

El uso de *tr* y *dos2unix*

LOST IN TRANSLATION

La herramienta *tr* es realmente útil. Este comando tan simple permite reemplazar cadenas en archivos de texto. Ya sea reemplazando letras o eliminando espacios en blanco, le sorprenderá la versatilidad de *tr*.

POR HEIKE JURIK

La utilidad *tr* reemplaza caracteres en archivos de texto. Para ello lee la entrada estándar y envía los resultados a la salida estándar. Por supuesto, podemos utilizar los operadores tradicionales para redirigir ambos flujos. De hecho, cuando *tr* brilla de verdad es en combinación con otras órdenes del intérprete de comandos.

Reemplazos Simples

El comando *tr* recibe dos cadenas como argumento y reemplaza todas las coincidencias del primer argumento con el segundo argumento. Puede sonar complicado, veamos el siguiente ejemplo. Con la siguiente sentencia cambiamos cada carácter “e” que haya en “Petronila”, por el carácter “a”:

```
$ echo Petronila | tr 'e' 'a' > Patronila
```

Por supuesto que podemos especificar la cadena sobre la que queremos reemplazar en un fichero. Con *tr '1' '2' < prueba.txt* reemplazaremos cualquier

“1” que aparezca en *prueba.txt* por “2” y enviaremos el resultado a la salida estándar. Debemos tener en cuenta que cada carácter que pasamos como argumento se interpreta de manera independiente, es decir, cada carácter se reemplaza por su contraparte en el segundo argumento. De esta manera, *tr 'abc' 'xyz'* reemplazará “a” por “x”, “b” por “y” y “c” por “z”. Si la segunda cadena es más corta que la primera, *tr* sustituirá los que no tengan contraparte por el último carácter del segundo argumento.

Por ejemplo, si escribimos *tr 'abc' 'z'* reemplazaremos cada “a”, cada “b” y cada “c” por “z”. Sin embargo, *tr* es incapaz de sustituir correctamente “ä” por “ae”. Al no ser las cadenas de la misma longitud, cada “ä” se sustituye sólo por “a”. Para este tipo de sustitución, se recomienda usar *sed*.

Mayúsculas/Minúsculas

tr puede ser extremadamente útil si necesitamos intercambiar mayúsculas y minúsculas. La mejor opción es definir dos argumentos como vectores de letras minúsculas y mayúsculas, por ejemplo:

```
tr 'a-z' 'A-Z' < prueba.txt
```

O si lo prefiere, también es posible del siguiente modo:

```
tr [:lower] [:upper:] > prueba.txt
```

Todo Salvo...

El comando *tr* tiene algunos parámetros que nos permiten tener mayor control. Por ejemplo, podemos usar la opción *-d* para borrar:

```
tr -d '0-9' < prueba.txt
```

que provoca que todos los números pasen a mejor vida. Si combinamos esta opción con *-c*, tenemos una manera aún mejor de eliminar contenido superfluo. Si queremos suprimir cualquier cosa excepto espacios en blanco, letras en mayúscula y letras en minúscula, usamos *-c* para indicarle a *tr* qué es lo que no tiene que borrar:

```
tr -c -d 'A-Z a-z' < prueba.txt
```

En combinación con *-s*, *tr* nos permite reducir el tamaño de un archivo, algo útil si tuviéramos, pongamos por caso, un archivo *.log* lleno de espacios en blanco. La opción *-s* aguarda el paso de

uno o dos argumentos. Por ejemplo, `tr -s ' ' < prueba.txt` borra todos los espacios en blanco de un archivo. Pero si sólo necesitamos eliminar los dobles espacios o tabulaciones, e insertar un espacio simple en su lugar, podemos suministrar dos argumentos a `tr -s`:

```
tr -s [:blank:] ' ' < prueba.txt
```

En este caso, `tr` reemplazará espacios en blanco contiguos o tabulaciones por espacios simples.

