Aprenda Linux, 101: Cadenas de texto y filtros

Manipulando texto en la línea de comando usando GNU textutils

Ian Shields
Senior Programmer
IBM

02-01-2012
(Primera publicación 02-01-2012)

Hay mucho más en cuanto a manipulación de texto que cortar y pegar, particularmente cuando usted no está usando una GUI. Estudie para el examen 101 del Linux Professional Institute Certification (LPIC), o aprenda por diversión. En este artículo, Ian Shields le introduce a la manipulación de texto en Linux® usando filtros del paquete GNU textutils. Hacia el final de este artículo, usted estará manipulando texto como un experto. [La primera línea del Listado 7 ha sido corregida, gracias a un lector alerta. -Ed.]

Ver más contenido de esta serie

Sobre esta serie

Esta serie de artículos le ayuda a aprender tareas de administración de sistema Linux. Usted también puede usar el material en estos artículos para prepararse para los exámenes (LPIC-1) de nivel 1 Linux Professional Institute Certification.

Vea nuestro mapa de ruta de la serie para una descripción de un enlace hacia cada artículo de esta serie. El mapa de ruta está en desarrollo y refleja los últimos objetivos (abril del 2009) para los exámenes LPIC-1: a medida que completemos artículos, los añadiremos al mapa de ruta. Mientras tanto, usted puede encontrar versiones anteriores de material similar , que soporta objetivos LPIC-1 anteriores a abril del 2009, en nuestros tutoriales de preparación para el examen de certificación LPI.

Prerrequisitos

Para aprovechar al máximo los artículos de esta serie usted debe contar con conocimientos básicos Linux y un sistema Linux funcional en el cual poder practicar los comandos que se cubren en este artículo. Algunas versiones diferentes del programa darán un formato de resultado diferente, por lo que sus resultados pueden no siempre verse igual a los listados y gráficas mostradas aquí.

Visión General

Este artículo le ofrece una introducción a los filtros, lo cual le permite construir interconexiones complejas para manipular texto usando los filtros. Usted aprenderá cómo mostrar texto, ordenarlo, contar palabras y líneas y traducir caracteres, entre otras cosas. usted también aprenderá cómo usar el editor de secuencia sed.

En este artículo usted aprenderá sobre los siguientes temas:

- Enviar archivos de texto y secuencias de resultado mediante filtros de herramienta de texto para modificar el resultado
- Usar comandos UNIX estándar que se encuentran en el paquete GNU textutils
- Usar el editor para hacer script de cambios complejos a archivos de texto

Este artículo le ayuda a prepararse para el Objetivo 103.2 del Tema 103 del examen 101 (LPIC-1) Junior Level Administration. El objetivo tiene un peso de 3. El material en este artículo corresponde al de abril del 2009 objetivos del examen 101. Usted siempre debe referirse a los objetivos para conocer los requerimientos definitivos.

Filtrado de texto

El filtrado de texto es el proceso de tomar una secuencia de texto de entrada y efectuar algo de conversión del texto antes de enviarlo hacia una secuencia de salida. Aunque tanto la entrada como la salida pueden provenir de un archivo, en los entornos Linux y UNIX® el filtrado se hace con mayor frecuencia mediante la construcción de una interconexión de comandos donde el resultado de un comando es interconectado o redirigido para ser usado como entrada para el siguiente. Las interconexiones y el redireccionamiento se cubren con mayor detalle en el artículo sobre secuencias, interconexiones y redirecciones (el cual usted puede encontrar en el mapa de ruta de la serie), pero por ahora observemos las interconexiones y el redireccionamiento de salida usando los operadores | y >.

Secuencias

Una secuencia no es nada más que una cadena de bytes que se puede leer o escribir usando funciones de biblioteca que oculten los detalles de un dispositivo subyacente de la aplicación. El mismo programa puede leer desde o escribir hacia una terminal, archivo o conector de red, en una forma independiente en cuanto a dispositivo, usando secuencias. Los entornos de programación modernos y shells usan tres secuencias de E/S estándar:

- stdin es la secuencia de entrada estándar, la cual proporciona entrada a los comandos.
- stdout es la secuencia de salida estándar, la cual muestra el resultado de los comandos.
- stderr es la secuencia de error estándar, la cual muestra resultados de error de los comandos.

