

Tutorial y Guía Breve del Comando Sed para Unix y Linux

V0.2 - AmericaTI.com

14 de Noviembre de 2006

Tabla de contenidos

Resumen	1
Autoría y Copyright	1
Revisiones	2
Introducción	2
Uso básico	2
Conociendo la operación de <code>sed</code>	4
Ejecutar un "script" de <code>sed</code> : opción <code>-f</code>	4
Cómo opera <code>sed</code> internamente	5
Condiciones para ejecución de comandos <code>sed</code>	5
Comandos de <code>sed</code>	6
Ejemplos de <code>Sed</code>	9
Substituciones simples: Cambiar Ajos por Frijoles	9
Usando comandos avanzados	11

Resumen

Este documento pretende proporcionar una visión general y comentar algunos aspectos interesantes del popular comando "`sed`" disponible en prácticamente todos los sistemas Unix y Linux. Para varios ejemplos se asume que el lector ya conoce el lenguaje de expresiones regulares.

Autoría y Copyright

Este documento tiene copyright (c) 2006 AmericaTI EIRL (www.americati.com.) Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la "GNU Free Documentation License, Version 1.2", excepto en lo mencionado en el siguiente párrafo. Esta licencia puede obtenerse en: <http://www.gnu.org/licenses/fdl.txt>

Si se desea crear un trabajo derivado o publicar este documento para cualquier propósito, por favor contactarnos (vía nuestra página web) a fin de tener la oportunidad de proporcionar una versión más reciente. De no ser esto posible, la última versión debería estar disponible en el sitio web AmericaTI.com. [<http://www.americati.com>]

Son bienvenidas todas las sugerencias y correcciones.

Este documento fue confeccionado con DocBook utilizando el preprocesador **Qdk** disponible en SourceForge.net. [<http://qdk.sourceforge.net>]

Revisiones

- 0.1 2006-10-11 Primera versión preliminar
- 0.2 2007-01-24 Ejemplos de comandos avanzados
- 0.3 2007-03-26 Agregados dos ejemplos

Introducción

Sed es considerado un editor de texto orientado a "flujo" (en contraposición a los clásicos editores "interactivos") el cual acepta como entrada un archivo o la entrada estándar; cada línea es procesada y el resultado es enviado a la salida estándar.

Para casos muy simples, se suele emplear la sintaxis:

```
sed comandos_sed archivo
```

donde "comandos_sed" corresponde a uno o más especificaciones acerca de qué hacer con las líneas que se leen desde "archivo"¹.

Uso básico

Tal vez la manera más simple de iniciarse en sed es mediante ejemplos:

Borrar la segunda línea

```
$ cat rata
1
2
3
4
5
6
$ sed '2d' rata
1
3
4
5
6
```

Como se aprecia, tenemos un archivo de texto tremendamente simple conteniendo seis líneas. Al ejecutar sed cada línea es sometida a los "comandos sed". En este

¹ Como de costumbre, de no especificarse el "archivo", se lee desde stdin.

caso, el único comando es "d" referido a la línea número dos (la cual contiene un "2".) Este comando elimina la línea en cuestión, mientras que el resto de líneas no es alterado y por omisión se envían a `stdout`.

Imprimir una línea

```
$ sed '2p' rata
1
2
2
3
4
5
6
```

En este ejemplo hemos solicitado la impresión (comando "p") de la línea número dos. Debido a que por omisión las líneas que ingresan son a la vez enviadas a `stdout`, resulta que la segunda línea es mostrada en dos oportunidades. Obviamente esto no suele ser lo más útil; en el siguiente ejemplo desactivaremos la "impresión automática" de las líneas con la opción "-n":

```
$ sed -n '2p' rata
2
```

Modificar líneas

El comando de sustitución "s" permite modificar líneas:

```
$ sed 's/5/t/' rata
1
2
3
4
t
6
```

El comando "s" recibe un "patrón de búsqueda" y un texto de reemplazo. En el ejemplo anterior hemos buscado el texto "5" (en todas las líneas) y éste ha sido reemplazado por la letra "t" (podría ser cualquier texto.)

El ampersand (&) se utiliza para reemplazar con "el texto hallado":

```
$ sed 's/5/&t/' rata
1
2
3
4
5t
```

6

En este caso, el texto hallado es un "5" por lo que el amperstand contiene este mismo valor. Así, el reemplazar con "&t" equivale a reemplazar con "5t". Esto realmente tiene utilidad cuando se utiliza una expresión regular:

```
$ sed 's/[2-5]/&t/' rata
1
2t
3t
4t
5t
6
```

En este caso el "texto hallado" toma respectivamente los valores "2", "3", "4" y "5", el cual es asumido por el amperstand, con lo que el reemplazo real es respectivamente "2t", "3t", "4t" y "5t".

