

Aprenda > tecnología Java

Introducción a la manipulación de sistemas basados en UNIX

Utilizar herramientas estándar



Brad Yoes

Publicado en 14-08-2012

Un postulado básico de la filosofía UNIX es crear programas (o procesos) que hacen una sola cosa (o una sola filosofía que exige pensar cuidadosamente acerca de las interfaces y las formas de unir esto suerte, más simples) para crear resultados útiles. Normalmente, datos textuales fluyen entre procesos. Se han desarrollado herramientas de procesamiento de texto cada vez más avanzadas. En el pasado había perl, después llegó python y ruby. Mientras que éstos y otros lenguajes son procesados, dichas herramientas no siempre están disponibles, especialmente en el entorno de producción. Como demostraciones de un grupo de comandos básicos de procesamiento de texto UNIX y se los combinan entre sí para resolver problemas que también pueden ser abordados con lenguajes de scripting, un ejemplo brinda más información que leer explicaciones interminables. Por favor, observe los comandos UNIX y similares a UNIX que hay disponibles, las banderas de comandos, el comportamiento y cómo difieren entre las implementaciones.

Uso de cat

El comando cat es uno de los comandos más básicos. Se usa para crear, adjuntar, mostrar y eliminar archivos.

Podemos crear un archivo con cat utilizando '>' para redirigir una entrada estándar (stdin) a un archivo. '>' trunca los contenidos del archivo de salida especificado. El texto ingresado después de este punto se especifica a la derecha del operador '>'. Control-d señala un fin-de-archivo, devolviendo el control al prompt.

```
1 $ cat > grocery.list
2 apples
3 bananas
4 plums
5 <ctrl-d>
6 $
```

Use el operador '>>' para adjuntar entradas estándar a un archivo existente.

Ejemplo de cat para adjuntar un archivo:

```
1 $ cat >> grocery.list
2 carrots
3 <ctrl-d>
```

Examine los contenidos del archivo grocery.list utilizando cat sin banderas. Note cómo los contenidos de la redirección y adjuntan ejemplos de operador.

Ejemplo de cat sin banderas:

```
1 $ cat grocery.list
2 apples
3 bananas
4 plums
5 carrots
```

El comando cat se puede utilizar para enumerar las líneas de un archivo.

Ejemplo de cat para contar líneas:

```
1 $ cat -n grocery.list
2      1 apples
3      2 bananas
4      3 plums
5      4 carrots
```

Uso de nl

El filtro nl lee líneas de stdin o de los archivos especificados. El resultado se escribe en stdout, en un archivo o a otro proceso. El comportamiento de nl se controla a través de varias opciones.

De manera predeterminada, nl cuenta las líneas que son similares a cat -n.

```

1 $nl grocery.list
2     1 apples
3     2 bananas
4     3 plums
5     4 carrots

```

Use la bandera `-b` para especificar las líneas a numerar. Esta bandera toma un “tipo” como `a` a qué líneas necesitan ser numeradas – use ‘a’ para numerar todas las líneas, ‘t’ le dice a `nl` que las líneas que sólo son espacio blanco, ‘n’ especifica que no hay que numerar líneas. En el ejemplo de patrón, `nl` numera las líneas especificadas por un patrón de expresión regular, en este caso las letras ‘a’ o ‘b’.

Ejemplo de `nl` para numerar líneas conforme a una expresión regular:

```

1 $ nl -b p^[ba] grocery.list
2     1 apples
3     2 bananas
4     plums
5     carrots

```

De manera predeterminada, `nl` separa el número de la línea del texto usando una tabulación como delimitador distinto, como por ejemplo el signo ‘=’.

Ejemplo de `nl` para especificar un delimitador:

```

1 $nl -s= grocery.list
2     1=apples
3     2=bananas
4     3=plums
5     4=carrots

```

Uso de `wc`

El comando `wc` (wordcount) cuenta el número de líneas, palabras (separadas por espacio blanco) especificadas o de `stdin`.

Ejemplos de uso de `wc`:

```

1 $wc grocery.list
2     4      4      29 grocery.list
3 $wc -l grocery.list
4     4 grocery.list
5 $wc -w grocery.list
6     4 grocery.list

```



Usando grep

El comando `grep` busca archivos especificados o `stdin` en busca de patrones que concuerden. La salida de `grep` es controlada por varias banderas de opción.

A modo de demostración, se ha creado un archivo nuevo para usar `grocery.list`.

```
1 $cat grocery.list2
2 Apple Sauce
3 wild rice
4 black beans
5 kidney beans
6 dry apples
```

developerWorks®

Aprenda

Desarrolle

Conéctese

Ejemplo del uso básico de `grep`:

Contenido

```
1 $ grep apple grocery.list grocery.list2
2 grocery.list:apples
3 grocery.list2:dry apples
```

... de cat

`grep` tiene un número considerable de banderas de opción. A continuación se presentan algunas de las opciones más comunes.

Uso de `wc`

Para mostrar el nombre de archivo (cuando se usa `-l`), `grep` puede mostrar el nombre de archivo y el número de líneas que coinciden con el patrón.

Corrientes, tuberías, redirecciones, tee, etc.

documento-aquí

Ejemplo de `grep` - contar el número de coincidencias:

```
1 $ grep -c apple grocery.list grocery.list2
2 grocery.list:1
3 grocery.list2:1
```

Uso de `colrm`

Al buscar en múltiples archivos, utilizar la opción `-l` para suprimir la impresión de nombres de archivo.

Uso de `expand` y `unexpand`

Ejemplo de `grep` - suprimir nombre de archivo:

Uso de `comm`, `cmp`, y `diff`

```
1 $ grep -h apple grocery.list grocery.list2
2 apples
3 dry apples
```



Ejemplo de grep – sin diferenciación de mayúsculas y minúsculas:

```
1 | $ grep -i apple grocery.list grocery.list2
2 |
3 | grocery.list:apples
4 | grocery.list2:Apple Sauce
5 | grocery.list2:dry apples
```

A veces sólo se necesita imprimir el nombre de archivo y no la línea de concordancia de patrón. Se puede hacer imprimiendo sólo los nombres de archivo que contienen líneas con un patrón concordante.

Ejemplo de grep – sólo nombres de archivo:

```
1 | $ grep -l carrot grocery.list grocery.list2
2 | grocery.list
```

Los números de línea se pueden proveer como parte del resultado. Use la opción -n para incluirlos.

Ejemplo de grep – incluir números de línea:

```
1 | $ grep -n carrot grocery.list grocery.list2
2 | grocery.list:4:carrots
```

Hay ocasiones en las que el resultado deseado son las líneas que no concuerdan con el patrón. Se puede hacer usando la opción -v.

