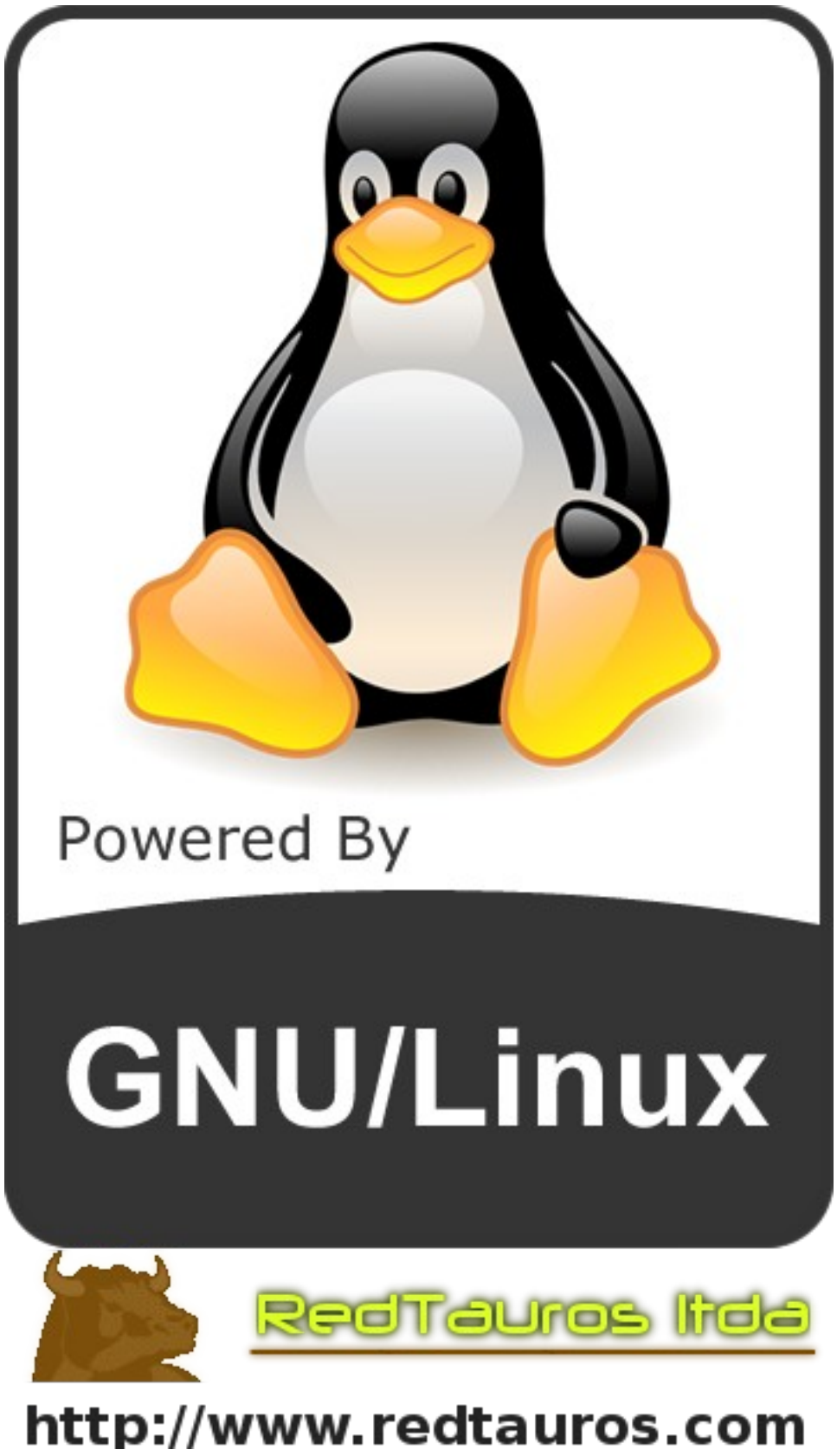

Powered By
GNU/Linux
RedTauros ltda
http://www.redtauros.com

Índice de contenido

EL SHELL: COMANDOS BÁSICOS DE LINUX

Introducción

Un intérprete de comandos es un programa que toma la entrada del usuario, por ejemplo las órdenes que teclea, y la traduce a instrucciones.
Podemos compararlo con el COMMAND.COM de MS-DOS, que realiza exactamente la misma tarea.
El intérprete de comandos (Shell) será una de las interfaces con Linux, y el X Window será otra interfaz que nos permite ejecutar órdenes usando el ratón y el teclado. Cuando accedemos al sistema entramos por defecto en el entorno gráfico de X Windows, el KDE. Para pasar al modo texto (intérprete de comandos) desde el modo gráfico hemos de pulsar la combinación:

<ctrl>+<alt>+<F1>
o bien:
<ctrl>+<alt>+<F2>
o bien:
<ctrl>+<alt>+<F3>
o bien:
<ctrl>+<alt>+<F4>
o bien:
<ctrl>+<alt>+<F5>
o bien:
<ctrl>+<alt>+<F6>

Esto hace que el sistema salga del modo gráfico y acceda a alguna de las seis consolas virtuales de Linux, a las cuales también se puede acceder cuando se arranca en modo de texto.
Para volver al modo gráfico hay que presionar <ctrl>+<alt>+<F7> o <ctrl>+<alt>+<F8> (Según la sesión en modo gráfico a la que deseemos regresar).
La segunda forma es más cómoda y menos radical, permitiendo acceder al shell desde el mismo entorno gráfico. Para esto hay que abrir un programa llamado terminal o consola, el kconsole (en el entorno KDE), o los xterm o gnome-terminal (en GNOME).

Además de ser un intérprete interactivo de los comandos que tecleamos, el Shell es también un lenguaje de programación, el cual nos permite escribir guiones que permiten juntar varias órdenes en un fichero. Similar a los ficheros batch de MS-DOS.

En Unix existen varios tipos de intérpretes de comandos. Los dos más importantes son:

- Sh o Bourne shell: utiliza una sintaxis similar a la usada en los primeros sistemas Unix.
- Csh o C shell: utiliza una sintaxis diferente a la de sh, similar al lenguaje de programación C.

En Linux también disponemos de varios intérpretes de comandos, con algunas diferencias respecto a los de Unix. Los más usados son:

- Bash o Bourne Again Shell: es equivalente a Bourne shell, pero con muchas características avanzadas de la C shell. Cualquier guión (script) escrito para Bourne funcionará en bash.
- Tsch (exTended C shell): es una versión extendida del C original.

Sólo aparecerán diferencias entre unos y otros a la hora de escribir guiones. Es decir, en lo que respecta a los comandos usuales es indiferente el tipo de intérprete de comandos usado.

NOCIONES BÁSICAS

El formato general de una orden en Linux es:

comando *[-opciones] [argumentos]*

A la hora de introducir los comandos hay que tener en cuenta las siguientes características:

• Los comandos hay que teclearlos exactamente.
• Las letras mayúsculas y minúsculas se consideran como diferentes (Case Sensitive).
• En su forma más habitual (los shells de Bourne o de Korn), el sistema operativo utiliza un signo de $ como prompt para indicar que está preparado para aceptar comandos, aunque este carácter puede ser fácilmente sustituido por otro u otros elegidos por el usuario. En el caso de que el usuario acceda como administrador este signo se sustituye por #.
• Cuando sea necesario introducir el nombre de un fichero o directorio como argumento a un comando, Linux, permite escribir las primeras letras del mismo y realiza un autorrellenado al presionar la tecla del tabulador. Si no puede distinguir entre diversos casos rellenará hasta el punto en el que se diferencien.

Por ejemplo, supongamos una carpeta con los siguientes directorios:

Programas
Documentos_proyecto
Documentos_privados

Al escribir ***cd*** *Pr<tab>* Linux rellenará el resto del contenido hasta escribir ***cd*** *Programas*. Por el contrario al escribir ***cd*** *D<tab>* escribirá ***cd*** *Documentos_*

COMANDOS BÁSICOS

passwd

Podemos cambiar la contraseña empleando la orden ***passwd***. Nos pedirá la contraseña anterior (current) y la nueva. Volverá a pedir una segunda vez la nueva para validarla. El usuario root podrá cambiar la contraseña de cualquier otro usuario. Sin embargo, los usuarios no privilegiados solamente podrán cambiar su propia clave. Sintaxis:

passwd *usuario*

date

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

date *[opción][formato]*

cal

Muestra el calendario del mes o año actual actual. Sintaxis:

cal *[mes][año]*

Por ejemplo,
- ***cal*** : muestra el calendario del mes actual.
- ***cal*** *1949* : muestra el calendario del año 1949.

- ***cal*** *05 1945* : muestra el calendario de Mayo de 1949.

who

Indica qué usuarios tiene el ordenador en ese momento, en qué terminal están y a qué hora iniciaron la sesión. Sintaxis :

who

whoami

Indica el usuario que está trabajando en la terminal actual. Sintaxis:

whoami

lastlog

Le permite al administrador del servidor conocer la última fecha de acceso, de los usuarios del sistema. Sintaxis:

whoami

uname

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

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.

logname

Indica el nombre del usuario conectado al sistema (el que ha hecho login). Sintaxis:

logname

man

Todos los manuales de Linux se encuentran dentro del propio sistema operativo. Este comando permite acceder a la información correspondiente a la orden que se le especifique como parámetro. Sintaxis:

man *comando*

Por ejemplo, ***man*** *who* hace aparecer por pantalla de forma paginada la ayuda correspondiente al comando *who*. Se puede navegar por estas páginas usando los cursores o con las tecla Intro o e del teclado. Para salir de ella, habrá que pulsar q.

info

Proporciona ayuda resumida acerca de un comando en cuestión. Sintaxis:

info *[comando]*

clear

Este comando se utiliza para limpiar la pantalla. Sintaxis:

clear

echo

Muestra por pantalla los argumentos que le pasamos. Sintaxis:

echo *[argumento1] [argumento2] ... [argumentoN]*

alias

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

alias *etiqueta='orden'*

Por ejemplo :

alias *apu='sudo apt-get update'*

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

CARACTERES COMODÍN O WILDCARDS

Una característica importante de la mayoría de los intérpretes de comandos en Linux es la capacidad para referirse a más de un fichero. Una forma de hacerlo es utilizando caracteres especiales llamados comodines.