Otro ejemplo muy similar:

```
$ sed 's/.*/&00/' rata
100
200
300
400
500
600
```

Conociendo la operación de sed

Ejecutar un "script" de sed: opción -f

En todos los ejemplos anteriores, hemos proporcionado un argumento en la línea de comandos que especificaba lo que deseábamos que `sed` realice ("los comandos `sed`"). En casos más sofisticados es usual utilizar un archivo de "script `sed`" que contenga los "comandos `sed`" a ejecutarse (especificado con la opción "-f"). Por ejemplo, en lugar de:

```
$ cat rata
1
2
3
4
5
6
$ sed -n '2p' rata
2
```

Podemos emplear un archivo auxiliar de "script":

```
$ cat prueba1.sed
2p
$ sed -n -f prueba1.sed rata
2
```

Cómo opera sed internamente

Se utilizan dos buffers en memoria: el "espacio de patrones" y el "espacio hold". Ambos inicialmente están vacíos.

El programa `sed` lee una línea de texto de la entrada y la deposita en el "espacio de patrones" (sin el "fin de línea" final.) A continuación se ejecutan los "comandos `sed`" que correspondan (que satisfacen ciertas condiciones) y finalmente el texto del "espacio de patrones" se envía a la salida estándar seguido por un "fin de línea" (excepto si se utiliza la opción "-n".) Este proceso se repite para todas las líneas de la entrada.

Como se aprecia, el contenido del "espacio de patrones" se pierde entre línea y línea de la entrada; por el contrario, el "espacio hold" se mantiene.

Condiciones para ejecución de comandos sed

Direccionar líneas

Existen dos métodos estándar:

- **NUMERO**: Un número corresponde a la línea de la entrada cuya posición corresponde al mismo ("\$" significa la última línea)
- **/REGEXP/**: Corresponde a las líneas que satisfacen la expresión regular especificada

Si no se especifica una dirección, se asume que el comando aplicará sobre todas las líneas.

Rangos

Tienen la forma:

```
dirección1,dirección2
```

Corresponde a aquellas líneas que se inician apenas se satisfaga la "dirección1" y termina apenas se satisfaga "dirección2".

Ejemplo:

```
$ cat rata
```

```
1
2
3
4
5
6
$ sed -n '/[23]/,/[56]/p' rata
2
3
4
5
```

Comandos de sed

Comentario:

Este comando se debería emplear sólo en la primera línea de un script de sed (por portabilidad):

```
$ cat prueba2.sed
# Un ejemplo sed
/[23]/,/[56]/p
$ sed -n -f prueba2.sed rata
2
3
4
5
```

Este comando no acepta condiciones de línea.

Terminar: q

Este comando termina el script en la línea cuya condición se satisface. Opcionalmente recibe un argumento numérico correspondiente al valor de retorno (por omisión es cero.)

```
$ cat prueba3.sed
# Un ejemplo sed
/[23]/,/[56]/p
4q 10
$ sed -n -f prueba3.sed rata
2
3
4
$ echo $?
10
```

Borrar espacio de patrones: d

Este comando borra el espacio de patrones e inmediatamente pasa a la siguiente línea de entrada.

```
$ cat prueba4.sed
# Un ejemplo sed
/[23]/,/[56]/d
diego@rat:~/escritos/sed$ sed -f prueba4.sed rata
1
6
```

Imprimir el espacio de patrones: p

Generalmente se usa con la opción "-n". Se ha mostrado en gran cantidad de ejemplos.

Agrupar comandos: { ... }

Esto permite ejecutar un conjunto de comandos en relación a una misma condición de línea:

```
$ cat prueba5.sed
# Un ejemplo sed
/[23]/,/[56]/{
p
p
}
$ sed -n -f prueba5.sed rata
2
2
3
3
4
4
5
5
```

Hacer substituciones: s/regexp/reemplazo/[opciones]

El "espacio de patrones" es analizado en búsqueda de la expresión regular "regexp". De ocurrir, ésta es reemplazada con "reemplazo". Si se proporciona la opción "g", este proceso se repite con todas las ocurrencias de "regexp" en el "espacio de patrones" (en caso contrario, sólo ocurre un reemplazo.) Finalmente, la opción "p" envía el "espacio de patrones" resultante a la salida estándar siempre que haya ocurrido reemplazo.

