Retorno al Shell

Daniel Alejandro Benitez

Muchas son las razones que podemos escuchar y dar acerca de los beneficios de saber operar GNU/Linux prescindiendo de la interfaz gráfica, pero sin lugar a dudas todos coincidimos en que las utilidades que podemos ejecutar desde la línea de comandos no sólo son muy poderosas, sino que en ocasiones simplifican las tareas que se realizan desde el entorno gráfico. Esto puede parecer una exageración, pero los invito a que sean ustedes los que lo comprueben.

Una buena forma de empezar es aprendiendo como están organizados los datos en un sistema GNU/Linux. Al hacerlo estaremos en condiciones de entender cual es la utilidad de cada directorio, su importancia y contenido.

Estándar de la Jerarquía del sistema de archivos (FHS – Filesystem Hierarchy Standard)

FHS (Filesystem Hierarchy Standard) es un estándar que define los nombres, la ubicación y los permisos de muchos tipos de archivos y directorios. Se trata de un documento que es una referencia para todo sistema compatible FHS. Sin embargo da lugar a algunas extensibilidades en algunas areas mientras que a otras no las cubre. Esto da cierta libertad al momento de aplicarlo, de allí que pueden existir pequeñas diferencias entre las distribuciones GNU/Linux en lo que a estructura del FHS se refiere.

Cabe aclarar que en GNU/Linux todo es un archivo, esto incluye archivos, directorios, todo tipo de dispositivo de hardware, particiones de los discos, etc; es decir absolutamente todo.

Este estándar habilita:

- Al software para predecir la ubicación de archivos y directorios instalados.
- A los usuarios a predecir la ubicación de los archivos y directorios instalados.
- Para mayor información sobre el estándar, dirigirse a: *http://www.pathname.com/fhs/*.

A continuación vamos a ver una descripción de cada uno de los directorios más importantes y su contenido.

El directorio raíz (/)

El contenido del directorio raíz debe ser adecuado para arrancar, recuperar y/o reparar el sistema. Todo surge a partir del directorio raíz (/) y es el único directorio en el nivel superior del árbol jerárquico de archivos.

El directorio /boot/

El directorio `/boot/` contiene archivos estáticos requeridos para arrancar el sistema, tales como el kernel de Linux y los archivos de configuración necesarios para dicha tarea. Estos archivos son esenciales para que el sistema arranque correctamente.

El directoio /bin/

En este directorio se encuentran archivos binarios ejecutables del sistema. Entre ellos el shell *bash*, las utilidades de configuración del sistema, utilidades para el manejo de archivos, entre otros. Los archivos contenidos aquí son para ser usados por todos los usuarios.

El directoio /sbin/

Aquí encontramos unicamente los ejecutables esenciales para el arranque, recuperación y reparación del sistema. En su mayoría sólo pueden ser ejecutados por el usuario root.

El directoio /dev/

Este directorio contiene archivos que representan los diferentes dispositivos de hardware y que son esenciales el correcto funcionamiento del sistema.

```
administrador@kubuntu-704:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 107
model name      : AMD Athlon(tm) 64 X2 Dual Core Processor 4000+
stepping        : 1
cpu MHz         : 1592.194
cache size      : 512 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme pse tsc msr pae cx8 sep pge cmov mmx fxsr sse sse2 mmx
ext up lahf_lm cmp_legacy svm cr8legacy ts fid vid ttp tm stc [6]
bogomips        : 2894.45
clflush size    : 32

administrador@kubuntu-704:~$
```

Figura 1. En la figura vemos el contenido del archivo `/proc/cpuinfo` que muetra información relacionada con el procesador

El directorio /etc/

`/etc/` está reservado para los archivos de configuración del sistema y de las distintas aplicaciones. Aquí no deben colocarse binarios, que si bien antiguamente se lo hacía, deberían colocarse en /sbin/ o en /bin/ según corresponda. Como ejemplo veremos el contenido de algunos de sus subdirectorios.

En `/etc/X11/` hay archivos de configuración del servidor de ventanas X, entre ellos archivos importantes como `xorg.conf`. `/etc/skel/` proporciona una forma de asegurarnos que todo nuevo usuario creado tenga la misma configuración inicial. El directorio `/etc/` es uno de los más importantes.

El directorio /lib/

El directorio `/lib/` debería contener sólo las bibliotecas (libraries) necesarias para ejecutar los binarios alojados en `/bin/` y en `/sbin/`. Estas bibliotecas compartidas son particularmente importantes para arrancar el sistema y ejecutar comandos.

El directorio /media/

Contiene los subdirectorios utilizados como puntos de montaje para medios removibles, por ejemplo CD-ROMs entre otros.

El directorio /mnt/

Este directorio está reservado para sistemas de archivos montados temporalmente, tales como montajes de NFS. Recordemos que para los medios removibles utilizamos el directorio `/media/`. No deben instalar programas en este directorio.

El directorio /proc/

Se trata de un directorio que contiene archivos "*especiales*" que extraen información del kernel o se la envían. Debido a la gran variedad de datos y a la cantidad de diferentes formas en que este directorio se comunica con el kernel, sería necesario dedicarle todo un artículo sólo a él. Es por ello que se da una descripción somera destacando algunos puntos que pueden ser útiles.