Ejemplo de grep – concordancia invertida:

```
1 | $ grep -v beans grocery.list2
2 | Apple Sauce
3 | wild rice
4 | dry apples
```

A veces el patrón deseado forma una “palabra” rodeada de espacio blanco u otros caracteres. La mayoría de las versiones de grep provee una opción -w para facilitar la escritura de búsquedas de palabras.

Ejemplo de grep – concordancia de palabra:

```
1 | $ grep -w apples grocery.list grocery.list2
2 | grocery.list:apples
3 | grocery.list2:dry apples
```



- `stdin = 0`
- `stdout = 1`
- `stderr = 2`

Ejemplo de fusión y división de corrientes estándar:

Ejemplo de redireccionamiento para hacer un archivo de copia de seguridad:

Ejemplo de tubería simple a nl:

6/31

5 | 4 carrots

Otro ejemplo mostrado antes fue cómo hacer una búsqueda sin diferenciación de mayúsculas un patrón. Esto se puede hacer usando redireccionamiento - en este caso desde `stdin`, o es similar al ejemplo anterior de tubería simple.

Ejemplo con `grep` - redireccionamiento `stdin` y conexión:

```
1 | $ grep -i apple < grocery.list2
2 | Apple Sauce
3 | dry apples
4 | $cat grocery.list2 | grep -i apple
5 | Apple Sauce
6 | dry apples
```

En algunas situaciones un bloque de texto será redireccionado a un comando o a un archivo. El mecanismo para lograr esto es el uso de 'documento-aquí' o 'here-doc'. Para incorporar documento utiliza el operador '`<<`' para redireccionar el texto siguiente hasta que alcanza el delimitador especificado después del operador `<<`.

Ejemplo de documento-aquí básico en la línea de comando:

```
1 | $ cat << EOF
2 | > oranges
3 | > mangos
4 | > pinapples
5 | > EOF
6 | oranges
7 | mangos
8 | pinapples
```

Este resultado se puede redireccionar a un archivo, en este ejemplo, el delimitador cambió a `c` (explicado después) se usa para cambiar a mayúsculas las letras con documento-aquí.

Ejemplo de documento-aquí básico redireccionado a un archivo:

```
1 | cat << ! > grocery.list3
2 | oranges
3 | mangos
4 | pinapples
5 | !
6 | $ cat grocery.list3
7 | oranges
8 | mangos
9 | pinapples
10 | $tr [:lower:] [:upper:] << !
11 | > onions
```



Usando head y tail

Los comandos `head` y `tail` se usan para examinar las partes superiores (cabecera) o inferiores (cola) de un archivo. Para mostrar la parte superior de dos líneas y la parte inferior de dos líneas de un archivo use los comandos `head -n 2` y `tail -n 2`, respectivamente. De forma similar, la opción `-c` muestra los primeros o los últimos caracteres de un archivo.

Ejemplo de uso básico de los comandos `head` y `tail`:

```
1 $ head -n2 grocery.list
2 apples
3 bananas
4 $ tail -n2 grocery.list
5 plums
6 carrots
7 $ head -c12 grocery.list
8 apples
9 bananas
10 $ tail -c12 grocery.list
11 plums
12 carrots
```

Un uso común del comando `tail` es la observación de archivos de registro o la salida de programas para detectar problemas, o para notar cuándo termina un proceso. La opción `-f` (`tail -f`) hace que `tail` siga mostrando resultados después de alcanzar el marcador fin-de-archivo, y continúe mostrando resultados cuando la salida se reinicia.

Usando tr

El comando `tr` se utiliza para traducir caracteres desde `stdin`, mostrándolos en `stdout`. En los conjuntos de caracteres y reemplaza caracteres del primer conjunto con caracteres del segundo conjunto. Las clases (conjuntos) de caracteres predefinidos están disponibles para ser usados por `tr`, y algunas de ellas se muestran a continuación.

Estas clases son:

- `alnum` - caracteres alfanuméricos
- `alpha` - caracteres alfabéticos
- `blank` - caracteres de espacio blanco
- `cntrl` - caracteres de control
- `digit` - caracteres numéricos



-
- `lower` - caracteres alfabéticos en minúsculas
 - `print` - caracteres imprimibles
 - `punct` - caracteres de puntuación
 - `space` - caracteres de espacio
 - `upper` - caracteres en mayúsculas
 - `xdigit` - caracteres hexadecimales

El comando `tr` puede traducir caracteres en minúsculas de una cadena a mayúsculas.

Ejemplo `tr` - cambiar una cadena a mayúsculas:

```
1 | $ echo "Who is the standard text editor?" |tr [:lower:] [:upper:]
2 | WHO IS THE STANDARD TEXT EDITOR?
```

`tr` se puede usar para eliminar caracteres con nombre de una cadena.

Ejemplo `tr` - eliminar caracteres de una cadena:

```
1 | $ echo 'ed, of course!' |tr -d aeiou
2 | d, f crs!
```

Use `tr` para traducir caracteres con nombre en una cadena a espacio. Cuando se encuentran múltiples en una secuencia, se traducen a un espacio simple.

El comportamiento de la bandera de opción `-s` difiere entre sistemas.

Ejemplo `tr` - traducir caracteres a un espacio:

```
1 | $ echo 'The ed utility is the standard text editor.' |tr -s astu ' '
2 | The ed ili y i he nd rd ex edi or.
```

La bandera de opción `-s` se puede usar para suprimir espacio blanco extra de una cadena.

```
1 | $ echo 'extra      spaces - 5' | tr -s [:blank:]
2 | extra spaces - 5
3 | $ echo 'extra      tabs - 2' | tr -s [:blank:]
4 | extra  tabs - 2
```



carro seguido por una línea nueva. Usar `tr` con algo de redireccionamiento es una forma de `tr` formato.

Ejemplo `tr` - quitar retorno de carro:

```
1 | $ tr -d '\r' < dosfile.txt > unixfile.txt
```

Uso de `colrm`

Al usar `colrm` se pueden cortar columnas de texto desde una corriente. En el primer ejemplo columna 4 hasta el final de la línea para cada línea de la tubería. Luego, el mismo archivo se columnas 4 a 5.

Ejemplo de `colrm` para quitar columnas:

```
1 | $ cat grocery.list |colrm 4
2 | app
3 | ban
4 | plu
5 | car
6 | $ cat grocery.list |colrm 4 5
7 | apps
8 | banas
9 | plu
10 | carts
```

Uso de `expand` y `unexpand`

El comando `expand` cambia las tabulaciones a espacios, mientras que `unexpand` cambia espacios a tabulaciones. Los comandos sacan el resultado de `stdin` o de los archivos nombrados en la línea de comando. Se puede establecer una o más posiciones de tabulador.

