar el crontab que está configurado:

```
>crontab -d
```

Definir el directorio en el que se almacenará el archivo de crontab. Para realizar esta operación se deben tener permisos de ejecución en dicho directorio:

```
>crontab -c dir
```

Instrucción para manejar el crontab de otros usuarios existentes en el sistema:

```
>crontab -u usuario
```

*Ángel González M.*

# apt → (advanced packets tool: herramienta avanzada de paquetes)

apt es la herramienta que utiliza Debian y sus derivadas (Ubuntu incluida), para gestionar los paquetes instalables disponibles en los repositorios.

[Ver enlace](#)

# aptitude → (aptitude: aptitud, habilidad)

Es una versión mejorada de apt y en Ubuntu ya no viene instalado por defecto. Nació como un front-end de apt, es decir, como una especie de aplicación gráfica y en modo texto para realizar todo lo que hace apt. Pero lo cierto es que sus características son mejores.

Para abrir el interfaz gráfico de aptitude, tan sólo hay que teclearlo en la terminal:

```
>aptitude
```

Por supuesto, también se puede usar exactamente igual que apt:

```
>aptitude search nombre_paquete
```

```
>aptitude install nombre_paquete
```

```
>aptitude remove nombre_paquete
```

```
>aptitude purge nombre_paquete
```

```
>aptitude update
```

```
>aptitude upgrade
```

# dpkg → (depackage: despaquetar)

Los paquetes cuando se instalan sufren un proceso de desempaquetado. En el fondo un paquete .deb contiene una serie de scripts de pre-instalación, post-instalación y los archivos en cuestión del paquete.

Este comando lo usaremos para instalar un paquete .deb que ya tengamos descargado en nuestro sistema. En muchas ocasiones hay una aplicación que no está en los repositorios y nos hemos bajado el .deb para instalarlo con el interfaz gráfico que corresponda (GDebi en el caso de GNOME). En el fondo estas interfaces gráficas están basadas en dpkg.

Si queremos instalar un paquete ya descargado mediante consola usaremos el argumento '-i' (i=install):

```
>dpkg -i nombre_paquete
```

Para desinstalarlo '-r' (r=remove):

```
>dpkg -r nombre_paquete
```

Para desinstalar el paquete y los ficheros de configuración "--purge" (purgar):

```
>dpkg -r --purge nombre_paquete
```

# Alien → (Alien: de otro país, de otro planeta)

Aunque Debian -y por extensión Ubuntu, Mint, ...- dispone de una ingente cantidad de paquetes en sus repositorios, puede que alguien tenga algún problema en encontrar una aplicación específica empaquetada como le interesa aunque ha visto el paquete que quiere para otras distros.

Alien es bastante práctico para estas situaciones ya que nos permite transformar un paquete de un gestor de paquetes determinado en otro. Por ejemplo podemos pasar de un .rpm (Red Hat) a .deb (Debian) y viceversa. Las extensiones soportadas son:

- \* deb (Debian)
- \* rpm (Red Hat)
- \* slp (Stampede)
- \* tgz (Slackware)
- \* pkg (Solaris)

Su uso es sencillo. Lo que debemos saber es el argumento que transformará el paquete original en la extensión objetivo:

“-to-deb” o “-d” → para transformar a .deb

“-to-rpm” o “-r” → para transformar a .rpm

“-to-tgz” o “-t” → para transformar a .tgz

“-to-pkg” o “-p” → para transformar a .pkg

“-to-slp” → para transformar a .slp

Como ejemplo, pasaremos un supuesto paquete de Red Hat llamado “programa.rpm” a “programa.deb”:

alien -d programa.rpm

Ángel González M.



# Comprimir y descomprimir archivos

Hay varios comandos:

- zip
- tar
- gzip
- gunzip
- rar
- bzip2
- bunzip

# Ejemplos típicos de compresion y descompresion

**bunzip2 archivo1.bz2:** descomprime el archivo con nombre 'archivo1.bz2'.

**bzip2 archivo1:** comprime un archivo con nombre 'archivo1'.

**gunzip archivo1.gz:** descomprime el archivo con nombre 'archivo1.gz'.

**gzip archivo1:** comprime un archivo llamado 'archivo1' y genera 'archivo1.gz'.

**gzip -9 archivo1:** comprime con compresión al máximo.

**rar a file1.rar test\_file:** crear un fichero rar llamado 'file1.rar'.

**rar a file1.rar file1 file2 dir1:** comprimir 'file1', 'file2' y 'dir1' simultáneamente.

**rar x archivo1.rar:** descomprime el archivo archivo1.rar.

**unrar x archivo1.rar:** descomprime el archivo archivo1.rar.

**tar -cvf destino.tar archivo1:** crear un tarball descomprimido llamado destino.tar.

**tar -cvf destino.tar archivo1 archivo2 dir1:** crear un archivo destino.tar que contiene los archivos y directorio 'archivo1', 'archivo2' y 'dir1'.