Al igual que en MS-DOS, el comodín * hace referencia a cualquier carácter o cadena de caracteres en el nombre del fichero. El intérprete de comandos sustituirá el asterisco por todas las combinaciones posibles provenientes de los ficheros en el directorio al cual nos estamos refiriendo. Se dice que está realizando una expansión de comodines.

El carácter ? es también comodín, aunque solamente expande un carácter. Con ambos caracteres existe una excepción. No afectarán a aquellos ficheros que comienzan por un punto, y que son ocultos para órdenes como ***ls***.

Además, podemos utilizar los corchetes para referirnos a un conjunto de caracteres o bien un rango de caracteres ASCII.

Ejemplos:

ls **n**

muestra todos los archivos y directorios, del directorio actual, que contienen el carácter n

ls ***

muestra todos los archivos y directorios del directorio actual

ls *tm?*

muestra todos los archivos y directorios del directorio actual que comienzan por tm y contienen tres caracteres

ls *tabla[123]a*

muestra todos los archivos y directorios del directorio actual que comienzan por tabla, seguidos del carácter 1, 2 ó 3, y terminan en a

ls *??base[A-Z][5-9]**

muestra todos los archivos y directorios del directorio actual que comienzan con dos caracteres cualesquiera, seguidos de la cadena base, a continuación una letra mayúscula, seguida de un número del 5 al 9 y por último una cadena de caracteres (uno, varios o ninguno)

ÓRDENES RELACIONADAS CON DIRECTORIOS

Directorio Personal

Como se ha visto anteriormente el directorio personal es un directorio con un determinado nombre asignado a un usuario. Los directorios personales habitualmente son subdirectorios de /home

Generalmente el nombre coincide con el del nombre de usuario, aunque puede no ser así, y varios usuarios pueden estar trabajando en el mismo directorio. Cada usuario de Linux puede crear una estructura en árbol de subdirectorios y archivos tan compleja como desee bajo su directorio personal pero normalmente nunca fuera de él.

ls

Permite mostrar el contenido de un directorio. ***ls*** Muestra los nombres de los ficheros y subdirectorios contenidos en el directorio en el que se está. Sólo se obtienen los nombres de los ficheros, sin ninguna otra información. Sintaxis:

ls *[-opciones][fichero]*

Opciones :

-a : Muestra todos los ficheros incluyendo algunos que ordinariamente están ocultos para el usuario (aquellos que comienzan por un punto). Recordemos que el fichero punto . indica el directorio actual y el doble punto .. el directorio padre, que contiene, al actual.
-l : Esta es la opción de lista larga: muestra toda la información de cada fichero incluyendo: protecciones, tamaño y fecha de creación o del último cambio introducido,...
-c : Muestra ordenando por día y hora de creación.
-t : Muestra ordenando por día y hora de modificación.
-r : Muestra el directorio y lo ordena en orden inverso.
-R : Lista también subdirectorios.
ls subdir : Muestra el contenido del subdirectorio subdir.
-l filename : Muestra toda la información sobre el fichero filename.

Las opciones anteriores pueden combinarse. Por ejemplo:

ls *-cr*

Muestra el directorio ordenando inversamente por fechas.
El comando ls admite los caracteres de sustitución o * y ?. Por ejemplo:

ls **.gif*

Muestra todos los nombres de ficheros que acaben en .gif, por ejemplo, dib1.gif, a.gif, etc.

ls *file?*

Muestra todos los ficheros cuyos nombres empiecen por file y tengan un nombre de cinco caracteres, por ejemplo: file1, file2, filea, etc.

mkdir

El comando ***mkdir*** (make directory) permite a cada usuario crear un nuevo subdirectorio. Sintaxis:

mkdir *subdirectorio*

donde subdirectorio es el nombre del directorio que se va a crear.

rmdir

Este comando borra uno o más directorios del sistema (remove directory), siempre que estos subdirectorios estén vacíos. Sintaxis:

rmdir *subdirectorio*

Por ejemplo, ***rmdir*** *subdir1*, donde subdir es el nombre del directorio que se va a eliminar.

cd

Este comando permite cambiar de directorio a partir del directorio actual de trabajo. Sintaxis:

cd *[directorio]*

Veamos algunas opciones:
cd – : cambia al último directorio en el que estuvimos antes del actual.
cd.. : cambia al directorio padre.
cd ~ : nos lleva a nuestra casa home.
cd : Nos sitúa nuevamente en el directorio personal del usuario.
cd / : cambia al directorio raíz

Nota: al contrario que en MS-DOS en Linux no existe la forma ***cd..*** sin espacio entre ***cd*** y los dos puntos.

pwd

El comando ***pwd*** (print working directory) visualiza o imprime la ruta del directorio en el que nos encontramos en este momento. Este comando es uno de los pocos que no tiene opciones y se utiliza escribiendo simplemente ***pwd***.

ACCESO A UNIDADES DE DISCO

Montaje y desmontaje

Linux a diferencia de Windows no utiliza letras ("a:", "c:", "d:", ...) para acceder a las distintas unidades de disco de un ordenador. En Linux para acceder al contenido de una unidad de disco o de un CD-ROM este tiene que haber sido previamente "montado". El montado se realiza mediante el comando ***mount***, con lo que el contenido de la unidad se pone a disposición del usuario en el directorio de Linux que se elija. La sintaxis de este comando es la siguiente:

mount *[-t tipo_de_sistema_ficheros] [dispositivo] directorio_de_montaje*

Por ejemplo para acceder al CD-ROM se tecleará el siguiente comando:

mount *-t iso9660 /dev/cdrom /mnt/cdrom*

donde -t iso9660 indica el tipo de sistema que usa la unidad de disco para guardar los ficheros (las más usuales son: iso9660 en el caso de un CD-ROM, vfat en el caso de Windows, y ext4 en el caso de Linux), /dev/cdrom indica el dispositivo que se va a montar. Todos los dispositivos están representados por un fichero del directorio /dev; por ejemplo, en el caso de un disquete será seguramente /dev/fd0, por último /mnt/cdrom es el directorio en el que se pondrá a disposición del usuario el contenido del CD-ROM. Para montar disquetes se suele utilizar el directorio /mnt/floppy (aunque esto depende de la versión de Linux que utilicemos).

De todas formas el usuario siempre puede crear un directorio vacío con el nombre que el elija para montar las unidades de disco que desee donde desee.

Cuando el usuario haya dejado de usar ese disco deberá "desmontarlo" mediante el comando ***umount*** antes de sacar el disquete o el CD-ROM.

umount */floppy*

En principio, para utilizar el comando mount especificando todos los parámetros hace falta ser administrador o root. Para que un usuario común pueda utilizar disquetes, CD-ROM, etc. hay que editar el fichero /etc/fstab. Por ejemplo para que cualquier usuario pueda acceder a un disquete habrá que indicar la siguiente línea:

/dev/fd0 /mnt/floppy vfat user,noauto 0 0

También habrá que asegurarse de que el directorio /mnt/floppy sea accesible por todos los usuarios.
Una vez seguidos los pasos anteriores cualquier usuario podrá "montar" un disquete escribiendo el siguiente comando:

mount */mnt/floppy*

Al igual que antes, el usuario deberá ejecutar el comando ***umount*** */mnt/floppy* antes de sacar el disquete.

Nota: En algunas distribuciones se montan las unidades en /media y en otras en /mnt.

Para montar una memoria USB, aunque la mayoría de las distribuciones la monta de manera automática, a veces en el ambiente de consola no siempre es así. Por eso lo

vamos a hacer paso a paso.

Para identificar en donde esta la memoria USB (despues de haber insertado la memoria USB), usamos el comando ***dmesg*** y buscamos las ultimas lineas:

dmesg

```
….
[ 6591.591987] scsi6 : usb-storage 1-3:1.0
[ 6591.592462] usbcore: registered new interface driver usb-storage
[ 6591.592465] USB Mass Storage support registered.
[ 6592.593661] scsi 6:0:0:0: Direct-Access     SanDisk  Cruzer Micro     0.1
PQ: 0 ANSI: 2
[ 6592.594951] sd 6:0:0:0: Attached scsi generic sg2 type 0
[ 6592.596252] sd 6:0:0:0: [sdb] 4001760 512-byte logical blocks: (2.04
GB/1.90 GiB)
[ 6592.597934] sd 6:0:0:0: [sdb] Write Protect is off
[ 6592.597946] sd 6:0:0:0: [sdb] Mode Sense: 03 00 00 00
[ 6592.597954] sd 6:0:0:0: [sdb] Assuming drive cache: write through
[ 6592.600236] sd 6:0:0:0: [sdb] Assuming drive cache: write through
[ 6592.600254]  sdb: sdb1
```

aquí vemos que se identifica como sdb:sdb1 (tiene una particion), esto quiere decir que usaremos sdb1. Ahora creamos una carpeta en donde montar la memoria.

mkdir *1*

Y montamos la memoria en esa carpeta:

sudo *mount /dev/sdb1 1/*

ÓRDENES RELACIONADAS CON FICHEROS

cp

Copia un fichero o ficheros en otro fichero o directorio. Sintaxis:

cp *fichero1 [fichero2] ... [ficheroN] destino*

donde [ficheroX] es el fichero a copiar y [destino] es el fichero o directorio de destino. Podemos utilizar . y .. para referirnos al directorio actual y al directorio padre respectivamente. Así pues, la orden ***cp*** *file1 file2*, hace una copia de file1 y le llama file2. Si file2 no existía, lo crea con los mismos atributos de file1. Si file2 existía antes, su contenido queda destruido y es sustituido por el de file1. El fichero file2 estará en el mismo directorio que file1.
Tanto file1 como file2 indican el nombre de un archivo, que puede incluir el la ruta al mismo si alguno de ellos no se encuentra en el directorio actual. Otra posibilidad es:

cp *file1 file2 namedir*

que hace copias de file1 y file2 en el directorio namedir.

mv

Se utiliza para el traslado y cambio de nombre de ficheros. Sintaxis:

cp *fichero1 [fichero2] ... [ficheroN] destino*

Como vemos, este comando tiene una forma similar al anterior. El comando mv realiza la misma función que el cp pero además destruye el fichero original. Así, si ejecutamos la orden

mv *file1 file2*

en definitiva se traslada el contenido de file1 a file2; a efectos del usuario lo que ha hecho es cambiar el nombre a file1, llamándole file2. De igual forma,

mv *file1 file2 namedir*

traslada uno o más ficheros (file1, file2,...) al directorio namedir conservándoles el nombre.