El caracter "&" en "reemplazo" equivale al texto de entrada que ha encajado con la expresión regular.

Ejemplo:

```
$ cat prueba6.sed
# Un ejemplo sed
/[23]/,/[[56]]/s/./x/p
ls/./a&b/p
$ sed -n -f prueba6.sed rata
alb
x
x
x
x
```

Ejemplo:

```
$ cat raton
estaba la rana cantando
debajo del agua
diego@rat:~/escritos/sed$ cat prueba7.sed
# Un ejemplo sed
1s/a/x/g
2s/a/x/
diego@rat:~/escritos/sed$ sed -f prueba7.sed raton
estxbx lx rxnx cxntxndo
debxjo del agua
```

Grabar en archivo: w

Este comando graba el contenido del "espacio de patrones" en el archivo especificado. En la primera ejecución el archivo es truncado (en el ejemplo, apreciar que el contenido original de `salida.txt` desaparece.)

```
$ cat salida.txt
xxx
$ cat prueba8.sed
# Un ejemplo sed
/[23]/,/[[56]]/{p
w salida.txt
}
$ sed -n -f prueba8.sed rata
2
3
4
5
$ cat salida.txt
2
3
4
5
```

El "espacio hold"

Estos comandos son algo extraños. Ver los ejemplos más adelante.

- x Intercambiar el contenido del "espacio hold" con el del "espacio de patrones"
- g Transferir el contenido del "espacio hold" hacia el del "espacio de patrones"
- h Transferir el contenido del "espacio de patrones" hacia el del "espacio hold"
- G Agregar un "fin de línea" al contenido del "espacio de patrones" y a continuación agregarle el contenido del "espacio hold"
- H Agregar un "fin de línea" al contenido del "espacio hold" y a continuación agregarle el contenido del "espacio de patrones"

Otros comandos

- P Imprimir el contenido del "espacio de patrones" hasta antes del primer "fin de línea"
- N Agregar un "fin de línea" al "espacio de patrones" y a continuación agregar el contenido de la siguiente línea de la entrada
- D Borrar caracteres en el "espacio de patrones" hasta el primer "fin de línea". Si no queda ningún contenido, reiniciar un ciclo normal; de lo contrario, el ciclo se reinicia con el contenido actual
- : etiq Permite especificar una "etiqueta" a continuación del ":"
- b etiq Genera un salto en la ejecución del script hacia la etiqueta especificada
- t etiq Genera un salto sólo si ha ocurrido una substitución exitosa en la última línea leída
- y/s1/s2/ Traduce los caracteres listados en 's1' a los listados en 's2'. Estos conjuntos de caracteres deben ser del mismo tamaño

Ejemplos de sed

Substituciones simples: Cambiar Ajos por Frijoles

Pretendemos intercambiar los ajos por frijoles en el siguiente texto:

```
$ cat ajos.txt
El trabajo dignifica al hombre
Las semillas del ajo tienen un fuerte olor a ajo
Ajolote es un animal extraño
Ajo para ahuyentar al vampiro!
Obrajosa, el pueblo de Doña Perfecta
```

Le rebajo el kilo de ajos

La substitución simple obviamente no funciona:

```
$ cat ajos1.sed
$ sed -f ajos1.sed ajos.txt
El trabfrijol dignifica al hombre
Las semillas del frijol tienen un fuerte olor a frijol
Ajolote es un animal extraño
Ajo para ahuyentar al vampiro!
Obrfrijolsa, el pueblo de Doña Perfecta
Le rebfrijol el kilo de frijols
```

Para ubicar "ajo" como una palabra independiente, podemos auxiliarnos de los espacios en blanco separadores:

```
$ cat ajos2.sed
s/ ajo / frijol /g
$ sed -f ajos2.sed ajos.txt
El trabajo dignifica al hombre
Las semillas del frijol tienen un fuerte olor a ajo
Ajolote es un animal extraño
Ajo para ahuyentar al vampiro!
Obrajosa, el pueblo de Doña Perfecta
Le rebajo el kilo de ajos
```

El resultado es mejor, pero se puede mejorar. Por ejemplo, el "ajo" de la segunda línea no es intercambiado por no tener un espacio en blanco a continuación. El patrón " ajo\$" podría ubicarlo, y de forma complementaria, deberíamos incluir el patrón "^ajo ":

```
$ cat ajos3.sed
s/ ajo / frijol /g
s/ ajo$/ frijol/
s/^ajo /frijol /
$ sed -f ajos3.sed ajos.txt
El trabajo dignifica al hombre
Las semillas del frijol tienen un fuerte olor a frijol
Ajolote es un animal extraño
Ajo para ahuyentar al vampiro!
Obrajosa, el pueblo de Doña Perfecta
Le rebajo el kilo de ajos
```

Observar que en estos casos no es necesario el sufijo "g" en el comando "s".