El kernel de Linux tiene dos funciones primarias: controlar el acceso a los dispositivos físicos y establecer como y cuándo los procesos interactuarán con estos dispositivos.

En los archivos contenidos en `/proc/` se puede encontrar mucha información importante con detalles sobre el hardware del sistema y cualquier proceso que se esté ejecutando actualmente.

Cabe destacar que los archivos a los que hacemos referencia son de un tipo llamado archivo *virtual*, y es por esta razón que a menudo se hace referencia a `/proc/` como un sistema de archivos virtual.

Estos archivos virtuales poseen cualidades únicas, una de ellas es que la mayoría tienen un tamaño de 0 bytes. Sin embargo, cuando se visualiza el archivo utilizando por ej. el comando `cat` o `more`, puede mostrar gran cantidad de información.

El directorio /opt/

Este directorio proporciona un área para guardar habitualmente paquetes de software de una aplicación amplia y estática.

Un paquete que coloca archivos en `/opt/` crea un directorio con el mismo nombre del paquete. Estos archivos son guardados de forma tal que se evita que esten esparcidos por todo el sitema de archivos, dándole al administrador una forma fácil de determinar cuál es el rol que cumple cada archivo dentro de un paquete en particular.

El directorio /usr/

El directorio `/usr/` es para archivos que puedan ser compartidos a través de muchas máquinas. Habitualmente `/usr/` se monta como sólo lectura. y en una partición aparte.

El directorio /var/

El FHS requiere que Linux sea capaz de montar `/usr/` en sólo lectura, por ello cualquier programa que escriba archivos log o que necesite los directorios `spool/` o `lock/` debería escribirlos en el directorio `/var/`. El FHS especifica que `/var/` es para archivos de datos variables. Esto incluye archivos y directorios spool, datos de administración, de registro, correo local y archivos temporales entre otros.

El directorio /sys/

`/sys/` utiliza el nuevo sistema de archivos virtual *sysfs* específico del kernel 2.6. El directorio `/sys/` contiene información similar a la que se encuentra en `/proc/`.

El directorio /home/

En `/home/` encontramos los directorios de cada uno de los usuarios del sistema.

Cada usuario posee un directorio personal en donde guarda su configuración personal, documentos, etc. Sin embargo, según el estándar FHS, ningún programa debería basarse en esta ubicación.

La lista de directorios es mucho más extensa y aquí mencionamos sólo algunos a modo de introducción.

Otros directorios importantes que encontramos son:

- `/root/` es el directorio "*home*" del administrador del sistema, es decir el usuario root.
- `/tmp/` contiene archivos temporales.
- `/lost+found/` es un directorio de archivos perdidos.

Para más información podemos consultar el estándar FHS en: *http://www.pathname.com/fhs/.*

O desde una terminal ingresando la siguiente orden:

```
$man -hier
```

El intérprete de comandos, el Shell

El shell o intérprete de comandos es el mecanismo que permite la comunicación entre el Kernel y el usuario. Su principal función es la de tomar las "*ordenes*" que ingresa el usuario para enviárselas al Kernel.

Existen muchos shells para Linux, entre los más conocidos encontramos el shell Bourne (bash), el shell Berkley C (csh) y el shell Korn (ksh). Siendo *bash* el más difundido de los tres y el que se incluye como shell predeterminado en la mayoría de las distribuciones *GNU/Linux.*

** El Kernel o núcleo es el corazón de un sistema operativo.*

La Terminal

La Terminal es una forma de manejar el sistema prescindiendo de la interfaz gráfica. Para ello nos presenta una pantalla en modo texto donde muestra el "prompt" o indicador del sistema. Esto significa que el interprete de comandos está listo para recibir las órdenes del usuario. No obstante a la hora de introducirlas debemos tener en cuenta que debemos distinguir entre mayúsculas y minúsculas y que generalmente los comandos van en minúsculas.

Existen dos formas de acceder a una terminal, la primera es utilizando una aplicación que la emula dentro de un entorno gráfico como son Konsole, la terminal de KDE o xterm de GNOME. La otra forma es presionando `Ctrl + Alt + F1` con lo que accederemos a una terminal completamente en modo texto. Existen seis terminales a las que podemos acceder utilizando la combinación `Ctrl + Alt + F1 a Ctrl + Alt + F6`. Al utilizar esta forma mostrará el indicador de inicio de Sesión (login) donde es necesario ingresar el nombre de usuario y la contraseña. Introducidos estos datos se carga la configuración se la cuenta correspondiente y el sistema queda listo para utilizarse. Para regresar al modo gráfico presionar `Ctrl + Alt + F7`.

Generalmente el indicador del sistema (prompt) finaliza con el carácter $ cuando se utliza la terminal como usuario normal y con # cuando se utiliza como usuario root (administrador del sistema).

Trabajar con la terminal nos provee un mecanismo muy poderoso y rápido con el que se pueden realizar un sin fin de tareas.

Los comandos

Todos los comandos en GNU/Linux son externos al intérprete de comandos, esto significa que por cada uno de ellos tendremos un archivo alojado en un determinado directorio dependiendo de su función (ej. `/bin/, /sbin/, /usr/bin/)`.

En general todos los comandos pueden ser ingresados acompañados de una o más opciones. El formato general es :

```
comando [opciones] [archivo/
directorio]
```

A continuación damos una introducción a algunos de ellos. Muchos de los comandos poseen muchas más opciones de las que se describen aquí y es por ello que intentamos enumerar las más comunes, dando al lector lo necesario para que profundize en aquellos casos que lo considere necesario.

