Comandos en Linux (contenido de ficheros)

Juan M. Alberola

1. Intérprete de Comandos Shell

Hasta ahora hemos visto comandos que nos permiten movernos por la estructura de directorios así como gestionar el contenido de directorios (crear, copiar, mover y borrar). En esta parte, nos centraremos en comandos para trabajar con el contenido de los ficheros.

1.1. Redirección

Los comandos tienen por defecto una entrada estándar (teclado) y una salida estándar (pantalla). Sin embargo, es frecuente cambiar esta entrada estándar o esta salida estándar para indicar otros elementos.

Por ejemplo, imaginemos que en vez de mostrar el contenido de un directorio en pantalla, queremos redireccionarlo a un fichero para que se guarde dicho contenido allí. La sintaxis del comando ls se completaría con el carácter `>` seguido del fichero donde queramos mostrar el resultado, por ejemplo *salida.txt* (Figura 1).

```
juanmi@portatil:~$ ls > salida.txt
juanmi@portatil:~$
```

Figura 1: Redirección

Hay que tener en cuenta que el comando `>` destruye el fichero destino, en este caso *salida.txt*. Si lo que queremos es añadir contenido al final del fichero debemos utilizar el doble carácter `>>`. Comprueba el resultado de hacer este comando.

1.2. Comando echo, cat y wc

El comando `echo` muestra por la salida estándar la cadena de texto que recibe como parámetro (Figura 2). Prueba este comando y observa como introduce un salto de línea. Si queremos evitar esto utilizaremos el parámetro `-n` (Figura 3).

```
juanmi@portatil:~$ echo "mensaje"
mensaje
juanmi@portatil:~$
```

Figura 2: Comando “echo”

```
juanmi@portatil:~$ echo -n "mensaje con salto de linea"
mensaje con salto de lineajuanmi@portatil:~$
```

Figura 3: Comando “echo”

El comando `cat` muestra el contenido de uno o más ficheros en el terminal. Prueba a volcar una cadena de texto a un fichero mediante el comando `echo` y utiliza el comando `cat` para visualizar este fichero (Figura 4).

```
juanmi@portatil:~$ cat salida.txt
Descargas
Documentos
Escritorio
examples.desktop
Imágenes
Música
Plantillas
Público
salida.txt
Vídeos
juanmi@portatil:~$
```

Figura 4: Volcado del fichero “salida.txt”

Si en vez de tener un fichero tenemos varios, podemos visualizar su contenido poniendo sus nombres seguidos de un espacio.

```
juanmi@portatil:~$ cat salida.txt salida2.txt
Descargas
Documentos
Escritorio
examples.desktop
Imágenes
Música
Plantillas
Público
salida.txt
Vídeos
total 48
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Descargas
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Documentos
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Escritorio
-rw-r--r-- 1 juanmi juanmi 8445 dic 28 10:42 examples.desktop
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Imágenes
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Música
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Plantillas
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Público
-rw-rw-r-- 1 juanmi juanmi    0 ene  5 17:17 salida2.txt
-rw-rw-r-- 1 juanmi juanmi  106 ene  5 17:13 salida.txt
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Vídeos
juanmi@portatil:~$
```

Figura 5: Visualización de varios ficheros

El comando `wc` cuenta las líneas, palabras o letras de un fichero (Figura 6). Para ello utilizamos los respectivos parámetros `-l`, `-w` y `-c`.

1.3. Tuberías

La tubería (representada por `|`) nos permite combinar varios comandos para ejecutarlos simultáneamente, donde el resultado del primero se envía a la entrada del segundo. Por ejemplo, si queremos saber cuántos ficheros (y directorios) tenemos en un directorio, podemos listar el contenido de un directorio de manera que en cada línea nos aparezca un fichero (`ls -l`), y después contar cuántas líneas tiene la salida del primer comando (`wc -l`). Utilizando una tubería podríamos concatenar los dos comandos (Figura 7).

```
juanmi@portatil:~$ cat salida2.txt
total 48
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Descargas
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Documentos
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Escritorio
-rw-r--r-- 1 juanmi juanmi 8445 dic 28 10:42 examples.desktop
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Imágenes
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Música
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Plantillas
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Público
-rw-rw-r-- 1 juanmi juanmi    0 ene  5 17:17 salida2.txt
-rw-rw-r-- 1 juanmi juanmi  106 ene  5 17:13 salida.txt
drwxr-xr-x 2 juanmi juanmi 4096 dic 28 11:00 Vídeos
juanmi@portatil:~$ wc -l salida2.txt
12 salida2.txt
juanmi@portatil:~$
```

Figura 6: Líneas del fichero

```
juanmi@portatil:~$ ls -l | wc -l
12
juanmi@portatil:~$
```

Figura 7: Tuberías

1.4. Comando ps

Un **programa** es un elemento estático representado por unas líneas de código que describen lo que hace. Por otro lado, un **proceso** es un elemento dinámico que representa una instancia de un programa en ejecución. En este sentido, el comando `ps` con la opción `aux`, nos muestra los procesos que hay ejecutándose en el sistema (Figura 8).

```
juanmi@portatil:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   3648  2088 ?        Ss   16:13   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    16:13   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    16:13   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   16:13   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S<   16:13   0:00 [kworker/u:0H]
root         8  0.0  0.0      0     0 ?        S    16:13   0:00 [migration/0]
```

Figura 8: Comando “ps”

Si hemos ejecutado el programa `emacs` para editar un fichero llamado `nombres.txt`, nos aparecerá una línea al ejecutar el comando `ps aux` (Figura 9).

```
juanmi    6456  3.3  0.8  85284 21988 pts/1    Sl   17:33   0:01 emacs nombres.txt
```

Figura 9: Comando “ps”

1.5. Comando kill y killall

Podemos matar un proceso que está en ejecución mediante el comando `kill`. Para ello, al ejecutar el comando `ps aux`, vemos que en la segunda columna hay un número que identifica a cada proceso (PID). Para matar un proceso concreto, usaremos el comando `kill` seguido del PID que queramos matar (Figura 10).

