Procesamiento básico de texto en Unix | ProgPLN

Víctor Peinado v.peinado@filol.ucm.es

17-23 de octubre de 2014

Referencias

- Unix for poets [1]

[1] Church, K. W. *Unix for poets* `http://web.stanford.edu/class/cs124/kwc-unix-for-poets.pdf`

Comandos para procesar ficheros de texto

Tras haber repasado en el documento anterior algunos de los comando básicos de gestión de ficheros en Unix, ahora vamos a presentar algunas de las herramientas para procesar ficheros de texto.

Recuerda que en el directorio `/opt/textos/` tienes varios ficheros de texto de ejemplo que puedes copiar en tu directorio de trabajo y jugar con ellos.

El comando `cp /opt/textos/*.txt ~/` copiará estos ficheros en tu directorio de usuario.

wc [OPCIONES] [RUTA/]FICHERO

Para mostrar estadísticas del tamaño de un fichero de texto podemos utilizar el comando `wc`.

Si se ejecuta sin ninguna opción, se nos muestran cuatro columnas con la siguiente información: número de líneas, número de palabras, número de caracteres y nombre del fichero.

Por ejemplo, para imprimir por pantalla las estadísticas de todos los ficheros del directorio actual que contengan la cadena 'palabras' en su nombre, ejecutamos:

```
user@mypc:~$ wc palabras*
26  26 174 palabras.txt
21  42 317 palabras-unicas-ordenadas.txt
21  42 317 palabras-unicas.txt
68 110 808 total
```

Entre las opciones más útiles de `wc`, están:

- `-l` imprime el número de líneas de un fichero.
- `-w` imprime el número de palabras de un fichero.

```
user@mypc:~$ wc -l palabras.txt
26 palabras.txt
user@mypc:~$ wc -w palabras-unicas.txt
42 palabras-unicas.txt
```

sort [OPCIONES] [RUTA/]FICHERO

Procesa un fichero de texto línea a línea y las muestra ordenadas. Las opciones determinan el tipo de ordenación empleado:

- `-d` orden alfabético, incluso para los números.
- `-n` orden numérico: coloca los números antes que las letras y ordena los dígitos de menor a mayor.
- `-f` no distingue entre mayúsculas y minúsculas.
- `-r` orden inverso.

ojo, distingue entre mayúsculas y minúsculas y tiene en cuenta los posibles espacios en blanco que aparezcan

Las opciones se pueden combinar, p. ej., para ordenar un fichero numéricamente de mayor a menor podemos ejecutar `sort -nr FICHERO`.

uniq [OPCIONES] [RUTA/]FICHERO

Procesa un fichero de texto línea a línea y elimina los duplicados, es decir, solo muestra una ocurrencia por línea. Algunas de las opciones más interesantes son:

- `-c` muestra el número de ocurrencias de cada línea.
- `-d` muestra solo las líneas duplicadas.
- `-i` ignora las diferencias entre mayúsculas y minúsculas

Ojo, `uniq` solo recuerda la última línea que ha visto y, en consecuencia, sólo es capaz de eliminar líneas duplicadas cuando aparecen todas juntas unas detrás de otras. Antes de eliminar líneas duplicadas con `uniq` es conveniente ordenarlas con `sort`.

grep [OPCIONES] PATRÓN FICHERO

Procesa un fichero de texto línea a línea e imprime solo aquellas que contengan el patrón de búsqueda especificado. Entre las opciones más interesantes, están:

- `-i` ignora las diferencias entre mayúsculas y minúsculas
- `-v` invierte el sentido del patrón, es decir, muestra solo las líneas que no coincidan con el patrón.
- `-c` imprime, en lugar de las líneas, el número de ocurrencias en las que el patrón coincide.