Nota: El signo $ que se antempone al comando en los ejemplos simboliza el prompt y no debe ser tipeado al probar los ejemplos.

Documentación y ayuda

El comando man se encarga de formatear y mostrar las páginas del manual del sistema. Es la manera más fácil y rápida de encontrar información acerca de cómo utilizar un comando. La forma más simple de utilizar man es ingresándolo seguido del comando que se desea consultar.

Ejemplo:

```
$ man date
```

Al darle ingreso a la orden anterior mostrará la ayuda correspondiente al comando date.

Para avanzar a la siguiente página de la ayuda bastará con presionar la tecla barra espaciadora o ENTER para desplazarnos línea a línea. Para salir hay que presionar la tecla q.

El comando man puede ir acompañado de opciones. Veamos algunos ejemplos:

- *-f:* utilizar esta opción equivale al comando whatis
- *-k:* hace que man cumpla la misma función que apropos
- *-h* o –help: muestra las opciones que pueden utilizarse con el comando man.
- *-a:* muestra secuencialmente todas las páginas que contienen el comando buscado.

Para más información sobre el uso de man y opciones podemos ingresar:

```
$ man man
```

La mayoría de los comandos cuentan con la opción -h (o también --help), que muestra en pantalla la sintaxis del comando junto con sus opciones principales. Esto es útil si ya se conoce la funcionalidad del comando pero no se recuerda alguna opción en particular:

```
$ man -h
```

Si se sabe el nombre del comando podemos utilizar whatis que devuelve una descripción breve del comando consultado.

whatis analizará la descripción de todos los comandos disponibles en busca de la cadena ingresada. En la información devuelta observaremos que algunos comandos van seguidos de un número o letra encerrado entre paréntesis, éstos hacen referencia al número de sección del manual. También puede ocurrir que devuelva más de un resultado para el mismo comando, lo que indica que hay más de un uso para dicho comando:

```
whatis equivale a  utilizar man -f.
```

Ejemplo:

```
$ whatis time
```

Si no se está seguro de lo que se desea consultar y al realizar la búsqueda con whatis o man -f no se ha obtenido nada útil, se puede intentar utilizar apropos. Este comando analiza las descripciones de la misma manera que whatis pero además presenta las correspondencias parciales con la cadena que se ingresó:

```
$ apropos time
```

apropos cumple la misma función que man *-k.*

Más ayuda

Info es el formato de documentación del proyecto *GNU*. Para acceder a esta documentación simplemente tipeamos el comando info desde la línea de comandos. Esto nos presentará un documento de texto con un índice de todos los documentos disponibles en este formato. Cada línea del documento precedida por un asterisco (*) identifica un hiperenlace (que tienen un comportamiento similar a los de *HTML*) que nos llevará a la sección correspondiente; para ello bastará con ubicarse sobre el item y presionar la tecla *ENTER*.

Podemos obtener ayuda acerca del uso de info presionando '?' y para salir de la aplicación presionando *'q'*. Otra forma de utilizar info es ingresarlo seguido del comando que se está buscando, de ésta manera irá directamente a la sección correspondiente.

Ejemplo:

```
$ info date
```

Documentación adicional

Generalmente en todas las distribuciones se incluye un directorio con documentos que pueden ser muy útiles. Su ubicación debería ser `/usr/share/doc/` , de no ser así probar `con /usr/doc/`.

En algunos casos la información contenida se reduce a un simple archivo 'leame' ('readme') mientras que en otros podemos encontrar información realmente útil.

```
administrador@kubuntu-704:~$ ls -l /usr/share/doc/bash
total 384
-rw-r--r-- 1 root root    799 2007-04-11 01:32 BUGS.bash_completion
-rw-r--r-- 1 root root  41158 2007-04-11 01:32 changelog.bash_completion.gz
-rw-r--r-- 1 root root  25656 2007-04-11 01:25 changelog.Debian.gz
-rw-r--r-- 1 root root 146846 2006-10-03 15:16 changelog.gz
-rw-r--r-- 1 root root  66631 2006-09-26 19:24 CHANGES.gz
-rw-r--r-- 1 root root   4926 2006-08-29 16:16 COMPAT.gz
drwxr-xr-x 2 root root   4096 2007-04-17 07:19 completion-contrib
-rw-r--r-- 1 root root  17613 2007-04-11 01:25 copyright
-rw-r--r-- 1 root root   1322 2007-04-11 01:25 FAQ
-rw-r--r-- 1 root root    727 2007-04-11 01:25 inputrc.arrows
-rw-r--r-- 1 root root   2927 1999-02-17 22:13 INTRO.gz
-rw-r--r-- 1 root root  16966 2006-07-06 15:23 NEWS.gz
-rw-r--r-- 1 root root   2961 2006-07-06 15:32 POSIX.gz
-rw-r--r-- 1 root root   1105 2007-04-11 01:25 README.abs-guide
-rw-r--r-- 1 root root   7355 2007-04-11 01:32 README.bash_completion.gz
-rw-r--r-- 1 root root   3037 2007-04-11 01:25 README.commands.gz
-rw-r--r-- 1 root root   3159 2007-04-11 01:32 README.Debian.gz
administrador@kubuntu-704:~$
```