El comando,

mv *namedir1 namedir2*

cambia el nombre del subdirectorio namedir1 por namedir2.
Hay que recalcar que el comando ***mv*** sirve así mismo para cambiar el nombre de los ficheros.

rm

Borrado de ficheros. Este comando elimina uno o más ficheros de un directorio en el cual tengamos permiso de escritura. Sintaxis:

rm *file1 [file2]*

Con este comando resulta facilísimo borrar ficheros inútiles, y desgraciadamente, también los útiles. Por eso es conveniente y casi imprescindible emplear lo opción *–i*, de la forma siguiente:

rm *–i file1 file2*

Con esta opción, Linux pedirá confirmación para borrar cada fichero de la lista, de si realmente se desea su destrucción o no. Se recomienda usar siempre este comando con esta opción para evitar el borrado de ficheros útiles. Por ejemplo, si se teclea, ***rm*** *–i superfluo* aparecerá en pantalla el aviso siguiente: remove superfluo? y habrá que contestar y (yes) o n (not). En este comando se pueden utilizar los caracteres comodines (* y ?), como por ejemplo, ***rm*** *fich** que borraría todos los ficheros del directorio actual que comiencen por fich. El comando ***rm*** * borrará todos los ficheros del directorio actual. Otra opción es *–r*, que borra directorios recursivamente (borran el directorio y todo su contenido).

file

Este comando realiza una serie de comprobaciones en un fichero para tratar de clasificarlo, mostrando sus características. Sintaxis:

file *fichero*

Tras su ejecución este comando muestra el tipo del fichero e información al respecto del mismo. Este comando se puede aplicar también a directorios.

cat

Visualización sin formato de un fichero. Este comando permite visualizar el contenido de uno o más ficheros de forma no formateada. También permite copiar uno o más ficheros como apéndice de otro ya existente. Algunas formas de utilizar este

comando son las siguientes:

cat *filename*

Saca por pantalla el contenido del fichero filename.

cat *file1 file2*

Saca por pantalla, secuencialmente y según el orden especificado, el contenido de los ficheros indicados.

cat *>file1*

Acepta lo que se introduce por el teclado y lo almacena en file1 (se crea file1). Para terminar se emplea <ctrl>d . Si volvemos a ejecutar la instrucción *cat >file1*, borrara el contenido previo y lo reemplazara por el nuevo.

pr

Visualización de ficheros con formato. Este comando, a diferencia de cat, imprime por consola el contenido de los ficheros de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo de las mismas. Está muy en relación con el comando lp de salida por impresora. Las formas más importantes que admite son las siguientes:

pr *file*

Produce una salida estándar de 66 líneas por página, con un encabezamiento de 5 líneas (2 en blanco, una de identificación y otras 2 líneas en blanco).

pr *–ln file*

Produce una salida de n líneas por página (cuando el tamaño de papel de impresora, por ejemplo, tiene un número de líneas distinto de 66)

pr *–p file*

Hace una pausa para presentar la página, hasta que se pulsa <return> para continuar

pr –t : file Suprime las 5 líneas del encabezamiento y las del final de página.
pr –wn : file Ajusta la anchura de la línea a n posiciones.
pr –d : file Lista el fichero con espaciado doble.

La salida de este comando es por la consola, pero puede redireccionarse a otro fichero, por ejemplo, si ejecutamos el comando:

pr *file1 > file2*

se crea un fichero nuevo llamado file2 que es idéntico a file1, pero con formato por páginas y columnas.

Comandos more y less

Estos comandos permiten visualizar un fichero pantalla a pantalla. El número de líneas por pantalla es de 23 líneas de texto y una última línea de mensajes, donde aparecerá la palabra more. Cuando se pulsa la barra espaciadora (el espacio en blanco), se visualizará la siguiente pantalla. Para salir de este comando (terminar la visualización) se pulsa <ctrl>d o q. El comando ***more*** muestra el contenido de los ficheros indicados, una pantalla cada vez. Sintaxis:

more *fichero1 [fichero2] ... [ficheroN]*

La teclas que nos permiten movernos por el fichero son: b va a la página anterior, barra espaciadora va a la página siguiente, flechas de cursor arriba y abajo, q finaliza la ejecución de more.

El comando ***less*** es muy similar al anterior pero, además, permite el desplazamiento a lo largo del texto empleando las teclas de cursores pudiendo desplazarse, además, hacia la izquierda o la derecha. Sintaxis:

less *fichero1 [fichero2] ... [ficheroN]*

grep

El comando ***grep*** localiza una palabra, clave o frase en un conjunto de directorios, indicando en cuáles de ellos la ha encontrado. Este comando rastrea fichero por fichero, por turno, imprimiendo aquellas líneas que contienen el conjunto de caracteres buscado. Si el conjunto de caracteres a buscar está compuesto por dos o más palabras separadas por un espacio, se colocará el conjunto de caracteres entre apóstrofes ('). Su sintaxis es la siguiente:

grep *[-opcion] 'conjuntocaracteres' file1 file2 file3*

siendo 'conjuntocaracteres' la secuencia de caracteres a buscar, y file1, file2, y file3 los ficheros donde se debe buscar. Veamos un nuevo ejemplo:

grep *'TRIANGULARIZACION MATRIZ' matrix.f scaling.f*

Este comando buscará TRIANGULARIZACION MATRIZ entre las líneas de los ficheros matrix.f
y scaling.f.

Las opciones principales del comando son:

c : lo único que se hace es escribir el número de las líneas que satisfacen la condición.
i : no se distinguen mayúsculas y minúsculas.
l : se escriben los nombres de los ficheros que contienen líneas buscadas.
n : cada línea es precedida por su número en el fichero.
s : no se vuelcan los mensajes que indican que un fichero no se puede abrir.
v : se muestran sólo las líneas que no satisfacen el criterio de selección.

A continuación se muestra una serie de ejemplos.

grep *'^d' text* : líneas que comienzan por d.
grep *'^[^d]' text* : líneas que no comienzan por d.
grep *-v '^C' file1 > file2* : quita las líneas de file1 que comienzan por C y lo copia en file2.

head

Muestra las primeras líneas del contenido de los archivos especificados; por defecto muestra las 10 primeras líneas. Sintaxis:

head *[-número] <fichero1> <fichero2> ... <ficheroN>*

Por ejemplo, ***head*** *-7 texto*, escribe por pantalla las 7 primeras líneas del fichero texto.

tail

Muestra las ultimas líneas del contenido de los archivos especificados; por defecto muestra las 10 primeras líneas. Sintaxis:

tail *[-número] <fichero1> <fichero2> ... <ficheroN>*

OTROS COMANDOS BÁSICOS

Espacio ocupado en el disco: Comandos du y df

El comando ***du*** permite conocer el espacio ocupado en el disco por un determinado directorio y todos los subdirectorios que cuelgan de él. Para usarlo basta simplemente colocarse en el directorio adecuado y teclear ***du***. Este comando da el espacio de disco utilizado en bloques. Para obtener la información en bytes se debe emplear el comando con la opción *-h*:

du *-h*

El comando ***df*** por el contrario informa del espacio usado por las particiones del sistema que se encuentren montadas.

IMPRESIÓN

Comando lpr

El comando ***lpr*** se emplea para imprimir una serie de ficheros. Si se emplea sin argumentos imprime el texto que se introduzca a continuación en la impresora por defecto. Por el contrario, ***lpr*** *nombre_fichero* imprime en la impresora por defecto el fichero indicado.

BÚSQUEDA DE FICHEROS.

Comando find

Con ***find*** podemos encontrar archivos (y, por tanto, directorios) basando su búsqueda en distintas características de los mismos. Sintaxis:

find *[camino...] [expresión]*

El número de opciones de ***find*** es muy elevado, por lo que se aconseja acudir al manual para mayor información (*man* ***find***). No obstante, destaquemos algunas de ellas:
-mount : Impide la búsqueda en subdirectorios montados con otro tipo de sistema de ficheros distinto al directorio inicial de búsqueda.
-amin n : El fichero fue accedido por última vez hace n minutos.
-anewer file : El fichero fue accedido después de la fecha de la última modificación del fichero file.
-atime n : El fichero fue accedido por última vez hace n*24 horas.
-cmin n : El estado del fichero cambió por última vez hace n minutos.
-cnewer file : El estado del fichero cambió por última vez después de la última modificación del fichero file.
-ctime : El estado del fichero cambio por última vez hace n*24 horas.
-empty : El fichero (o directorio) está vacío.
-fstype tipo : El fichero se encuentra en el sistema de ficheros tipo.
-gid n : EL GID (identificativo de grupo) del fichero es n
-group name : El nombre del grupo al que pertenece el fichero es name.
-inum n : El fichero tiene como número de inodo n.

