Tutorial BASH n°4
Comandos avanzados

INDICE

1. Comandos ps 3
2. Comando sed 4
2.1. Uso básico de sed 4
2.2. Utilizar Expresiones Regulares 6
2.3. Ejemplos de uso de sed 8
3. Comando du 9
4. Comando df 10

1. Comandos ps

El comando ps es el que permite informar sobre el estado de los procesos. **ps** está basado en el sistema de archivos /proc, es decir, lee directamente la información de los archivos que se encuentran en este directorio. Tiene una gran cantidad de opciones, incluso estas opciones varían dependiendo del estilo en que se use el comando. Estas variaciones sobre el uso de ps son las siguientes:

- Estilo **UNIX**, donde las opciones van precedidas por un guion -
- Estilo **BSD**, donde las opciones no llevan guion
- Estilo **GNU**, donde se utilizan nombres de opciones largas y van precedidas por doble guion –

Sea cual sea el estilo utilizado, dependiendo de las opciones indicadas, varias columnas se mostrarán en el listado de procesos que resulte, estas columnas pueden ser entre muchas otras, las siguientes (y principales):

p o **PID**	Process ID, número único o de identificación del proceso.
P o **PPID**	Parent Process ID, padre del proceso
U o **UID**	User ID, usuario propietario del proceso t o TT o
TTY	Terminal asociada al proceso, si no hay terminal aparece entonces un '?'
T o **TIME**	Tiempo de uso de cpu acumulado por el proceso c o CMD. El nombre del programa o comando que inició el proceso
RSS	Resident Sise, tamaño de la parte residente en memoria en kilobytes
SZ o **SIZE**	Tamaño virtual de la imagen del proceso
NI	Nice, valor nice (prioridad) del proceso, un número positivo significa menos tiempo de procesador y negativo más tiempo (-19 a 19)
C o **PCPU**	Porcentaje de cpu utilizado por el proceso
STIME	Starting Time, hora de inicio del proceso
S o **STAT**	Status del proceso, estos pueden ser los siguientes
	R runnable, en ejecución, corriendo o ejecutándose
	S sleeping, proceso en ejecución pero sin actividad por el momento, o esperando por algún evento para continuar
	T sTopped, proceso detenido totalmente, pero puede ser reiniciado
	Z zombie, difunto, proceso que por alguna razón no terminó de manera correcta, no debe haber procesos zombies
	D uninterruptible sleep, son procesos generalmente asociados a acciones de IO del sistema
	X dead, muerto, proceso terminado pero que sigue apareciendo, igual que los Z no deberían verse nunca

2. Comando sed

Sed es un editor de textos en línea de comando. Toma como entrada uno o más ficheros, los procesa según los parámetros especificados, y finalmente escribe el resultado en la salida estándar.

Sed procesa su entrada línea a línea, lo cual quiere decir que no es apto para alterar ficheros binarios, ni ficheros con líneas extremadamente largas. Su funcionamiento puede describirse de la siguiente manera: Veamos algunos ejemplos de uso de sed.

Normalmente un administrador de sistemas tiene que modificar archivos de configuración, tablas y otros archivos de texto repetidamente para realizar cambios y ajustes. Linux trae una gran cantidad de editores de texto tanto para la línea de comandos (emacs, vim, nano, etc) como gráfico (gedit, kate, etc), este tipo de editores son útiles cuando vamos a editar un par de textos. Pero ¿qué pasa cuando queremos modificar una gran cantidad de archivos o queremos automatizar modificaciones a través de un script?, en este caso es que entra el comando sed.

La versión GNU de sed viene incluida por defecto en la gran mayoría (sino en todas) las distribuciones de GNU/Linux.

2.1. Uso básico de sed

El formato básico del comando sed es:

sed opciones [script] [archivo(s)]

Características principales de sed:

- Si no se le da ningún archivo sed toma la entrada estandar (stdin).
- sed filtra línea por línea, no letra por letra.
- La salida por defecto de sed es la salida estándar (stdout).

Para borrar una línea hacemos lo siguiente:

sed 'nº_de_línead' fichero

Podemos indicar un número de línea concreto. Por ejemplo:

sed '1d' fichero

Podemos indicar un intervalo de líneas a borrar. Por ejemplo:

sed '3,5d' fichero

También podemos indicar que queremos borrar desde una determinada línea en adelante:

sed '3,$d' fichero

Otro ejemplo útil es borrar las líneas en blanco de un fichero:

sed '/^$/d' fichero

A la hora de borrar, también podemos especificar una cadena, de tal forma que el comando borrará todas las líneas que contengan esa cadena. Ejemplo:

cat fichero | sed '/^[ ]*$/d' > ficherodestino

Lo anterior borrará todas las líneas en blanco de fichero.