El "Ajo" de la cuarta línea no ha sido reemplazado por iniciarse con una "A" mayúscula. Podemos corregirlo con facilidad:

```
$ cat ajos4.sed
s/ ajo / frijol /g
```

```
s/ ajo$/ frijol/  
s/^ajo /frijol /  
s/ Ajo / Frijol /g  
s/ Ajo$/ Frijol/  
s/^Ajo /Frijol /  
$ sed -f ajos4.sed ajos.txt  
El trabajo dignifica al hombre  
Las semillas del frijol tienen un fuerte olor a frijol  
Ajolote es un animal extraño  
Frijol para ahuyentar al vampiro!  
Obrajosa, el pueblo de Doña Perfecta  
Le rebajo el kilo de ajos
```

Los "ajos" en plural de la última línea se pueden procesar de un modo similar:

```
$ cat ajos5.sed  
s/ ajo / frijol /g  
s/ ajo$/ frijol/  
s/^ajo /frijol /  
s/ Ajo / Frijol /g  
s/ Ajo$/ Frijol/  
s/^Ajo /Frijol /  
s/ ajos / frijoles /g  
s/ ajos$/ frijoles/  
s/^ajos /frijoles /  
s/ Ajos / Frijoles /g  
s/ Ajos$/ Frijoles/  
s/^Ajos /Frijoles /  
$ sed -f ajos5.sed ajos.txt  
El trabajo dignifica al hombre  
Las semillas del frijol tienen un fuerte olor a frijol  
Ajolote es un animal extraño  
Frijol para ahuyentar al vampiro!  
Obrajosa, el pueblo de Doña Perfecta  
Le rebajo el kilo de frijoles
```

Usando comandos avanzados

Salto

El siguiente script permite invertir el orden de los caracteres de cada línea (en el ejemplo, provenientes de la entrada estándar.)

```
$ sed -f rever.sed  
123456  
654321  
abracadabra  
arbadacarba  
$ cat rever.sed
```

```
s/^.*$/&\n/
:n
s/^\([^\\n]\)\(\(.*\)\n\(\(.*\)\$\/\2\n\1\3/
tn
s/\n//
```

El script hace uso de expresiones regulares agregando un separador artificial (un salto de línea) al final de cada una de éstas; luego se crea un "loop" entre el ":n" (etiqueta) y el "tn" (saltar a etiqueta "n" si la substitución fue exitosa); cada substitución lleva un caracter del inicio hacia después del salto de línea; este caracter del inicio debe ser distinto del salto de línea a fin de que la substitución falle cuando cuando se hayan desplazado todos los caracteres.

Tras salir del loop, una última substitución elimina el salto de línea agregado.

Unir líneas de entrada

Tenemos un archivo correspondiente a un catálogo de libros:

```
cat libros.txt
Archivo de Libros

-----

La abeja Maya

Un cuento de insectos y arácnidos

-----

Los perros hambrientos
Un clásico de Ciro alegría
-----

Cien años de soledad
Quién no lo conoce?
La mejor época de GGM
con Macondo, Melquiades, y otros
```

Como se aprecia, la estructura corresponde a un grupo de guiones que separan cada entrada; luego de los guiones hay una línea en blanco, y la siguiente línea contiene el título de cada libro. El siguiente script extrae los títulos:

```
$ cat libros.sed
/^-----/{
N
N
s/\(.*\)\n\(\(.*\)\n\(\(.*\)\)/\3/
p
}
```

```
$ sed -n -f libros.sed libros.txt
La abeja Maya
Los perros hambrientos
Cien años de soledad
```

La primera línea busca líneas con guiones; para cada una de éstas, se procede a agregar (al buffer de patrones) el contenido de las dos líneas que siguen (con un salto de línea intermedio) mediante el comando `N`. Finalmente, la substitución separa el buffer de patrones en tres expresiones regulares separadas por saltos de línea y lo substituye por el tercero de éstos (la última línea leída, el título.) Finalmente, el resultado se imprime con `'p'` (que a la sazón agrega un conveniente salto de línea.)