Figura 2. Aquí vemos el resultado de ejecutar el comando del ejemplo anterior

Cómo navegar por el Sistema de Archivos

Al comienzo de éste artículo describimos algunos de los directorios más importantes del sistema. A continuación veremos algunos comandos que sirven para desplazarse a traves de ellos. Pero primero hay dos conceptos que debemos tener en claro: qué es una *ruta de acceso* y qué son las rutas de acceso *absosultas* y *relativas*. De una manera simplificada, una *ruta de acceso* es el trayecto más corto de directorios por el que tenemos que desplazarnos desde el directorio raíz para llegar a un directorio o archivo en el que se está trabajando en ese instante. Cada nombre de directorio se separa con el carácter /. Todas las rutas de acceso *absolutas* deben comenzar con el carácter / es decir referencian al directorio ráiz, mientras que las *relativas* no; y se utilizan para indicar que el directorio referenciado es un subdirectorio del actual.

¿En que directorio me encuentro?: pwd

El comando `pwd` (present working directory – directorio actual de trabajo) sirve para saber en qué directorio se encuentra el usuario en ese momento. Al introducirlo sabrá exactamente donde se encuentra ubicado.

Ejemplo:

```
$ pwd
/home/usuario
```

Esto es principalmente útil cuando en el prompt no muestra la ruta de acceso.

Cómo cambiar de directorio: cd

El comando `cd` permite cambiar del directorio actual a otro especificado, haciendo que se abandone el actual y que pase a serlo aquel que se especificó en el comando `cd`.

Ejemplos:

```
$ pwd
/home/usuario
```

Si ejecutamos:

```
$ cd /usr/share/doc
```

Y luego hacemos:

```
$ pwd
/usr/share/doc
```

Comprobamos que luego de ejecutar el comando `cd` el directorio actual pasó a ser `/usr/share/doc`.

Generalmente al abrir una terminal se parte del directorio home del usuario con el que está trabajando.

Para facilitar el trabajo, GNU/Linux no obliga a recordar la ruta de acceso al directorio personal, sino que permite utilizar el carácter ~ para hacer referenciarlo. De esta forma cuando se esté en un directorio que no sea el propio, podremos regresar al home rapidamente tipeando.

Ejemplo:

```
$ cd ~
```

Existe otra manera más eficaz de cambiar de directorio y es utilizando los comandos `pushd` y `popd`.

`pushd` cambia al directorio especificado colocando el actual en la pila de directorios (una pila es un tipo de almacenamiento de datos). Luego se pueden hacer todos los cambios de directorios necesarios y cuando se desee volver al primer directorio bastará con ingresar el comando `popd`.

Ejemplos:

```
$ pwd
/home/usuario
$ cd /usr/share/doc
$ pwd
/usr/share/doc
$ cd /etc
$ pwd
/etc
$ popd
$ pwd
/home/usuario
```

Al hacer `popd` regresa al directorio desde donde se ejecutó `pushd` y luego la pila de directorios se limpia.

Nota: el comando `pwd` no es necesario utilizarlo, en el ejemplo se lo incluye sólo para hacerlo más claro.

Cómo mostrar el contenido de un directorio: ls

`ls` sirve para obtener un listado de los archivos de un directorio. Si no se indica lo contrario mostrará el contenido del directorio actual. Sintaxis:

```
ls [opciones] [archivo..]
```

A continuación se verán algunas las opciones más comunes:

- *-C* - Lista los archivos en columnas, ordenados verticalmente.
- *-F* - Añade detras cada nombre de directorio un '/', y detras cada nombre de un ejecutable un '*'.
- *-R* - Lista recursivamente los subdirectorios encontrados.
- *-d* - Lista nombres de directorios como otros archivos, en vez de listar sus contenidos.
- *-l* - Permite visualizar más información acerca de los archivos mostrados. Al principio puede parecer un poco complicado pero a continuación damos algunos conceptos para facilitar su comprensión.

En la primer línea del listado muestra el total de elementos del directorio en cuestión.

Luego vienen las líneas en las que se detalla el contenido del directorio mostrándose una línea por cada elemento.

Los primeros 10 carácteres que aparecen en la parte izquierda corresponden a al información sobre los atributos de cada archivo. El primer carácter determina si se trata de un archivo (en cuyo caso el primer será una a) o directorio (en este caso el carácter es una `d`). Pueden haber casos donde el primer carácter sea una l que indica que se trata de vínculos o nombres de archivos alternativos para otros archivos o directorios. Los nueve carácteres restantes estan agrupados en tres grupos de tres letras que indican si el archivo es de lectura, escritura y ejecutable para el usuario, para el grupo y para del resto de los usuarios.

Luego de éste grupo de 10 carácteres aparece un número que se refiere a los vínculos del archivo. En general es un dato poco importante para la mayoría de los usuarios.

A continuación aparece el nombre del propietario y el nombre del grupo al que pertenece el archivo o directorio. El siguiente dato es la cantidad de espacio en disco que ocupa el archivo o directorio. En el caso de tratarse de un directorio, no se muestra el espacio que ocupan los archivos contenidos en él sino el espacio que ocupa el archivo de datos que controla el directorio. Este valor se mide en bytes. Despúes del tamaño viene la fecha y la hora en que se modificó por última vez.