Ejemplo de `expand` y `unexpand`:

```
1 | $ cat grocery.list|head -2|nl|nl
2 |      1      1  apples
3 |      2      2  bananas
4 | $ cat grocery.list|head -2|nl|nl|expand -t 5
5 |      1      1  apples
6 |      2      2  bananas
7 | $ cat grocery.list|head -2|nl|nl|expand -t 5,20
8 |      1      1  apples
9 |      2      2  bananas
```



Uso de comm, cmp, y diff

Para demostrar estos comandos se han creado dos archivos nuevos.

Crear archivos para la demostración:

```
1 cat << EOF > dummy_file1.dat
2 011 IBM 174.99
3 012 INTC 22.69
4 013 SAP 59.37
5 014 VMW 102.92
6 EOF
7 cat << EOF > dummy_file2.dat
8 011 IBM 174.99
9 012 INTC 22.78
10 013 SAP 59.37
11 014 vmw 102.92
12 EOF
```

El comando `diff` compara dos archivos, informando las diferencias que hay entre ellos. `diff` admite varias opciones. En el ejemplo siguiente, se muestra primero un `diff` predeterminado seguido por uno que ignora espacio blanco, luego termina con un ejemplo de la bandera de opción `-i` que ignora mayúsculas y minúsculas cuando hace las comparaciones.

Ejemplos del comando `diff`:

```
1 $ diff dummy_file1.dat dummy_file2.dat
2 1,2c1,2
3 < 011 IBM 174.99
4 < 012 INTC 22.69
5 ---
6 > 011 IBM 174.99
7 > 012 INTC 22.78
8 4c4
9 < 014 VMW 102.92
10 ---
11 > 014 vmw 102.92
12
13 $ diff -w dummy_file1.dat dummy_file2.dat
14 2c2
15 < 012 INTC 22.69
16 ---
17 > 012 INTC 22.78
18 4c4
19 < 014 VMW 102.92
20 ---
21 > 014 vmw 102.92
22
23 $ diff -i dummy_file1.dat dummy_file2.dat
```



```

27 ---
28 > 011 IBM 174.99
29 > 012 INTC 22.78

```

El comando `comm` compara dos archivos pero se comporta de forma bastante distinta que `diff`. El resultado - sólo líneas de `file1` (columna 1), sólo líneas de `file2` (columna 2) y líneas comunes. Las banderas de opción se pueden utilizar para suprimir columnas de salida. Probablemente se quiere suprimir la columna 1 y la columna 2, mostrando únicamente las líneas que son comunes a ambos archivos. Como se ve en el ejemplo de comando `comm` abajo.

Ejemplo de comando `comm`:

```

1 $ comm dummy_file1.dat dummy_file2.dat
2      011 IBM 174.99
3 011 IBM 174.99
4 012 INTC 22.69
5      012 INTC 22.78
6      013 SAP 59.37
7 014 VMW 102.92
8      014 vmw 102.92
9
10 $ comm -12 dummy_file1.dat dummy_file2.dat
11 013 SAP 59.37

```

El comando `cmp` también compara dos archivos. Sin embargo, a diferencia de `comm` o `diff`, el comando `cmp` (predeterminada) informa el primer número de byte y de línea donde los dos archivos difieren.

Ejemplo de comando `cmp`:

```

1 $ cmp dummy_file1.dat dummy_file2.dat
2 dummy_file1.dat dummy_file2.dat differ: char 5, line 1

```

Usar fold

Cuando se usa el comando `fold` las líneas se dividen en un ancho especificado. Originalmente se usaba para ayudar a dar formato a texto para dispositivos de salida con ancho fijo que no tenían la bandera de opción `-w` permite la especificación del ancho de una línea para usar en lugar de anchos predeterminados.

Ejemplo de uso de `fold`:

```

1 $ fold -w8 dummy_file1.dat
2 011 IBM

```

```
6 | 013 SAP
7 | 59.37
8 | 014 VMW
9 | 102.92
```

Usando paste

El comando paste se utiliza para alinear archivos lado a lado, fusionando registros de cada a se usa redireccionamiento se pueden crear nuevas filas uniendo cada registro de un archivo

Crear archivos para la demostración:

```
1 | cat << EOF > dummy1.txt
2 | IBM
3 | INTC
4 | SAP
5 | VMW
6 | EOF
7 | cat << EOF > dummy2.txt
8 | 174.99
9 | 22.69
10 | 59.37
11 | 102.92
12 | EOF
```

Ejemplo 1 de paste - líneas de múltiples archivos:

```
1 | $ paste dummy1.txt dummy2.txt grocery.list
2 | IBM      174.99  apples
3 | INTC     22.69  bananas
4 | SAP      59.37  plums
5 | VMW      102.92  carrots
```

Hay una bandera de opción -s para procesar los archivos de a uno (en serie) en lugar de hacerlos como las columnas se alinean con las filas del ejemplo anterior.

Ejemplo 2 de paste - líneas de múltiples archivos:

```
1 | $ paste -s dummy1.txt dummy2.txt grocery.list
2 | IBM      INTC      SAP      VMW
3 | 174.99    22.69     59.37    102.92
4 | apples    bananas   plums     carrots
```



condensado en una línea sola, use un delimitador para separar los campos retornados (tabu predeterminado). En este ejemplo, el comando `find` se usa para localizar directorios donde bibliotecas de 64 bits, y desarrolla una ruta adecuada para adjuntar a una variable `$LD_LIBRARY_PATH`.

Ejemplos de `paste` - con delimitador:

```
1 $ find /usr -name lib64 -type d | paste -s -d:
2 /usr/lib/qt3/lib64:/usr/lib/debug/usr/lib64:/usr/X11R6/lib/X11/locale/lib64:/u
3 lib64:/usr/lib64:/usr/local/ibm/gsk7_64/lib64:/usr/local/lib64
4
5 $ paste -d, dummy1.txt dummy2.txt
6 IBM,174.99
7 INTC,22.69
8 SAP,59.37
9 VMW,102.92
```

Usando bc

Para usar una forma sencilla de hacer cálculos en shell, considere `bc`, la “calculadora básica”. Algunos shells ofrecen formas de hacer cálculos nativamente, otros dependen de `expr` para usar `bc`, los cálculos se pueden transportar entre shells y sistemas UNIX, pero tenga cuidado proveedores.