-links n : El fichero tiene n enlaces.
-mmin n : Los datos del fichero sufrieron su última modificación hace n minutos.
-mtime n : Los datos del fichero sufrieron su última modificación hace n*24 horas.
-name criterio : Busca por nombre de fichero según criterio, el cual admite caracteres comodín.
-newer file : El fichero fue modificado después que file.
-path criterio : Busca por ruta de fichero según criterio, el cual admite caracteres comodín
-size n : Busca por tamaño de fichero, el cual será n.
-type c : El fichero es del tipo c, el cual puede ser:
b bloque especial
c carácter especial.
d directorio
p nombre de tubería (FIFO)
f fichero normal.
l enlace simbólico
s socket
D puerta (enlace Solaris)
-uid n : EL UID (identificativo de usuario) del fichero es n
-used n : El fichero fue accedido por última vez n días después del cambio de su estado.
-used name : El nombre del usuario al que pertenece el fichero es name.

Veamos algunos ejemplos de su funcionamiento:

Ejemplo 1: Buscar todos los archivos que hay en el directorio actual y en sus subdirectorios con extensión .txt

find *. -name "*.txt"*

Ejemplo 2: Buscar los ficheros con permisos 777

find ** -perm 777*

ENLACES A FICHEROS.

Comando ln

Los enlaces nos van a permitir realizar copias de los ficheros con otro nombre, para poder acceder a ellos desde lugares distintos a su ubicación original, con un ahorro de espacio muy importante con respecto al comando ***cp***. Nuestro sistema identifica a los ficheros mediante un número denominado inodo, que les asigna en el momento de su creación.

Es decir, un directorio lo que contiene realmente es una lista de números de inodo con sus correspondientes nombres de fichero. De esta forma, cada nombre de fichero es un enlace a un inodo particular; por ello, cada inodo está asociado a un conjunto de información guardada en el disco, que puede tener asignados distintos nombres, y a la que podremos acceder desde distintos lugares del árbol de directorios si así lo deseamos.

En este sentido, podremos crear dos tipos distintos de enlaces a ficheros: *enlaces duros* y *enlaces simbólicos*.

El comando ***ln*** nos servirá para crear ambos tipos de enlaces.

Enlaces duros (hard links)

Si utilizamos el comando ***ln*** sin especificar ninguna opción, por defecto crearemos un enlace duro. La sintaxis es la siguiente:

ln *<fichero> [nombre del enlace]*
ln *<fichero1> <fichero2> ... <ficheroN> <nombre de directorio>*

Obviamente, el fichero o ficheros para los que deseamos crear un enlace duro deberán existir. Así mismo, si el último argumento es el nombre de un directorio que existe, crearemos un enlace duro a cada fichero, dentro del directorio, y con el mismo nombre de fichero.

Si solamente especificamos el fichero que queremos enlazar, y no indicamos ningún nombre para el enlace, éste se creará con el mismo nombre que el fichero a enlazar. Los cambios que realicemos en el fichero enlazado o en el enlace, se reflejarán en el resto, ya que todos tendrán el mismo número de inodo, y por lo tanto hacen referencia al mismo conjunto de información.

La ventaja de utilizar enlaces duros radica en que el comando ***rm*** únicamente borrará aquel fichero que le indiquemos. La información solamente se borrará por completo cuando borremos todos los enlaces a un inodo.

La desventaja con respecto a los enlaces simbólicos es que sólo permite crear enlaces dentro del mismo sistema de ficheros.

Los directorios . y .. son enlaces duros al directorio actual y a su directorio padre respectivamente.

Ejemplo:
1 – Creamos el fichero pruebaln con la orden ***cat***.

cat *> pruebaln*
hola
CTRL+ D

2 – Creamos un enlace a pruebaln que se llame penlace.

ln *pruebaln penlace*

3 – Veamos las características de estos ficheros con la orden ***ls***. Utilizamos la opción *–i* para ver el número de inodo. Ambos tendrán el mismo número de inodo con dos enlaces.

ls *–i pruebaln penlace*

4 – Modificamos pruebaln y comprobamos si también se modifica penlace.

cat *>>pruebaln*
adios
CTRL+ D
*$**cat** pruebaln*
*$**cat** penlace*

5 – Modificamos penlace y comprobamos si también se modifica pruebaln.

*$**cat** >>penlace*
otra vez hola
CTRL+ D
*$**cat** penlace*
*$**cat** pruebaln*

6 – Eliminamos pruebaln y comprobamos si penlace permanece y contiene la información correspondiente.

*$**rm** pruebaln*
*$**cat** penlace*

7 – Si utilizamos la orden ***ls*** *–i*, vemos que penlace sigue con el mismo número de inodo, que ahora solamente tendrá un enlace.

Nota : Un INODO es una estructura de datos, por así decirlo una tabla que contiene información sobre un fichero. Cada fichero se identifica por un número de inodo. Este número es único dentro de todo el sistema de ficheros. Dentro de cada inodo existe la siguiente información:
**** Número de inodo***
**** Tipo de fichero***
**** Propietario de dicho fichero***
**** Permisos del fichero***
**** Fecha de creación del mismo***
Puedes ver toda esta información si ejecutas un ls –l en cualquier directorio de tu sistema.

Enlaces simbólicos

Si utilizamos la opción *–s* con el comando ***ln***, es decir ***ln*** *–s*, crearemos un enlace simbólico. La sintaxis en este caso es la misma que utilizamos para crear enlaces duros. Podemos encontrar una similitud entre este tipo de enlaces y los accesos directos que estamos acostumbrados a crear con los sistemas Windows.

En el caso de los enlaces simbólicos, cada fichero tendrá un número de inodo distinto. Sin embargo, al igual que con los enlaces duros, todos los cambios que se realicen en uno de los ficheros se verán reflejados en el resto.

Si borramos el fichero enlazado, el enlace simbólico perderá toda la información, puesto que su inodo apunta a un número de inodo que ya no existe. Sin embargo, podremos crear enlaces simbólicos a ficheros de otros sistemas de archivos.

Ejemplo:
1 – Aún tenemos el fichero penlace. Creamos un enlace duro a penlace que se llame pruebaln.

ln *penlace pruebaln*

2 – Con la orden ls –li vemos que ambos tienen el mismo inodo, y que este inodo tiene dos enlaces.

ls *–li pruebaln penlace*

3 – Creamos un enlace simbólico a penlace que se llame penlacesim.

ln *–s penlace penlacesim*

4 – Con la orden ls –li vemos que tienen distinto número de inodo. Además, el inodo de penlacesim sólo tiene un enlace, y el inodo de penlace sigue teniendo dos. En la línea correspondiente a penlacesim vemos que aparece el fichero al que apunta, y la letra l (“ele”) al inicio.

ls *–li pruebaln penlace penlacesim*

5 – Cambiamos penlace y comprobamos si cambia penalcesim.

cat *>>penlace*

otra vez adios
CTRL+ D
*$**cat** penlacesim*

6 – Por último borramos penlace. Comprobamos que pruebaln permanece y que no podemos ver el contenido de penlacesim, el sistema nos dirá que no existe. Para que desaparezca totalmente tenemos que borrarlo.

***rm** penlace*
***cat** pruebaln*
***cat** penlacesim*
***rm** penlacesim*
***rm** pruebaln*

AGRUPACIÓN Y COMPRESIÓN DE FICHEROS

Comandos tar y gzip/gunzip

Tanto el comando ***tar*** como ***gzip*** son ampliamente empleados para la difusión de programas y ficheros en Linux.

tar : Este comando agrupa varios ficheros en uno solo o "archivo", mientras que el segundo los comprime. En conjunto estos dos programas actúan de forma muy similar a programas como Winzip. Su sintaxis es:

***tar** [opciones][ficheros]*

El modo en el que se escriben las opciones de tar es un poco especial. El guión inicial, por ejemplo, no es necesario. Las opciones más comunes para tar son:

-c creación de archivadores nuevos.
-x extracción de archivos de un archivador existente.
-v muestra los archivos mientras se agregan o se extraen.
-t muestra el contenido de un archivo tar.
-f el siguiente argumento es el archivador a crear, del que queremos extraer archivos o mostrar un listado.

Para crear un nuevo archivo se emplea:

***tar** -cvf nombre_archivo.tar fichero1 fichero2 ...*

donde fichero1, fichero2 etc. son los ficheros que se van a añadir al archivo tar. Si se desea extraer los ficheros se emplea:

***tar** -xpvf nombre_archivo.tar fichero1 ...*

Veamos algunos ejemplos:

***tar** cvf escritorio.tar Desktop*

empaqueta el contenido de Desktop en un archivador nuevo escritorio.tar

***tar** xvf escritorio.tar Desktop/Floppy.desktop*

extrae del archivo escritorio.tar el fichero indicado

***tar** xvf escritorio.tar*

extrae todo el contenido del archivo escritorio.tar

tar *tvf escritorio.tar*

muestra un listado largo del contenido del archivo escritorio.tar

Hay que tener en cuenta, a la hora de extraer el contenido de un archivador (al fichero tar resultante se le suele llamar así), si el archivador se creó conservando el nombre del directorio de origen. Es posible que se sobrescriba el contenido de los ficheros originales.