Para finalizar, en último lugar aparece el nombre del archivo.

- *- h* : Utilizada con -l muestra los bytes ocupados por los arhivos en otra unidad (Ej. 14K, 13M, 4G). Equivale a utilizar la opción `--human`-readable.
- *- a* :De manera predeterminada en la lista mostrada por ls no se incluyen todos los archivos a menos que especifique la opción `-a` (all - todos) , esto hace que en

el listado aparezcan los archivos cuyos nombres empiecen con un punto ' . ' (por lo general estos nombres se utilizan para archivos de control y configuración).

El comando ls tiene más opciones y pueden consultarse haciendo:

```
$ man ls
```

Ejemplos:

```
$ pwd
/home/usuario
$ ls -l -h
$ ls -t
$ ls -l /usr/share/doc/bash
```

Búsqueda de archivos

Para localizar un archivo es necesario conocer algo sobre él. Puede ser parte del nombre, la fecha de creación o el tamaño.

El comando find

Si disponemos de algunos de los datos antes mencionados (nombre, fecha o tamaño), se puede utilizar el comando `find`, que se encargará de analizar todo el sistema de archivos en busca de aquellos que se correspondan con el criterio de búsqueda especificado.

Sintaxis:

```
find [path] [opciones] [cadena a
buscar]
```

Path: es la ruta de acceso desde la cuál se comenzará la búsqueda.

Opciones:

- *-h , --help* : muestra la ayuda del comando.
- *-name* : Se realiza una búsqueda por nombre. Distingue entre mayúsculas y minúsculas.
- *-iname* : igual que *-name* pero NO se distingue entre mayúsculas y minúsculas.
- *-ctime n* : Muestra los archivos cuyo status haya sido cambiado en el tiempo especificado en el sufijo *n* y está expresado en días.
- *-print* : indica que muestre por pantalla el resultado de la búsqueda. No suele se necesario especificar esta opción.
- *-type* c : Donde c puede ser reemplazada por d si es un directorio, f (file) si se trata de un archivo o por l si es link simbólico. De esta manera se visualizarán unicamente los archivos del tipo especificado.
- *-size n*: Para buscar archivos de un tamaño n. Puede ir seguido de alguno de los siguientes sufijos:
  - c para bytes.
  - K para kilobytes.
  - M para megabytes.
  - G para gigabytes.
- *-user usuario* : Restringe la búsqueda a los archivos del usuario especificado en '*usuario*'.
- *-group grupo* : Restringe la búsqueda a los archivos del grupo especificado en '*grupo*'.

Ejemplos:

```
$ find /usr/share/doc/ -name '*.html'
$ find / -type f -name 'sound*'
$ find / -ctime 2
$ find ~/ -ctime +5
$ find /usr/share/doc -size 1024k
```

El comando locate

Este comando es muy fácil de usar. Para ejecutarlo simplemente hay que incluir el nombre del archivo a buscar. *Locate* se ejecuta de forma muy rápida debido a que no busca el archivo por todo el sistema de archivos sino que lo hace dentro de una base de datos dedicada a este fin.

Sintaxis:

```
locate [archivo]
```

La desventaja que tiene el comando *locate* es que en caso de no estar actualizada la base de datos no encontrará lo que se busca a pesar de existir. En ese caso habrá que utilizar el comando *updatedb* para actualizarla.

Ejemplo:

```
$ locate soundcard
```

Como resultado de la búsqueda anterior devolverá todos los nombres de archivos (y directorios) que contienen la palabra *soundcard.*

Cómo buscar una cadena de carácteres dentro de un archivo

Sin dudas el comando `grep` es uno de los más potentes y útites del sistema operativo.

`grep` recorre el cuerpo de los archivos en busca de la palabra o cadena especificada mostrando las líneas que concuerdan con el patrón introducido.

Sintaxis:

```
grep [opciones] [cadena a buscar]
archivo...
```

Opciones principales :

- *-c:* Muestra la cantidad de líneas que satisfacen la condición.
- *-i*: no se distinguen entre mayúsculas y minúsculas.
- *-l*: se muestran los nombres de los archivos que contienen líneas buscadas.
- *-n*: cada línea devuelta es precedida por su número dentro del archivo.
- *-s*: no se muetsran los mensajes que indican que un archivo no existe o no es accesible.
- *-v*: se muestran sólo las líneas que NO satisfacen el criterio de búsqueda.

Ejemplos:

```
$grep -c 'GNU' /usr/share
   /doc/bash/copyright
27
```

Al ejecutar el comando anterior devuelve la cantidad de líneas (27) que contienen la cadena 'GNU':

```
$grep -n 'gnu' /usr/share
   /doc/bash/copyright
7: bash  ftp.gnu.org:
   /pub/gnu/bash
      /bash-3.1.tar.gz
```

En este ejemplo muestra que la línea que contiene la cadena *´gnu*' precedida por el número de la misma (7):

```
$grep -n 'GNU' /usr/share
   /doc/bash/copyright
```

Cómo trabajar con archivos y directorios

Elimación de archivos y directorios: rm.

El comando *rm* se utiliza para eliminar tanto archivos como directorios. Por defecto rm no elimina directorios.