Ejemplo de `bc` – cálculos simples:

```
1 $ echo 2+3 | bc
2 5
3
4 $ echo 3*3+2 | bc
5 11
6
7 $ VAR1=$(echo 2^8 | bc)
8 $ echo $VAR1
9 256
10
11 $ echo "(1+1)^8" | bc
12 256
```

`bc` puede realizar más que estos simples cálculos. Es un intérprete que define sus propias funciones internas y definidos por usuarios, de manera similar a un lenguaje de programación. De manera que los dígitos a la derecha de los decimales. Aumente la precisión del resultado utilizando `scale`. La muestra el ejemplo, `bc` escala a números grandes e instrumenta una precisión proporcional a la base de conversión para los números de entrada y salida. En el ejemplo siguiente:

- el soporte de números grandes se ilustra calculando 2 a la 128
- la función interna `sqrt()` es necesaria para calcular la raíz cuadrada de 2
- Desde `ksh`, calcule e imprima un porcentaje

Ejemplo de `bc` – más cálculos:

```

1  $ echo "obase=16; 2^8-1"|bc
2  FF
3
4  $ echo "99/70"|bc
5  1
6
7  $ echo "scale=20; 99/70"|bc
8  1.41428571428571428571
9
10 $ echo "scale=20;sqrt(2)"|bc
11 1.41421356237309504880
12
13 $ echo 2^128|bc
14 340282366920938463463374607431768211456
15
16 $ printf "Percentage: %2.2f%%\n" $(echo .9963*100|bc)
17 Percentage: 99.63%
```

La página principal de `bc` es detallada e incluye ejemplos.

Usando `split`

Una tarea útil del comando `split` es dividir grandes archivos de datos en archivos más pequeños. Un ejemplo se muestra que `BigFile.dat` tiene 165782 líneas que usan el comando `wc`. La bandeja muestra el número máximo de líneas de cada archivo de salida. `split` permite especificar un prefijo | salida, debajo, `BigFile_` es el prefijo especificado. Otras opciones permiten el control del sufijo. La opción `-p` permite que se produzcan divisiones a una expresión regular como el comando `cs`. Consulte la página principal para obtener más información.

Ejemplo de `split`:

```

1  $ wc BigFile.dat
2  165782  973580 42557440 BigFile.dat
3
4  $ split -l 15000 BigFile.dat BigFile_
5
6  $ wc BigFile*
7  165782  973580 42557440 BigFile.dat
8  15000    87835 3816746 BigFile_aa
9  15000    88483 3837494 BigFile_ab
```



```

13      7514    43817 1908914 BigFile_af
14      248296 1459578 63725149 total

```

Usando cut

El comando `cut` se utiliza para ‘recortar’ secciones basadas en columnas de un archivo o de `stdin`. Corta datos en bytes (`-b`), caracteres (`-c`), o campos (`-f`) según lo especifique la lista. posiciones de byte/carácter son especificadas usando listas separadas por comas y guiones posición o el campo deseado si sólo se necesita el resultado de uno. Se puede especificar un guión de modo que 1-3 imprima los campos (o posiciones) 1 a 3, -2 imprima desde el comienzo (ó byte/carácter 2), y 3- especifique a `cut` que imprima el campo (ó posición) 3 hasta el final múltiples campos usando una coma. Otras banderas que pueden ser útiles son `-d` para especificar el delimitador, `-s`, para suprimir líneas sin delimitadores.

Ejemplos de `cut`:

```

1  $ cat << EOF > dummy_cut.dat
2  # this is a data file
3  ID,Name,Score
4  13BA,John Smith,100
5  24BC,Mary Jones,95
6  34BR,Larry Jones,94
7  36FT,Joe Ruiz,93
8  40RM,Kay Smith,91
9  EOF
10
11 $ cat dummy_cut.dat |cut -d, -f1,3
12 # this is a data file
13 ID,Score
14 13BA,100
15 24BC,95
16 34BR,94
17 36FT,93
18 40RM,91
19
20 $ cat dummy_cut.dat |cut -b6-
21 s is a data file
22 me,Score
23 John Smith,100
24 Mary Jones,95
25 Larry Jones,94
26 Joe Ruiz,93
27 Kay Smith,91
28
29 $ cat dummy_cut.dat |cut -f1- -d, -s
30 ID,Name,Score
31 13BA,John Smith,100
32 24BC,Mary Jones,95
33 34BR,Larry Jones,94
34 36FT,Joe Ruiz,93
35 40RM,Kay Smith,91

```



Usando uniq

El comando `uniq` se usa típicamente para enumerar listas de forma única desde una fuente (archivo o `stdin`). Para operar apropiadamente, las líneas duplicadas deben ser posicionadas Normalmente, la entrada al comando `uniq` se ordena, por lo tanto las líneas duplicadas se a que se utilizan con el comando `uniq` son `-c`, para imprimir el conteo del número de veces qu puede utilizar para mostrar una instancia de líneas duplicadas.

Ejemplos de `uniq`:

```

1  $ cat << EOF > dummy_uniq.dat
2
3  13BAR  Smith  John   100
4  13BAR  Smith  John   100
5  24BC   Jone   Mary   95
6  34BRR  Jones  Larry  94
7  36FT   Ruiz   Joe    93
8  40REM   Smith  Kay    91
9  13BAR   Smith  John   100
10 99BAR   Smith  John   100
11 13XIV   Smith  Cindy  91
12
13
14 EOF
15
16 $ cat dummy_uniq.dat | uniq
17
18 13BAR   Smith  John   100
19 24BC    Jone   Mary   95
20 34BRR   Jones  Larry  94
21 36FT    Ruiz   Joe    93
22 40REM   Smith  Kay    91
23 13BAR   Smith  John   100
24 99BAR   Smith  John   100
25 13XIV   Smith  Cindy  91
26
27 $ cat dummy_uniq.dat | sort |uniq
28
29 13BAR   Smith  John   100
30 13XIV   Smith  Cindy  91
31 24BC    Jone   Mary   95
32 34BRR   Jones  Larry  94
33 36FT    Ruiz   Joe    93
34 40REM   Smith  Kay    91
35 99BAR   Smith  John   100
36
37 $ cat dummy_uniq.dat | sort |uniq -d
38
39 13BAR   Smith  John   100
40
41 $ cat dummy_uniq.dat | sort |uniq -c
42 3
43 3 13BAR   Smith  John   100
44 1 13XIV   Smith  Cindy  91
45 1 24BC    Jone   Mary   95

```



49 | 1 99BAR Smith John 100

Usando sort

Para ordenar filas en `stdin` o un archivo en un orden en particular tal como alfabético o numérico. De manera determinada, el resultado de `sort` se escribe en `stdout`. Las variables del `LC_COLLATE`, ó `LANG` pueden afectar el resultado de `sort` y otros comandos. Observe cómo registros duplicados separados – un doble de IBM y el otro doble es una línea vacía.