Entre Dos Mundos

Si hemos tenido ocasión de intercambiar ficheros entre sistemas Windows y Linux, habremos comprobado que suelen aparecer caracteres extraños al final de las líneas. Y si, por ejemplo, tratamos de abrir un archivo ASCII creado en Windows con el editor Vim, encontraremos extraños caracteres ^M. La razón de todo esto es que los dos sistemas usan símbolos distintos para el carácter nueva línea. Mientras que Windows usa `\r\n` para nueva línea, Linux únicamente usa `\n`.

`tr` puede ayudarnos a migrar archivos ASCII entre los dos sistemas. Mediante:

```
tr -d '\r' < textowin > textolinux
```

Eliminamos el carácter `\r` extra del final de cada línea, con lo que tenemos la con-

versión. La opción `-d` indica a `tr` que elimine el carácter no deseado. Con el operador `<` logramos procesar `textowin`, y con `>` enviamos el resultado "limpio" al archivo llamado `textolinux`.

Conversores Alternativos

Las herramientas `dos2linux` y `unix2dos` son también útiles si necesitamos convertir de acá para allá entre sistemas Linux y Windows. Para convertir un archivo de texto Windows al formato adecuado para Linux, simplemente tecleamos:

```
$ dos2unix -n textowin > textolinux
dos2unix: converting file >
textowin to textolinux >
in UNIX format ...
```

Esta sentencia usa la opción `-n`, que nos permite especificar tanto un fichero de entrada como uno nuevo de salida. La página *man* contiene más detalles y consejos. La opción `-k` mantiene la información de fecha y hora original, y la opción `-o` escribe los cambios directamente en el archivo original. El comando `unix2dos` hace exactamente lo mismo, pero en sentido contrario:

```
unix2dos -n textolinux textowin
```

En algunos sistemas, estos comandos son enlaces simbólicos a los programas `fromdos` y `todos`, respectivamente. Estos programas tienen una sintaxis ligeramente distinta y parámetros también diferentes. Es recomendable usar la opción `-b` para crear una copia de seguridad, incluso si estamos creando un archivo nuevo. También debemos tener en cuenta que la opción `-a` elimina todos los retornos de carro cuando se le indica a `fromdos` (y no sólo los precedidos por un avance de línea). Si especificamos `-a` con el comando `todos`, convertiremos todos los avances de línea en pares CR-LF (retorno de carro-avance de línea). El

comportamiento por defecto en este caso será convertir sólo aquellas líneas que no están precedidas de un retorno de carro.

Los dos ejemplos de `dos2unix` y `unix2dos` mostrados anteriormente tendrán en este caso el siguiente aspecto:

```
cat textowin | fromdos -a >
textolinux
cat textolinux | todos -a >
textowin
```

Combinados

El comando `tr` realmente brilla en todo su esplendor cuando se usa en combinación con otras órdenes de la línea de comandos. Por ejemplo, imaginemos un gran número de archivos con espacios en blanco en el nombre de archivo y queremos reemplazar estos espacios con guiones bajos. Un bucle `for`, el comando `mv` y `tr` nos ayudarán a realizar la operación:

```
$ for i in *; do mv -v "$i" >
`echo $i | tr ' ' '_'`; done
'archivo con espacio' -> >
'archivo_con_espacio'
'archivo con espacio 2' -> >
'archivo_con_espacio_2'
```

Si traducimos esto a cristiano, la sentencia quedaría algo como: para todos los archivos en el directorio actual, renombramos los archivos tal y como nos indica el resultado de aplicarle `tr`. ■

GLOSARIO

Nueva línea: La sintaxis para la nueva línea en los ordenadores está basada en las máquinas de escribir. Hay un botón de avance de línea y otro de retorno de carro. Los sistemas operativos tienen distintos métodos de tratar la nueva línea. Mientras que Linux usa un simple avance de línea (`\n`), DOS y Windows le añaden un retorno de carro (`\r\n`).

LA AUTORA

Heike Jurzik estudió Alemán, Informática e Inglés en la Universidad de Colonia, Alemania. Descubrió Linux en 1996 y quedó fascinada con la potencia de la línea de comandos desde ese momento. En su tiempo libre puede que la encuentre en clases de folclore irlandés, o visitando Irlanda.