Interconectando con |

Las entradas pueden provenir de parámetros que usted suministra a los comandos, y el resultado puede mostrarse en su terminal. Muchos comandos de procesamiento de texto (filtros) pueden tomar entradas o de la secuencia de entrada estándar o de un archivo. Para usar el resultado de un comando, command1, como entrada para un filtro, command2, usted conecta los comandos usando el operador (|). El Listado 1 muestra cómo interconectar el resultado de `echo` para ordenar una pequeña lista de palabras.

Listado 1. Interconectando el resultado de echo a la entrada de sort

```
[ian@echidna ~]$ echo -e "apple\npear\nbanana"|sort
apple
banana
pear
```

Cualquiera de los comandos puede tener opciones o argumentos. Usted también puede usar | para redirigir el resultado del segundo comando de la interconexión hacia un tercer comando, y continuar así. La construcción de interconexiones de comandos extensas, cada una con una capacidad limitada, es una forma común Linux y UNIX para lograr tareas. Algunas veces usted también verá el guión (-) usado en lugar de un nombre de archivo como argumento para un comando, queriendo decir que el resultado debe provenir de stdin más que de un archivo.

Redireccionamiento de resultado con >

Aunque es interesante poder crear una interconexión de varios comandos y ver el resultado en su terminal, hay veces en que usted desea guardar el resultado en un archivo. Usted hace esto con el operador de redireccionamiento de resultado (>).

Por el resto de la sección estaremos usando algunos archivos pequeños, así que vamos a crear un directorio llamado lpi103-2 y luego cd dentro de ese directorio. Luego usaremos > para redirigir el resultado del comando `echo` hacia un archivo llamado text1. Todo esto se muestra en el Listado 2. Note que el resultado no aparece en la terminal porque ha sido redirigido al archivo.

Listado 2. Redirigiendo un resultado de un comando a un archivo

```
[ian@echidna ~]$ mkdir lpi103-2
[ian@echidna ~]$ cd lpi103-2
[ian@echidna lpi103-2]$ echo -e "1 apple\n2 pear\n3 banana" > text1
```

Ahora que tenemos un par de herramientas básicas para interconectar y redirigir, observemos algunos comandos comunes de procesamiento de texto y filtros en UNIX y Linux. Esta sección le muestra algunas de las capacidades básicas; revise las páginas man apropiadas para conocer más sobre estos comandos.

Cat, od y split

Ahora que usted ha creado el archivo test1, es posible que usted desee ver qué hay en él. Use el comando `cat` (expresión corta para concatenar) para mostrar el contenido del un archivo en stdout. El Listado 3 verifica el contenido del archivo creado arriba.

Listado 3. Mostrando el contenido de un archivo con cat

```
[ian@echidna lpi103-2]$ cat text1
1 apple
2 pear
3 banana
```

El comando `cat` toma la entrada de stdin si usted no especifica un nombre de archivo (o si lo especifica - como el nombre de archivo). Vamos a usar esto junto con el redireccionamiento de resultado para crear otro archivo de texto como se muestra en el Listado 4.

Listado 4. Creando un archivo de texto con cat

```
[ian@echidna lpi103-2]$ cat >text2
9       plum
3       banana
10      apple
```

Muchos filtros pequeños

Otro ejemplo de un filtro pequeño es el comando tac . El nombre es el inverso de cat, y la función también es la inversa de cat, en cuanto a que el archivo es mostrado en orden inverso. Intente ejecutar **tac text2 text1**
usted mismo.

En el Listado 4, cat continúa leyendo de stdin hasta el final del archivo. Use la combinación **Ctrl-d** (mantenga **Ctrl** presionado y presione **d**) para señal de fin de vida. Esta es la misma combinación de teclas para salir del bash shell. Use la tecla tab para alinear los nombres de las frutas en una columna.

¿Recuerda que cat es el nombre corto de concatenar? Usted puede usar cat para concatenar varios archivos juntos para mostrarlos. El Listado 5 muestra los dos archivos que acabamos de crear.

Listado 5. Concatenando dos archivos con cat

```
[ian@echidna lpi103-2]$ cat text*
1 apple
2 pear
3 banana
9       plum
3       banana
10      apple
```

Cuando usted muestra estos dos archivos de texto usando cat, notará diferencias de alineación. Para aprender qué causa esto, usted necesita observar los caracteres de control que están en el archivo. Sobre estos se actúa en resultado de visualización de texto más que presentando alguna representación del carácter de control en sí, así que necesitaremos volcar el archivo en un formato que le permita encontrar e interpretar estos caracteres especiales. Las herramientas de texto GNU incluyen un comando od (o Octal Dump) para este propósito.