Ejemplo: Nos situamos en el directorio raíz como root. Si archivamos los ficheros /etc/group y /etc/passwd:

tar *cvf backup.tar /etc/group /etc/passwd*

estamos conservando los nombres del directorio al que pertenecen. Por lo tanto, para extraer estos ficheros nos tendremos que situar en el directorio raíz:

cd /
tar *xvf backup.tar /etc/group /etc/passwd*

Sin embargo, si archivamos los ficheros group y passwd estando en /etc:

tar *cvf /backup.tar group passwd*

no guardamos la ruta, por lo que para extraer los ficheros tendremos que situarnos en ella:

cd /
cd /etc
tar *xvf /backup.tar group passwd*

gzip/gunzip : Al contrario que tar que agrupa varios ficheros en uno, gzip comprime un único fichero con lo que la información se mantiene pero se reduce el tamaño del mismo. El uso de gzip es muy sencillo:

gzip *[opciones] fichero*

con lo que se comprime fichero (que es borrado) y se crea un fichero con nombre fichero.gz.
La opción más común es:

```
-1 a -9     grado de compresión, mínimo y máximo respectivamente.
-d          descomprimir el fichero .gz
```

Si lo que se desea es descomprimir un fichero se emplea entonces:

gzip *-d fichero.gz*

recuperando el fichero inicial.

Otra posibilidad sería utilizar el comando ***gunzip*** para la descompresión, de la siguiente forma:

gunzip *fichero.gz*

Como se ha comentado al principio es típico emplear ***tar*** y ***gzip*** de forma consecutiva, para obtener ficheros con extensión tar.gz o tgz que contienen varios ficheros de forma comprimida (similar a un fichero zip). El comando ***tar*** incluye la opción *z* para estos ficheros de forma que para extraer los ficheros que contiene:

tar *-zxf fichero.tar.gz*

CAMBIO DE MODO DE LOS FICHEROS

comandos chmod, chown y chgrp

Cada usuario es dueño de su directorio personal y será dueño también de los archivos que incluya en él.
Un usuario en Linux podrá configurar permisos en sus archivos. Por ello, distinguiremos por un lado tres categorías de usuarios, y por otro los tipos de permisos que cada uno de ellos puede tener sobre un archivo y/o directorio.

chown

Puede utilizar chown para cambiar el propietario al cual pertenece un archivo o directorio. Puede especificarse tanto el nombre de un usuario, así como un número de identidad de usuario (UID). De manera opcional, utilizando un signo de dos puntos (:) o bien un punto (.), permite especificar también un nombre de grupo.

Opciones:

- -R De manera descendente cambia el propietario de los directorios junto con todos sus contenidos. De manera opcional también permite cambiar el grupo al cual pertenecen.
- -v (o --verbose) Salida más descriptiva.
- --version Ver el número de versión del programa.
- --dereference Actúa sobre enlaces simbólicos en lugar de hacerlo sobre el destino.
- -h (o --no-dereference) En el caso de enlaces simbólicos, cambia el propietario del destino en lugar del propio enlace.
- --reference Cambia el el propietario de un archivo, tomando como referencia el propietario de otro.

chgrp

Puede utilizar chgrp para cambiar el grupo al cual pertenece un archivo o directorio. Puede especificarse tanto el nombre de un grupo, así como un número de identidad de grupo (GID).

Opciones:

- -R De manera descendente cambia el grupo al cual pertenecen los directorios junto con todos sus contenidos.
- -v (o --verbose) Salida de chgrp más descriptiva.
- --version Ver el número de versión del programa.
- --dereference Actúa sobre enlaces simbólicos en lugar de hacerlo sobre el destino.
- -h (o --no-dereference) En el caso de enlaces simbólicos cambia el grupo del destino en lugar del propio enlace.
- --reference Cambia el grupo de un archivo tomando como referencia el propietario de otro.

Ejemplos:

- Ejecute lo siguiente para realizar el cambio de propietario a fulano sobre el archivo algo.txt.

  chown fulano algo.txt

- Ejecute lo siguiente para realizar el cambio de propietario a fulano y el

grupo desarrollo sobre el archivo algo.txt.

chown fulano:desarrollo algo.txt

- Ejecute lo siguiente para realizar el cambio de propietario a fulano y el grupo mail del sub-directorio Mail junto con todo su contenido.

  chown -R fulano:mail Mail

- Ejecute lo siguiente para realizar el cambio de grupo a desarrollo sobre el archivo algo.txt.

  chgrp desarrollo algo.txt

Categorías de usuarios

• Dueño del archivo (u).
• Grupo dueño (g), formado por todos los usuarios que son miembros de un grupo asociado al archivo.
• Resto de usuarios (o), todos los usuarios que no son ni el dueño ni miembros del grupo dueño.

Tipos de permisos

- Lectura (r de Read, leer): para un archivo permite leer su contenido, para un directorio permite que se muestren los archivos que contiene.
- Escritura (w de Write, escribir): para un archivo permite que se modifique su contenido, para un directorio permite agregar y quitar archivos.
- Ejecución (x de eXecute, ejecutar): para un archivo permite su ejecución, para un directorio permite que el usuario lo recorra (que entre y pase por él) – si no tiene permiso de lectura, aunque pueda entrar no podrá ver el contenido.
- Cuando ejecutamos el comando ls -l, podemos ver la configuración de permisos de los archivos:
  - El primer carácter indica el tipo de archivo: “d” si es directorio, “-“ si es un archivo regular, “l” si es un enlace simbólico.
  - Los siguientes nueve caracteres indican los permisos para el dueño, el grupo dueño y otros (rwxrwxrwx); si aparece un guión, indica que el permiso correspondiente no está habilitado.
  - El siguiente número indica el número de vínculos.
  - Nombre del dueño y nombre del grupo dueño.
  - Tamaño en bytes.
  - Fecha de la última modificación.
  - Nombre del archivo.

chmod

Para cambiar los permisos de un archivo o directorio del servidor, tienes que utilizar el comando ***chmod***.

Para ver la ayuda, digitamos:

chmod *--help*

Este comando en concreto tiene varias sintaxis permitidas. De entre ellas por ejemplo puedes utilizar:

chmod *[opciones] modo-en-octal fichero*

Las opciones podemos indicarlas o no, según queramos. Opciones típicas son:

* -R para que mire también en los subdirectorios de la ruta.
* - v para que muestre cada fichero procesado
* - c es como -v, pero sólo avisa de los ficheros que modifica sus permisos

El modo en octal es un número en base 8 (octal) que especifique el permiso. Los números en octal se especifican empezando el número por un 0. Por ejemplo, 0777 es indica todos los permisos posibles para todos los tipos de usuario. 0666 indica que se dan permisos de lectura y escritura, pero no de ejecución. 0766 indica que se dan permisos de lectura y escritura, pero sólo tienen permiso de ejecución para los usuarios que son dueños del archivo. 0755 indica permisos para lectura y ejecución, pero escritura sólo para el usuario que es dueño del archivo.

Por ejemplo:

chmod *0777 archivo.txt*

Asigna todos los permisos al archivo archivo.txt

chmod *0666 **

Asigna permisos de lectura y escritura, no de ejecución a todos los archivos y directorios del directorio donde ejecutamos el comando.

chmod *-R 0644 **

Esto da permisos a todos los archivos y directorios del directorio donde se invoca el comando y de todos los directorios que cuelgan de él. Los permisos asignados son de lectura a todos los usuarios, de escritura sólo al dueño del archivo y de ejecución a nadie.

Otro modo de trabajo con chmod

Luego también se pueden asignar permisos de otra manera, utilizando otra posible sintaxis de chmod, que tal vez resulte más útil si no queremos tratar con los valores en octal.

chmod *[opciones] modo[,modo]… fichero*

Para ello tenemos que tener claros los distintos grupos de usuarios:

* u: usuario dueño del fichero
* g: grupo de usuarios del dueño del fichero
* o: todos los otros usuarios
* a: todos los tipos de usuario (dueño, grupo y otros)

También hay que saber la letra que abrevia cada tipo de permiso:

* r: se refiere a los permisos de lectura
* w: se refiere a los permisos de escritura
* x: se refiere a los permisos de ejecución

Ejemplos

chmod *o=rwx **

Asigna permisos de lectura, escritura y ejecución para los usuarios "otros" a todos los archivos de la carpeta

chmod *a=rwx fichero.txt*

Asigna todos los permisos a todos los usuarios para el archivo fichero.txt

chmod *go= **

Quita todos los permisos para los usuario del grupo y los usuarios otros.

chmod *u=rwx,g=rw,o= **

Da todos los permisos al dueño del fichero, a los del grupo del dueño le asigna permisos de lectura y escritura y a los otros usuarios les quita todos los permisos.

Nota:un espacio después de la coma "," en los distintos modos de permisos que se indiquen hace fallar el comando.

chmod *a=r **

Da permisos únicamente de lectura a todos los tipos de usuario.

De un modo parecido a lo que acabamos de ver, también se pueden añadir o quitar permisos con los operadores + y −. Para ello se indica el tipo de usuario y el permiso que se resta o añade. Algo como esto:

chmod *a-wrx **

Esto quita todos los permisos a todos los tipos de usuario.

chmod *a+r,gu+w **

Este comando asigna permisos de lectura a todos los usuarios y permisos de escritura al dueño del archivo y el grupo del dueño.

chmod *u=w,a+r **

Este comando asigna permisos de escritura al usuario dueño y a todos los usuarios les añade permiso de lectura.

GESTIÓN DE USUARIOS Y GRUPOS

Introducción

Ya sabemos que Linux es un sistema multiusuario y por lo tanto distingue diferentes usuarios. Cada usuario recibe una cuenta que incluirá toda la información necesaria (nombre de usuario, directorio inicial, etc.). Además de las cuentas dadas a personas, existen cuentas especiales definidas por el sistema que tienen privilegios especiales. La más importante es la cuenta raíz (administrador), con el nombre de usuario root.

Normalmente, los usuarios normales están restringidos, de forma que los permisos de los ficheros en el sistema están preparados para que no puedan borrar o modificar ficheros en directorios compartidos por todos los usuarios. Estas restricciones desaparecen para root. El usuario root puede leer, modificar o borrar cualquier fichero en el sistema, cambiar permisos y pertenencias, etc. Por lo tanto, podemos deducir que la gestión de los usuarios solamente puede realizarla el usuario root.