También podemos matar todos los programas de un tipo, mediante la orden `killall` seguida del nombre del proceso (Figura 11).

```
juanmi@portatil:~$ kill -9 6456
juanmi@portatil:~$
```

Figura 10: Comando “kill”

```
juanmi@portatil:~$ killall -9 emacs
[2]+  Terminado (killed)      emacs nombres.txt
juanmi@portatil:~$
```

Figura 11: Comando “killall”

1.6. Comando grep

El comando `grep` nos permite hacer una búsqueda de cadenas de caracteres. Este comando tiene multitud de utilidades. A modo de ejemplo, si tenemos un fichero con una serie de nombres, podemos filtrar aquellas líneas que contengan una cadena concreta. Este comando se suele utilizar combinado con una tubería con otros comandos, como el `cat` (Figura 12).

```
juanmi@portatil:~$ cat nombres.txt
Juan Miguel
Manolo
Juan Jose
Vicente
Antonio
juanmi@portatil:~$ cat nombres.txt | grep "Juan"
Juan Miguel
Juan Jose
juanmi@portatil:~$
```

Figura 12: Comando “grep”

Sin embargo, también podemos utilizar este comando sin tubería. Por ejemplo, si queremos ver todas las cadenas que aparezcan en los ficheros de nuestro directorio actual, podemos hacerlo según la Figura 13. Esto nos mostrará los ficheros que tienen las cadenas buscadas.

```
juanmi@portatil:~$ grep Juan ./*
./nombres2.txt:Juan Miguel
./nombres2.txt:Juan Jose
./nombres.txt:Juan Miguel
./nombres.txt:Juan Jose
juanmi@portatil:~$
```

Figura 13: Comando “grep”

Si además queremos buscar en subdirectorios, utilizaremos el parámetro `-R` (Figura 14).

```
juanmi@portatil:~$ grep -R "Juan" ./*
./directorio_nuevo/nombres.txt:Juan Miguel
./directorio_nuevo/nombres.txt:Juan Jose
./nombres2.txt:Juan Miguel
./nombres2.txt:Juan Jose
./nombres.txt:Juan Miguel
./nombres.txt:Juan Jose
juanmi@portatil:~$
```

Figura 14: Comando “grep”

1.7. Comando find

El comando `find` nos permite buscar ficheros por su nombre. Para ello, especificamos dónde queremos buscar (un . es el directorio actual), seguido del parámetro `-name` para indicar el nombre a buscar y `- print` para indicar que queremos que nos muestre dónde se encuentra. El resultado de la búsqueda son los directorios dónde está el fichero buscado (Figura 15).

```
juanmi@portatil:~$ find . -name "nombres.txt" -print
./directorio_nuevo/nombres.txt
./nombres.txt
juanmi@portatil:~$
```

Figura 15: Comando "find"

Si no nos acordamos del nombre completo, podemos utilizar el caracter `*` para que nos encuentre todos los ficheros que cumplan un patrón (Figura 16).

```
juanmi@portatil:~$ find . -name "nombre*" -print
./directorio_nuevo/nombres.txt
./nombres2.txt
./nombres.txt
juanmi@portatil:~$
```

Figura 16: Comando "find"

1.8. Comando head y tail

El comando `head` seguido de un número `-n`, nos muestra las `n` primeras líneas de un fichero. Este comando también se suele utilizar combinado con tuberías (Figura 17).

```
juanmi@portatil:~$ cat salida.txt
Descargas
Documentos
Escritorio
examples.desktop
Imágenes
Música
Plantillas
Público
salida.txt
Vídeos
juanmi@portatil:~$ cat salida.txt | head -2
Descargas
Documentos
juanmi@portatil:~$
```

Figura 17: Comando "head"

De una forma similar, el comando `tail` nos muestra las `n` últimas líneas de un fichero (Figura 18).

Estos dos comandos también se suelen utilizar concatenados (Figura 19).

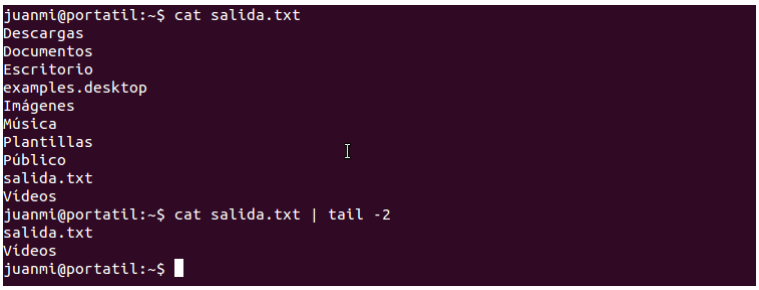

Figura 18: Comando “tail”

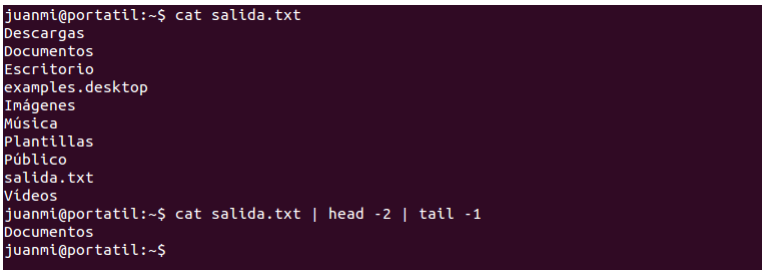

Figura 19: Comandos “head” y “tail”