Ejemplo de `sort` – comportamiento predeterminado:

```

1  $ cat << EOF > dummy_sort1.dat
2
3  014 VMW, 102.92
4  013 INTC, 22.69
5  012 sap, 59.37
6  011 IBM, 174.99
7  011 IBM, 174.99
8
9  EOF
10
11 $ sort dummy_sort1.dat
12
13
14 011 IBM, 174.99
15 011 IBM, 174.99
16 012 sap, 59.37
17 013 INTC, 22.69
18 014 VMW, 102.92

```

`sort` tiene una bandera excelente para sustituir el comando `uniq` en muchas circunstancias. el archivo, quita las filas duplicadas de modo que se produzca un listado de filas únicas de re

Ejemplo de `sort` – ordenar único:

```

1  $ sort -u dummy_sort1.dat
2
3  011 IBM, 174.99
4  012 sap, 59.37
5  013 INTC, 22.69
6  014 VMW, 102.92

```

A veces se necesita el ordenamiento revertido de la entrada. De manera predeterminada, se ordena en orden descendente (numérico) y en orden alfabético para los datos de caracteres. Use la bandera de opción `-r` para ordenamiento predeterminado.

```
1 $ sort -ru dummy_sort1.dat
2 014 VMW, 102.92
3 013 INTC, 22.69
4 012 sap, 59.37
5 011 IBM, 174.99
```

Distintas situaciones requieren que un archivo sea ordenado en ciertos campos o “claves”. A la bandera de opción `-k` que permite especificar una llave de ordenamiento por posición. Los campos se ordenan de forma determinada.

Ejemplo de sort – ordenar con una clave:

```
1 $ sort -k2 -u dummy_sort1.dat
2
3 011 IBM, 174.99
4 013 INTC, 22.69
5 014 VMW, 102.92
6 012 sap, 59.37
```

Cuando la diferenciación de mayúsculas y minúsculas representa un problema, `sort` provee la bandera de opción `-f` que ignora mayúsculas y minúsculas al hacer comparaciones. Cuando se combinan banderas múltiples, algunas versiones de UNIX necesitan que estas banderas estén especificadas en un orden determinado.

Ejemplo de sort – ordenar ignorando mayúsculas y minúsculas:

```
1 $ sort -k2 -f -u dummy_sort1.dat
2
3 011 IBM, 174.99
4 013 INTC, 22.69
5 012 sap, 59.37
6 014 VMW, 102.92
```

Hasta ahora, todos los ordenamientos han sido del tipo alfabético. Cuando necesite ordenar numéricamente, uso la bandera de opción `-n`.

Ejemplo de sort – ordenamiento numérico:

```
1 $ sort -n -k3 -u dummy_sort1.dat
2
3 013 INTC, 22.69
4 012 sap, 59.37
5 014 VMW, 102.92
6 011 IBM, 174.99
```



Ejemplo de sort – ordenar campo usando delimitador no predeterminado:

```
1 $ sort -k2 -t"," -un dummy_sort1.dat
2
3 013 INTC, 22.69
4 012 sap, 59.37
5 014 VMW, 102.92
6 011 IBM, 174.99
```

Usando join

Cualquier persona familiarizada en la escritura de búsquedas de base de datos reconoce la `join` igual que la mayoría de los comando UNIX, el resultado se muestra en `stdout`. Para “combinar” campos especificados de dos archivos línea por línea. Si hay campos especificados, `join` comienza desde el comienzo de cada línea. El separador de campos predeterminado es el espacio blanco (simplemente un espacio o espacios adyacentes). Cuando se produce una concordancia de campos, se muestran los resultados por cada par de líneas con los campos concordantes. Para obtener resultados legibles, los campos deben estar ordenados por campos para que puedan concordar. No todos los sistemas implementan `join` de esta manera.

Este ejemplo utiliza `-t` para especificar un separador de campo y muestra cómo combinar dos archivos (predeterminado) delimitado por comas. Los operadores de base de datos deberían reconocer esta sintaxis interna que muestra sólo las filas concordantes.

Ejemplo de join – usando delimitador de campo no predeterminado:

```
1 cat << EOF > dummy_join1.dat
2 011,IBM,Palmisano
3 012,INTC,Otellini
4 013,SAP,Snabe
5 014,VMW,Maritz
6 015,ORCL,Ellison
7 017,RHT,Whitehurst
8 EOF
9
10 cat << EOF > dummy_join2.dat
11 011,174.99,14.6
12 012,22.69,10.4
13 013,59.37,26.4
14 014,102.92,106.1
15 016,27.77,31.2
16 EOF
17
18 cat << EOF > dummy_join3.dat
19 IBM,Armonk
20 INTC,Santa Clara
21 SAP,Walldorf
22 VMW,Palo Alto
```



```

26
27 $ join -t, dummy_join1.dat dummy_join2.dat
28 011,IBM,Palmisano,174.99,14.6
29 012,INTC,Otellini,22.69,10.4
30 013,SAP,Snabe,59.37,26.4
31 014,VMW,Maritz,102.92,106.1

```

Para especificar campos para “combinar” en cada archivo se puede usar la bandera de opción `-j1 2` ó `-j1 2` especifica el segundo campo del archivo uno, el primer campo del segundo archivo, y también una combinación interna para concordar únicamente con las filas.

Ejemplo de join – campos especificados:

```

1 $ join -t, -j1 1 -j2 2 dummy_join3.dat dummy_join1.dat
2 IBM,Armonk,011,Palmisano
3 INTC,Santa Clara,012,Otellini
4 SAP,Walldorf,013,Snabe
5 VMW,Palo Alto,014,Maritz
6 ORCL,Redwood City,015,Ellison

```

Continuando con la idea de hacer ejemplos relativos a bases de datos, las banderas pueden especificar una combinación externa de tabla izquierda. Left outer join incluye todas las filas del primer archivo que no concuerdan en el segundo archivo o tabla. Use `-a` para incluir todas las filas del archivo especificado.