Otro de los usos interesantes es borrar los espacios al comienzo de cada línea:

sed 's/^ *//g' fichero

O borrar los espacios al final de cada línea:

sed 's/ *$//g' fichero

Otro de los usos más interesantes de sed es sustituir cadenas. Podemos sustituir una cadena por otra de la siguiente manera:

sed 's/cadena1/cadena2/' fichero

Al ejecutar el comando anterior, se sustituye la primera cadena que encuentra por la segunda. Pero, si lo que queremos es sustituir todas las cadenas que encuentre, en cada una de las líneas, añadimos el parámetro g:

sed 's/cadena1/cadena2/g' fichero

Por otra parte, también podemos hacer que sustituya la cadena1 por la cadena2 en un número de línea concreto:

sed '5 s/USUARIO/usuario/g' fichero

Con cadenas de texto normales la cosa es sencilla, pero al que más y al que menos le resulta complicado cuando lo que hay que sustituir son caracteres especiales como el tabulador: \t o el carácter de nueva línea: \n. Pero veamos como tampoco es complicado: Imaginemos que tenemos un fichero con campos en los que el separador es el tabulador y queremos sustituir este carácter separador por otro carácter separador, como por ejemplo el punto y coma (;). Lo haremos de la siguiente manera:

sed 's/\t/;/g' fichero

Para borrar línea o conjunto de líneas se puede ejecutar sed de la siguiente forma:

sed -e '1d' texto1

En el terminal, se debería ver el texto sin la primera línea. La opción -e que es para editar, luego colocamos '1d' que le dice a sed que tome la primera ocurrencia de línea y la borre (d=delete) y luego le pasamos el nombre del archivo texto1. Hay que tener en cuenta que sed no modifica el archivo original como tal, simplemente nos muestra la modificación en la salida estándar (stdout), si queremos guardar la modificación en un nuevo archivo debemos redireccionar la salida.

Ahora que pasa si queremos borrar más de una línea, pues afortunadamente sed permite que le demos rangos, si queremos borrar de la línea 2 a la línea 5 ejecutamos: **sed -e '2,5d' texto1**

Este comando es muy parecido al anterior solo que en esta oportunidad le estamos dando un rango de la línea 2 a la línea 5. Hay que considerar que sed toma en cuenta las líneas en blanco también.

2.2. Utilizar Expresiones Regulares

Como casi todos los comandos en Linux sed también soporta expresiones regulares las cuales son muy útiles para filtrar patrones, al utilizar estas expresiones regulares es que podemos utilizar sed a su máximo potencial.

Vamos a ver un ejemplo y después entramos en detalle. Supongamos que queremos eliminar todas las líneas en blanco de nuestro archivo de texto1 utilizaríamos el siguiente comando:

sed -e '/^$/d' texto1

Lo único que hemos cambiado en el comando es la expresión regular /^$/ esta expresión quiere decir todas las líneas que estén en blanco. Muy útil cuando queremos ver un archivo que tiene muchas líneas en blanco y queremos compactar todo para verlo mejor.

Otro comando útil es para eliminar los comentarios que comienzan normalmente con # en los archivos de configuración para eliminarlos ejecutamos:

sed -e '/^#/d' /etc/services | less

Esto elimina todas las líneas que comienzan con #, pero sed imprime el resultado en la salida estándar (stdout) y no en el archivo.

Las expresiones regulares son un grupo de caracteres y reglas que sirven para filtrar patrones de letras, números, símbolos y condiciones como por ejemplo al principio de la línea o al final, un carácter o varios, etc.