**tar -tf archivo.tar:** mostrar los contenidos de archivo.tar.

**tar -xvf archivo.tar:** extraer un tarball.

**tar -xvf archivo.tar -C /tmp:** extraer un tarball en la ruta /tmp.

**tar -cvfj destino.tar.bz2 dir1:** crear un tarball comprimido con bzip2.

**tar -xvfj archivo.tar.bz2:** descomprimir un archivo tar comprimido con bzip2

**tar -cvfz destino.tar.gz dir1:** crear un tarball comprimido con gzip.

**tar -xvfz archivo.tar.gz:** descomprimir un archivo tar comprimido con gzip.

**tar -xvfz archivo.tar.gz dir1/archivo1.txt :** extraer solo el archivo1.txt que está en el directorio dir1

**tar -xvfz archivo.tar.gz dir1/dir2/dir3/ --strip-components=3 :** Extraer los archivos contenidos en dir3 y eliminar la ruta dir1/dir2/dir3/ (3 componentes) de los archivos al extraerlos.

**zip destino1.zip archivo1:** crear un archivo comprimido en zip.

**zip -r destino1.zip archivo1 archivo2 dir1:** comprimir, en zip, varios archivos y directorios de forma simultánea

**unzip archivo1.zip:** descomprimir un archivo zip.

Ángel González M.

# Ejemplos típicos de uso de RPM

**rpm -ivh package.rpm:** instalar un paquete rpm.

**rpm -ivh --nodeeps package.rpm:** instalar un paquete rpm ignorando las peticiones de dependencias.

**rpm -U package.rpm:** actualizar un paquete rpm sin cambiar la configuración de los ficheros.

**rpm -F package.rpm:** actualizar un paquete rpm solamente si este está instalado.

**rpm -e package\_name.rpm:** eliminar un paquete rpm.

**rpm -qa:** mostrar todos los paquetes rpm instalados en el sistema.

**rpm -qa | grep httpd:** mostrar todos los paquetes rpm con el nombre “httpd”.

**rpm -qi package\_name:** obtener información en un paquete específico instalado.

**rpm -qg “System Environment/Daemons”:** mostrar los paquetes rpm de un grupo software.

**rpm -ql package\_name:** mostrar lista de ficheros dados por un paquete rpm instalado.

**rpm -qc package\_name:** mostrar lista de configuración de ficheros dados por un paquete rpm instalado.

**rpm -q package\_name --whatrequires:** mostrar lista de dependencias solicitada para un paquete rpm.

**rpm -q package\_name --whatprovides:** mostrar la capacidad dada por un paquete rpm.

**rpm -q package\_name --scripts:** mostrar los scripts comenzados durante la instalación /eliminación.

**rpm -q package\_name --changelog:** mostrar el historial de revisiones de un paquete rpm.

**rpm -qf /etc/httpd/conf/httpd.conf:** verificar cuál paquete rpm pertenece a un fichero dado.

**rpm -qp package.rpm -l:** mostrar lista de ficheros dados por un paquete rpm que aún no ha sido instalado.

**rpm --import /media/cdrom/RPM-GPG-KEY:** importar la firma digital de la llave pública.

**rpm --checksig package.rpm:** verificar la integridad de un paquete rpm.

**rpm -qa gpg-pubkey:** verificar la integridad de todos los paquetes rpm instalados.

**rpm -V package\_name:** chequear el tamaño del fichero, licencias, tipos, dueño, grupo, chequeo de resumen de MD5 y última modificación.

**rpm -Va:** chequear todos los paquetes rpm instalados en el sistema. Usar con cuidado.

**rpm -Vp package.rpm:** verificar un paquete rpm no instalado todavía.

**rpm2cpio package.rpm | cpio --extract --make-directories \*bin\*:** extraer fichero ejecutable desde un paquete rpm.

**rpm -ivh /usr/src/redhat/RPMS/`arch`/package.rpm:** instalar un paquete construido desde una fuente rpm.

**rpmbuild --rebuild package\_name.src.rpm:** construir un paquete rpm desde una fuente rpm.

*Ángel González M.*

# Ejemplos: Actualizador de paquetes YUM (Red Hat, Fedora y similares)

**yum install package\_name:** descargar e instalar un paquete rpm.

**yum localinstall package\_name.rpm:** este instalará un RPM y tratará de resolver todas las dependencias para ti, usando tus repositorios.

