Curso Linux Admin

Comandos Básicos de Sistemas Linux

Temario

Guía de Comandos GNU/LINUX

Comandos: órdenes que el usuario ejecuta en una consola de texto desde el prompt. Este indica que el intérprete está listo para trabajar. Al tipear una orden, el intérprete de comandos (en nuestro caso bash), lo transforma en una pregunta entendible para el kernel y transforma la respuesta que el núcleo da en algo entendible para nosotros.

Bash, actúa como un intérprete entre el kernel y el usuario.

Los comandos tienen modificadores que brindan más información al usuario de lo que arroja cada comando por sí sólo. El usuario necesita de información específica u ordenada de una determinada manera así que, por lo general, da mucho uso a cada una de las opciones disponibles.

Comando pwd (Print Working Directory)

Lo primero que tenemos que conocer es donde estamos posicionados en el sistema. El prompt no nos muestra toda la ruta sino el último directorio. Es conveniente saber en qué directorio estamos a partir de "/", la raíz del sistema operativo.

Tenemos que recordar que inmediatamente después del login nos posicionamos en directorio *home*

En el ejemplo que mostraremos a continuación el directorio en el que estamos es /root ya que nos hemos logueado como superusuario (root).

Ejemplo de Comando pwd: Imprimir directorio de trabajo.

```
equipo1:~# pwd
/root
```

Ahora que ya sabemos dónde estamos, es posible que deseemos movernos a otro Directorio, entonces debemos usar el comando cd (cambiar de directorio).

Comando cd (Change Directory)

El comando cd nos permite dirigirnos hacia el directorio que nosotros queramos.

En el ejemplo vamos a ir a /tmp:

```
equipo1:~# cd /tmp
equipo1:/tmp#
```

Si tipeamos cd volvemos siempre a nuestro home, no importa cuán lejos estemos de él.

Ejemplo:

```
equipo1:/tmp# cd <enter>
equipo1:~#
```

Modificadores del comando cd

```
equipo1:~# cd -
```

Este modificador hace que el usuario siempre se mueva entre los últimos dos directorios visitados.

Ejemplo No 1:
En el siguiente ejemplo nos cambiamos primero al directorio tmp, luego al directorio /etc, y después con el comando anterior nos movemos entre estos dos últimos.

```
equipo1:~# cd /tmp
equipo1:/tmp# cd /etc
equipo1:/etc#  cd -
equipo1:/tmp# cd -
equipo1:/etc#
```

Para saber dónde queremos ir tenemos primero que saber dónde estamos parados. (Para el que no sabe dónde va... nunca soplan buenos vientos!)

El directorio en el cual nos encontramos se llama punto "." y el que está antes de este se llama también punto "." Se dice que es el padre del que estamos parados. Si queremos ir de un directorio, al directorio padre, ejecutamos el comando cd ... y nos vamos un directorio hacia arriba.

```
equipo1:/etc# cd ..
equipo1:/#
```

Ejemplo No 2:
Ahora pongamos un ejemplo sobre el árbol de directorios.

```
/
|
/------var
|  /---------log
|         /------apache2
|  /--------backups
```

Supongamos ahora que estamos parados en el directorio apache2

```
equipo1:/etc# cd /var/log/apache2
```

Queremos ir al directorio que está a la misma altura de log, que se llama backups podríamos ejecutar este comando:

```
equipo1:/var/log/apache2# cd ../../backups
```

Es lo mismo que ejecutar dos veces cd .. y después cd backups.

Ahora que ya aprendimos cómo llegar hasta donde queremos ir, pasaremos a estudiar la forma de visualizar lo que hay en los directorios.

Comando ls (list)

Este comando permitirá ver el contenido que presenta un directorio.

Para eso nos muestra la información con colores a fin de que podamos identificar mejor los ficheros que allí se encuentran.

```
equipo1:/var/log/apache2# cd /
equipo1:/# ls
```

```
equipo1:/# ls
bin  boot  cdrom  dev  etc  home  initrd.img  lib  lost+found  media  mnt
opt  proc  root  sbin  selinux  srv  sys  tmp  usr  var  vmlinuz
```

Aquí nos muestra el contenido del directorio / en color azul porque son directorios.

COLORES	
Blanco	*Archivos de textos o binarios no ejecutables*
Verde	*Archivos de textos ejecutables (scripts) o binarios ejecutables*
Celeste	*Links Simbólicos*
Rojo con letras blancas parpadeantes	*Links simbólicos rotos*
Amarillo	*Dispositivos*
Violeta	*Archivos de imagenes o archivos temporales*

Práctica

Realizar un ls de los siguientes directorios, identificando los colores que los objetos poseen.

```
/var/log/apache2
/etc/init.d
/etc/rc1.d
/dev
/usr/share/apache2/icons
```

El comando ls presenta muchos modificadores. A continuación veremos algunos de ellos con ejemplos.

