

Conversión en la shell

# CONVERSIÓN EN COMANDOS

Sebastian Duda, Fotolia

Los juegos de caracteres o las nuevas líneas diferentes pueden complicar el intercambio de datos entre usuarios Linux. Un par de herramientas de conversión de la shell nos evitarán el quebradero de cabeza.

**POR HEIKE JURZIK**

**S**i frecuentemente intercambiamos documentos de texto entre sistemas Windows y Linux, sólo necesitaremos unos cuantos comandos de la shell para convertir los caracteres de nueva línea diferentes. *dos2unix* y *unix2dos* convierten con un único comando. Si es un juego de caracteres el que causa problemas, serán *iconv* y *recode* los que convertirán el revoltijo de letras en algo más aceptable.

## ¿Qué Está Pasando?

Las extensiones de ficheros son bastante superfluas. Linux diferencia los tipos de ficheros por su contenido y no por cómo acaba su nombre. Reconoce, por ejemplo, un fichero MP3 como tal, incluso si su nombre indica algo completamente distinto.

La Figura 1 muestra el administrador de ficheros de Gnome, Nautilus, que identifica tanto el fichero MP3

camuflado (*chicken.txt* mostrado en el ejemplo) como el fichero LaTeX pretendiendo ser un fichero gráfico PNG (*brief.png*).

La herramienta de la línea de comandos *file* ayuda a identificar los ficheros correctamente (Listado 1). Los inspecciona y realiza varias pruebas para reconocer el tipo de fichero. Usualmente sólo coge los primeros bytes para tomar una decisión correcta. Si *file* no llega a ninguna conclusión, nos lo dirá con lo que sigue:

```
$ file ficheroraro
ficheroraro: data
```

El programa suministra una buena cantidad de información sobre los ficheros de texto. La salida en el Listado 2 muestra que el texto no lo es todo. Tres ficheros de texto generan tres mensajes completamente diferentes de *file*: uno es UTF-8, otro usa terminadores de línea Windows (CRLF, véase “El Final de la Línea...”), y el otro es ISO-8859. En las secciones siguientes examinare-

### Listado 1: Fichero

```
01 $ file *
02 letter.png: LaTeX 2e
03 document text
04 chicken.txt: MP3 file with
05 ID3 version 2.3.0 tag
06 test.txt: ISO-8859 text
```

### Listado 2: Text no es Texto

```
01 $ file *
02 utf8.txt: UTF-8 Unicode
03 text
04 win.txt: ISO-8859 text,
05 with CRLF line terminators
06 iso 8859.txt: ISO-8859 text
```

mos los programas que modifican estos diferentes tipos de texto para facilitar el intercambio de datos con otros sistemas.

### El Final de la Línea...

... es a menudo donde comienzan los problemas, porque diferentes sistemas operativos usan métodos diferentes para marcar los finales de línea. La sintaxis, básicamente tomada de las máquinas de escribir manuales, controla el escenario del final-de-línea, siendo completamente diferente en Linux que en Windows. Mientras los sistemas Linux añaden un nueva línea (`\n` = "new line"), Windows usa una combinación de nueva línea y retorno de carro (`\r` = "return").

Esta pequeña aunque crítica diferencia es muy notable cuando abrimos un fichero de texto en un editor. Un fichero de texto creado en Windows estará lleno de caracteres `^M` si lo abrimos en Vim.

Las herramientas de la shell *dos2unix* ("DOS a Unix") y *unix2dos* ("Unix a DOS") nos dan una solución para esto. Tal y como sugieren sus nombres, convierten ficheros de un formato a otro.

Algunas distribuciones Linux colocan las herramientas en sus propios paquetes. Las diferentes versiones de Debian actuales ofrecen el paquete *tofrodos* con los programas *fromdos* y *todos*. Dos enlaces simbólicos de `/usr/bin/dos2unix` y `/usr/bin/unix2dos` señalan a las herramientas, permitiendo a los usuarios trabajar con *dos2unix* y *unix2dos*, o con *fromdos* y *todos*, cuando sea necesario. Las programas hacen básicamente lo mismo, aunque con un par de diferencias.