Sintaxis:

```
rm [opciones] archivo
```

Opciones principales:

- *-f*: No pide confirmación. No escribe mensajes de diagnóstico.
- *-i* : Pide confirmación de borrado por cada archivo.
- *-r* , -R o *--recursive* : Borra recursivamente directorios.

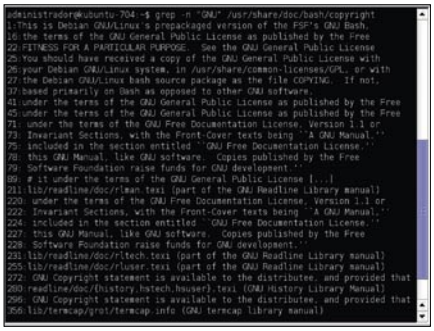

Figura 3. Al ejecutar el ejemplo anterior visualizaremos en pantalla algo similar a lo mostrado en está figura

Ejemplo:

```
$ rm -i *.png
```

Al dar entrada al comando anterior, borrará del directorio actual todos los archivos cuya extensión sea *.png* solicitando confirmación por cada uno de ellos. De no tener los permisos necesarios mostrará el mesaje '*permiso denegado*' y el archivo no será borrado.

Para evitar equivocaciones es conveniente utilizar la opción -i siempre que se desee eliminar más de un archivo.

Cómo crear un directorio: mkdir

Los directorios son muy útiles para organizar los archivos. Para crear un directorio debemos utilizar el comando `mkdir`.

Sintaxis:

```
mkdir [opciones] directorio..
```

Opciones:

- *-m modo, --mode=modo*: Crea los directorios con los permisos especificados en 'modo'
- *-p, --parents* : Crea los directorios padre que falten para cada argumento directorio. No hace caso de argumentos que correspondan a directorios existentes. Por ejemplo, si existe un directorio */prueba* y se ingresa `mkdir /prueba` da error, pero `mkdir -p /prueba` no lo da.
- *-verbose*: Muestra un mensaje para cada directorio creado. Esta opción es útil cuando se utiliza con --parents.

Ejemplos:

```
$ pwd
/home/usuario
$ mkdir prueba
```

En el ejemplo anterior se crea el directorio 'prueba' a partir del directorio actual, es decir que '*prueba*' pertenecerá al directorio '*usuario*' que a su vez pertenece a '*home*' y éste al raíz.

```
$ mkdir -p prueba/archivos_pdf
```

En este ejemplo se crea el directorio '*archivos_pdf*' dentro del directorio '*prueba*' que pertenece al actual; pero en caso de no existir este último también lo creará para luego crear '*archivos_pdf*'.

Recordar que el comando `pwd` no es necesario ingresarlo y simplemente se lo introduce para hacer más claro el ejemplo.

```
$ mkdir prueba varios
```

Al darle entrada al comando creará los directorios '*prueba*' y '*varios*' ambos tendrán la misma jerarquía y pertenecerán al directorio actual.

Borrar un directorio: rmdir (remove directory)

Este comando borra uno o más directorios siempre y cuando estos estén vacíos.

Sintaxis:

```
rmdir [opciones]
   nombre_directorio...
```

nombre_directorio: es el nombre del directorio a eliminar. En caso de que se quiera eliminar más uno, cada nombre deberá ir separado por un espacio.

Opciones:

- `-p` : si el directorio incluye más de un directorio, borra todos comenzando desde el de nivel inferior hasta llegar al de nivel superior.

Copia de archivos: cp

Este comando permite copiar un archivo en otro nuevo archivo destino, o bien copiar uno o más archivos en un único directorio destino.

Sintaxis:

```
cp [opciones] archivo
   _origen archivo_destino
```

Opciones:

- *-i*, --interactive: Modo interactivo. Pregunta si se desea sobreescribir un archivo destino existente.
- *-p*: Preserva los permisos, el propietario y el grupo de los archivos originales, más la fecha y hora de última modificación y el de último acceso.
- *-f, --force*: Elimina los archivos de destino que ya existan sin pedir confirmación.
- *-u, --update*: Copia un archivo solamente cuando el origen es más nuevo que el destino o cuando el destino no existe.
- *-v, --verbose*: Muestra el nombre del archivo que se está copiando a medida que va procesando.
- *-b, --backup*: Hace copias de respaldo de archivos que están a punto de ser sobreescritos o borrados.
- *-r*: Copia directorios recursivamente. Todos los nombres de archivo de origen serán tratados como directorios y se copiarán recursivamente en el directorio destino.

Ejemplos:

```
$ cp *.pdf documentos
```

En el ejemplo anterior copia todos los archivos de extensión .pdf en el directorio *documentos* que pertenece al directorio actual:

```
$ cp imagen.png lafoto
```

Aquí se copia el archivo *imagen.png* en otro llamado *lafoto*. Si existe un directorio llamado *lafoto* el archivo se copiará dentro de ese directorio:

```
$ cp imagen.png
   imagen2.png
```

En este ejemplo se copiará el archivo llamado *imagen.png* a otro *imagen2.png*. Si el archivo destino existe será reemplazado por el nuevo:

```
$ cp -r /var/log
   /samba prueba
```

Al ejecutar el ejemplo anterior, se crearía un directorio *samba* en el directorio *prueba* y se copiaría el contenido del directorio */var/log/samba* en el nuevo directorio *prueba/samba*.