Mostrar la antepenúltima línea de un archivo

Como primer paso, el script que se muestra a continuación se limita a imprimir las últimas tres líneas de un archivo:

```
$ sed -n -f ultimos.sed libros.txt
Quién no lo conoce?
La mejor época de GGM
con Macondo, Melquiades, y otros
$ cat ultimos.sed
H
g
3,$ { s/[^\n]*\n// ; h }
$p
```

Los comandos `"H"` y `"g"` se ejecutan para todas las líneas. En el primer caso, el espacio `"hold"` va acumulando las líneas leídas (el espacio de patrones) separadas por saltos de línea; a continuación, el comando `"g"` transfiere esta acumulación al espacio de patrones. Para las primeras tres líneas no se hace nada más.

En las subsiguientes líneas, la substitución elimina la línea más `"antigua"` que se encuentra al principio del espacio de patrones, y es copiada nuevamente al espacio `hold` con `"h"`. En otras palabras, el espacio `hold` va acumulando una `"ventana"` de tres líneas a través del archivo. Al procesarse a la última línea, el contenido del buffer de patrones (que coincide con el de `hold`) se imprime.

Con una ligera variación obtendremos la antepenúltima línea buscada:

```
$ sed -n -f antepenultimo.sed libros.txt
Quién no lo conoce?
$ cat antepenultimo.sed
H
g
3,$ { s/[^\n]*\n// ; h }
${ s/\([^^\n]*\).*\/\1/ ; p }
```

La única diferencia radica en que al procesarse la última línea, la substitución deja únicamente la primera línea de la `"ventana"` en el buffer de patrones; esta coincide con la antepenúltima.

Subrayar la primera línea de un archivo

Cierto comando produce el siguiente listado (no importa el comando, sólo tome en cuenta la salida):

```
$ echo "show databases" | mysql -u pafuser
Database
information_schema
ktest
mysql
paftest
sugarcrm
superapof
```

Como debe sospechar, se trata de un listado de "bases de datos", lo cual es indicado en la primera línea de la salida con la palabra "Database"; es evidente que esta primera línea corresponde al "título" de dicha salida, y podríamos "subrayarlo" para hacerlo notar mejor:

```
$ echo "show databases" | mysql -u pafuser | sed -f subrayado.sed
Database
-----
information_schema
ktest
mysql
paftest
sugarcrm
superapof
```

El script `sed` es muy sencillo:

```
cat subrayado.sed
# subrayar la primera linea
l{h
s/./-/g
H
g
}
```

Todo ocurre exclusivamente al leer la primera línea (y sólo la primera.) La línea leída (en el espacio de patrones) se copia al buffer "hold" con el comando 'h'; luego cada carácter del buffer de patrones es reemplazado por un guión o dash ("-") de modo tal que el subrayado encaje perfectamente con la longitud del título. Después agregamos "el subrayado" del espacio de patrones a la primera línea (en el buffer hold) mediante el comando "H" (ahora el espacio hold contiene la primera línea subrayada); finalmente transferimos el espacio hold al espacio de patrones con el comando "g" (también se pudo usar "x" puesto que el espacio hold ya no se va a utilizar.)

Dos columnas

Asumiendo que en ejemplo anterior tuviesemos muchas filas en el resultado, podríamos desear obtener una salida a dos columnas (exceptuando el título.) El siguiente script realiza el efecto:

```
$ cat 2col.sed
# subrayar la primera linea
1{h
s/./-/g
H
g
}
2,${s/./&\t/
N
s/\n//
}
```

Como se aprecia, para las líneas (empezando de la segunda) se agrega un tabulador y luego se fuerza la lectura de una línea adicional (comando "N"); lamentablemente, de forma automática se introduce un "salto de línea" el cual debe ser eliminado o se perdería el efecto deseado, para lo cual se hace la última substitución.

El resultado es:

```
$ echo "show databases" | mysql -u pafuser | sed -f 2col.sed
Database
-----
information_schema      ktest
mysql      pafatest
sugarcrm      superapof
```

El tabulador funciona como separador, pero la salida no tiene buena apariencia. El siguiente script es una variación del anterior en el que se fuerza la primera columna a 20 caracteres completados con espacios:

```
$ cat 2colnotab.sed
# subrayar la primera linea
1{h
s/./-/g
H
g
}
2,${s/./& /
s/\(. \{20\}\).*/\1/
N
s/\n//
}
```

La salida ahora luce más legible:

```
$ echo "show databases" | mysql -u pafuser | sed -f 2colnotab.sed
```

Database

information_schema	ktest
mysql	paftest
sugarcrm	superapof