Hay varias opciones para el od, como la opción -A para controlar el radix de los desplazamientos de archivo -t para controlar la forma del contenido del archivo presentado. El radix puede ser especificado como o, (octal, el predeterminado), d (decimal), x (hexadecimal), o n (sin desplazamientos mostrados). Usted puede mostrar el resultado como octal, hex, decimal, punto flotante, ASCII con escapes de barra inversa, o caracteres con nombre (nl para newline, ht para horizontal tab, etc.). El Listado 6 muestra algunos de los formatos disponibles para volcado del archivo de ejemplo text2.

Listado 6. Volcado de archivos con od

```
[ian@echidna lpi103-2]$ od text2
0000000 004471 066160 066565 031412 061011 067141 067141 005141
0000020 030061 060411 070160 062554 000012
0000031
[ian@echidna lpi103-2]$ od -A d -t c text2
0000000   9  \t   p   l   u   m  \n   3  \t   b   a   n   a   n   a  \n
0000016   1   0  \t   a   p   p   l   e  \n
0000025
[ian@echidna lpi103-2]$ od -A n -t a text2
  9 ht   p   l   u   m nl   3 ht   b   a   n   a   n   a nl
  1   0 ht   a   p   p   l   e nl
```

Notas:

1. La opción -A de cat proporciona una forma alternativa para ver dónde están sus tabulaciones y finales de línea. Consulte la página man para más información.
2. Si usted ve espacios en lugar de tabulaciones en su archivo text2, consulte Expand, unexpand y tr, más adelante en este artículo para ver cómo intercambiar entre etiquetas y espacios en un archivo.
3. Si usted tiene un trasfondo de sistema principal, tal vez esté interesado en la herramienta hexdump, la cual hace parte de un conjunto de herramientas diferente. Este no se cubre aquí, por lo que debe consultar las páginas man.

Nuestros archivos de ejemplo son bastante pequeños, pero algunas veces usted tendrá archivos grandes que necesita dividir en piezas más pequeñas. Por ejemplo, usted puede querer descomponer un archivo de gran tamaño en pequeñas porciones del tamaño de un CD para poder escribirlo en CD para enviarlo por correo a alguien que pueda crear un DVD para usted. El comando split hará esto de tal forma que el comando cat se pueda usar para recrear el archivo fácilmente. De forma predeterminada, los archivos que resulten del comando split tienen un prefijo en su nombre 'x' seguido por un sufijo como 'aa', 'ab', 'ac', ..., 'ba', 'bb', etc. Las opciones permiten que usted cambie estos predeterminados. Usted también puede controlar el tamaño de los archivos de resultado y si los archivos de resultado contienen líneas completas o solo conteos de bytes.

El Listado 7 ilustra la división de nuestros dos archivos de texto con diferentes prefijos para los archivos de resultado. Dividimos text1 en archivos que contienen como máximo dos líneas, y text2 en archivos que contienen como máximo 18 bytes. Luego usamos cat para mostrar algunas de las piezas individualmente, así como para mostrar un archivo completo usando globbing, en cual se cubre en el artículo sobre administración básica de archivo y directorio (el cual usted puede encontrar en el mapa de ruta de la serie).

Listado 7. Dividiendo y recombinando con split y cat

```
[ian@echidna lpi103-2]$ split -l 2 text1
[ian@echidna lpi103-2]$ split -b 17 text2 y
[ian@echidna lpi103-2]$ cat yaa
9       plum
3       banana
1[ian@echidna lpi103-2]$ cat yab
0       apple
[ian@echidna lpi103-2]$ cat y* x*
9       plum
3       banana
10      apple
1 apple
2 pear
3 banana
```

Note que el archivo de división llamado yaa no terminó con un carácter de línea nueva, por lo que nuestro prompt fue desplazado después de que usamos cat para mostrarlo.

Wc, head y tail

Cat muestra el archivo completo. Esto está bien para pequeños archivos como nuestros ejemplos, per suponga que usted tiene un archivo grande. Bien, primero usted puede querer usar el comando wc (Word Count) para ver qué tan grande es el archivo. El comando wc muestra el número de líneas, palabras y bytes de un archivo. Usted también puede encontrar el número de bytes usando ls -l. El Listado 8 muestra el listado de directorio de formato extenso mostrando nuestros dos archivos de texto, así como el resultado de wc.