Ejemplo de join – left outer join:

```

1 $ join -t, -a1 dummy_join1.dat dummy_join2.dat
2 011,IBM,Palmisano,174.99,14.6
3 012,INTC,Otellini,22.69,10.4
4 013,SAP,Snabe,59.37,26.4
5 014,VMW,Maritz,102.92,106.1
6 015,ORCL,Ellison
7 017,RHT,Whitehurst

```

Full outer joins incluye todas las filas de ambos archivos o tablas, sin importar si los campos coinciden. Full outer join incluye una combinación externa completa especificando ambos archivos con la bandera de opción `-a1 -a2`.

Ejemplo de join – full outer join:

```

1 $ join -t, -a1 -a2 -j1 2 -j2 1 dummy_join1.dat dummy_join3.dat
2 IBM,011,Palmisano,Armonk
3 INTC,012,Otellini,Santa Clara
4 SAP,013,Snabe,Walldorf
5 VMW,014,Maritz,Palo Alto
6 ORCL,015,Ellison,Redwood City
7 EMC,Hopkinton

```



Usando sed

El editor de flujos sed es una herramienta útil de análisis y manipulación de texto que sirve para archivos o flujos de datos. Lee texto línea por línea, aplicando los comandos especificados en una línea determinada, el resultado va a los comandos stdout. El comando sed usa operaciones de edición de texto de un almacenamiento intermedio, adjuntar o insertar texto en un almacenamiento intermedio, transformar texto basado en expresiones regulares y más.

Un ejemplo básico de sustitución sed muestra la bandera de opción -e utilizada para especificar una edición. Se pueden especificar múltiples expresiones o ediciones para una sola ejecución sed. La "s" al principio de la edición indica que esto es un comando de sustitución. El patrón de reemplazo se indica primero el patrón "IBM" a reemplazar. Luego, aparece el patrón de reemplazo "/". Por último, "g" indica que hay que hacer el cambio globalmente en el almacenamiento intermedio. La tercera demostración de este ejemplo ilustra una combinación de tres ediciones: reemplaza comas por guiones bajos, espacios por guiones bajos, y quitar caracteres de dos puntos – note cómo los caracteres de escape.

Ejemplo de sed – sustitución básica / ediciones múltiples:

```
1 $ echo "IBM 174.99" | sed -e 's/IBM/International Business Machines/g'
2 International Business Machines 174.99
3
4 $ echo "Oracle DB" | sed -e 's/Oracle/IBM/g' -e 's/DB/DB2/g'
5 IBM DB2
6
7 $ echo "C:\Program Files\PuTTY\putty.exe" | sed -e 's/\\\/_/' -e 's/ /_/' -e
8 C/Program_Files/PuTTY/putty.exe
```

En el ejemplo siguiente se configura un archivo para demostrar otro recurso de sed. Además de otro uso frecuente de sed. El comando UNIX grep es un filtro empleado comúnmente; no es de manipular el texto en la línea de comando. Este ejemplo muestra cómo utilizar el comando sed que comienzan con "#" ó espacio blanco y luego "#". Se muestra un ejemplo de grep utilizando una referencia.

Ejemplo de sed - filtrado:

```
1 cat << EOF > dummy_sed.txt
2 # top of file
3 # the next line here
4 # Last Name, Phone
5 Smith, 555-1212
6 Jones, 555-5555 # last number
```



```

10 Smith, 555-1212
11 Jones, 555-5555 # last number
12
13 $ grep -v ^[:space:]*# dummy_sed.txt
14 Smith, 555-1212
15 Jones, 555-5555 # last number

```

Para entender mejor el comportamiento de sed se muestran algunos patrones más. Se crean patrones que puedan actuar en cierto texto. El primer patrón sed muestra cómo eliminar los últimos 4 caracteres de un archivo (presentados en el archivo). Luego, el patrón elimina todos los caracteres a la derecha de una extensión de archivo. Debajo se muestra un patrón para eliminar líneas vacías. Un carácter de escape que el patrón de búsqueda sea utilizado como parte del resultado. En este ejemplo, IBM es utilizado y especificado como parte del resultado usando el símbolo et. El último patrón demostrado es cómo se puede usar sed para eliminar retornos de carro de un archivo de texto transferido desde un sistema Windows. Se ingresa “^M” en el script o en la línea de comando presionando primero Control-v y luego p. Como que las características del terminal pueden afectar el ingreso de la combinación control-v, se

Ejemplo de sed – más patrones:

```

1  cat << EOF > filelist.txt
2  PuTTY.exe
3
4  sftp.exe
5  netstat.exe
6  servernames.list
7  EOF
8
9  $ sed 's/....$//' filelist.txt
10 PuTTY
11
12 sftp
13 netstat
14 servernames.
15
16 $ sed 's/\..*$//g' filelist.txt
17 PuTTY
18
19 sftp
20 netstat
21 servernames
22
23 $ sed '/^$/d' filelist.txt
24 PuTTY.exe
25 sftp.exe
26 netstat.exe
27 servernames.list
28
29 $ echo "IBM 174.99" | sed 's/IBM/&-International Business Machines/g'
30 IBM-International Business Machines 174.99
31
32 $ cat dosfile.txt | sed 's/^M//' > unixfile.txt

```

mostrar cada línea de entrada como parte del resultado. En el primer ejemplo, sed opera en modo `stream` (líneas 4 a 7). Note cómo sólo se muestra la primera fila o tabla presentada del archivo (líneas 4 a 7). Luego se muestra la última línea del archivo. La mayoría de las versiones de sed permiten que los patrones especiales se apliquen al comando. Note que en el resultado sólo las comas, y no los comentarios, son

Ejemplo de sed – rangos de direcciones:

```

1  cat << EOF > dummy_table.frag
2
3  <p>This, is a paragraph.</p>
4  <table border="1">
5  <tr>
6  <td>row 1, 1st cell</td>
7  <td>row 1, 2nd cell</td>
8  </tr>
9  <tr>
10 <td>row 2, 1st cell</td>
11 <td>row 2, 2nd cell</td>
12 </tr>
13 </table>
14 <!--This, is another comment. -->
15 EOF
16
17 $ sed -n 4,7p dummy_table.frag
18 <tr>
19 <td>row 1, 1st cell</td>
20 <td>row 1, 2nd cell</td>
21 </tr>
22
23 $ sed -n -e 1p -e \ $p dummy_table.frag
24 <!--This, is a comment. -->
25 <!--This, is another comment. -->
26
27 $ sed '/^&lt;table/,/^&lt;\/table/s/,//g' dummy_table.frag
28 <!--This, is a comment. -->
29 <p>This, is a paragraph.</p>
30 <table border="1">
31 <tr>
32 <td>row 1 1st cell</td>
33 <td>row 1 2nd cell</td>
34 </tr>
35 <tr>
36 <td>row 2 1st cell</td>
37 <td>row 2 2nd cell</td>
38 </tr>
39 </table>
40 <!--This, is another comment. -->

```

Los patrones que están dentro de una expresión pueden ser agrupados y luego referenciados. Esto puede resultar útil en una variedad de contextos tales como el intercambio de valores. Las corchetes y las corchetes se utilizan para resaltar patrones en la expresión y se los debe separar con una barra invertida (pattern-here\). El patrón se referencia en otras partes de la expresión utilizando \n donde n

Patrón	Comentarios
<code>/^#.*\$/d</code>	eliminar de líneas de resultado que comiencen con <code>#</code>
<code>/^\$/d</code>	eliminar de líneas vacías del resultado
<code>s/\([[:a-z:]]*\):\(.*)\n/\2:\1 /</code>	Este argumento marca la primera cadena de caracteres en minúsculas y luego marca la cadena de caracteres que le sigue a resultado, estas cadenas marcadas cambian de posición.