Caracteres especiales:

Carácter	Descripción
^	Corresponde al inicio de la línea
$	Corresponde al final de la línea
.	Filtra un sólo carácter
*	Filtra cero o más caracteres
[]	Filtra todo el rango dentro de los []

Para entender un poco mejor vamos a ver unos ejemplos todos estos los pueden colocar dentro de las ' ' en el comando sed para filtrar según sea el caso:

Expresión Regular	**Descripción**
/./	Filtra cualquier línea que tenga al menos 1 carácter
/.../	Filtra cualquier línea que tenga al menos 3 carácter
/^#/	Filtra cualquier línea que comience con #
/^$/	Filtra cualquier línea en blanco
/)$/	Filtra cualquier línea que termine con ')' (sin espacios)
/) *$/	Filtra cualquier línea que termine con ')' seguido por cero o más espacios
/[xyz]/	Filtra cualquier linea que contenga las letras 'x' 'y' o 'z' en minúsculas
/^[DEF]/	Filtra cualquier línea que comience con 'D', 'E' o 'F'

Para Imprimir línea o conjuntos de líneas, también podemos usar sed. Hasta ahora solo hemos borrado líneas pero si queremos que imprima las líneas que coinciden con nuestro filtro. Vamos a ver un ejemplo, supongamos que queremos imprimir las líneas que comienzan con d y e minúsculas entonces ejecutamos lo siguiente:

sed -n -e '/^[de]/p' texto1

Tenemos dos elementos nuevos en nuestro comando, el primero es la opción -n que hace que sólo se imprima lo que coincida con nuestro filtro y hemos cambiado el comando 'd' por el comando 'p' (print) que dice que imprima el espacio del patrón. Si ejecutan el comando anterior deberían obtener sólo dos líneas.

2.3. Ejemplos de uso de sed

Para borrar una línea hacemos lo siguiente:

sed 'nº_de_línead' fichero

Podemos indicar un número de línea concreto. Por ejemplo:

sed '1d' fichero

Podemos indicar un intervalo de líneas a borrar. Por ejemplo:

sed '3,5d' fichero

También podemos indicar que queremos borrar desde una determinada línea en adelante:

sed '3,$d' fichero

Otro ejemplo útil es borrar las líneas en blanco de un fichero:

sed '/^$/d' fichero

A la hora de borrar, también podemos especificar una cadena, de tal forma que el comando borrará todas las líneas que contengan esa cadena. Ejemplo:

cat fichero | sed '/^[ ]*$/d' > ficherodestino

Lo anterior borrará todas las líneas en blanco de fichero.

Otro de los usos interesantes es borrar los espacios al comienzo de cada línea:

sed 's/^ *//g' fichero

O borrar los espacios al final de cada línea:

sed 's/ *$//g' fichero

Otro de los usos más interesantes de sed es sustituir cadenas. Podemos sustituir una cadena por otra de la siguiente manera:

sed 's/cadena1/cadena2/' fichero

Al ejecutar el comando anterior, se sustituye la primera cadena que encuentra por la segunda. Pero, si lo que queremos es sustituir todas las cadenas que encuentre, en cada una de las líneas, añadimos el parámetro g:

sed 's/cadena1/cadena2/g' fichero

Por otra parte, también podemos hacer que sustituya la cadena1 por la cadena2 en un número de línea concreto:

sed '5 s/USUARIO/usuario/g' fichero

Con cadenas de texto normales la cosa es sencilla, pero al que más y al que menos le resulta complicado cuando lo que hay que sustituir son caracteres especiales como el tabulador: \t o el carácter de nueva línea: \n. Pero veamos como tampoco es complicado: Imaginemos que tenemos un fichero con campos en los que el separador es el tabulador y queremos sustituir este carácter separador por otro carácter separador, como por ejemplo el punto y coma (;). Lo haremos de la siguiente manera:

sed 's/\t/;/g' fichero

3. Comando du

El comando du permite conocer el espacio ocupado en el disco por un determinado directorio y todos los subdirectorios que cuelgan de él.

La sintaxis de este comando es:

du [opciones] ... [fichero]...

Los usos más habituales del comando du son:

du	Al ejecutar el comando du sin parámetros, nos muestra el espacio de disco utilizado por el directorio donde nos encontramos, dado en número de bloques.
du –h	Al ejecutar du con la opción —h nos muestra el espacio en disco que ocupa el directorio, pero en número de bytes.
du <directorío>	Nos muestra el número de bloques lógicos de 1 KB que ocupa el directorio indicado
du -a	Nos muestra el número de bloques lógicos de 1 KB que ocupan los ficheros contenidos en el directorio actual, o el directorio que se pase como argumento.
du -s	obtenemos un resumen sin que aparezcan los detalles

4. Comando df

El comando **df** por el contrario informa del espacio total, ocupado y disponible para cada uno de los sistemas de ficheros. La sintaxis de este comando es:

df [opción] ... [fichero] ...