En este sistema operativo podemos ver que tanto los archivos como los directorios pueden tener varios puntos pero cuando el punto está adelante los transforma en ocultos. Por ejemplo .hola no se verá con un ls normal, al utilizar el modificador "a" podemos ver todos los archivos ocultos que se encuentran en ese directorio.

¿Por qué queremos tener archivos ocultos?

Generalmente los archivos ocultos son los que contienen información acerca de configuraciones. Al no ser visibles fácilmente podemos garantizar que no serán borrados en forma accidental.

Veamos un ejemplo:

```
equipo1:/var/log/apache2# cd <enter>
equipo1:~# ls
```

Ejecutemos un ls simple y después ejecutemos un ls -a para ver la diferencia.

```
equipo1:~# ls
apt.txt dbootstrap_settings instalar install-report.template

equipo1:~# ls -a
.  .gconf .profile instalar .gconfd .ssh install-report.template .aptitude
.gnome .viminfo  .bash_history .gnome2 apt.txt .bashrc .gnome2_private
dbootstrap_settings
```

Aquí podemos ver la diferencia: la mayoría de los archivos precedidos por un ¨.¨ (punto) son archivos de configuración. Si aparecieran de esta forma podrían ser borrados sin querer. De este modo quedan resguardados.

Otra opción interesante es el parámetro -l

Al listar con el modificador -l obtenemos la siguiente información de los archivos y directorios.

```
equipo1:~# ls -l
total 16
-rw-r--r-- 1 root root 269 May 26 15:55 apt.txt
-rw-r--r-- 1 root root 183 May 26 15:13 dbootstrap_settings
-rwx--x--x 1 root root 307 May 26 15:28 instalar
-rw-r--r-- 1 root root 1336 May 26 15:13 install-report.template
```

Está dispuesta en columnas para que nos resulte más sencillo reconocerla. Veamos cómo queda cada columna:

1. La primera letra que aparece en la 1ra columna indica si lo que estamos viendo es un archivo o un directorio.
   -rwxr-xr-x En esta línea lo que estamos viendo es un archivo ya que no hay nada en la primera columna.
   drwxrwxr-x En este caso la letra d inicial indica que estamos en presencia de un directorio. A continuación veremos los permisos que el archivo o directorio poseen.
2. Esta columna nos dice si este objeto posee links que lo estén apuntando.
3. Usuario dueño del objeto.
4. Grupo dueño del objeto.
5. Tamaño en bytes del objeto.
6. Fecha de creación del objeto.

Para ver el tamaño en una unidad entendible podemos combinar dos parámetros:

```
equipo1:~# ls -lh
total 16K
-rw-r--r-- 1 root root 269 May 26 15:55 apt.txt
-rw-r--r-- 1 root root 183 May 26 15:13 dbootstrap_settings
-rwx--x--x 1 root root 307 May 26 15:28 instalar
-rw-r--r-- 1 root root 1.4K May 26 15:13 install-report.template
```

Ahora podemos ver mejor cual es el tamaño real de cada uno. Es interesante ver qué pasa con los links al usar ls -l:

```
equipo1:~# cd /etc/rc2.d
equipo1:/etc/rc2.d# ls -l
lrwxrwxrwx 1 root root 13 May 26 14:54 K14ppp -> ../init.d/ppp
lrwxrwxrwx 1 root root 14 May 26 15:52 K15bind -> ../init.d/bind
lrwxrwxrwx 1 root root 14 May 26 15:52 K20dhcp -> ../init.d/dhcp
lrwxrwxrwx 1 root root 15 May 26 15:53 K20mysql -> ../init.d/mysql
lrwxrwxrwx 1 root root 15 May 26 15:56 K20samba -> ../init.d/samba
lrwxrwxrwx 1 root root 16 May 26 15:33 K20xprint -> ../init.d/xprint
lrwxrwxrwx 1 root root 17 May 26 15:53 K91apache2 -> ../init.d/apache2
lrwxrwxrwx 1 root root 18 May 26 14:55 S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx 1 root root 15 May 26 14:55 S11klogd -> ../init.d/klogd
...
```

Otra opción interesante del ls es la opción -R que nos muestra que es lo que hay dentro de los

