1. 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.

- En cualquier GNU/Linux tenemos la llamada terminal o consola que abre un shell o intérprete de comandos. En Ubuntu la abrimos buscando en en el Dash o tablero de Unity: "**Terminal**" o pulsando la combinación de teclas `Ctrl`+`Alt`+`T`
- También se puede pasar al **modo texto** (intérprete de comandos) desde el modo gráfico pulsando:`Ctrl`+`Alt`+`F1` o bien con: `F2 F3 F4 F5 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).

Enlaces de interés:
GNU Emacs, Manuales Online
Una introducción rápida a *GNU Emacs*

2. NOCIONES BÁSICAS

En una terminal:

- Las aplicaciones con nombres compuestos se escriben con guión entre las palabras (ej. compizconfig-settings-manager).
- Para los nombres de archivos y directorios que contienen espacios en blanco hay que envolverlos en comillas dobles (ej. "nombre archivo") o simples (ej. 'nombre archivo').

Un consejo: Para no haceros un lío, nunca uséis espacios en blanco en los nombres de carpetas y archivos y sustituirlo por un guión bajo (mis_imágenes) o un guión medio (mis-imágenes)

- Los espacios en blanco se utilizan únicamente para separar ordenes (ej. para instalar varios paquetes: `sudo apt-get install avidemux k3b kde-i18n-es k3b-i18n`, vemos que dichos paquetes están separados por espacios en blanco entre ellos).
- La ruta "/home/tu_usuario" se puede cambiar por el símbolo "**~**" (para escribirlo, pulsar la combinación de teclas `Alt Gr`+`Ñ`), que viene a sustituirlo en la línea de ordenes, sea cual sea el nombre del usuario

Cuando tecleamos una orden, el intérprete de comandos sigue una serie de pasos:
1. Busca el nombre de la orden y comprueba si es una orden interna.
2. Comprueba si la orden es un alias, es decir, un nombre sustitutorio de otra orden.
3. Si no se cumple ninguno de los casos anteriores, busca el programa correspondiente y lo ejecuta.
4. Si el intérprete de comandos no puede encontrar la orden que hemos tecleado, muestra un mensaje de error.

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.

• En su forma más habitual, 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.

La Terminal guarda un **HISTORIAL** y podéis ver cómo funciona en:
http://ubuntu-guia.blogspot.com/2010/08/historial-terminal-consola-ubuntu.html

3. MANUALES DE COMANDOS

- **`man` ⟶ (manual: manual)**

Nos ofrece el **manual** de cualquier comando en la propia terminal.

En esta guía he pretendido hacer solo una breve introducción de los comandos y sus argumentos más utilizados y sus posibilidades son muchas más, por ello os animo a que lo utilicéis siempre que tengáis alguna duda.

Para utilizarlo, basta con ejecutar "man" seguido del comando del que deseamos saber más o simplemente recordar:

```
man comando
```

En ocasiones la información que nos ofrece man puede llegar a ser excesiva. Casi todos los comandos y aplicaciones aceptan el argumento “--help” o "-h" para que muestre cierta ayuda más resumida. Por ejemplo con "apt-get":

```
apt-get --help
```

o

```
apt-get -h
```

En Ubuntu, los manuales están en Inglés pero podéis ponerlos en español (no todo está traducido). Para ello:

1. Instalar los paquetes de idioma español:

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

2. Recargar el idioma con:

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

3. Reiniciar la terminal y por ejemplo para ver el manual del comando "`ls`", ejecutamos:

```
man ls
```

Más información aquí.

4. COMANDOS RELACIONADOS CON ARCHIVOS Y DIRECTORIOS

- **`ls` → (list: listar)**

Nos muestra el contenido de la carpeta que le indiquemos después.

La sinapsis del comando sería:

```
ls [opciones] [ruta]
```

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 ambio 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.
`--color` → Muestra el contenido del directorio coloreado.

Ejemplos:

Si queremos que nos muestre lo que contiene el directorio o carpeta "/etc":

```
ls /etc
```

Si no ponemos nada interpretará que lo que queremos ver es el contenido de la carpeta donde estamos actualmente:

```
ls
```

Además acepta ciertos argumentos que pueden ser interesantes:

Para mostrar todos los archivos y carpetas, incluyendo los ocultos:

```
ls -a
```

Para mostrar los archivos y carpetas junto con los permisos que tiene, lo que ocupa, su dueño, ...:

```
ls -l
```

Además se pueden solapar los argumentos:

Si quisiéramos mostrar los archivos de la misma forma que antes, pero que muestre también los ocultos:

```
ls -la
```

- **`file`**

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

La sinapsis del comando sería:

```
file [OPCIÓN...] [ARCHIVO...]
```

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

- **`cd` → (change directory: cambiar directorio)**

Lo utilizamos para cambiar de directorio o carpeta en la terminal.