Listado 8. Usando wc con archivos de texto

```
[ian@echidna lpi103-2]$ ls -l text*
-rw-rw-r--. 1 ian ian 24 2009-08-11 14:02 text1
-rw-rw-r--. 1 ian ian 25 2009-08-11 14:27 text2
[ian@echidna lpi103-2]$ wc text*
 3  6 24 text1
 3  6 25 text2
 6 12 49 total
```

Las opciones le permiten controlar el resultado de wc o mostrar otra información como la máxima longitud de línea. Vea la página man para detalles.

Dos comandos le permiten mostrar la primera parte (head) o la última parte (tail) de un archivo. Estos comandos son head y tail . Estos se pueden usar como filtros, o pueden tomar un nombre de archivo como argumento. De forma predeterminada estos pueden mostrar la primera (o la última) de 10 líneas del archivo o secuencia. El Listado 9 usa el comando dmesg para mostrar mensajes de inicialización, en conjunto con wc, tail y head para descubrir que hay 791 mensajes, luego para mostrar los 10 últimos de ellos y finalmente para mostrar los seis mensajes comenzando a 15 del final. Algunas líneas pueden haber sido truncadas en este resultado (indicado por ...).

Listado 9. Usando wc, head y tail para mostrar mensajes de inicialización

```
[ian@echidna lpi103-2]$ dmesg|wc
   791    5554   40186
[ian@echidna lpi103-2]$ dmesg | tail
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput12
generic-usb 0003:04B3:310B.0009: input,hidraw1: USB HID v1.00 Mouse [HID 04b3:310b] on us
b-0000:00:1a.0-2.4/input0
usb 3-2.4: USB disconnect, address 11
usb 3-2.4: new low speed USB device using uhci_hcd and address 12
usb 3-2.4: New USB device found, idVendor=04b3, idProduct=310b
usb 3-2.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
usb 3-2.4: configuration #1 chosen from 1 choice
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput13
generic-usb 0003:04B3:310B.000A: input,hidraw1: USB HID v1.00 Mouse [HID 04b3:310b] on us
b-0000:00:1a.0-2.4/input0
usb 3-2.4: USB disconnect, address 12
[ian@echidna lpi103-2]$ dmesg | tail -n15 | head -n 6
usb 3-2.4: USB disconnect, address 10
usb 3-2.4: new low speed USB device using uhci_hcd and address 11
usb 3-2.4: New USB device found, idVendor=04b3, idProduct=310b
usb 3-2.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
usb 3-2.4: configuration #1 chosen from 1 choice
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput12
```

Otro uso común de tail es para seguir a un archivo usando la opción -f , normalmente con un conteo de línea de 1. Usted puede usar esto cuando tenga un proceso de segundo plano que esté generando un resultado en un archivo y usted desee ingresar y ver cómo lo está haciendo. En este modo, tail s e ejecutará hasta que usted lo cancele (usando **Ctrl-c**), mostrando líneas a medida que son escritas en el archivo.

Expand, unexpand y tr

Cuando creamos nuestros archivos text1 y text2, creamos text2 con caracteres de tabulación. Algunas veces usted puede desear intercambiar tabulaciones por espacios y viceversa. Los comandos expand y unexpand

hacen esto. La opción -t de ambos comandos le permiten establecer los topes de tabulación. Un solo valor establece tabulaciones repetidas a ese intervalo. El Listado 10 muestra cómo expandir las tabulaciones en el text2 por espacios individuales y otra secuencia whimsical de expand y unexpand que desalinea el texto en text2.

Listado 10. Usando expand y unexpand

```
[ian@echidna lpi103-2]$ expand -t 1 text2
9 plum
3 banana
10 apple
[ian@echidna lpi103-2]$ expand -t8 text2|unexpand -a -t2|expand -t3
9         plum
3         banana
10      apple
```

Desafortunadamente, usted no puede usar unexpand para reemplazar los espacios en text1 por tabulaciones, pues unexpand requiere por lo menos dos espacios para convertir a tabulaciones. Sin embargo, usted puede usar el comandotr , el cual traduce caracteres de un conjunto (set1) hacia los caracteres correspondientes en otro conjunto (set2). El Listado 11 muestra cómo usar tr para traducir espacios en tabulaciones. Como tr es netamente un filtro, usted genera entrada para este usando el comando cat . Este ejemplo también ilustra el uso de - para significar entrada estándar para cat, así que podemos concatenar el resultado de tr y el archivo text2.