Cómo mover archivos y directorios: mv

El comando *mv* permite mover o renombrar archivos o directorios.

Si el último argumento nombra a un directorio existente, mv mueve cada uno de los archivos a ese directorio. De lo contrario, si sólo se dan dos nombres de archivos, renombra el primero al segundo. Es un error que el último argumento no sea un directorio y se den más de un archivo como origen.

Sintaxis:

```
mv [opciónes] archivo
   _origen... archivo_destino
```

Opciones:

- *-i*: Pide confirmación cuando el destino existe.
- *-f, --force*: Borra los ficheros de destino existentes sin preguntar al usuario.
- *-v, --verbose*: Muestra el nombre del archivo que se está moviendo a medida que va procesando.

Ejemplos:

```
$ mv foto1.png
   foto2.png   documentos
```

En este ejemplo mueve los archivos *foto1.png* y *foto2.png* al directorio *documentos* conservando los nombres originales de los archivos:

```
$ mv *.pdf documentos
```

Aquí se mueven todos los archivos de extensión *.pdf* al directorio *documentos* que pertenece al directorio actual:

```
$ mv foto1.png foto.png
```

En el ejemplo anterior se le cambia el nombre al archivo *foto1.png* de manera que se su nuevo nombre sea *foto.png*:

```
$ mv documentos
   mis_documentos
```

Si el nombre origen es un directorio, se cambiará el nombre por el especificado en destino.

En nuestro caso, y asumiendo que el directorio *documentos* existe, se le cambia el nombre de manera que su nuevo nombre sea *mis_documentos*.

Cómo visualizar archivos: cat

Una forma fácil de visualizar el contenido de archivos es utilizando el comando `cat`. También puede utilizarse junto con un operador de redireccionamiento para crear un archivo que sea el resultado de varios archivos concatenados. `cat` muestra el contenido de los archivos cuyos nombres se introduzcan, uno a continuación del otro sin hacer pausas durante el despliegue de la información. Esto puede ser un problema cuando se trata de archivos muy extensos.

Sintaxis:

```
cat [opciones]
   nombre_de_archivo
```

Opciones:

- *-E*: muestra el signo $ al final de cada línea.
- *-n, --number*: numera todas las líneas mostradas.

Ejemplos:

```
$cat mitexto
```

Muestra en pantalla el contenido del archivo *mitexto*:

```
$cat documento1  documento2
```

Muetra en pantalla, secuencialmente y según el orden especificado, el contenido de los archivos indicados:

```
$cat documento1
  documento2 > documento3
```

El contenido de los archivos especificados (*documento1 y documento2*) es grabado en *documento3*:

```
$cat documento1
   documento2 >> documento3
```

El contenido de *documento1* y *documento2* es añadido al final de *documento3*:

```
$cat >mitexto
```

Acepta lo que se introduce por el teclado y lo graba en *mitexto* (se crea *mitexto*). Para finalizar se debe presionar `<ctrl>d`:

```
$cat /etc/passwd
```

En el ejemplo anterior muestra el contenido del archivo *passwd* que se encuentra en el directorio */etc*.

Visualización de archivos por páginas: more y less

Como dijimos, el comando cat sirve para visualizar el contenido de archivos comodamente cuando no son muy grandes. Pero existen otros comandos que han sido especialmente creados para visualizar archivos extensos desde la línea de comandos, estos son: more y less.

Para utlizar more, solo hay que ingresar el comando seguido del nombre del archivo.

Sintaxis:

```
more [opciones] nombre_archivo
```

Opciones:

- *-num*: Esta opción especifica un entero que indica el tamaño de la pantalla (en líneas).
- *-p*: No realizar desplazamiento. En lugar del desplazamiento, limpia toda la pantalla y para después mostrar el texto.
- *-d*: more mostrará el mensaje "[Press space to continue, '*q*' to quit.]" (pulsa espacio para continuar, 'q' para salir) y en vez de emitir un pitido cada vez que se pulse una tecla ilegal mostrará "[Press 'h' for instructions.]" (Pulse 'h' para obtener instrucciones).
- *-s*: Reducir múltiples líneas en blanco a una.

Las siguientes son opciones que se utilizan dentro del comando more, es decir cuando se está visualizando el contenido de un archivo:

- Barra espaciadora: avanza a la página siguiente.
- Tecla q: sale del programa more y vuelve a la línea de comandos.
- Tecla s: Pasa a la siguiente línea de texto. Se utiliza desplazarse por el texto lentamente.
- Tecla f: Avanza toda una página del archivo.
- Tecla / <patrón>: Busca y avanza hasta la cadena de texto especificada por patrón.
- Tecla b: retrocede una página del archivo.
- Tecla ? o h: muestra la ayuda para el comando more.

El comando less

less al igual que *more*, permite visualizar el contenido de un archivo de texto aunque es un visor más versátil y moderno que more. Una de sus principales diferencias es la capacidad que tiene less para moverse por el interior de los archivos que se estan visualizando.

Sintaxis:

```
less [opciones] nombre_archivo
```

A continuación se dan algunos comandos que permitirán moverse por el archivo visualizado:

- Barra espaciadora: avanza a la página siguiente.
- *Tecla b*: retrocede una página.
- *Tecla q*: Cierra el programa less y vuelve a la línea de comandos.