subdirectorios que están dentro de un directorio.

```
equipo1:/etc/rc2.d$ ls -R /etc/apache2/
/etc/apache2/:
apache2.conf  apache2.conf.dpkg-dist  conf.d  envvars  httpd.conf  mods-
available  mods-enabled     ports.conf  sites-available  sites-enabled

/etc/apache2/conf.d:
charset  security

/etc/apache2/mods-available:
actions.conf        authn_file.load       cgid.conf          deflate.load
ident.load         mime_magic.load           python.load
unique_id.load   actions.load        authnz_ldap.load      cgid.load
dir.conf              imagemap.load         negotiation.conf
rewrite.load                      userdir.conf     alias.conf
authz_dbm.load        cgi.load              dir.load         include.load
negotiation.load      setenvif.conf               userdir.loadalias.load
authz_default.load    charset_lite.load  disk_cache.conf  info.conf
     php5.conf            setenvif.load                usertrack.load
     asis.load           authz_groupfile.load  dav_fs.conf
     disk_cache.load  info.load          php5.load                sick-
hack-to-update-  modules  version.loadauth_basic.load     authz_host.load
dav_fs.load        dump_io.load     ldap.load           proxy_ajp.load
speling.load                      vhost_alias.load auth_digest.load
authz_owner.load      dav.load              env.load
log_forensic.load     proxy_balancer.load  ssl.conf     authn_alias.load
authz_user.load       dav_lock.load      expires.load     mem_cache.conf
     proxy.conf           ssl.load
authn_anon.load     autoindex.conf        dav_svn.conf
ext_filter.load       mem_cache.load        proxy_connect.load
status.confauthn_dbd.load         autoindex.load        dav_svn.load
file_cache.load       mime.conf             proxy_ftp.load
status.loadauthn_dbm.load      cache.load               dbd.load
filter.load           mime.load          proxy_http.load
substitute.loadauthn_default.load      cern_meta.load         deflate.conf
headers.load          mime_magic.conf       proxy.load
suexec.load

/etc/apache2/mods-enabled:
alias.conf       authz_default.load    autoindex.conf  dav_svn.conf
env.load              negotiation.load      proxy_connect.load
setenvif.conf    alias.load            authz_groupfile.load  autoindex.load
dav_svn.load  mime.conf          php5.conf         proxy_http.load
setenvif.load    auth_basic.load  authz_host.load       cgi.load
dir.conf         mime.load         php5.load         proxy.load
status.confauthn_file.load  authz_user.load       dav.load        dir.load
     negotiation.conf  proxy.conf        python.load         status.load

/etc/apache2/sites-available:
default  default.dpkg-dist  default-ssl  trac

/etc/apache2/sites-enabled:
000-default
```

En este ejemplo podemos ver que dentro del directorio /etc/apache2 tenemos cinco subdirectorios llamados: **mods-availble, mods-enabled, sites-available, sites-enabled y ssl.**

Comando du (disk usage)

El comando du nos dice qué espacio ocupa cada archivo en el disco y también cuál es el total del directorio. Podemos usar el parámetro h para que nos muestre la unidad de medida que está usando.

Ejemplo:

```
equipo1:~# du -h /etc/samba
16K /etc/samba/
```

En este ejemplo obtenemos la información de cuánto ocupa el directorio /etc/samba en el disco.
Veamos otra opción interesante

```
equipo1:~# du -h /etc/samba/*
4.0K /etc/samba/gdbcommands
8.0K /etc/samba/smb.conf
```

Al usar el * nos muestra cuál es el peso de cada uno de los archivos que se encuentran en el directorio pasado como argumento.

Comando df (disk free)

Este comando nos permite ver cuál es el espacio libre que nos queda por file system (particiones) y cuánto es el usado. Si a esto le agregamos el parámetro h lo podremos observar con unidades.

Ejemplo:

```
equipo1:/$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             5.6G  3.5G  1.9G  66% /
tmpfs                 126M     0  126M   0% /lib/init/rw
udev                   10M   72K   10M   1% /dev
tmpfs                 126M     0  126M   0% /dev/shm
```

Comando history

Este comando nos permite observar cuáles son los comandos que hemos usado. Para darnos esta información el comando history lee lo que dice el archivo oculto .bash_history que se encuentra en los homes de los usuarios. Veremos a continuación como es la salida por pantalla de este comando.

Ejemplo:

```
equipo1:~# history
1 vi /boot/grub/grub.conf
2 init 6
3 ls
4 cd /boot/grub
5 vi grub.conf
6 man grub
7 man grub.conf
8 vi grub.conf
9 init 6
10 cd /tmp
11 ls
```

Ejemplo 2:
Supongamos que queremos ejecutar un comando que está en la línea 11.

¿Qué podemos hacer? En vez de tipear el comando nuevamente podemos tipear en la consola el número de línea anteponiéndole el carácter "!".
Probemos...

```
equipo1:~# !11
ls
apt.txt dbootstrap_settings instalar install-report.template:
```

Este archivo es muy importante porque contiene todos los comandos que el usuario fue ejecutando durante sus sesiones. Permanece oculto para que los usuarios no lo borren accidentalmente.

Comando mkdir (make directory)

Con este comando creamos nuevos directorios vacios. En este ejemplo veremos cómo hacerlo.

```
equipo1:~# mkdir nuevodir
```

Con la opción mkdir -p podemos crear un árbol completo de directorios:

```
equipo1:~# mkdir -p dir1/dir2/dir3
```

La línea de comandos anterior nos permite crear el siguiente árbol:

```
/
|------/dir1
        |-----/dir2
       |------/dir3
```

Comando rmdir (Remove directory)

El comando rmdir permite borrar directorios, si los mismos cumplen con la condición de encontrarse vacios.

```
equipo1:~# rmdir nuevodir
```

También podemos usar la opción rmdir -p para borrar un árbol de directorios cumpliendo con la condición también de vacios. Si los directorios contienen archivos, tendremos que utilizar el comando rm que veremos más adelante.

```
equipo1:~# rmdir -p dir1/dir2/dir3
```

Comando cp (copy)

El comando cp se usa para copiar archivos. Veamos su aplicación:

Ejemplo: Creamos un directorio que se llama prueba y luego vamos a copiar a ese directorio algunos archivos.