**yum update package\_name.rpm:** actualizar todos los paquetes rpm instalados en el sistema.

**yum update package\_name:** modernizar / actualizar un paquete rpm.

**yum remove package\_name:** eliminar un paquete rpm.

**yum list:** listar todos los paquetes instalados en el sistema.

**yum search package\_name:** Encontrar un paquete en repositorio rpm.

**yum clean packages:** limpiar un caché rpm borrando los paquetes descargados.

**yum clean headers:** eliminar todos los ficheros de encabezamiento que el sistema usa para resolver la dependencia.

**yum clean all:** eliminar desde los paquetes caché y ficheros de encabezado.

# Ejemplos: Paquetes Deb (Debian, Ubuntu y derivados)

**dpkg -i paquete.deb:** instalar / actualizar un paquete deb.

**dpkg -r nombre\_del\_paquete:** eliminar un paquete deb del sistema.

**dpkg -l:** mostrar todos los paquetes deb instalados en el sistema.

**dpkg -l | grep httpd:** mostrar todos los paquetes deb con el nombre “httpd”

**dpkg -s nombre\_del\_paquete :** obtener información en un paquete específico instalado en el sistema.

**dpkg -L nombre\_del\_paquete :** mostrar lista de ficheros dados por un paquete instalado en el sistema.

**dpkg --contents paquete.deb :** mostrar lista de ficheros dados por un paquete no instalado todavía.

**dpkg -S /bin/ping :** verificar cuál paquete pertenece a un fichero dado

# date → (date: fecha)

Muestra por pantalla el día y la hora, permitiendo, además, el cambio de la misma.

La sinapsis del comando sería:

**>date [OPCIÓN]... [+FORMATO]**

o bien:

**>date [-u|--utc|--universal] [MMDDhhmm[[SS]AA][.ss]]**

Para ver las opciones, ejecutar:

**>date --help**

# Paquetes de idioma

1. Instalar los paquetes de idioma español:

```
>sudo apt install manpages-es manpages-es-extra
```

2. Recargar el idioma con:

```
>export LANG=es_ES.UTF-8
```

# cal → (calendar: calendario)

Muestra el calendario del mes o año actual actual.

La sinapsis del comando sería:

**>cal [mes] [año]**

Por ejemplo,

**>cal** → muestra el calendario del mes actual.

**>cal 2014** → muestra el calendario del año 2014.

**>cal 05 2015** → muestra el calendario de Mayo de 2015.

Ángel González M.



## who → (who: quien)

Indica qué usuarios tiene el ordenador en ese momento, en qué terminal (tty) está y a qué hora iniciaron la sesión.

La sinapsis del comando sería:

**>who [OPCIÓN]...**

**>who -a:** mostrar quien está registrado, e imprimir hora del último sistema de importación, procesos muertos, procesos de registro de sistema, procesos activos producidos por init, funcionamiento actual y últimos cambios del reloj del sistema.

*Ángel González M.*

# whoami → (who I am: quien soy)

Indica el usuario que está trabajando en la terminal actual.

La sinapsis del comando sería:

**>whoami**

# ¿Qué es el kernel?

Es el núcleo del sistema operativo

# uname

Proporciona el nombre del sistema en el que se está trabajando.

La sinapsis del comando sería:

**>uname [-opciones]**

Como opciones principales tenemos:

-a → indica, además, la versión, fecha y tipo de procesador.

-m → indica, además, el tipo de de procesador.

-r → indica, además, la versión.

-v → indica, además, la fecha.

# Ejemplo comando uname

mostrar la arquitectura de la máquina (2).

```
>uname -m
```

mostrar la versión del kernel usado.

```
>uname -r
```

mostrar los componentes (hardware) del sistema.

```
>dmidecode -q
```

# ¿Cómo se que versión del kernel tengo?

`uname -r`

# alias

Asigna un nombre o etiqueta a la ejecución de un comando con sus opciones.

La sinapsis del comando sería:

**>alias etiqueta='orden'd**

La orden alias solamente, muestra todos los alias que hay creados. La orden unalias elimina el alias especificado.