- Flecha arriba / abajo: retrocede o avanza una línea.
- */<patrón>*: Busca y avanza hasta la cadena encontrada. Esta búsqueda se realiza desde la posición en que se encuentre situado en ese momento hasta el final del archivo.
- *?<patrón>*: Busca hacia atrás, desplazándose hasta la cadena encontrada.
- *Tecla h*: Muestra la ayuda para el comanando.

Visualizar partes de un archivo: head y tail

head muestra sólo las 10 primeras líneas de cada archivo especificado.

Sintaxis:

```
head [opciones] archivo1..
```

Opciones:

- `-c, --bytes=[-]N` : muestra los primeros N bytes de cada archivo especificado. Si se antepone el signo ' - ' al número muestra los últimos N bytes.
- `-n, --lines=[-]N`: muestra la cantidad de líneas especificadas por el parámetro N. Si N está precedido del signo ' - ' , mostrará las últimas líneas del archivo.

El comando *tail* es inverso a *head*, es decir en lugar de mostrar las primeras líneas de cada archivo, muestra las 10 del final.

Ejemplos:

```
$head /usr/share/doc
   /grub/AUTHORS
```

En el ejemplo se visualizan las 10 primeras líneas del archivo AUTHORS:

```
$head -n15 /usr/share  /doc/grub/
AUTHORS
```

Como en el ejemplo anterior, pero en lugar de mostrar 10 líneas mostrará 15.

Sintaxis:

```
tail [opciones] archivo1..
```

Opciones:

- `-c, --bytes=N` : muestra los últimos N bytes de cada archivo especificado.
- `-n, --lines=N`: muestra la cantidad de líneas especificadas por el parámetro N.

Ejemplos:

```
$tail -n15 /usr/share
   /doc/grub/AUTHORS
```

En el ejemplo se visualizan las 15 últimas líneas del archivo AUTHORS.

Cómo contar líneas, palabras y carácteres: wc

Podemos utilizar el comando wc para saber de forma rápida y sencilla cuantas líneas, palabras y carácteres tiene una archivo. Si se especifica más de un archivo muestra también el total de todas las líneas.

Sintaxis:

```
wc [opciones] archivo
```

Opciones:

- *-c*: muestra sólo la cantidad de bytes.
- *-l*: muestra sólo la cantidad de líneas.
- *-m*: muestra sólo la cantidad de carácteres.
- *-w*: muestra sólo la cantidad de palabras.
- *-L*: muestra sólo la longitud de la palabra más larga.

Ejemplos:

```
$wc /usr/share/doc/grub/AUTHORS
  54   278  2056 /usr/share/doc/grub/
AUTHORS
```

En el ejemplo muestra la cantidad de líneas (54), palabras (278) y carácteres (2056) que contiene el archivo AUTHORS.

```
$wc -w /usr/share/doc/grub/AUTHORS
278 /usr/share/doc/grub/AUTHORS
```

Al darle entrada al comando se visualizará unicamente la cantidad de palabras que hay en el archivo especificado. 278 en este caso.

Resumen de otros comandos útiles

file: Este comando realiza una serie de comprobaciones en un archivo para tratar de identificar su tipo. Tras su ejecución este comando muestra el tipo del archivo e información acerca del mismo.

Ejemplo:

```
$ file /usr/share/doc
   /grub/AUTHORS
/usr/share/doc/grub
   /AUTHORS: ASCII English text
```

date: Muestra por pantalla el día y la hora. Si se tiene privilegios de superusuario, también permite cambiar la fecha y hora actuales.

Uptime: Devuelve la hora actual, el número de usuarios que han iniciado una sesión, el tiempo que lleva el sistema funcionando y la cantidad de carga que ha tenido que soportar el sistema. *cal*: Muestra el calendario en formato tradicional.

Ejemplos:

```
$cal 2007
```

Mostrará el calendario para todo el año 2007.

```
$cal 04 1998
```

Muestra el calendario de abril de 1998.

```
$cal -3
```

Muestra el calendario del mes actual, el anterior y el próximo. *who* Muestra los usuarios que tienen una sesión iniciada en el sistema junto con el nombre de la terminal de control y la fecha y hora en que iniciaron y la IP desde la que están conectados. *clear* Este comando limpia la pantalla de la consola. *free*: muestra la cantidad de memoria libre y usada del sistema. *free* muestra la cantidad total de memoria fisica y de intercambio (swap) presente en el sistema, así como la memoria compartida y los buffers usados por el nucleo .

Conclusión

GNU/Linux dispone de un gran número de utilidades y debido a esto es imposible explicar cada una de ellas en este artículo. Por ello, se han tratado de incluir principalmente comandos que pueden ser útiles para aquellos usuarios que estan haciendo sus primeras incursiones en GNU/Linux, pero al mismo tiempo se han incluido conceptos que consideramos les permitirán construir una buena base que sirva de sustento para los conocimientos que vayan adquiriendo. ■

Sobre el autor

Daniel A. Benitez es desarrollador de software desde hace 20 años, ha trabajado en distintas tecnologías y sistemas operativos. Su primer contacto con Linux fue en el año 1995, y a partir del 2000 lo adoptó como su sistema operativo. Actualmente brinda consultoría a distintas empresas sobre desarrollo de software, TI y seguridad informática. Se le puede escribir a la dirección: *danielbenitez.itpro@sion.com*