Por ejemplo, imprime del fichero `palabras.txt` solo aquellas líneas que contengan la cadena `mente`.

```
user@mypc:~$ grep "mente" palabras.txt
```

Para realizar la operación contraria, es decir, imprimir solo aquellas líneas que no contienen la cadena `mente`, podemos utilizar `grep` con la opción `-v`.

```
user@mypc:~$ grep -v "mente" palabras.txt
```

tr [OPCIONES] PATRÓN1 PATRÓN2 < FICHERO

El comando `tr` permite procesar un fichero de texto y transformar o realizar sustituciones entre los caracteres del patrón `PATRÓN1` con los caracteres del patrón `PATRÓN2`. Veamos algunos ejemplos de uso:

Para procesar nuestro fichero de palabras y sustituir cualquier aparición de una `m` en una `M`, ejecutamos:

```
user@mypc:~$ tr "m" "M" < palabras.txt
```

Para sustituir las vocales en minúsculas a mayúsculas, ejecutamos:

```
user@mypc:~$ tr "aeiou" "AEIOU" < palabras.txt
```

Si queremos pasar a mayúsculas un texto escrito en minúsculas, podemos especificar un rango de caracteres (de la *a* a la *z*):

```
user@mypc:~$ tr "a-z" "A-Z" < palabras.txt
```

Fíjate en el comportamiento de estos alias: `a-z` para las letras del alfabeto en minúsculas, `A-Z` para las letras mayúsculas, `A-z` para el alfabeto completo en mayúsculas y minúsculas, `0-9` para los dígiyos, o incluso subconjuntos de estos caracteres como `a-m` y `5-9`

Para sustituir cualquier dígito que encontremos por una `X`, podemos ejecutar:

```
user@mypc:~$ tr "0-9" "X" < palabras.txt
```

Un uso muy del comando `tr` muy interesante para tareas de procesamiento de texto es utilizarlo para *tokenizar* o segmentar en palabras un fichero de texto. Para ello, necesitaremos especificar determinadas opciones de manera que sustityamos cualquier carácter que no sea `-c` una letra mayúscula o minúscula (`A-Za-z`) por un único (`-s`) retorno de carro (`\n`). Comprueba qué hace la siguiente intrucción:

```
user@mypc:~$ tr -sc "A-Za-z" "\n" < texto.txt > texto-tokenizado.txt
```

La instrucción anterior no tendrá en cuenta como palabras o *tokens* los caracteres con diacríticos propios del español o las secuencias de dígitos que encuentre. Para separar correctamente las palabras en español y considerar también como *tokens* los números, necesitamos modificar el primer patrón:

```
user@mypc:~$ tr -sc "A-Za-zÁÉÍÓÚÜáéíóúüñÑ0-9" "\n" < texto.txt > texto-tokenizado.txt
```

Entrada, salida y pipelines *para encadenar comandos*

Encadenando herramientas

Hasta ahora hemos visto comandos para realizar operaciones sencillas sobre ficheros. La mayoría de las herramientas de Unix son similares: consisten en herramientas pequeñas que realizan muy bien una operación muy concreta y determinada, y funcionan de la siguiente manera:

- toman uno o varios ficheros como entrada
- los manipulan; y
- generan un salida, que podemos mostrar por pantalla o redirigirla a otro sitio.

La verdadera potencia y versatilidad de estas herramientas la obtenemos cuando encadenamos unas con otras, seleccionando como entrada de una de estas instrucciones la salida de otra. Para ello podemos hacer uso de algunos símbolos especiales que veremos a continuación.

Entrada estándar < *[RUTA/]FICHERO*

La mayoría de los comandos de Unix que hemos visto hasta el momento toman como entrada el nombre de fichero que especifiquemos. En otros casos, es obligatorio especificar la entrada con el símbolo <.

```
user@mypc:~$ cat < palabras.txt        # equivalente a cat palabras
user@mypc:~$ sort -nr < palabras.txt   # equivalente a sort -nr palabras.txt
user@mypc:~$ tr "a" "A" < palabras.txt # es obligatorio el uso de <
```

Salida estándar > *[RUTA/]FICHERO*

Cuando no especificamos la salida de un comando se asume por defecto que se trata de la impresión por pantalla. Hemos visto anteriormente que podemos redirigir la salida de un comando a un fichero de texto.