Listado 11. Usando tr

```
[ian@echidna lpi103-2]$ cat text1 |tr ' ' '\t'|cat - text2
1       apple
2       pear
3       banana
9       plum
3       banana
10      apple
```

Si usted no está seguro de lo que está sucediendo en los últimos dos ejemplos, intente usando od para terminar cada etapa de la interconexión de turno; por ejemplo,
cat text1 |tr ' ' '\t' | od -tc

Pr, nl y fmt

El comando pr se usa para dar formato a los archivos para impresión. El encabezado predeterminado incluye el nombre de archivo y la fecha y hora de creación de archivo, junto con un número de página y dos líneas de encabezado en blanco. Cuando el resultado es creado a partir de múltiples archivos o de la secuencia de entrada estándar, la fecha y hora actuales se utilizan en lugar del nombre de archivo y la fecha de creación. Usted puede imprimir archivos lado a lado en columnas y controlar muchos aspectos de formateo a través de opciones. Como es usual, refiérase a la página man para detalles.

El comando nl numera las líneas, lo cual puede ser conveniente cuando se estén imprimiendo archivos. Usted también puede numerar líneas con la opción -n del comando cat . El Listado 12 muestra cómo imprimir nuestro archivo de texto y luego cómo numerar text2 e imprimirlo lado a lado con el text1.

Listado 12. Numerando y dando formato para impresión

```
[ian@echidna lpi103-2]$ pr text1 | head


2009-08-11 14:02                text1                Page 1


1 apple
2 pear
3 banana


[ian@echidna lpi103-2]$ nl text2 | pr -m - text1 | head


2009-08-11 15:36                                Page 1


    1  9      plum                 1 apple
    2  3      banana                2 pear
    3  10      apple                3 banana
```

Otro comando útil para formatear texto es el comando fmt , el cual da formato al texto para que se ajuste a los márgenes. Usted puede unir varias líneas cortas así como dividir otras largas. En el Listado 13, creamos text3 con una sola línea extensa de texto usando variantes del recurso de historial !#:* para guardar la escritura de nuestra frase cuatro veces. También creamos text4 con una palabra por línea. Luego usamos cat para mostrarlos sin formato incluyendo un carácter '$' que se ve, para mostrar finales de línea. Finalmente, usamos fmt para darles formato hasta un ancho máximo de 60 caracteres. De nuevo, consulte la página man para detalles sobre opciones adicionales.

Listado 13. Dando formato para una longitud máxima de línea

```
[ian@echidna lpi103-2]$ echo "This is a sentence. " !#:* !#:1->text3
echo "This is a sentence. " "This is a sentence. " "This is a sentence. ">text3
[ian@echidna lpi103-2]$ echo -e "This\nis\nanother\nsentence.">text4
[ian@echidna lpi103-2]$ cat -et text3 text4
This is a sentence.  This is a sentence.  This is a sentence. $
This$
is$
another$
sentence.$
[ian@echidna lpi103-2]$ fmt -w 60 text3 text4
This is a sentence.  This is a sentence.  This is a
sentence.
This is another sentence.
```

Sort y uniq

El comando sort ordena la entrada usando la secuencia de clasificación (LC_COLLATE) del sistema. El comando sort también puede fusionar archivos que ya están ordenados y verificar si un archivo ya está ordenado o no.

El Listado 14 ilustra el uso del comando sortpara ordenar nuestros dos archivos de texto después de traducir los espacios en blanco por tabulaciones en text1. Como el orden de clasificación es por carácter, es posible que le sorprendan los resultados. Afortunadamente, el comando sort puede ordenar por valores numéricos o por valores de carácter. Usted puede especificar esta elección para todo el registro o para cada campo. A menos que usted especifique un separador de campo diferente los campos están delimitados por espacios en

blanco o por tabulaciones. El segundo ejemplo del Listado 14 muestra el ordenamiento del primer campo numéricamente y el segundo por orden de clasificación (alfabéticamente). Este también ilustra el uso de la opción `-u` para eliminar cualquier línea duplicada y conservar únicamente las líneas que sean únicas.