Podemos usarlo con rutas absolutas o relativas.

En las absolutas le indicamos toda la ruta desde la raíz (/). Por ejemplo, estemos donde estemos, si escribimos en consola …

```
cd /etc/apt
```

… nos llevará a esa carpeta directamente. Del mismo modo si escribimos …

```
cd /
```

… nos mandará a la raíz del sistema de ficheros.

Las rutas relativas son relativas a algo, y ese algo es la carpeta donde estemos actualmente. Imaginad que estamos en /home y queremos ir a la carpeta "Imágenes" dentro de vuestra carpeta personal. Con escribir …

```
cd Imágenes
```

… nos situará allí. Como véis hemos obviado el "/home/carpeta_personal" inicial ya que si no lo introducimos toma como referencia el directorio donde estamos, que es ese.
¿Y qué sucede si escribimos tan sólo …

```
cd
```

Sí, sólo “cd”. Esto lo que hace es que te lleva a tu carpeta personal directamente, estemos donde estemos. Es algo realmente muy práctico, muy simple y que no todos conocen.

- **`mkdir` → (make directory: hacer directorio)**

Crea una carpeta o directorio con el nombre que le indiquemos.

Nuevamente podemos usar rutas absolutas y relativas. Podemos indicarle toda la ruta que le precede al directorio que queremos crear:

```
mkdir /home/carpeta_personal/nueva_carpeta
```

O si estamos ya en la carpeta que lo va a contener basta con poner tan sólo el nombre de la nueva carpeta. Por ej. si ya estamos en /home/carpeta_personal:

```
mkdir nueva_carpeta
```

- **`rm` → (remove: borrar)**

Borra el archivo o la carpeta que le indiquemos.

Como antes se puede indicar la ruta completa o el nombre del archivo. Esto a partir de ahora lo vamos a obviar, creo que ya ha quedado claro con los dos comandos anteriores.

Para borrar un archivo:

`rm nombre archivo`
Para borrar un directorio o carpeta vacía:

`rm nombre carpeta`
Para borrar un directorio o carpeta que contiene archivos y/o otras carpetas que pueden, a su vez, contener más carpetas y archivos:

`rm -r nombre_carpeta`
Otras opciones:
"`-f`", no te pide una confirmación para eliminar.
"`-v`", va mostrando lo que va borrando.

También existe el comando "rmdir" para borrar carpetas o directorios:

`rmdir nombre_directorio`
Pero solo borrará directorios vacíos. Para borrar un directorio no vacío, junto con todo lo que tenga debajo, emplear "rm -r".

- **`cp` → (copy: copiar)**

Copia el archivo o directorio indicado donde le digamos.

Aquí podemos también jugar con las rutas, tanto para el fichero origen, como en el del destino. También podéis cambiar el nombre que le queréis poner a la copia.

La sinapsis del comando sería:

`cp [/ruta/de/original...] [/ruta/de/copia...]`
Por ejemplo, en nuestra carpeta personal vamos a crear una copia de seguridad "sources.list.backup", de nuestros repositorios "/etc/apt/sources.list". Lo voy a explicar según donde estemos colocados en la terminal, para comprender lo primordial que es saber en todo momento el directorio donde estamos colocados en la terminal:

- Si estamos colocados en nuestra carpeta personal, debemos de poner la ruta absoluta del original y la ruta relativa de la copia:

`cp /etc/apt/sources.list sources.list.backup`
- Si nos colocamos en el directorio que contiene el archivo original (`cd /etc/apt`), debemos de poner la ruta relativa del original y la ruta absoluta de la copia:

`cp sources.list /home/tu_usuario/sources.list.backup`
Nota: no olvides cambiar "tu_usuario" por el nombre de tu usuario o en su defecto sustituye "/home/tu_usuario" por el símbolo "**~**" (pulsar la combinación de teclas `Alt Gr`+`Ñ`). Sería así:

`cp sources.list ~/sources.list.backup`
- Si estuviéramos en cualquier otro directorio o simplemente para no tener problemas, escribimos las dos rutas absolutas:

`cp /etc/apt/sources.list /home/tu usuario/sources.list.backup`
Nota: no olvides cambiar "tu_usuario" por el nombre de tu usuario o en su defecto sustituye "/home/tu_usuario" por el símbolo "**~**" (pulsar la combinación de teclas `Alt Gr`+`Ñ`). Sería así:

```
cp /etc/apt/sources.list ~/sources.list.backup
```

- **`mv` → (move: mover)**

Es igual que el anterior, sólo que en lugar de hacer una copia, mueve directamente el archivo con el nombre que le indiquemos, pudiendo ser otro distinto al original:

La sinapsis del comando sería idéntica a copiar:

```
mv [/ruta/de/original...] [/ruta/de/destino...]
```

Ejemplo para mover un "archivo.flv" del directorio "/tmp" (temporales) a nuestra carpeta personal y de paso cambiarle el nombre a "mi_archivo.flv". Lo pongo con las dos rutas absolutas para no repetir todo lo anterior.

```
mv /etc/archivo.flv /home/tu_carpeta/mi_archivo.flv
```

O en su defecto sustituye "/home/tu_usuario" por el símbolo "**~**" (`Alt Gr`+`Ñ`)

Otro uso muy práctico que se le puede dar es para **renombrar** un archivo. Basta con indicar el nuevo nombre en el segundo argumento con la misma ruta del primero. En este ejemplo suponemos que ya estamos en la carpeta que lo contiene:

```
mv archivo.flv mi_archivo.flv
```

- **`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
```

- **`find` → (find: encontrar)**

Busca archivos o carpetas en la ruta que le indiques:

La sinapsis del comando sería:

```
find [/directorio/donde/buscar...] [-expresión] [búsqueda]
```

Donde "expresión" es el tipo de búsqueda y siempre se le antepone el signo "**-**"

La expresión "`-name`" sería para realizar una búsqueda por nombre. Por ejemplo, para buscar en todo el sistema de archivos o raíz "/" las carpetas y archivos que se llamen "pepino". Sería:

```
find / -name pepino
```

Si tuviéramos la seguridad de que se encuentra en /var por ejemplo, se lo indicaríamos:

```
find /var -name pepino
```

Si no estamos muy seguros del nombre podemos indicárselo con comodines. Supongamos que el nombre de lo que buscamos contiene “pepi”, en la misma carpeta de antes:

```
find /var -name *pepi*
```

Otra expresión sería "`-size`" para realizar la búsqueda por tamaño. Por ejemplo podemos decirle que encuentre los archivos/carpetas de más de 1500 KB:

```
find / -size +1500
```

Se pueden conbinar varios atributos para afinar la búsqueda. Por ejemplo, buscar los archivos/carpetas que contienen el nombre “pepi” y tienen menos de 1000 KB:

```
find / -name *pepi* -size -1000
```

La opción "2>/dev/null" es muy interesante para que no muestre los errores de "Permiso denegado". Por ejemplo para buscar en la raíz "/" el archivo "gdmflexiserver":

```
find / -name gdmflexiserver 2>/dev/null
```

- **`grep` → (localizar)**

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 ('). S

La sinapsis del comando sería:

```
grep [OPCIÓN] 'conjuntocaracteres' [ARCHIVOS...]
```

siendo 'conjuntocaracteres' la secuencia de caracteres a buscar, y file1, file2, y file3 los ficheros donde se debe buscar. Veamos Ejemplo para buscar TRIANGULARIZACION MATRIZ entre las líneas de los ficheros matrix.f y scaling.f.:

```
grep 'TRIANGULARIZACION MATRIZ' matrix.f 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.

- **`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:

Sacar por pantalla el contenido del fichero filename:

```
cat filename
```

Sacar por pantalla, secuencialmente y según el orden especificado, el contenido de los ficheros indicados (file1 y file2):

```
cat file1 file2
```

Aceptar lo que se introduce por el teclado y lo almacena en file1 (se crea file1):

```
cat >file1
```

5. COMANDOS RELACIONADOS CON SISTEMA Y ADMINISTRACIÓN

- **`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
```

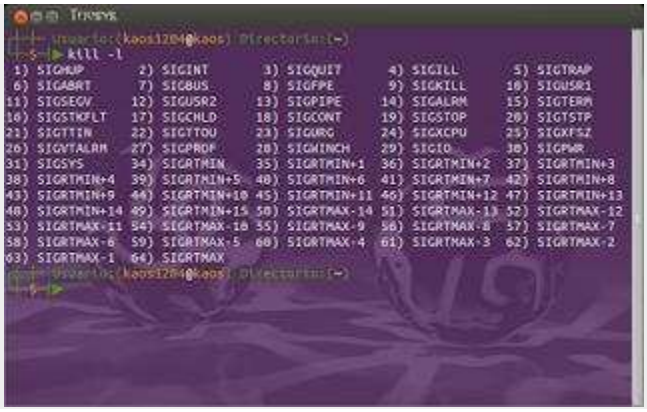
…nos mostrará un listado de todos los procesos, su PID a la izquierda y su nombre a la derecha. Si queremos más información:

```
ps aux
```

- **`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
```



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
```

Si quisieramos 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 -CONT  3760
```

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 Ubuntu 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
```

Más información en:
http://www.ubuntu-guia.com/2012/08/comandos-su-y-sudo.html
http://www.ubuntu-guia.com/2010/09/activar-desactivar-root-ubuntu.html

- **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
```

Más información en:
http://www.ubuntu-guia.com/2012/08/comandos-su-y-sudo.html

- **passwd → (password: contraseña)**

Cambia las contraseñas de cuentas de usuario.

Los usuarios normales solo pueden cambiar la contraseña de su propia cuenta y el superusuario puede cambiar todas.

La sinapsis del comando sería:

```
passwd [opciones] [USUARIO]
```

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

Si se especifica nombre-usuario, se cambiará la contraseña de dicho usuario (para esto se debe ser root), sino, la del usuario que ejecuta el comando. La mecánica de cambio de contraseña tiene 3 pasos:

- Ingresar la contraseña antigua.
- Ingresar la contraseña nueva.
- Repetir la contraseña nueva para confirmar.

- **`apt` → (advanced packets tool: herramienta avanzada de paquetes)**

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

Merece una guia solo para él y poidéis verla en:
http://www.ubuntu-guia.com/2011/01/comando-apt-get-en-ubuntu.html

- **`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-get:

```
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 despaquetaje. 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- 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)
* slm (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 "pepino.rpm" a "pepino.deb":

```
alien -d pepino.rpm
```

• `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
```

• `cal` → (calender: 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.

- **`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]...
```

- **`whoami`** → **(who I am: quien soy)**

Indica el usuario que está trabajando en la terminal actual.

La sinapsis del comando sería:

```
whoami
```

- **`finger`**

Presenta una información completa de los usuarios conectados a la red.

La sinapsis del comando sería:

```
finger [-lmsp] [user ...] [user@host ...]
```

- **`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.

- **`alias`**

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

La sinapsis del comando sería:

```
alias etiqueta='orden'
```

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

- **clear**

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

La sinapsis del comando sería:

```
clear
```

6. 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)

7. ACCESO A UNIDADES DE DISCO: MONTAJE Y DESMONTAJE

Linux a diferencia de Windows no utiliza letras ("C:", "D:", ...) para acceder a las distintas unidades de disco de un ordenador. 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 sinapsis del comando sería:

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

Por ejemplo para acceder al CD-ROM se teclearía 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 ext2 (3 o 4) 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,
"**/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).

En el caso de Ubuntu, el comando mount admite directamente los directorios /cdrom, /cdrom1, /floppy, ... para el montaje de nuestras unidades, por lo que lo único que habría que escribir, para montar por ejempo la disquetera, sería:

```
mount /floppy
```

Si omitimos el tipo de sistema de ficheros y/o el dispositivo, Ubuntu toma la información correspondiente del fichero /etc/fstab, el cual contiene información de los distintos sistemas de ficheros del equipo.

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. Siguiendo con el ejemplo de la disquetera en Ubuntu, debería escribir:

```
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: Existen en la actualidad distribuciones (por ejemplo, SuSE Linux) que realizan este proceso de forma automática, por lo que las unidades de disquete y CD-ROM quedan accesibles a todos los usuarios de una forma sencilla, empleando los comandos:

```
mount /mnt/floppy
umount /mnt/floppy
```

Siempre que /mnt/floppy sea la ruta adecuada.
Para desmontar una partición empleamos el comando "umount":

```
umount /dev/sdxX
```

Donde "xX" es la partición que queremos desmontar. Por ej. "sda5" que es la partición "5" del disco duro "a".