Ahora, si lo queremos de forma permanente, esto lo ponemos dentro del fichero **~/.bashrc** el cual está en nuestro **/home**, y si no está, pues lo creamos (*siempre con el punto delante*). Cuando ya tengamos añadida la línea del **alias** en este fichero, simplemente ponemos en consola:

**\$ . .bashrc**

**o**

**source ~/.bashrc**

Ángel González M.



# Alias con parámetros

A los alias se le pueden pasar parámetros con \$1, \$2, etc

Ej

```
> alias editar='pico $1'
```

Ahora podemos llamar a editar pasando como parámetro un fichero

```
> editar fichero.txt
```

# Borrar un alias

```
unalias nombre_del_alias
```

# clear

Este comando se utiliza para limpiar la pantalla de la terminal.

La sinapsis del comando sería:

**>*clear***

# Como se sale de una sesión en la consola

**>exit**

# ¿De qué forma puedo averiguar el uso de memoria que consume mi ordenador?

>free

# ¿Con qué comando se monitorea la memoria y uso de CPU?

**>top**

**Para salir pulsamos la tecla q**

# ¿Puedo tener más de una consola abierta?

Sí en modo gráfico es fácil (abres muchas consolas).

Y en modo consola, se puede alternar entre ellas con el atajo de teclado alt +f1, f2, f3 según a la consola que queramos cambiar

# ¿Donde se escriben los comandos?

En el prompt



# ¿Cómo se le llama al intérprete de comandos?

Una **Shell** de Unix o también **shell**, es el término usado en informática para referirse a un intérprete de comandos, el **cual** consiste en la interfaz de usuario tradicional de los sistemas operativos basados en Unix y similares como GNU/Linux.

El **interprete de comandos** es el programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el sistema operativo. En otras palabras el objetivo de cualquier **intérprete de comandos** es ejecutar los programas que el usuario teclea en el prompt del mismo.

El nombre del **interprete** Bourne en la mayoría de los UNIX es /bin/sh (donde sh viene de "shell", **interprete** de **comandos** en ingles). ... Bajo **Linux** hay algunas diferencias en los interpretes de **comandos** disponibles. Dos de los más usados son el "Bourne Again Shell" o "Bash" (/bin/bash) y Tcsh (/bin/tcsh).

# ¿Qué comando reinicia el equipo?

reboot

# ¿Comando para apagar mi ordenador?

halt

o

poweroff

# Comando para apagar en un tiempo determinado

## **shutdown**

- a Control access to the shutdown command using the control access file /etc/shutdown.allow. See Access Control below for more information.
- k Do not shut down, but send the warning messages as if the shutdown were real.
- r Reboot after shutdown.
- h Instructs the system to shut down and then halt.
- P Instructs the system to shut down and then power down.
- H If -h is also specified, this option instructs the system to drop into boot monitor on systems that support it.
- f Skip fsck after reboot.
- F Force fsck after reboot.
- n Don't call init to do the shutdown of processes; instruct shutdown to do that itself.

The use of this option is discouraged, and its results are not always predictable.

- c Cancel a pending shutdown. (This does not apply to "shutdown now", which does not wait before shutting down.) With this option, it is not possible to give the time argument, but you can still specify an explanatory message that will be sent to all users.
- t sec Tell init to wait sec seconds between sending processes the warning and the kill signal, before changing to another runlevel.

*Ángel González M.*

# ¿Cómo puedo obtener información detallada de un comando?

con el comando **man "nombrecomando"**

Ejemplo:

**man ping**: mostrar las páginas del manual de un comando ping, usar la opción '-k' para encontrar cualquier comando relacionado.

# Desloguearse (cerrar sesión)

>logout

# Avanzado: Monitoreando y Depurando

top: mostrar las tareas de linux usando la mayoría cpu.

ps -eafw: muestra las tareas Linux.

ps -e -o pid,args --forest: muestra las tareas Linux en un modo jerárquico.

pstree: mostrar un árbol sistema de procesos.

kill ID\_Processo: llamar el cierre de un proceso y esperar a que termine normalmente.

kill -9 ID\_Processo: forzar el cierre de un proceso y terminarlo.

kill -1 ID\_Processo: forzar un proceso para recargar la configuración.

killall httpd : Matar todos los procesos con nombre httpd o el nombre indicado.

pkill -u user1 : Matar todos los procesos del usuario user1.

lsof -p \$\$: mostrar una lista de ficheros abiertos por procesos.

lsof /home/user1: muestra una lista de ficheros abiertos en un camino dado del sistema.

strace -c ls >/dev/null: mostrar las llamadas del sistema hechas y recibidas por un proceso.