Por ejemplo, para ordenar alfabéticamente el fichero `palabras.txt` y guardar el resultado en otro fichero de nueva creación llamado `palabras-ordenadas.txt`, ejecutamos la siguiente instrucción, indicando explícitamente la entrada y la salida.

```
user@mypc:~$ sort < palabras.txt > palabras-ordenadas.txt
```

Recuerda que podemos redirigir la salida de dos maneras:

- > redirige la salida a un fichero. Si no existe, lo crea. Si existía previamente, lo sobreescribe.
- >> redirige la salida a un fichero. Si no existe, lo crea. Si existe, concatena el resultado al contenido anterior.

*Tubería (*pipeline*) comando* | *comando*

Como hemos visto, muchas de las herramientas de Unix ejecutan tareas sencillas sobre ficheros de texto y generan una salida.

La verdadera potencia de estas herramientas radica en la posibilidad de encadenar varias de ellas, haciendo que una herramienta

tome como entrada y procese la salida de otra herramienta que hemos ejecutado previamente.

El *pegamento* que nos permite encadenar herramientas desde la línea de comandos es la tubería o *pipeline* `|`.

Imaginemos que queremos procesar línea a línea una fichero de texto formado por un listado de palabras. Dicho listado contienen palabras desordenadas y numerosos duplicados. Podemos encadenar tres herramientas como `sort`, `uniq` y `grep` para generar un nuevo listado de palabras ordenadas, únicas y que contengan la cadena `mente` con el siguiente comando:

```
user@mypc:~$ sort palabras.txt | uniq | grep "mente" > ordenadas.txt
```

Podemos realizar la misma operación sin necesidad de almacenar la salida en ningún fichero. Para revisar el resultado, nos basta con redirigir la salida al paginador `less` del siguiente modo:

```
user@mypc:~$ sort palabras.txt > | uniq | grep "mente" | less
```

Ejemplos de la potencia de los pipelines

Procesamos un fichero de texto línea a línea, impriendo solo las líneas que contengan palabras terminadas en *-mente* (susceptibles de ser adverbios) y, para facilitar su localización en el texto original, las imprimimos con el número de línea.

```
user@mypc:~$ cat -n texto.txt | grep "mente"
```

Procesamos un fichero de texto línea a línea, separándolo en palabras, buscando aquellas palabras que terminen en *-mente*, calculamos su frecuencia y lo redirigimos al paginador.

```
user@mypc:~$ cat texto.txt | tr -sc "a-zA-Z0-9" "\n" | grep "mente" | sort | uniq -c | less
```

Procesamos un fichero de texto línea a línea, separándolo en palabras, ordenamos alfabéticamente, contabilizamos frecuencias de aparición, las ordenamos de mayor a menor y las almacenamos en un fichero.

```
user@mypc:~$ cat texto.txt | tr -sc "a-zA-Z0-9" "\n" | sort | uniq -c | sort -nr > texto.pals.frec.txt
```

Cómo buscar ayuda

No es sencillo recordar las opciones disponibles para cada comando (solo algunos *übergeeks* lo consiguen), así que es habitual echar mano de los comandos de la shell.

Tenemos dos tipos de ayuda que podemos consultar desde la propia línea de comandos.

Todos los comandos tienen una opción `--help` (a veces también `-h`) que podemos ejecutar para acceder a la ayuda en formato abreviado. Para acceder a la ayuda del comando `cat`, ejecuta:

```
user@mypc:~$ cat --help
```

El comando `man` (de *manual*) nos da acceso al manual completo de cada comando. Para abrir las páginas del manual del comando `grep` ejecuta:

El comando `man` abre la ayuda en el paginador `less`. Recuerda que para salir de `less` hay que pulsar `q`.

```
user@mypc:~$ man grep
```