Listado 14. Ordenamiento por carácter y numérico

```
[ian@echidna lpi103-2]$ cat text1 | tr ' ' '\t' | sort - text2
10      apple
1       apple
2       pear
3       banana
3       banana
9       plum
[ian@echidna lpi103-2]$ cat text1|tr ' ' '\t'|sort -u -k1n -k2 - text2
1       apple
2       pear
3       banana
9       plum
10      apple
```

Note que todavía tenemos dos líneas que tienen la palabra "apple", porque la prueba del carácter único es llevada a cabo en ambas claves de ordenamiento, k1n y k2 en nuestro caso. Piense en cómo modificar o añadir pasos a la interconexión anterior para eliminar la segunda ocurrencia de 'apple'.

Otro comando llamado `uniq` nos da otra forma para controlar la eliminación de líneas duplicadas. El comando `uniq` normalmente funciona sobre archivos ordenados, y elimina líneas **consecutivas** idénticas de cualquier archivo, esté ordenado o no. El comando `uniq` también puede ignorar algunos campos. El listado 15 ordena dos archivos de texto usando el segundo campo (nombre de fruta) y luego elimina las líneas que sean idénticas, iniciando en el segundo campo (es decir, nos saltamos el primer campo cuando estemos probando con `uniq`).

Listado 15. Usando uniq

```
[ian@echidna lpi103-2]$ cat text1|tr ' ' '\t'|sort -k2 - text2|uniq -f1
10      apple
3       banana
2       pear
9       plum
```

Nuestro ordenamiento fue por orden de clasificación, por lo que `uniq` nos da la línea "10 apple" en lugar de "1 apple". Intente añadiendo una clasificación numérica en el campo clave 1 para ver cómo cambiar esto.

Cut, paste y join

Ahora observemos tres comandos adicionales que manejan los campos en datos textuales. Estos comandos son particularmente útiles para manejar datos de tablas. El primero es el comando `cut` el cual extrae campos de archivos de texto. El delimitador de campo predeterminado es el carácter de tabulación. El Listado 16 usa `cut` para separar las columnas de text2 y luego usa un espacio como delimitador de resultado, lo cual es una forma exótica de convertir la tabulación de cada línea en un espacio.

Listado 16. Usando cut

```
[ian@echidna lpi103-2]$ cut -f1-2 --output-delimiter=' ' text2
9 plum
3 banana
10 apple
```

El comando `paste` pega líneas de dos o más archivos lado a lado, similar a la forma en que el comando `pr` fusiona archivos usando su opción `-m` . El Listado 17 muestra el resultado de pegar nuestros dos archivos de texto.

Listado 17. Pegando archivos

```
[ian@echidna lpi103-2]$ paste text1 text2
1 apple 9       plum
2 pear  3       banana
3 banana        10      apple
```

Estos archivos muestran el pegado simple, pero `paste` puede pegar datos de uno o más archivos de diversas y variadas formas. Consulte la página man para detalles.

Nuestro comando final de manipulación de campo es `join`, el cual une campos con base en un campo que coincida. Los archivos deben ser ordenados en el campo join. Como text2 no está en orden numérico, podríamos ordenarlo y luego join uniría las dos líneas que tengan un campo join coincidente (el campo con valor 3 en este caso).

Listado 18. Uniendo archivos con campos de unión

```
[ian@echidna lpi103-2]$ sort -n text2|join -j 1 text1 -
3 banana banana
join: file 2 is not in sorted order
```

¿Entonces, qué salió mal? Recuerda que usted aprendió sobre ordenamiento numérico y de caracteres en la sección Sort y uniq . La unión se lleva a cabo en caracteres coincidentes de acuerdo al orden de clasificación local. Este no trabaja sobre campos numéricos a menos que los campos sean todos de la misma longitud.

Usamos la opción `-j 1` para unir en el campo 1 de cada archivo. El campo a usar para la unión podrá especificarse separadamente para cada archivo. Usted puede, por ejemplo, unir con base en el campo 3 de un archivo y del campo 10 de otro.

Vamos a crear también un nuevo archivo, text5, ordenando el text1 en el segundo campo (nombre de la fruta) y luego reemplazando espacios por tabulaciones. Si luego ordenamos text2 en su segundo campo y unimos eso con text5 usando el segundo campo de cada archivo como el campo para unir, deberemos tener dos coincidencias (apple y banana). El Listado 19 ilustra esta unión.