Un ejemplo de crear un montaje automático de una partición en el arranque de Ubuntu:

http://www.ubuntu-guia.com/2011/08/montar-particion-inicio-ubuntu-1104.html

8. OTROS COMANDOS BÁSICOS

• `du y df` → (Espacio ocupado en el disco)

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 ejecutar:

```
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:

```
df
```

Como el anterior, da el espacio en bloques. Para obtener la información en bytes se debe emplear el comando con la opción "-h":

```
df -h
```

• `lpr` → (Impresión)

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.

• `ln` → (Enlaces a ficheros)

Los enlaces nos van a permitir realizar copias de los ficheros (archivos o carpetas) 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. La sintaxis es la siguiente:

```
ln [opciones] origen [dest]
```

```
ln [opciones] origen... directorio
```

ENLACES DUROS (HARD LINKS)

Si utilizamos el comando ln sin especificar ninguna opción, por defecto crearemos un enlace duro.

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
```
Pulsamos Enter, escribimos algo, por ejemplo "*hola*" y pulsamos Enter y `Ctrl`+`D` para guardarlo.
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.

kaos1310@kaos:~$ **ls -i pruebaln penlace**
2753739 penlace 2753739 pruebaln
4 – Ahora modificamos pruebaln añadiendo otra línea ...

```
cat >> pruebaln
```
Pulsamos Enter, escribimos algo, por ejemplo "*adios*", pulsamos Enter y `Ctrl`+`D` para guardarlo.
... y comprobamos si también se modifica penlace:

kaos1310@kaos:~$ **cat pruebaln**
hola
adios
kaos1310@kaos:~$ **cat penlace**
hola
adios

5 – Ahora modificamos penlace añadiendo otra línea ...

```
cat >> penlace
```

Pulsamos Enter, escribimos algo, por ejemplo "*otra vez hola*", pulsamos Enter y `Ctrl`+`D`para guardarlo.
... y comprobamos si también se modifica pruebaln.

kaos1310@kaos:~$ **cat penlace**
hola
adios
otra vez hola
kaos1310@kaos:~$ **cat pruebaln**
hola
adios
otra vez hola
6 – Eliminamos pruebaln ...

```
rm pruebaln
```

... y comprobamos si penlace permanece y contiene la información correspondiente.

kaos1310@kaos:~$ **cat penlace**
hola
adios
otra vez hola
7 – Si utilizamos la orden **`ls -i`**, vemos que penlace sigue con el mismo número de inodo, que ahora solamente tendrá un enlace:

kaos1310@kaos:~$ **ls -i penlace**
2753739 penlace

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 Win2.

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.

kaos1310@kaos:~$ **ls -li pruebaln penlace**
2753739 -rw-r--r-- 2 kaos1310 kaos1310 25 dic 21 10:40 penlace
2753739 -rw-r--r-- 2 kaos1310 kaos1310 25 dic 21 10:40 pruebaln
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 de los permisos.

kaos1310@kaos:~$ **ls -li pruebaln penlace penlacesim**
2753739 -rw-r--r-- 2 kaos1310 kaos1310 25 dic 21 10:40 penlace
*2783398 **l**rwxrwxrwx 1 kaos1310 kaos1310 7 dic 21 11:00 **penlacesim** -> penlace*
2753739 -rw-r--r-- 2 kaos1310 kaos1310 25 dic 21 10:40 pruebaln
5 – Cambiamos penlace y comprobamos si cambia penalcesim.

```
cat >> penlace
```
Pulsamos Enter, escribimos algo, por ejemplo "*otra vez adios*", pulsamos Enter y `Ctrl`+`D`para guardarlo.

kaos1310@kaos:~$ **cat penlacesim**
hola
adios
otra vez hola
otra vez adios
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, además borramos pruebaln para dejar todo como estaba sin las pruebas que hemos hecho.

```
rm penlace
```
kaos1310@kaos:~$ **cat pruebaln**
hola
adios
otra vez hola
otra vez adios
kaos1310@kaos:~$ **cat penlacesim**
cat: penlacesim: No existe el archivo o el directorio

```
rm penlacesim
rm pruebaln
```

*9.5. 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

os 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
```

*9.6. 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.

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 nombre_archivo, podemos ver la configuración de permisos del archivo nombre_archivo:

- 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.

1. Comando chmod -> Para cambiar los permisos de un fichero se emplea el comando chmod, que tiene el formato siguiente:

```
chmod [quien] oper permiso files
```

donde:

- quien -> Indica a quien afecta el permiso que se desea cambiar. Es una combinación cualquiera de las letras **"u"** para el usuario, **"g"** para el grupo del usuario, **"o"** para los otros usuarios, y **"a"** para todos los anteriores. Si no se da el quien, el sistema supone **"a"**.

- oper -> Indica la operación que se desea hacer con el permiso. Para dar un permiso se pondrá un +, y para quitarlo se pondrá un -. Si quiero dar exactamente unos permisos, pondremos =.
- permiso -> Indica el permiso que se quiere dar o quitar. Será una combinación cualquiera de las letras anteriores : r,w,x,s.
- files -> Nombres de los ficheros cuyos modos de acceso se quieren cambiar.

Por ejemplo, para quitar el permiso de lectura a los usuarios de un fichero llamado fichero.txt el comando a utilizar es:

```
chmod a -r fichero.txt
```

Los permisos de lectura, escritura y ejecución tienen un significado diferente cuando se aplican a directorios y no a ficheros normales. En el caso de los directorios el permiso r significa la posibilidad de ver el contenido del directorio con el comando ls; el permiso w da la posibilidad de crear y borrar ficheros en ese directorio, y el permiso x autoriza a buscar y utilizar un fichero concreto.

2. Comando chown -> Por otra parte, el comando chown se emplea para cambiar de propietario ("change owner") a un determinado conjunto de ficheros. Este comando sólo lo puede emplear el actual propietario de los mismos. Los nombres de propietario que admite Linux son los nombres de usuario, que están almacenados en el fichero /etc/passwd.

La forma general de utilización del comando chown es:

```
chown newowner file1 file2 ...
```

3. Comando chgrp -> Análogamente, el grupo al que pertenece un fichero puede ser cambiado con el comando chgrp, que tiene una forma general similar a la de chown,

```
chgrp newgroup file1 file2...
```

Los grupos de usuarios están almacenados en el fichero /etc/group.

10. GESTIÓN DE USUARIOS Y GRUPOS

10.1. 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.

10.2. 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).

10.3.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:
a) 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.
b) Entorno linuxconf:
Desde una sesión en modo consola podemos escribir la orden linuxconf para acceder a un entorno que nos permite gestionar todo Linux, y por lo tanto la gestión de usuarios.
Entramos en Users, User accounts (cuentas de usuario), y podremos ver todas las cuentas de usuario que están definidas en el sistema.
Para añadir una nueva cuenta, con el tabulador seleccionamos el botón Add.
Escribiremos el Login, Full name, group, Home directory, User ID, por ejemplo un cero en Must deep # days y siete en Warm # days before expiration. Con el tabulador nos posicionamos en el botón Accept.
Para salir de linuxconf pulsamos el botón Dismiss y Quit.
c) 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
#adduser usuario
```

Habremos creado un usuario sin contraseña. Para habilitar su cuenta comprobamos si en el campo clave de los ficheros /etc/passwd y /etc/shadow existen signos de admiración cerrada (!), en ese caso debemos borrarlos. Posteriormente establecemos una contraseña si lo deseamos.

Si queremos crear un usuario a nuestra medida utilizaremos la siguiente sintaxis:

```
#useradd/adduser –r -p -u -g -c -d -s
```

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.
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.

10.4. Otras órdenes para la gestión de usuarios y grupos

chfn: permite cambiar el nombre completo del usuario:

```
#chfn –f
```

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

```
#groupadd [-g GID] [-f]
```

-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 ] [-n ]
```

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

```
#id

#id usuario
```

11. REDIRECCIONAMIENTO Y TUBERÍAS

11.1. 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 al gunos 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
$

11.2. Redireccionamiento de la entrada y la salida

Al igual que en MS-DOS, podemos utilizar los caracteres > y <>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
$sort <. 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). *10.3. 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.

12. LA EDICIÓN DE TEXTO. EL EDITOR vi

12.1. 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. La versión proporcionada con Mandrake es una versión mejorada de vi (vim – VI iMproved).
Para comenzar con vi y editar un fichero de texto emplearemos la sintaxis:

```
$vi
```

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.
• En modo órdenes o modo comandos no podremos insertar texto. Nos permitirá usar ciertas órdenes de edición de ficheros o cambiar a otros modos.
• 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.
• 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.

12.2. 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.

12.3. 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.

12.4. 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 : 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).

12.5. 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.
• /: desplaza el cursor hacia delante hasta que encuentra el texto 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/: borra todo desde la posición del cursor hasta que encuentra el texto cadena.
d0: borra todo hasta el inicio de la línea actual.

12.6. 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?: 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.

12.7. 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.

12.8. Editar otros ficheros

Si estamos editando un fichero con vi, podemos editar otro escribiendo :e 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.

12.9. 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.