Conceptos de gestión de usuarios

La información que el sistema mantiene acerca de cada usuario es la siguiente:

- Nombre de usuario: es un identificador único dado a cada usuario del sistema. Es la cadena de caracteres con la que el usuario se identifica al entrar en el sistema. Se pueden utilizar letras, dígitos y los caracteres _ (guión bajo) y . (punto). Ejemplo: simmd.
- User ID o UID: es un número único dado a cada usuario del sistema. Su número debe ser mayor que el del último usuario creado en el sistema.
- Group ID o GID: número identifica el grupo al que pertenece el usuario. El número ha de ser el mismo para todos los usuarios que formen el grupo. Cada usuario puede pertenecer a uno o más grupos definidos por el administrador del sistema. Aunque la importancia real de las relaciones de grupo es la relativa a los permisos de ficheros.
- Clave: el sistema almacena la contraseña del usuario encriptada. El comando passwd nos permitirá asignar y cambiar las claves de los usuarios.
- Nombre completo: puede ser el nombre real del usuario, su número de teléfono, su dirección, etc. Es decir, guarda información real sobre el sistema.
- Directorio inicial: es el directorio al que accede el usuario al entrar en el sistema. Cada usuario debe tener su propio directorio inicial, normalmente situado bajo /home. En principio será el único directorio en el que el usuario podrá guardar su información personal, programas, etc. Ejemplo: /home/simmd.
- Intérprete de inicio: es el intérprete de comandos que arranca para el usuario cuando se conecta al sistema. Ejemplos: /bin/bash, /bin/tcsh.

El fichero que contiene toda esta información relativa a los usuarios es el fichero /etc/passwd. Este fichero contiene una línea por cada usuario del sistema, y su estructura es la siguiente:

nombre:clave encriptada:UID:GID:nombre completo:directorio de inicio:intérprete

Ejemplo:

simmd:x:501:501:simmd:/home/simmd:/bin/bash

En el caso de los grupos, la información sobre ellos se encuentra en el fichero /etc/group. Hay varios grupos definidos en el sistema (root, bin, sys, mail, etc) que se utilizan para permisos de ficheros del sistema. Los usuarios no deben pertenecer a ninguno de estos grupos. El formato de cada línea del fichero /etc/group es el siguiente:

nombre del grupo:clave:GID:otros miembros

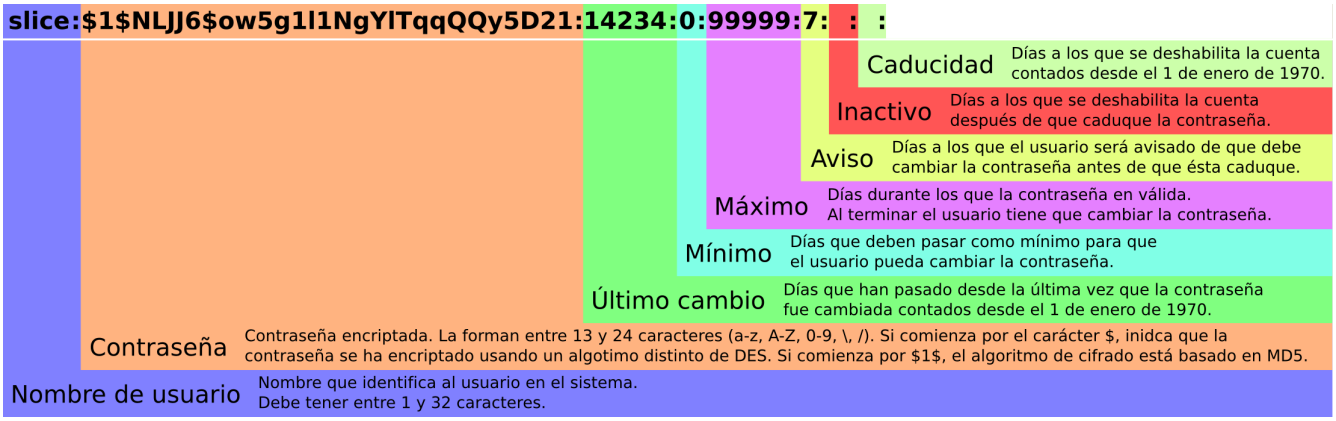
La clave del grupo no suele utilizarse.

En /etc/passwd cada usuario tiene un GID. Sin embargo, como los usuarios pueden pertenecer a otros grupos, podemos añadir su nombre de usuario en el campo otros miembros (separados unos usuarios de otros por comas) de todos aquellos grupos no definidos por el sistema a los que queremos que pertenezca. Podemos conocer a qué grupos pertenece un usuario utilizando la orden groups. (El grupo con GID 100 suele ser el grupo users).

En /etc/shadow se encuentran las claves encriptadas. El archivo sólo lo puede ver y editar el *root* por lo que para ver su contenido tenemos que ejecutar el siguiente comando:

sudo *pico /etc/shadow*

Cada línea de este archivo está formada por columnas, separadas por ':', con información sobre la contraseña de un único usuario.



Tipos de cuentas

Usuario root

- También llamado superusuario o administrador.
- Su UID (User ID) es 0 (cero).
- Es la única cuenta de usuario con privilegios sobre todo el sistema.
- Acceso total a todos los archivos y directorios con independencia de propietarios y permisos.
- Controla la administración de cuentas de usuarios.
- Ejecuta tareas de mantenimiento del sistema.
- Puede detener el sistema.
- Instala software en el sistema.
- Puede modificar o reconfigurar el kernel, controladores, etc.

Usuarios especiales

- Ejemplos: bin, daemon, adm, lp, sync, shutdown, mail, operator, squid, apache, etc.
- Se les llama también cuentas del sistema.
- No tiene todos los privilegios del usuario root, pero dependiendo de la cuenta asumen distintos privilegios de root.
- Lo anterior para proteger al sistema de posibles formas de vulnerar la seguridad.
- No tienen contraseñas pues son cuentas que no están diseñadas para iniciar sesiones con ellas.
- También se les conoce como cuentas de "no inicio de sesión" (nologin).
- Se crean (generalmente) automáticamente al momento de la instalación de Linux o de la aplicación.
- Generalmente se les asigna un UID entre 1 y 100

Usuarios normales

- Se usan para usuarios individuales.
- Cada usuario dispone de un directorio de trabajo, ubicado generalmente en /home.
- Cada usuario puede personalizar su entorno de trabajo.

- Tienen solo privilegios completos en su directorio de trabajo o su HOME.
- Por seguridad, es siempre mejor trabajar como un usuario normal en vez del usuario root, y cuando se requiera hacer uso de comandos solo de root, utilizar el comando *su*.
- En las distribuciones actuales de Linux se les asigna generalmente un UID superior a 1000.

Añadir nuevos usuarios y borrar usuarios

Podemos añadir usuarios al sistema de varias formas. La más engorrosa de todas es hacerlo a mano. Es decir, añadimos al fichero /etc/passwd la línea correspondiente al usuario.
Seguidamente le asignamos una clave con la orden passwd y finalmente establecemos el dueño, grupo dueño y permisos para el directorio /home/usuario que deberemos crear. Además, debemos actualizar /etc/shadow con pwconv.

Sin embargo, existen varias utilidades que nos permiten crear usuarios de una forma mucho más cómoda e intuitiva:

- Servidor Xwindow. Entorno KDE

  Iniciamos una sesión como root en el entorno gráfico. Seleccionamos: Menú K > Configuración > Usuarios > Crear usuarios (variará según la distribución)

  Aparecerá un cuadro de diálogo con los usuarios normales que hay creados en el sistema y toda la información correspondiente a cada uno de ellos. Desde aquí podremos añadir, borrar y modificar usuarios y grupos. Una vez realizados los cambios oportunos debemos guardarlos, como si de un documento se tratase.

- Servidor Xwindow. Entorno Gnome

  Iniciamos una sesión en el entorno gráfico. Seleccionamos: Sistema > Administración > Usuarios y Grupos > Añadir o eliminar (variará según la distribución)

  Aparecerá un cuadro de diálogo con los usuarios normales que hay creados en el sistema y toda la información correspondiente a cada uno de ellos. Desde aquí podremos añadir, borrar y modificar usuarios y grupos. Una vez realizados los cambios oportunos debemos guardarlos, como si de un documento se tratase.

- Useradd/Adduser

  Las órdenes useradd y adduser también nos permiten añadir nuevas cuentas de usuario desde el modo consola. Podemos crear un usuario con las características por defecto:

  useradd *usuario*

  Habremos creado un usuario sin contraseña. Para habilitar su cuenta le asignamos una contraseña asi :

  passwd *usuario*

  Si queremos crear un usuario a nuestra medida utilizaremos la siguiente sintaxis: ***useradd*** *-r <nombre usuario> -p <clave> -u <UID> -g <GID> -c <información> -d <direct. Inicial> -s <shell>*

  Lo único que nos quedará por hacer será crear el directorio /home del

usuario y establecer los permisos pertinentes.

Para borrar un usuario desde el modo consola:

userdel *[-r] usuario*

Si utilizamos la opción -r también eliminaremos el directorio home del usuario o directorio inicial.

En algunas distribuciones, es mas sencillo este proceso, solo digitamos:

adduser *usuario*

y seguimos las instrucciones.

Nota : Una forma de deshabilitar una cuenta de usuario sin tener que borrarla es escribir ! en el campo clave del usuario en el fichero /etc/shadow o /etc/passwd.

Otras órdenes para la gestión de usuarios y grupos

chfn : permite cambiar el nombre completo del usuario:

chfn *-f <nuevo nombre> <usuario>*

groups : muestra todos los grupos a los que pertenece el usuario.
groupadd : permite añadir un nuevo grupo. Sintaxis:

groupadd *[-g GID] [-f] <nombre del grupo>*

-f : obliga al sistema a informar si se producen errores (por ejemplo cuando el grupo que queremos crear ya existe).