Con este comando creamos el directorio prueba

```
equipo1:~# mkdir prueba
```

Ahora nos cambiamos de directorio

```
equipo1:~# cd prueba
```

Una vez allí copiaremos el archivo /etc/samba/smb.conf a este directorio. En este ejemplo el punto es el destino.

```
equipo1:~/prueba# cp /etc/samba/smb.conf .
```

Usaremos el punto para indicar que queremos copiar el archivo ó directorio origen al directorio en donde estamos parados. En este caso estamos creando una copia de seguridad del archivo origen cambiándole de nombre en el destino.

```
equipo1:~/prueba# cp /etc/samba/smb.conf ./smb.conf.bak
```

Una opción interesante del comando cp es el parámetro -a que hace una copia exacta de los directorios y subdirectorios. También de los permisos o links que pudiera haber en el directorio de origen.

```
equipo1:~# cp -a /var/log /root
```

Comando mv (move)

Este comando se utiliza para renombrar archivos ya que hace una réplica del original pero le cambia el nombre.

```
equipo1:~# mv smb.conf.bak httpd.conf
```

Comando rm (remove)

Cuando lo que queremos es borrar archivos tenemos que usar este comando. Debemos tener en cuenta que desde la línea de comandos no tenemos papelera de reciclaje, y tampoco hay un undelete, así que cuando borramos, nunca más podemos recuperar el archivo original.

```
equipo1:~/prueba# rm smb.conf
<Nos preguntará:>
rm remove: `smb.conf.bak'? y
```

Si tipeamos “y” , el archivo se borra definitivamente.

Este comando generalmente tiene un alias que es rm -i por eso es que nos pregunta siempre si estamos seguros.
Este comando sirve también para borrar directorios completos. En ese caso tenemos dos opciones -r, de recursividad. Siempre pregunta antes de borrar y el -f, que fuerza el borrado SIN PREGUNTAR.

Advertencia: Cuidado con este comando cuando se ejecute como root, ya que podríamos perder toda la información del disco duro.

Ejemplo:
Vamos a crear un directorio llamado prueba

```
equipo1:~# mkdir prueba
equipo1:~# cd prueba
```

Después nos moveremos hasta un directorio anterior y tenemos que ejecutar rm con la opción -r.

```
equipo1:~# rm -r prueba
```

Comando touch

El comando **touch** se utiliza para la creación de archivos vacios. En caso de que estos ya existan bajo el mismo nombre, los actualiza cambiándoles la fecha de creación en el sistema.
Ejemplos del comando:

```
equipo1:~# touch uno
equipo1:~# touch dos
equipo1:~# touch tres
equipo1:~# touch cuatro
```

Hemos creado los archivos “uno, dos, tres y cuatro”

Comando ln

En GNU/LINUX con este comando podremos crear en el sistema accesos directos.

- **Acceso Directo**: link de acceso rápido a determinados archivos o directorios del sistema. Una de sus propiedades es conservar la información de la ruta que lleva al archivo o directorio destino.

Existen dos tipos de link:

- **Link hard**: Su tarea es realizar un enlace desde un ínodo* determinado hasta los bloques que componen el archivo.
- **Link Softs**: Su tarea es crear un enlace desde un ínodo* hasta otro ínodo.

El Sistema Operativo utiliza, en sus script de arranque muchos links del tipo soft. En la unidad anterior pudimos apreciar que en él se detallaban todos los servicios que se levantarán en el nivel de corrida 2.
Cada uno de ellos en realidad está representado por un link al directorio del deamon específico y no es el archivo propiamente dicho. Veamos un ejemplo:

```
equipo1:~# cd /etc/r2.d/
equipo1:~# ls -l
total 0
lrwxrwxrwx 1 root root 13 May 26 14:54 K14ppp -> ../init.d/ppp
lrwxrwxrwx 1 root root 14 May 26 15:52 K15bind -> ../init.d/bind
lrwxrwxrwx 1 root root 14 May 26 15:52 K20dhcp -> ../init.d/dhcp
lrwxrwxrwx 1 root root 15 May 26 15:53 K20mysql -> ../init.d/mysql
lrwxrwxrwx 1 root root 15 May 26 15:56 K20samba -> ../init.d/samba
lrwxrwxrwx 1 root root 16 May 26 15:33 K20xprint -> ../init.d/xprint
lrwxrwxrwx 1 root root 17 May 26 15:53 K91apache2 -> ../init.d/apache2
lrwxrwxrwx 1 root root 18 May 26 14:55 S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx 1 root root 15 May 26 14:55 S11klogd -> ../init.d/klogd
lrwxrwxrwx 1 root root 15 May 26 14:54 S20exim4 -> ../init.d/exim4
lrwxrwxrwx 1 root root 15 May 26 14:54 S20inetd -> ../init.d/inetd
```

Un ínodo, nodo-i, nodo índice o i-node en inglés es una estructura de datos

propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo UNIX como es el caso de Linux. Un ínodo contiene las características (permisos, fechas, ubicación, pero NO el nombre) de un archivo regular, directorio, o cualquier otro objeto que pueda contener el sistema de ficheros.