strace -f -e open ls >/dev/null: mostrar las llamadas a la biblioteca.

watch -n1 'cat /proc/interrupts': mostrar interrupciones en tiempo real.

last reboot: mostrar historial de reinicio.

lsmod: mostrar el kernel cargado.

free -m: muestra el estado de la RAM en megabytes.

smartctl -A /dev/sda: monitorear la fiabilidad de un disco duro a través de SMART.

smartctl -i /dev/sda: chequear si SMART está activado en un disco duro.

tail /var/log/dmesg: mostrar eventos inherentes al proceso de carga del kernel.

tail /var/log/messages: mostrar los eventos del sistema.

w : Mostrar todas las sesiones de terminales y listar los usuarios que las invocaron

w | grep pts | sort -k1 : Mostrar las sesiones de terminales ordenadas por usuario

*Ángel González M.*

# Los archivos `/etc/bashrc`, `/etc/profile`, `~/.bashrc`, `~/.bash_profile`, `~/.bash_logout`.

Para todos los usuarios: (Se necesita permisos de root para editar/modificar estos archivos)

- **`/etc/profile`** --> Se ejecuta cuando cualquier usuario inicia la sesión.
- **`/etc/bashrc` o `/etc/bash.bashrc`**--> Se ejecuta cada vez que cualquier usuario ejecuta el programa bash

Para nuestro usuario particular:

- **`~/.bash_profile`** --> Se ejecuta el `.bash_profile` de juanito cuando juanito inicia su sesión.
- **`~/.bashrc`** --> Se ejecuta el `.bashrc` de juanito cuando juanito ejecuta el programa bash.
- **`~/.bash_logout`** es el fichero leído por Bash, cuando salimos del sistema.



# Ejecutar un servicio, programa o script cuando linux arranca

Creamos un script con las tareas a realizar en la siguiente localización

<http://www.instructables.com/id/Nodejs-App-As-a-RPI-Service-boot-at-Startup/>

```
>sudo nano /etc/init.d/miScript.sh
```

Damos los permisos adecuados. Los permisos 755 indican que el usuario root puede leer, escribir y ejecutar el archivo, los demás usuarios solamente pueden leer y ejecutar.

```
>sudo chmod 755 /etc/init.d/miScript.sh
```

En este caso ln -s nos sirve para crear un enlace simbólico, posteriormente es la ruta de origen del archivo y después el destino. En esta última parte es muy importante la sintáxis, la primera letra 'S' significa que el script se debe de ejecutar, el número '88' significa el orden de ejecución que puede ir desde 0 hasta 99 y al final una palabra de referencia para saber a qué proceso o programa apunta este enlace.

```
>sudo ln -s /etc/init.d/miScript.sh /etc/rc2.d/S88miServicio
```

Una vez hecho esto tendremos nuestro programa ejecutándose cada vez que el sistema arranque.

# Otros comandos útiles

`apropos ...keyword`: mostrar una lista de comandos que pertenecen a las palabras claves de un programa; son útiles cuando tú sabes qué hace tu programa, pero desconoces el nombre del comando.

`whatis keyword`: muestra la descripción de lo que hace el programa.

`gpg -c file1`: codificar un fichero con guardia de seguridad GNU.

`gpg file1.gpg`: decodificar un fichero con Guardia de seguridad GNU.

`ldd /usr/bin/ssh`: mostrar las bibliotecas compartidas requeridas por el programa ssh.

`chsh`: cambiar el comando Shell.

`chsh --list-shells`: es un comando adecuado para saber si tienes que hacer remoto en otra terminal.

# \$USER

Hace referencia al nombre usuario

echo \$USER

## ; concatenar comando

Ejecuta un comando tras otro. Cuando acaba un comando se ejecuta el siguiente.

comando1 ; comando2 ; comando3

Ejemplo:

ls ; mkdir carta ; cd carta

# && concatenar comando

Ejecutar el comando2 solo si se ejecuta el comando1:  
comando1 && comando2

Ejemplo:

mkdir directorio && cd directorio

# || concatenar comando

Ejecutar el comando2 solo si no se ejecuta el primero  
comando1 || comando2

Ejemplo

```
mkdir directorio || echo 'error'
```

# & comando en segundo plano

Si queremos ejecutar un comando en segundo plano (background) terminaremos el comando con el símbolo &

Ej `rm directorio &`

!!

Este comando repite el último comando que has escrito

Muy util si nos hemos olvidado de anteponer el sudo al comando anterior.

**# sudo !!**