Ejemplo de sed – agrupar patrones:

```

1  cat << EOF > sed_chown_example.txt
2  # use sed to swap the group:owner to owner:group
3
4  sudo chown dba:jdope oraenv.ksh
5  sudo chown staff:jdope sysenv.ksh
6  ...
7  EOF
8
9  $ sed '/^#.*$/d;/^$/d;s/\([[:a-z:]]*\):\(.*)\n/\2:\1 /' sed_chown_example.txt
10 sudo chown jdope:dba oraenv.ksh
11 sudo chown jdope:staff sysenv.ksh
12 ...

```

Usando awk

El programa `awk` puede ser un manipulador de texto útil – realiza tareas tales como el análisis de texto. Toma su entrada de `stdin` o de archivos y de forma predeterminada, muestra el resultado. Hay una gran variedad de releases disponibles para `awk` con distintos nombres como `nawk` y `gawk`. El comportamiento de los releases de proveedores de `awk` varía. `awk` es distinto a otros comandos revisados en este programa. Este lenguaje provee funciones internas de cálculo, manipulación de cadenas de texto. Los programadores también pueden definir sus propias funciones creando bibliotecas o scripts autónomos. Dado que `awk` contiene tantos recursos para demostrar, sólo mostramos algunos. Consulte la sección [Recursos](#) o las páginas principales para obtener más información.



Al comienzo de este ejemplo se usa `awk` como filtro para imprimir únicamente sistemas de archivos de Linux. De manera predeterminada, `awk` utiliza espacio blanco para identificar columnas separadas.

posiblemente para enviar un email o para escribir como parte de un mensaje en un archivo o de cómo crear una concordancia utilizando una comparación numérica.

Ejemplo de awk - filtro:

```

1  $ df -k
2  Filesystem            1K-blocks      Used Available Use% Mounted on
3  /dev/sda1             61438632  61381272     57360 100% /
4  udev                  255788      148     255640   1% /dev
5  /dev/mapper/datavg     6713132   3584984   3128148  54% /data
6  rmthost1:/archives/backup -          -          - - /backups
7  rmthost1:/archives/    -          -          - - /amc
8  rmthost1:/archives/data2 -          -          - - /data2
9
10 $ df -k |awk '$5 ~ /100%/ {print $0}'
11 /dev/sda1             61438632  61381272     57360 100% /
12
13 $ df -k |awk '$5 ~ /100%/ {printf("full filesystem: %s, mountpoint: %s\n",$6,
14 full filesystem: /, mountpoint: /dev/sda1
15
16 $ df -k |awk '$4 > 30000000 {print $0}'
17 Filesystem            1K-blocks      Used Available Use% Mounted on
18 /dev/mapper/datavg     6713132   3584984   3128148  54% /data

```

A veces los datos no están delimitados por espacio blanco. Tome, por ejemplo, el archivo `/etc/passwd` por el carácter de dos puntos “:”. Este ejemplo muestra cómo awk utiliza la bandera `-F` para la UID de las primeras 5 entradas presentadas en `/etc/passwd`. Luego, la función `substr()` de los primeros tres caracteres de la columna 1 del archivo `/etc/passwd`.

Ejemplo de awk - separador de campo / función de cadena:

```

1  $ cat /etc/passwd |awk -F: '{printf("%s %s\n", $1,$3)}' |head -5
2  root 0
3  daemon 1
4  bin 2
5  sys 3
6  adm 4
7
8  cat /etc/passwd |awk -F: '{printf("%s \n", substr($1,1,3))}' |head -5
9  roo
10 dae
11 bin
12 sys
13 adm

```

En varias ocasiones, los administradores de sistema o los programadores escriben scripts para un tipo de tarea. Aquí hay un ejemplo de un programa awk para obtener el promedio de la tercera columna de un archivo. El cálculo se hace manualmente sumando los datos de la columna 3 a una variable interna especial que awk usa para hacer seguimiento de cuántos registros fueron procesados.

Ejemplo de awk – programa / cálculo:

```

1  cat << EOF > dummy_file2.dat
2  011  IBM 174.99
3  012  INTC 22.78
4  013  SAP 59.37
5  014  vmw 102.92
6  EOF
7
8  $ cat avg.awk
9  awk 'BEGIN {total=0;}
10      {printf("tot: %.2f arg3: %.2f NR: %d\n",total, $3, NR); total+=$3;
11      END {printf "Total:%.3f Average:%.3f \n",total,total/NR}}'
12
13 $ cat dummy_file2.dat | avg.awk
14 tot: 0.00 arg3: 174.99 NR: 1
15 tot: 174.99 arg3: 22.78 NR: 2
16 tot: 197.77 arg3: 59.37 NR: 3
17 tot: 257.14 arg3: 102.92 NR: 4
18 Total:360.060 Average:90.015

```

Operaciones de cadenas basadas en shell

Shell puede ser un lenguaje de programación poderoso. Al igual que awk, shell ofrece una amplia variedad de operaciones de cadena, funcionalidad de cálculo, matrices, control de flujo y operaciones de control de flujo. Los ejemplos que muestran cómo extraer partes de una cadena de un lado. La operación no cambia el resultado y frecuentemente se utiliza como asignación de una variable. Use el signo percent “%” para truncar la derecha del patrón, y use el signo numeral “#” para truncar la izquierda.