Si no especificamos un GID, el sistema asigna el menor GID que corresponde a este grupo.

groupdel : borra el grupo cuyo nombre indiquemos junto a la orden.
groupmod : permite modificar el GID y el nombre del grupo. Sintaxis:

groupmod *[-g <nuevo GID>] [-n <nuevo nombre>] <nombre actual>*

id : muestra UID y GID del usuario y los grupos a los que pertenece el usuario conectado al
sistema. Sintaxis:

id
id *usuario*

REDIRECCIONAMIENTO Y TUBERÍAS

Introducción

Muchos comandos de Unix toman su entrada de la ENTRADA ESTÁNDAR (stdin) y envían su salida a la SALIDA ESTÁNDAR (stdout). El intérprete de comandos configura el sistema de forma que la stdin es el teclado y la stdout la pantalla.

Veamos algunos ejemplos que ilustren esta cuestión:

Ejemplo 1: Si al comando cat no le pasamos argumentos, actuará mostrando en

pantalla todo lo que hayamos tecleado antes de un Intro. Para indicarle al sistema que queremos finalizar la ejecución de cat, pulsamos la combinación de teclas CTRL+ D.

```
$cat
      hola        lo que recibe de la stdin
      hola        lo que devuelve a la stdout
      CTRL+ D     fin de cat
$
```

Ejemplo 2: El comando sort actúa de forma parecida. Si lo ejecutamos e introducimos un conjunto de líneas desde la stdin, cuando pulsemos la combinación CTRL+ D devolverá a la stdout las mismas líneas pero de forma ordenada.

```
$sort
      méndez
      luque
      rodríguez
CTRL+ D           fin de entradas
      luque
      méndez
      rodríguez
$
```

Redireccionamiento de la entrada y la salida

Podemos utilizar los caracteres > y < para redireccionar la salida y la entrada estándar respectivamente.
Para redireccionar la stdout podemos utilizar:

a) Redirección destructiva: crea un nuevo fichero, o sobrescribe el contenido de uno que ya existe, con aquella información que el comando recibe de la stdin.

   $comando >fichero

b) Redirección no destructiva: crea un nuevo fichero o añade al final del contenido de uno que ya existe la información que el comando recibe de la stdin.

   $comando >>fichero

Veamos algunos ejemplos:

Ejemplo 1:

```
$cat >pruebacat
      hola
      adios
      CTRL+ D
      $cat pruebacat
      hola
      adios
$cat >>pruebacat
      otra vez hola
      otra vez adios
      CTRL+ D
$cat pruebacat
      hola
      adios
      otra vez hola
```

Ejemplo 3

```
$cat >pruebasort
      3
      2
      1
      CTRL+ D
$cat pruebasort
      3
      2
      1
$sort pruebasort >pruebacat
$cat pruebacat
      1
      2
      3
$
```

otra vez adios *$*	
Ejemplo 2: *$sort >pruebasort* *3* *2* *1* *CTRL+ D* *$cat pruebasort* *1* *2* *3* *$sort >>pruebasort* *6* *5* *2* *CTRL+ D* *$cat pruebasort* *1* *2* *3* *2* *5* *6* *$*	Ejemplo 4 (redirección de stdin) *$cat >pruebasort* *3* *2* *1* *CTRL+ D* *$sort <pruebasort* *1* *2* *3* *$cat pruebasort* *3* *2* *1* *$*

Para redireccionar la stdin utilizamos el carácter <. Con cat no tiene mucho sentido, ya que el resultado que se obtiene es igual al que produce sin utilizar este redireccionamiento. Con sort, mostrará en pantalla el contenido del fichero que le indiquemos de forma ordenada. (Ver el ejemplo 4 anterior).

Tuberías (pipes)

Hemos visto que el comando sort, aunque simple, actúa como filtro, devolviendo a la salida lo que recibe desde la entrada de forma ordenada. Las tuberías pueden ser utilizadas para combinar comandos, de forma que la salida del primero es enviada a la entrada del segundo y así sucesivamente. De esta forma, podemos aplicar un filtro a la stdout del comando ls enviándola a la stdin de sort. Lo que conseguimos es conectar una cadena de comandos en una tubería.

Para crear las tuberías utilizamos el carácter | (barra vertical, carácter de canalización).

Ejemplos:

ls /usr/bin | more	muestra el contenido de /usr/bin por pantallas.
ls |sort -r	muestra un listado del directorio actual ordenado alfabéticamente de mayor a menor.
ls |sort -r |head -1	veremos el primer fichero de un listado del directorio actual ordenado alfabéticamente de mayor a menor.

LA EDICIÓN DE TEXTO.

El editor vi

Introducción

En Linux existen muchos editores de texto disponibles (vi, Emacs, joe), sin embargo será el visual editor (vi) el único que encontraremos en cualquier sistema Unix. vi fue el primer editor de pantalla completa que existió y, aunque no es fácil de usar, es una herramienta extremadamente potente.

Para comenzar con vi y editar un fichero de texto emplearemos la sintaxis:

vi *<nombre de fichero>*

En la pantalla, de 24 líneas, aparecerá una columna de "~" que indican el final del fichero. En la parte inferior veremos el nombre del nuevo fichero. En un principio no podremos insertar texto, ya que vi arranca en el modo órdenes, uno de los tres posibles modos de operación: modo órdenes, modo inserción, modo última línea.

1. En modo órdenes o modo comandos no podremos insertar texto. Nos permitirá usar ciertas órdenes de edición de ficheros o cambiar a otros modos.
2. Al modo de inserción, que nos permitirá escribir y desplazarnos por el archivo, se accede desde el modo comando por ejemplo con la orden i. Para volver al modo comando pulsamos la tecla Esc.
3. El modo última línea, o modo ex, proporciona ciertas órdenes extendidas a vi, como por ejemplo salir de vi guardando o sin guardar los cambios realizados en el archivo (:wq :q!). Para acceder a este modo, tecleamos : desde el modo comando. Para salir de él ejecutamos una orden o borramos todo, incluidos los dos puntos.

Insertar texto

Si estamos en modo órdenes podemos pasar al modo de inserción de varias formas:

- Tecla i: para insertar texto desde la posición en la que se encuentra el cursor.
- Tecla a: para insertar texto comenzando detrás de la posición actual del cursor.
- Tecla A: para insertar texto comenzando al final de la línea actual.
- Tecla I (i mayúscula): para insertar texto comenzando al inicio de la línea actual.
- Tecla o: para insertar texto debajo de la línea actual.
- Tecla O: para insertar texto por encima de la línea actual.

En la parte inferior de la pantalla aparecerá la cadena –INSERT—indicándonos que estamos en el modo de inserción. Podremos borrar y suprimir texto, además de movernos por el archivo con las flechas del cursor.

Borrar texto

Además de las teclas de retroceso y suprimir, podemos utilizar otras órdenes para borrar desde el modo comando:

- Tecla x: borra el carácter en el que se encuentra situado el cursor.
- Tecla X: borra el carácter que está a la izquierda del cursor.
- Teclas dd: borra la línea en la que se encuentra el cursor.
- Tecla dw: borra la palabra en la que se encuentra el cursor.
- Tecla o: para insertar texto debajo de la línea actual.

- Tecla O: para insertar texto por encima de la línea actual.

Modificar texto

Desde el modo comando podemos reemplazar o sustituir parte del texto:

- Tecla r: permite sustituir el carácter en el que se encuentra el cursor.
- Tecla R: en la parte inferior de la pantalla aparecerá la cadena –REPLACE--, que nos indica que podemos reemplazar el texto hasta que pulsemos la tecla Esc. Es decir, R es similar al modo de inserción, con la diferencia de que en lugar de insertar texto lo sobrescribe.
- Teclas :r <fichero>: inserta en el fichero que estamos editando el contenido del fichero que indicamos.
- Tecla ~: cambia de mayúsculas a minúsculas, o viceversa, el carácter en el que se encuentra el cursor (F10 cambia uno, F11 cambia tres, F12 cambia cuatro).

Órdenes de desplazamiento

Además de las flechas del cursor, podemos movernos por el documento desde el modo comando utilizando una serie de órdenes:

- Tecla h: un carácter a la izquierda.
- Tecla j: un carácter abajo.
- Tecla k: un carácter arriba.
- Tecla l (ele minúscula): un carácter a la derecha.
- Tecla e: al final de la palabra actual.
- Tecla b: al inicio de la palabra actual.
- Tecla w: al inicio de la palabra siguiente.
- Tecla 0 (cero): al inicio de la línea actual.
- Tecla $: al final de la línea actual.
- /<cadena>: desplaza el cursor hacia delante hasta que encuentra el texto cadena.
- ?<cadena>: desplaza el cursor hacia atrás hasta que encuentra el texto cadena.
- Tecla H: va al comienzo del archivo.
- Tecla G: va al final del archivo.
- CTRL+ f: avanza una pantalla.
- CTRL+ b: va una pantalla hacia atrás.

Cada uno de los comandos de movimiento puede estar precedido por un número, de forma que tenemos la posibilidad de movernos a una palabra, línea o carácter arbitrarios. Además, podemos asociar órdenes de desplazamiento con otras órdenes como por ejemplo borrar.
Ejemplos:

- 10G : va a la línea 10 del fichero.
- dG : borrará todo, desde la posición del cursor hasta el final del fichero.
- d$: borrará todo desde la posición del cursor hasta el final de la línea.
- 3e : moverá el cursor tres palabras hacia delante.
- d3b : borrará tres palabras hacia atrás.
- d/<cadena> : borra todo desde la posición del cursor hasta que encuentra el texto cadena.
- d0 : borra todo hasta el inicio de la línea actual.