Creando links de tipo hard

Crearemos un hard link a un archivo del directorio log y lo guardaremos en el directorio home de root. Este comando tiene dos parámetros:

- el archivo destino
- el nombre del link

```
equipo1:~# ln /var/log/messages logins
```

logins es un enlace a /**var**/**log**/**messages** por lo tanto nunca nos ocupará espacio físico en el disco duro.

Creando links de tipo soft

Cuando creamos el link hard, debemos indicar cuál es el archivo o directorio destino y el nombre del link. Esta misma función haremos cuando necesitemos crear un Link soft con la particularidad de la opción **-s** para identificarlo.

Link 1

```
equipo1:~# ln -s /var/log/messages logins1
```

Vamos a hacer un link soft a un directorio :

Link 2

```
equipo1:~# ln -s /var/log logins2
lrwxrwxrwx 1 root root 17 jul 29 18:00 logins1 -> /var/log/messages
lrwxrwxrwx 1 root root 8  jul 29 18:01 logins2 -> /var/log
```

Ahora vamos a hacer **cd** al link2 y veremos qué se nos imprime en pantalla.

El comando **ls -l** nos permitirá ver todos los soft links que hemos creado y a qué archivo o directorio apuntan.

Ejemplo:

```
equipo1:~# ls -l
-rwxr-xr-x 1 root root 782335 jul 21 17:44 icewm-1.2.7.tar.gz
```

A la hora de utilizar los links todo depende del destino...
Si el destino es un archivo podemos implementar los dos tipos de links mientras que si el destino es un directorio solo podemos aplicar los soft.

Comando shutdown

Este comando tiene la siguiente sintaxis:

```
shutdown [opciones] tiempo [mensaje]
```

El comando **shutdown** presenta una única propiedad: **apagar el equipo**.

Opciones más utilizadas del comando:

- **shutdown -r**: reiniciará el equipo
- **shutdown -c**: cancelación de la orden shutdown
- **shutdown -h**: apagará el equipo (sin que éste se vuelva a iniciar)
- **shutdown -k**: enviará el mensaje de apagado del equipo, sin apagarlo.

Ejemplo:

```
shutdown -r 30 "Reiniciando el sistema"
```

Provocará un reinicio rápido dentro de 5 minutos, enviando el mensaje "Reiniciando el sistema" a todos los usuarios conectados.

Comando halt

Este comando tiene la siguiente sintaxis:

```
halt [opciones]
```

El comando **halt** se utiliza para apagar el equipo.

Opciones más utilizadas del comando:

- **halt -f**: Fuerza el apagado el equipo.

Comando reboot

Este comando tiene la siguiente sintaxis:

```
reboot [opciones]
```

El comando **reboot** se utiliza para reiniciar el equipo.

Opciones más utilizadas del comando:

- **reboot -f**: Fuerza el reinicio del equipo.

Comando poweroff

Este comando tiene la siguiente sintaxis:

```
poweroff [opciones]
```

El comando **poweroff** se utiliza para apagar el equipo.

Opciones más utilizadas del comando:

- **poweroff -f**: Fuerza el apagado el equipo.

Comando cat

Este comando concatena (**catenate**) archivos y los imprime en la salida estándar. Si no se le pasa ningún argumento lee de la entrada estándar.
Existe también **zcat** que hace lo mismo pero con archivos compactados dentro del sistema. El comando **cat** puede ser interesante para ficheros pequeños, pero en general el comando **less** resulta de mayor utilidad en la mayoría de casos.

Ejemplo:

```
# cat /etc/passwd /etc/shadow
```

Comandos more y less

Los comandos more y less paginan (dividen en páginas) uno o varios archivos y los muestran en la terminal. De no indicárseles un archivo, paginan la entrada estándar. Se diferencian en las facilidades que brindan.

more: es más restrictivo en cuanto al movimiento dentro del texto, viasualiza sucesivamente el porcentaje del archivo visto hasta el momento. Cuando se alcanza el final del último archivo a paginar, more termina automáticamente.

Ejemplo:

```
equipo1:~# more /etc/apache2/apache2.conf
```

less: acepta el empleo de todas las teclas de movimiento tradicionales, es decir, nos permitirá ir tanto hacia arriba como hacia abajo del archivo.