Listado 19. Uniendo archivos con campos de unión

```
[ian@echidna lpi103-2]$ sort -k2 text1|tr ' ' '\t'>text5
[ian@echidna lpi103-2]$ sort -k2 text2 | join -1 2 -2 2 text5 -
apple 1 10
banana 3 3
```

Sed

Sed es el editor de secuencia. Varios artículos developerWorks, así como muchos libros y capítulos de libros sobre sed, están disponibles (vea Recursos). Sed es extremadamente poderosa y las tareas que puede llevar a cabo solo están limitadas por su imaginación. Esta breve introducción debe abrirle el apetito sobre sed, pero no pretende ser completa ni exhaustiva.

Como con muchos de los comandos de texto que hemos mirado hasta ahora, sed puede funcionar como un filtro o tomar su entrada desde un archivo. Output es la secuencia de salida estándar. Sed carga líneas desde la entrada hasta el espacio patrón, aplica comandos de edición sed al contenido del espació patrón y luego escribe el espacio patrón en el resultado estándar. Sed puede combinar varias líneas en el espacio patrón y puede escribir en un archivo, resultado seleccionado de solo escritura.

Sed usa sintaxis de expresión regular para buscar y reemplazar texto de forma selectiva en el espacio patrón, así como para controlar sobre cuáles líneas de texto deben operar conjuntos de comandos de edición. Las expresiones regulares se cubren de forma más completa en el artículo sobre búsqueda de archivos de texto usando expresiones regulares (el cual usted puede encontrar en el mapa de ruta de la serie). Un almacenamiento intermedio en espera le proporciona almacenamiento temporal para texto. El almacenamiento intermedio en espera puede reemplazar el espacio patrón, ser añadido al espacio patrón, o ser intercambiado con el espacio patrón. Sed tiene un conjunto de comandos limitado, pero estos combinados con sintaxis de expresión regular y con el almacenamiento intermedio en espera constituyen capacidades sorprendentes. Un conjunto de comandos sed es llamado normalmente un script sed.

El Listado 20 muestra tres scripts sed simples. En el primero, usamos el comando `s` (sustituir) para sustituir una 'a' en letra mayúscula por letra minúscula en cada línea. Este ejemplo solo reemplaza la primera 'a', por lo que en el segundo ejemplo añadimos la banderilla 'g' (de global) para hacer que sed cambie todas las ocurrencias. En el tercer script, introducimos el comando `d` (delete) para eliminar una línea. En nuestro ejemplo usamos una dirección de 2 para señalar que solo la línea 2 debe eliminarse. Separamos los comandos usando punto y coma (;) y usamos la misma sustitución global que usamos en el segundo script para reemplazar 'a' por 'A'.

Listado 20. Scripts sed iniciales

```
[ian@echidna lpi103-2]$ sed 's/a/A/' text1
1 Apple
2 peAr
3 bAnana
[ian@echidna lpi103-2]$ sed 's/a/A/g' text1
1 Apple
2 peAr
3 bAnAnA
[ian@echidna lpi103-2]$ sed '2d;$s/a/A/g' text1
1 apple
3 bAnAnA
```

Además de operar sobre líneas individuales, sed can puede operar sobre un rango de líneas. El inicio y el final del rango es separado por una coma (,) y puede especificarse como un número de línea, una expresión regular o un signo de moneda ($) para el final del archivo. Dada una dirección o un rango de direcciones, usted puede agrupar varios comandos entre llaves { y }, para hacer que estos comandos operen solo sobre las líneas seleccionadas por el rango. El Listado 21 ilustra dos formas de aplicar nuestra sustitución global

únicamente a las dos últimas líneas de nuestro archivo. Este también ilustra el uso de la opción `-e` para añadir múltiples comandos al script.

Listado 21. Direcciones Sed

```
[ian@echidna lpi103-2]$ sed -e '2,${' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
[ian@echidna lpi103-2]$ sed -e '/pear/,/bana/{' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
```

Los scripts sed pueden almacenarse en archivos. De hecho usted probablemente desee hacer esto para scripts usados con frecuencia. Recuerde que antes usamos el comando `tr` para cambiar los espacios en blanco del text1 por tabulaciones. Hagamos ahora eso con un script sed almacenado en un archivo. Usaremos el comando `echo` para crear el archivo. Los resultados se muestran en el Listado 22.