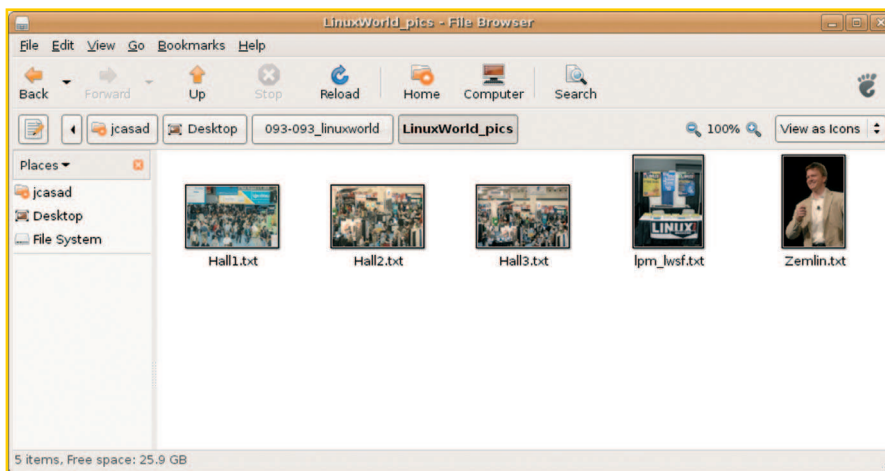


Figura 1: Sin trampas, Linux reconoce los tipos de fichero por su contenido y no por los sufijos de éstos.

En este artículo me referiré a estas herramientas como *dos2unix/unix2dos* (porque aún podemos usar esta sintaxis en sistemas Debian) y señalaré dónde difieren las opciones.

### Está bien lo que Bien Acaba

Para reparar un fichero de texto creado en Windows para su uso en Linux escribimos:

```
$ dos2unix win.txt
```

La herramienta es muy extrovertida y le dice a los usuarios qué está pasando detrás en forma de salida de línea de comandos:

```
$ dos2unix: converting file win.txt to UNIX format
```

La versión de Debian no nos da ninguna información de manera predefinida, aunque podemos configurar la opción `-v` (verbose) para hacerle hablar:

```
dos2unix: Converting win.txt
```

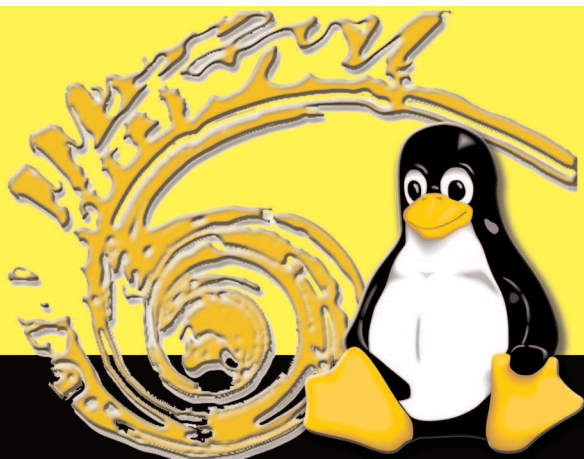
Como el comando procesa el fichero original en cualquier distribución que usemos y no crea un backup automático, deberíamos usar la opción Debian de configuración del parámetro `-b` para decirle a *dos2unix* que cree uno (un fichero con un sufijo `.bak`):

```
$ dos2unix -b win.txt
$ ls
win.bak win.txt
```

Usuarios con otras distribuciones pueden establecer el parámetro `-n` para especificar un nombre diferente para el fichero de salida. La sintaxis es *dos2unix -n fichero\_entrada fichero\_salida*, por ejemplo:

```
$ dos2unix -n win.txt linux.txt
```

`-k`, (`-p` en Debian) es otra opción útil; le dice a Linux que mantenga la marca temporal original para el



# Asociación Linux Español

fichero. Todos estos parámetros de la línea de comandos funcionan de la misma manera para *unix2dos*, aunque convierten en la otra dirección.

## Confusión en el Juego de Caracteres

Algunas distribuciones insisten en usar juegos de caracteres antiguos, mientras que otras confían en el recién llegado multilingüe. El debate entre ISO.8859-1 (5) versus UTF-8 ha alcanzado estado de culto en foros y listas de correo, similar a las perennes discusiones sobre el mejor editor de texto Linux. Todas las distribuciones modernas soportan ahora UTF-8 por defecto.

Si no estamos preparados, o simplemente no es posible cambiar por algunas razones, podemos volver a convertir a ISO-8859-1(5) con un par de pasos simples. Sin embargo, advertimos que nos enfrentaremos con renderizaciones extrañas de caracteres especiales cuando necesitamos intercambiar HTML o ficheros de texto. Los cuadros vacíos en los menús o diálogos, acentos rotos o diéresis, y muchos otros errores hacen que el cambio sea poco amigable por muy escrupuloso que seamos.