Cortar, copiar y pegar

Utilizaremos las órdenes y (Yank) y d (Delete) para copiar y cortar texto

respectivamente.
Combinaremos estas dos órdenes con las de desplazamiento para copiar o cortar conjuntos de caracteres, líneas, palabras. Para pegar el texto que hemos copiado o cortado utilizaremos las órdenes p (para insertar el texto después del cursor) y P (para insertar el texto antes del cursor).

Ejemplos:
- y?<cadena> : copiará todo desde la posición del cursor hacia atrás, hasta que encuentre el texto cadena.
- d15l : cortará 15 caracteres desde la posición del cursor hacia la derecha.
- y$: copiará todo desde la posición del cursor hasta el final del párrafo actual.

Guardar y salir

- Para salir sin guardar los cambios escribimos :q!.
- Para salir guardando los cambios escribimos :wq o ZZ o :x.
- Para guardar los cambios sin salir escribimos :w.

Editar otros ficheros

Si estamos editando un fichero con vi, podemos editar otro escribiendo :e <fichero> desde el modo comando. Para poder utilizar esta orden tendremos que indicarle a vi si queremos guardar o no los cambios del primer fichero; es decir, utilizaremos :w y luego :e, o bien :e! Directamente si no queremos guardar los cambios. Dejaremos de editar el primero y pasaremos al segundo.

Ejecutar comandos del intérprete

Podemos insertar, en el fichero que estamos editando, la salida de un comando. Para ello utilizamos la orden :r! y a continuación el comando que queremos ejecutar. Por ejemplo,

:r! ls -i

inserta un listado del directorio actual con números de inodo al final del párrafo actual.
También podemos ejecutar una orden desde vi y volver al editor una vez que ésta finalice. Utilizaremos la orden :!. Por ejemplo,

:! ls -i

mostrará en pantalla el mismo listado que en el ejemplo anterior, aunque en este caso los resultados no se insertarán en el fichero.
Incluso podemos dejar temporalmente vi e iniciar el intérprete de comandos para ejecutar otras órdenes. Para salir del intérprete y regresar a vi utilizamos la orden exit. Para iniciar el intérprete usamos la orden :shell. Por ejemplo, es posible que queramos consultar la página de manual de vi y guardarla en un fichero.

El editor PICO

Pico – sencillo editor de texto en el estilo de Pine Composer

Sintaxis

pico *[opciones] [fichero]*

Descripción

Pico es un sencillo editor de texto basado en el editor del sistema de mensajes Pine. Al igual que en Pine, los comandos son desplegados en la parte inferior de la pantalla, y se proporciona ayuda sensible al contexto. Conforme los caracteres son tecleados se insertan inmediatamente en el texto.

Los comandos de edición se introducen empleando combinaciones con la tecla control. Como una solución para los programas de comunicación que toman ciertos caracteres de control, se puede simular la tecla control presionando ESCAPE dos veces, seguida del carácter de control deseado, por ejemplo "ESC ESC c" sería el equivalente a teclear ctrl-c. El editor cuenta con cinco características básicas: justificación de párrafos, búsqueda, cortar/pegar por bloque, un corrector ortográfico, y un navegador de ficheros.

La justificación de párrafo (o llenado) se efectúa en el párrafo que contiene al cursor en ese momento, o, si el cursor se encuentra entre líneas, en el párrafo inmediatamente inferior. Los párrafos se delimitan con líneas en blanco, o con líneas que comiencen con un espacio en blanco o un tabulador. Se puede eliminar la justificación inmediatamente después de haber sido efectuada empleando la combinación de teclas control-U.

Las búsquedas de cadenas no diferencian entre mayúsculas y minúsculas. Una búsqueda comienza en la posición actual del cursor y abarca hasta el final del texto. La búsqueda mas reciente se ofrece como valor por defecto en las búsquedas subsiguientes.

Los bloques de texto pueden ser movidos, copiados o borrados con el adecuado uso de los comandos para marcar (ctrl-^), borrar (ctrl-k) y restaurar (ctrl-u). El comando borrar elimina el texto entre la "marca" y la posición actual del cursor, y lo coloca en el buffer "cortar". El comando restaurar efectúa un "pegado" en la posición actual del cursor.

El corrector ortográfico examina todas las palabras en el texto. Este presenta cada palabra incorrectamente escrita para su corrección al tiempo que la resalta en el texto. La corrección ortográfica puede ser cancelada en cualquier momento. Alternativamente, *pico* puede cambiar a la rutina de corrección por defecto, una rutina definida por la variable de ambiente SPELL. La rutina de reemplazo debe leer la entrada estándar y escribir en la salida estándar.

El navegador de ficheros se ofrece como una opción en los prompts de los comandos "Read File" y "Write Out". Su propósito es ayudar en la búsqueda de ficheros específicos y navegar en la jerarquía de directorios. Los nombres de ficheros con sus tamaños y nombres de directorios en el directorio de trabajo actual se presentan para su selección. El directorio actual de trabajo se presenta en la línea superior de la pantalla mientras la lista de los comandos disponibles ocupa las dos últimas. Se da soporte a varias funciones básicas de manipulación de ficheros: renombrado de ficheros, copiado, y borrado.

Ayuda más específica está disponible en el sistema de ayuda en línea de *pico*.

Opciones

+*n*
Provoca que *pico* inicie con el cursor colocado *n* líneas dentro del fichero. (Nota: no dejar espacio entre el signo "+" y el número)

-b
Habilita la opción de reemplazar las ocurrencias de un texto utilizando el comando "Where is"

-d
Reasigna la tecla "borrar" de manera que el carácter sobre el que está el cursor se elimina en lugar del carácter que esté a su izquierda.

-e
Habilita el completamiento de nombre de fichero.

-f
Utilizar teclas de función para los comandos. Esta opción es soportada únicamente en conjunción en el UW Enhanced NCSA telnet.

-h
Lista las opciones válidas de la línea de comandos.

-j
Habilita el comando "Goto" en el navegador de ficheros. Esto permite al comando indicar explícitamente a *pilot* que directorio visitar.

-g
Habilitar el modo "Show Cursor" en el navegador de ficheros. Provoca que el cursor sea colocado antes de la selección actual en lugar de ser colocado en la esquina inferior izquierda del despliegue.

-k
Provoca que el comando "Cut Text" elimine los caracteres desde la posición del cursor hasta el final de la línea en lugar de eliminar la línea completa.

-m
Habilita la funcionalidad del ratón. Esto solo funciona cuando *pico* se ejecuta desde una venta xterm del sistema X Window.

-n*n*
La opción -n*n* habilita la notificación de nuevo correo. El argumento *n* es opcional y especifica cada cuanto, en segundos, se verifica el buzón en busca de nuevo correo. Por ejemplo, -n60 hace que *pico* busque nuevos mensajes de correo cada minuto. El intervalo por defecto es 180 segundos, mientras que el mínimo permitido es 30. (Nota: no se debe dejar espacio entre "n" y el numero)

-o *dir*
Establece el directorio de operación. Únicamente los ficheros dentro de este directorio son accesibles. De igual manera, el navegador de ficheros se limita al sub-árbol de directorios del directorio especificado.

-r*n*
Establece la columna usada para limitar el margen derecho del comando "Justify"

-s *corrector*
Especificar un programa *corrector* alterno para usar cuando se verifique la ortografía.

-t
Habilitar el modo "herramienta". Útil para cuando *pico* se utiliza como editor dentro de otras herramientas (ejm. Elm, Pnews). *Pico* no confirmará para salir, y no renombrará el buffer con el comando "Write Out".

-v
Ver el fichero únicamente, inhabilita las funciones de edición.

-w
Inhabilita el acomodamiento de palabras (lo que permite editar líneas de gran longitud)

-x
Inhabilitar el menú de teclas en la parte inferior de la pantalla.

-z
Habilitar la suspensión de *pico* con ^Z

-q
Las definiciones termcap o terminfo para la introducción de secuencias de escape se utiliza en preferencia de las secuencias definidas por defecto. Esta opción solo está disponible si *pico* se compiló con TERMCAP_WINS habilitado.

Por último, cuando una instancia de *pico* está corriendo y es desconectada (es decir, recibe una SIGHUP), *pico* guardará el trabajo actual, si es necesario, antes de salir. El trabajo se guarda con el nombre de fichero actual con un ".save" añadido al final. Si el trabajo actual no tiene nombre, se guarda como "pico.save".

Eliminando Paquetes por Consola en Ubuntu:

Una de las maneras que tenemos en Ubuntu de desinstalar paquetes es mediante el comando **apt-get remove**. Lo que hace este comando es desinstalar el software que ya no usemos pero nos deja los archivos de configuración y los archivos que hayamos descargado para su instalación. Entonces vamos a hacer una desinstalación en tres pasos:

1- **sudo apt-get remove** "nombre-del-paquete"
2- **sudo apt-get purge** "nombre-del-paquete"
3- **sudo apt-get clean** "nombre-del-paquete"

Lo que estamos haciendo es lo siguiente, en el paso (1) uno le pedimos que nos desinstale el paquete elegido, en el paso (2) dos que borre los archivos de configuración de ese paquete y por último, en el paso (3) tres, que elimine del disco los archivos descargados para la instalación (si pensamos volver a instalar este paquete y nos sobra espacio en el disco no haría falta hacer este último paso, pero si queremos mantener nuestro Ubuntu limpio, sería recomendable.)

Buscar paquetes desde consola:

Muchas veces necesitamos buscar paquetes, o el nombre del paquete que queremos instalar. Para esto tenemos esta instrucción la cual nos ayudara a encontrar la respuesta:

```
apt-cache search vim | more
```

otra forma, mas puntual:

```
apt-cache search vim | grep vim-
```

Fuente:

http://www.adslfaqs.com.ar/crear-alias-en-bash-terminal-para-linux-ubuntu-debian/
http://espanol-linux-man-pages.coding-school.com/man/X_pico-1