Listado 22. Condensado de un sed

```
[ian@echidna lpi103-2]$ echo -e "s/ /\t/g">sedtab
[ian@echidna lpi103-2]$ cat sedtab
s/ /    /g
[ian@echidna lpi103-2]$ sed -f sedtab text1
1       apple
2       pear
3       banana
```

Existen muchos sed condensados útiles como el del Listado 22. Consulte Recursos para enlaces hacia algunos.

Nuestro ejemplo final usa el comando `=` para imprimir números de línea y luego filtrar nuevamente el resultado usando sed para imitar el efecto del comando `nl` para numerar líneas. El Listado 23 usa `=` para imprimir números de línea y luego usa el comando `N` para leer una segunda línea de entrada dentro del espacio patrón y final mente remueve el carácter de línea nueva (\n) que hay entre las dos líneas del espacio patrón.

Listado 23. Numerando líneas con sed

```
[ian@echidna lpi103-2]$ sed '=' text2
1
9       plum
2
3       banana
3
10      apple
[ian@echidna lpi103-2]$ sed '=' text2|sed 'N;s/\n//'
19      plum
23      banana
310     apple
```

¿No es exactamente lo que deseábamos! Lo que realmente querríamos sería que nuestros números se alinearan en una columna con algún espacio antes de las líneas del archivo. En el Listado 24, ingresaremos

varias líneas de comandos (note el > prompt secundario). Estudie ele ejemplo y refiérase a la explicación de abajo.

Listado 24. Numerando líneas con sed - segundo round

```
[ian@echidna lpi103-2]$ cat text1 text2 text1 text2>text6
[ian@echidna lpi103-2]$ ht=$(echo -en "\t")
[ian@echidna lpi103-2]$ sed '=' text6|sed "N
> s/^/      /
> s/^.*\(.....\)\n/\1$ht/"
     1  1 apple
     2  2 pear
     3  3 banana
     4  9       plum
     5  3       banana
     6  10      apple
     7  1 apple
     8  2 pear
     9  3 banana
    10  9       plum
    11  3       banana
    12  10      apple
```

Estos son los pasos que seguimos:

1. Primero usamos `cat` para crear un archivo de doce líneas de dos copias de cada uno de nuestros archivos text1 y text2. No es divertido dar formato a los números que hay en las columnas si no tenemos números de dígitos que difieran.
2. El bash shell usa la tecla de tabulación para completar comandos, por lo que puede ser útil tener un carácter de tabulación cautivo que pueda usar cuando desee una tabulación real. Usamos el comando `echo` para lograr esto y guardamos el carácter en la variable shell 'ht'.
3. Creamos una secuencia que contenga números de línea seguidos por líneas de datos como hicimos antes y lo filtramos a través de una segunda copia de sed.
4. Leemos una segunda línea en el espacio patrón
5. Damos un prefijo a nuestro número de línea al inicio del espacio patrón (denotado por ^) con seis espacios en blanco.
6. Cuando sustituimos toda la línea hasta la nueva línea con los últimos seis caracteres antes de la nueva línea más un carácter de tabulación. Esto alineará nuestros números de línea en las seis primeras columnas de la línea de resultados. Note que la parte izquierda del comando 's' usa '\(' y '\)' para marcar los caracteres que deseamos usar en la parte derecha. En la parte derecha, referenciamos el primer conjunto marcado así (y solo ese conjunto en este ejemplo) como \1. Note que nuestro comando está contenido entre comillas dobles (") de manera que la sustitución ocurra para $ht.

La versión 4 de sed contiene documentación en formato `info` e incluye muchos ejemplos excelentes. Estos no están incluidos en la versión más antigua 3.02. GNU sed aceptará `sed --version` para mostrar la versión.

Sobre el autor

Ian Shields

Ian Shields trabaja en múltiples proyectos Linux para la zona Linux developerWorks. Es programador sénior en IBM en Research Triangle Park, NC. Se unió a IBM en Canberra, Australia, como Ingeniero de Sistemas en 1973 y desde entonces ha trabajado en sistemas de comunicaciones e informática ubicua en Montreal, Canadá, y en el RTP, NC. Cuenta con varias patentes y ha publicado bastantes artículos. Su diploma de pre-grado es en matemática pura y filosofía de la Universidad Nacional de Australia. Tiene una maestría y un doctorado en ciencias de la computación de la Universidad Estatal de Carolina del Norte. Aprenda más sobre Ian en el perfil de Ian en My developerWorks.