El juego de caracteres y los convertidores de formato *recode* y *iconv* evitan dolores de cabeza a los usuarios. A pesar de que *iconv* se encuentra incluido en la mayoría de las distribuciones, por lo que podemos usar cualquier administrador de paquetes que prefiramos, deberemos instalar *recode*.

El parámetro *-l* nos dice qué formatos soportan las herramientas; para evitar una larga lista desplazándose por nuestra pantalla deberíamos direccionar la salida a un paginador, como en *recode -l | less* o *iconv -l | less*.

## Reaplicar Diariamente

Ambas herramientas aceptan ficheros en la línea de comandos o funcionarán como filtros. Cuando trabajan como filtros aceptan los datos desde entradas estándar y envían los resultados a salidas estándar (Figura 2).

Si introducimos en el prompt

```
$ iconv -f UTF -8
```

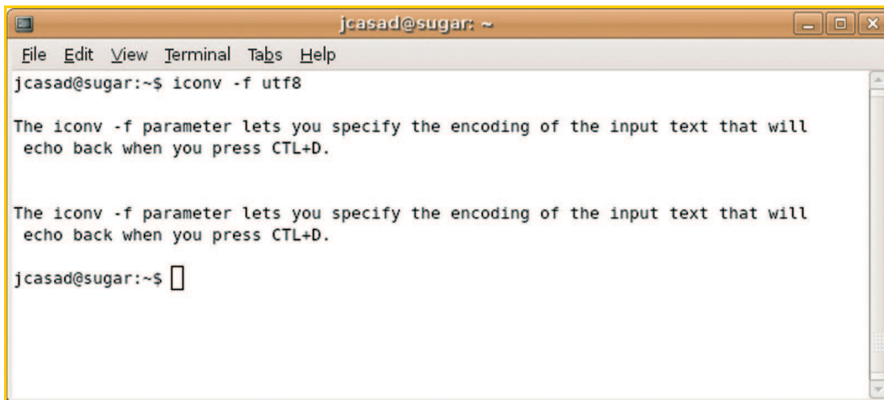


Figura 2: Para evitar confusión de código, si no estás preparado para cambiarte a UTF-8, puedes ejecutar un comando en la shell para convertirte.

la herramienta sabe que el texto de entrada está codificado en UTF.

Si luego escribimos texto en la línea de comandos o lo copiamos desde una aplicación a la shell y seguidamente pulsamos Ctrl + D para dejar de leer la entrada, *iconv* convertirá nuestra entrada y presentará los resultados en una salida estándar.

Por defecto, el programa usará el código de nuestro locale actual (*echo \$LANG [1]*). Si esto está configurado a *es\_ES@euro*, por ejemplo, *iconv* nos dará los resultados en ISO-8859-15 (Latin-9, que contiene el carácter euro junto a los caracteres suministrados por ISO-8859-1).

*Recode* funciona de forma parecida: si introducimos *recode utf8*, el programa esperará texto codificado-utf-8 y salida ISO-8859-1 (Latin 1) por defecto.

Si necesitamos un tipo de salida diferente, podemos pasarle el juego de caracteres requerido a *iconv* usando la opción *-t*. Por ejemplo:

```
$ iconv -f UTF-8 -t MAC
```

*Recode* espera una sintaxis ligeramente diferente: el formato de salida separado por dos puntos del formato de entrada (las versiones viejas del programa esperan dos puntos)

```
$ iconv -f UTF-8 utf8. >
txt
$ recode utf8..latin9 >
utf8.txt
```

Mientras *iconv* presenta los resultados en la shell, *recode* sobrescribe el original.

Los operadores de redirección son aquí útiles [2]:

```
$ recode utf8..latin9 >
< utf8.txt
> iso8859.txt
```

La aplicación para *iconv* es similar. Si preferimos no usar operadores, podemos configurar la opción *-o* seguida del fichero de salida:

```
$ iconv -f UTF-8 >
utf8.txt -o iso
8859.txt
```

## Conclusiones

Los beneficios de estas conversiones en la shell son obvios.

Si necesitamos convertir un importante número de ficheros para nuestro sistema, podemos usar los trucos de Bash normales:

```
$ for i in ~/download/ >
*.txt; do
recode utf8..lat9 >
$i; done
```

Un bucle *for* le dice a *recode* que convierta múltiples ficheros en una sola pasada. Evidentemente, esto es algo que también funciona con *iconv*.

## RECURSOS

[1] "Línea de Comandos: Variables de Entorno" por Heike Juezik, Linux Magazine, Número 32, pág. 81.

[2] "Línea de Comandos: Flujo de Datos" por Heike Juezik, Linux Magazine, Número 34, pág. 83.