Ejemplo de script shell – extracción de cadena:

```

1  $ cat string_example1.sh
2  #!/bin/sh
3  FILEPATH=/home/w/wyoes/samples/ksh_samples-v1.0.ksh
4  echo '${FILEPATH}'      = ' ${FILEPATH}'      " # the full filepath"
5  echo '${#FILEPATH}'    = ' ${#FILEPATH}'      " # length of the string"
6  echo '${FILEPATH%.*}'  = ' ${FILEPATH%.*}'    " # truncate right of the last dot"
7  echo '${FILEPATH%%.*}' = ' ${FILEPATH%%.*}'    " # truncate right of the first dot"
8  echo '${FILEPATH%/w*}' = ' ${FILEPATH%/w*}'    " # truncate right of the first w"
9  echo '${FILEPATH#*/*/}' = ' ${FILEPATH#*/*/}'  " # truncate left of the third slash"
10 echo '${FILEPATH##*/}' = ' ${FILEPATH##*/}'    " # truncate left of the last slash"
11
12 $ ./string_example1.sh
13 ${FILEPATH}=/home/w/wyoes/samples/ksh_samples-v1.0.ksh # the full filepath
14 ${#FILEPATH} = 42                                     # length of the string
15 ${FILEPATH%.*}=/home/w/wyoes/samples/ksh_samples-v1.0 # truncate right of the last dot
16 ${FILEPATH%%.*}=/home/w/wyoes/samples/ksh_samples-v1 # truncate right of the first dot
17 ${FILEPATH%/w*}=/home                                # truncate right of the first w
18 ${FILEPATH#*/*/}=wyoes/samples/ksh_samples-v1.0.ksh  # truncate left of the third slash
19 ${FILEPATH##*/}=ksh_samples-v1.0.ksh                 # truncate left of the last slash

```

Como ejemplo, supongamos que un administrador necesita cambiar la extensión de un minúsculas. Como los servidores UNIX diferencian mayúsculas y minúsculas, algunas aplica minúsculas, o tal vez el administrador simplemente trate de estandarizar las extensiones de cantidad de archivos a mano o a través de GUI puede llevar horas. A continuación se presentará un método para resolver este problema. El ejemplo está compuesto de dos archivos. Primero se crea un árbol de directorio de muestra y se pobra el árbol con algunos archivos. Luego se crea un script para configurar un árbol de directorio de muestra y poblar el árbol con algunos archivos. Tan pronto como se requieren cambios de extensión. El segundo script llamado `fix_extension.ksh` lee la lista de extensiones, cuando es apropiado. Como parte del comando `mv`, el operador de cadena `%s` del último punto "." en el nombre de archivo (trunca la extensión). Ambos scripts también utilizan `find` para mostrar qué se logró después de la ejecución.

Ejemplo de script shell – cambiar extensión de archivo:

```

1  $ cat setup_files.ksh
2  mkdir /tmp/mv_demo
3  [ ! -d /tmp/mv_demo ] && exit
4
5  cd /tmp/mv_demo
6  mkdir tmp JPG 'pictures 1'
7  touch a.JPG b.jpg c.Jpg d.jPg M.jpG P.jpg JPG_file.JPG JPG.file2.jPg file1.Jf
8  pic 2.Jpg' 10.JPG.bak 'pictures 1/photo.JPG' JPG/readme.txt JPG/sos.JPG
9
10 find . -type f | grep -i "\.jpg$" | sort | tee file_list.txt
11
12 $ ./setup_files.ksh
13 ./JPG.file2.jPg
14 ./JPG/sos.JPG
15 ./JPG_file.JPG
16 ./M.jpG
17 ./P.jpg
18 ./a.JPG
19 ./b.jpg
20 ./c.Jpg
21 ./d.jPg
22 ./file1.JPG.Jpg
23 ./pictures 1/photo.JPG
24 ./tmp/pic 2.Jpg
25
26 $ cd /tmp/mv_demo
27 $ cat /tmp/fix_extension.ksh
28 while read f ; do
29     mv "${f}" "${f%.*}.jpg"
30 done < file_list.txt
31
32 find . -type f | grep -i "\.jpg$" | sort
33
34 $ /tmp/fix_extension.ksh
35 ./JPG.file2.jpg
36 ./JPG/sos.jpg
37 ./JPG_file.jpg
38 ./M.jpg
39 ./P.jpg
40 ./a.jpg
41 ./b.jpg
42 ./c.jpg

```



46 | `./tmp/pic 2.jpg`

Con el ánimo de crear herramientas útiles y reutilizables, el ejemplo de cambio de extensión generalizado. Algunas mejoras que me vienen a la mente serían pasar una corriente de nombre como parte de una tubería. Se podrían agregar banderas de opción para especificar las extensiones (como .mp3 ó .mov), y cómo dar formato a la extensión de archivo, en cuanto a minúsculas, posibilidades están limitadas únicamente por la imaginación y el tiempo del programador.


Resumen

UNIX ofrece una amplia variedad de herramientas para hacer análisis de texto nativamente, necesidad de depender de intérpretes especiales que pueden no estar instalados. Este artículo sondeo de comandos si se lo compara con la frecuencia de su uso. Los comandos de este artículo demostrados, ya que algunos sistemas implementan banderas o se comportan de forma diferente. Ciertamente, UNIX ofrece más comandos y formas de lograr las mismas tareas – “Hay más comandos”

Recursos para Descargar

 [PDF de este contenido](#)

Temas relacionados

- Vaya a la página de inicio de [KornShell](#) para encontrar software e información.
- Consulte [Advanced Bash-Scripting Guide - Manipulating Strings](#) para aprender más acerca de las cadenas.
- Base de datos [join](#) explicada
- Vaya a [sed \\$HOME page](#) para obtener más información.
- Aprenda más sobre [Sed](#).
- Aprenda cómo utilizar el editor de flujo UNIX, [sed](#).
- [Gawk: Effective AWK Programming](#): Lea el manual de usuario estándar de GAWK.
- Aprenda más con este tutorial: [Awk - An Introduction and Tutorial by Bruce Bar](#) 
- Lea una introducción a [awk](#)
- [The Open Group Base Specifications Issue 7 \(awk\)](#)

- [Pruebe software IBM](#) gratuitamente. Descargue una versión de prueba, inicie sesión en el producto en un entorno de recinto de seguridad o acceda al mismo a través de la nube. E productos IBM.

Comentarios

[Inicie Sesión](#) o [Regístrese](#) para agregar comentarios.

☐ Reciba notificaciones de los comentarios

developerWorks

Acerca de

Informar abusos

Aviso de términos legales de terceros

Síguenos

Únete

Universidad

Startups (Inglés)

Business Partners (Inglés)

Seleccione un idioma

English

中文

日本語

Русский

Português (Brasil)

Español

한글

Descargas

Tutoriales & entrenamientos

Contacto

Privacidad

Condiciones de uso

Accesibilidad

Comentarios

Preferencias de

A