Ejemplo:

```
equipo1:~# less /etc/apache2/apache2.conf
```

En caso de que nuestro sistema no tenga instalado el comando, escribimos el comando **apt-get install less**, para instalarlo.

less nos permite navegar el archivo, por ejemplo escribimos /**localhost.** Nos va a marcar todas las palabras que haya encontrado, con la letra n minúscula nos vamos a ir moviendo hacia abajo hasta ir viendo todas las palabras que dicen **localhost**, con la N hacen lo mismo pero para arriba.

Ejemplos:

- **q**: permite interrumpir el proceso y salir
- /**p**: realiza búsquedas del patrón p dentro del texto. Para repetir la búsqueda del mismo patrón sólo es necesario escribir /
- **[n]b**: en **more** permite regresar n páginas (por defecto n=1)
- **[n]f**: en **more** se adelantan n páginas y en **less**, n líneas.

El **man**, para dar formato a su salida, utiliza por defecto el paginador **less**. Existen además los comando **zless** y **zmore** que permiten paginar a los archivos compactados sin necesidad de descompactarlos previamente en nuestro disco.

Comando tail y head

Los comandos **tail** y **head** muestran respectivamente el final y el comienzo (10 líneas por defecto) de uno o varios archivos. De no especificarse al menos un archivo toman la entrada estándar.

Este comando tiene la siguiente sintaxis:

- Muestra el Final: `tail [opciones] [archivos]`
- Muestra el comienzo: `head [opciones] [archivos]`

Opciones más utilizadas del comando:

- **tail -f**: para el caso de tail se ejecuta de forma sostenida. Significa que visualiza hasta el final del archivo hasta que se interrumpa el proceso (ctrl-c)
- **tail -q**: nos coloca los encabezamiento con el nombre de los archivos cuando se indican varios (quiet)
- **tail -<n>**: imprime las n últimas (primeras) líneas en lugar de las diez establecidas por defecto

El comando **head**, lo usamos para ver el encabezado, es decir las primeras 15 líneas del archivo.
Ejemplo:

```
equipo1:/tmp# head /etc/inittab
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $
# The default runlevel
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
```

Si queremos indicarle a head cuántas líneas quieren que les muestre pueden escribir lo siguiente:

```
equipo1:/tmp# head -n 20 /etc/inittab
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $
# The default runlevel.
id:2:initdefault:
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
# What to do in single-user mode.
~~:S:wait:/sbin/sulogin
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
```

```
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
```

tail (significa cola) y es muy interesante para ver que esta sucediendo en el sistema operativo en tiempo real ya que con la opción **-f** nos muestra un archivo a medida que se le van agregando líneas.

```
equipo1:/tmp# tail /etc/inittab
# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Ejemplo 2:

```
equipo1:/tmp# tail /var/log/messages
Mar 18 16:04:43 equipo1 kernel: 8139cp: 10/100 PCI Ethernet driver v1.2
(Mar 22, 2004)
Mar 18 16:04:43 equipo1 kernel: pci_hotplug: PCI Hot Plug PCI Core
version:0.5
Mar 18 16:04:43 equipo1 kernel: pci_hotplug: PCI Hot Plug PCI Core version:
0.5
Mar 18 16:04:43 equipo1 kernel: eth0: link up, 10Mbps, half-duplex, lpa
0x0000
Mar 18 16:04:58 equipo1 xfs: ignoring font path element
/usr/lib/X11/fonts/cyrillic/ (unreadable)
Mar 18 16:04:58 equipo1 xfs: ignoring font path element
/usr/lib/X11/fonts/CID(unreadable)
Mar 18 16:05:01 equipo1 squid[1188]: Squid Parent: child process 1192
started
Mar 18 16:05:03 equipo1 Xprt_64: No matching visual for GlcontextMode with
visual class = 0 (32775), nplanes = 8
Mar 18 16:24:41 equipo1 -- MARK --
Mar 18 16:44:41 equipo1 -- MARK - -
```

Este es uno de los comandos más usados por los usuarios por presentar la propiedad de mirar el archivo en el momento en que va creciendo. Cuando un archivo crece los últimos datos se agregan al final. Supongamos que queremos saber si los correos están saliendo, ejecutamos el comando **tai**l al archivo que contiene la cola de los correos salientes:

```
tail -f /var/log/auth.log
```

Ahora nos logueamos en otra consola y apreciaremos que aparece la entrada del nuevo logueo en el fichero **auth.log**, lo importante es que no salgamos del comando **tail**.

```
equipo1:/tmp# tail -f /var/log/auth.log
Mar 18 16:17:01 equipo1 CRON[1652]: (pam_unix) session closed for user root
Mar 18 16:39:01 equipo1 CRON[1715]: (pam_unix) session opened for user root
by(uid=0)
Mar 18 16:39:02 equipo1 CRON[1715]: (pam_unix) session closed for user root
Mar 18 16:43:42 equipo1 login[1316]: (pam_unix) session opened for user
root by LOGIN(uid=0)
Mar 18 16:43:42 equipo1 login[1316]: ROOT LOGIN on `tty2'
Mar 18 17:00:46 equipo1 login[1317]: (pam_unix) session opened for user
root by LOGIN(uid=0)
Mar 18 17:00:46 equipo1 login[1317]: ROOT LOGIN on `tty3'
Mar 18 17:02:07 equipo1 login[1318]: (pam_unix) session opened for user
```

```
root by LOGIN(uid=0)
Mar 18 17:02:07 equipo1 login[1318]: ROOT LOGIN on `tty4'
Mar 18 17:02:45 equipo1 login[1319]: (pam_unix) session opened for user
ciua by LOGIN(uid=0)
```

Aquí podemos apreciar que nos hemos logeado varias veces.

Ejemplo 3:

En este ejemplo haremos lo siguiente: “en una terminal ejecutamos el comando tail”

```
home:~# tail -f /var/log/messages
Mar 3 05:43:13 home -- MARK --
Mar 3 06:03:14 home -- MARK --
Mar 3 06:23:14 home -- MARK --
Mar 3 06:30:23 home syslogd 1.4.1#16: restart.
Mar 3 06:43:14 home -- MARK --
Mar 3 06:50:49 home kernel: inserting floppy driver for 2.4.26- 1-386
Mar 3 06:50:49 home kernel: Floppy drive(s): fd0 is 1.44M
Mar 3 06:50:49 home kernel: FDC 0 is a National Semiconductor PC87306
Mar 3 06:51:03 home kernel: end_request: I/O error, dev 02:00 (floppy),
sector 0
Mar 3 07:03:14 home -- MARK -
```

Nos vamos a loguear en otra consola y retornemos a la terminal donde ejecutamos **tail**. Observamos que la última línea tiene que ver con el **loguin** producido. **Esto lo hace un comando sumamente interesante.**

Comando wc

La función del comando **wc** es imprimir el número de líneas, palabras y bytes de uno o varios archivos. En el caso de varios archivos hace también un resumen de los totales. Si no especificamos un fichero toma la entrada estándar.

Opciones más utilizadas del comando:

- **wc -l**: sólo cuenta líneas
- **wc -c**: sólo cuenta bytes
- **wc -w**: sólo cuenta palabras

Ejemplo:

```
equipo1:/tmp# wc /var/log/auth.log
269 3140 22269 /var/log/auth.log
```

Esto nos da tres columnas como se puede visualizar en el ejemplo anterior. La primera es la cantidad de líneas, la segunda es la cantidad de palabras y la tercera es la cantidad de caracteres.

Comando diff

Puede determinar las diferencias entre **test1** y **test2** al ejecutar el comando **diff**:

Si test1 contiene:

```
Está en un laberinto de
pequeños pasajes sinuosos
que son todos iguales.
```

Y test2 contiene:

```
Está en un laberinto de
pequeños pasajes sinuosos
que son todos diferentes.
```

El comando indicará las diferencias que ha encontrado, con el número de línea, y señalará (con un < y un >) en qué archivo se ha detectado la diferencia:

```
3c3 Los números de línea pertinentes
< que son todos iguales. La versión de test1
---
> que son todos diferentes. La versión de test2
```

Observemos cómo nos indica el comando **diff** que si elimina la línea **"<"** y agrega la línea **">",** los dos archivos serían iguales.

Comandos updatedb y locate

Comenzaremos ejecutaremos el comando **updatedb** para actualizar la base de datos. Es una buena idea poner este comando en **cron** como veremos posteriormente para que se ejecute automáticamente todos los días a una hora en la que el sistema no tenga mucha carga. Esto nos permitirá agilizar nuestras búsquedas.

A continuación, el uso de **locate**. Dicho comando nos ayuda a buscar archivos por nombre, y en todo el sistema de archivos, es mucho menos flexible que **find** (lo cual es perfectamente comprensible una vez que hemos comprendido su funcionamiento)

Ejemplos:

```
# locate so
[... muuucha salida ...]

# locate '*.so'
[... las librerías del sistema ...]
```

La similitud con **find** es que los criterios con wildcards deben ser entrecomillados con comillas simples.

Diferencias entre los comandos **locate** y **find**:

- no nos permite especificar directorio, busca en toda la ruta al archivo (no sólo en el nombre del archivo)
- si le pasamos un criterio sin wildcards, devolverá todos los nombres de archivo cuya ruta completa contenta ese criterio (nótese en el primer comando de ejemplo), a diferencia de **find** que sólo devolvería los archivos con ese nombre exacto

Comando find

El comando **find** es uno de los más poderosos en un sistema GNU/Linux.
Nos permite buscar de forma recursiva en un directorio a todos los archivos que cumplan ciertas condiciones. Las condiciones pueden estar relacionadas con el nombre de los archivos, el tamaño, los permisos, el tipo, las fechas de acceso y modificación, etc.

Este comando tiene la siguiente sintaxis:

```
find [camino] [opciones]
```

Opciones más utilizadas del comando:

- **find -name <expresión>**: permite especificar patrones para los nombres de los archivos a buscar.
- **find -type <tipo>**: permite indicar el tipo de archivo a buscar. Este puede ser **d** para directorios, **f** para archivos regulares, **l** para enlaces simbólicos, **b** para dispositivos de bloque, **c** para dispositivos de carácter, **p** para tuberías y **s** para sockets
- **find -size +/-<n>**: permite indicar el tamaño máximo y/o mínimo de los archivos a buscar. Por defecto el tamaño se expresa en bloques de 512 bytes, pero si se precede este por un carácter **c** se referirá a bytes, **k** a kilobytes, **w** a palabras de dos bytes y **b** a bloques
- **find -perm [+|-]<modo>**: permite referirse a aquellos archivos cuyos permisos sean exactamente modo, incluya todos los de modo (signo -) o incluya alguno de los de <modo> (signo +). **El valor de <modo> se expresa en forma numérica**
- **find -nouser/-nogroup**: permite referirse a aquellos archivos o directorios que no posean dueño o grupo asociado
- **find -exec <comando>;** : permite definir un comando a ejecutarse para cada resultado de la búsqueda. La cadena {} se sustituye por el nombre de los archivos encontrados. El caracter "**;** " permite indicar la finalización del comando.

Ejemplo 1:
Busca en el directorio /etc todos los archivos con extensión .conf

```
home:~# find /etc -name '*.conf'
```

Ejemplo 2:
Busca los archivos cuyo tamaño esté entre 10Mb y 20Mb

```
home:~# find / -size +10240k -size -20480k
```

Ejemplo 3:
Busca los directorios que posean el permiso t (sticky).

```
home:~# find -perm +1000 -type d
```

Ejemplo 4:
Busca en toda la jerarquía de directorios aquellos que no posean ni dueño ni grupo asociado

```
home:~# find / -nogroup -nouser
```

Ejemplo 5:
Busca todos los archivos que se nombren core y los borra interactivamente. Los signos \ se utilizan para proteger de la interpretación del shell.

```
home:~# find / -name core -exec rm -i \{\} \;
```

Comando whereis

Este comando busca archivos dentro de la variable de entorno **$PATH**. Es muy útil para buscar comandos ya que nos brinda información adicional. También nos muestra desde donde se configura si es que es configurable y donde está la documentación.

Este comando tiene la siguiente sintaxis:

```
whereis <busqueda>
```

Ejemplo:

```
home:~# whereis updatedb

updatedb: /usr/bin/updatedb /etc/updatedb.conf
/usr/share/man/man1/updatedb.1.gz
```

Comando which

Le pasamos como primer argumento un comando, y nos dirá, de los directorios que existen en **$PATH**, en cuál de ellos está.

Ejemplo:

```
home:~#which ls
/bin/ls
```

Comando grep

El comando **grep** (**Globally Regular Expressions Pattern**) busca patrones en archivos. Por defecto devuelve todas las líneas que contienen un patrón determinado en uno o varios archivos. Utilizando las opciones se puede variar mucho este comportamiento. Si no le pasamos ningún archivo como argumento hace la búsqueda en su entrada estándar.

Este comando tiene la siguiente sintaxis:

```
grep [opciones] <patrón> [archivos]
```

Opciones más utilizadas del comando:

- **grep -c**: devuelve sólo la cantidad de líneas que contienen al patrón
- **grep -i**: ignora las diferencias entre mayúsculas y minúsculas
- **grep -H**: imprime además de las líneas, el nombre del archivo donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un archivo
- **grep -l**: cuando son múltiples archivos sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes
- **grep -v**: devuelve las líneas que no contienen el patrón
- **grep -r**: busca en un directorio de forma recursiva
- **grep -n**: imprime el número de cada línea que contiene al patrón

Ejemplo 1:

Busca dentro de los archivos /**usr**/**share**/**doc** en forma recursiva las líneas que contienen la palabra **linux**.

```
home:~# grep -H -r linux /usr/share/doc
```

Ejemplo 2:

Busca dentro del archivo /**var**/**log**/**messages** las líneas que contienen la palabra **log** y nos dice cual es dicha línea.

```
home:~# grep -n log /var/log/messages
```

Ejemplo 3:

Busca al usuario llamado **fabián** dentro de /**etc**/**passwd**

```
home:~# grep -i fabian /etc/passwd
fabian:x:1001:1001:,,,:/home/fabian:/bin/bash
```

Ejemplo 4:

Busca la cantidad de veces que aparece la palabra en un archivo.

```
home:~# grep -c root /etc/group
```

Ejemplo 5:

Busca la coincidencia mess dentro de los archivos del directorio log y nos muestra quien lo contiene.

```
home:~# grep -l -r -i mess /var/log
```

Muchos de los comandos propuestos en esta clase son de uso cotidiano para el Administrador de red. En su práctica diaria encontrará su triunfo profesional. No deje de consultarle a su tutor todo lo que crea necesario.

CONSEJO: Practique mucho estos comandos ya que se usarán mucho a lo largo del curso así como en la operación y administración del sistema.