Linux Básico
Laboratorio: Direccionamiento y tuberías

Presentado por:
Fabián Alberto Sánchez Ruiz

Presentado a:
Jesús Manuel Arrieta Madrid

Universidad de Córdoba
Facultad de Ingeniería

Montería - 2020

Taller: Direccionamiento y tuberías en Linux

Introducción

El siguiente documento es un taller en donde se evidencia el manejo y utilidad de las sentencias de direccionamiento y las tuberías en sistema Linux, con capturas y direcciones, además de múltiples ejemplos. Todo el procedimiento, incluyendo las capturas de pantalla, fueron realizadas en la distribución de Linux Debian 10 Buster.

1. Redireccionamiento y pipelines

En las terminales de cualquier sistema Linux podemos redireccionar la información que nos retorne como salida un comando cualquiera, como por ejemplo **cat** o **ls**, y mandársela a un archivo para que almacene esa información. Esto es posible gracias a una serie de símbolos sintácticos que existen la terminal de Linux para estas funciones de redireccionamiento.

- Símbolo **>:** Este símbolo se pone delante de un comando de retorno y a continuación del nombre de un fichero. La cadena de texto que devuelva el comando será guardada dentro del archivo, remplazando lo que el archivo tenía anteriormente. Si el fichero no existe, se creará automáticamente con el nombre que se le haya puesto.

- Símbolo **>>:** Este símbolo hace básicamente lo mismo que el anterior, pero con este no se sobrescribirá la información que pueda haber tenido o no el archivo, sino que simplemente se agrega al final del archivo sin borrar nada de su información anterior.

- Símbolo **<:** Como se puede imaginar, este símbolo sirve para lo contrario que los dos anteriores. El fichero no sufrirá modificación alguna, sino que se usará su información como entrada para el comando que se le haya asignado.

Ahora, unos ejemplos de uso de los redireccionamientos:

```
fabiloco@miPC:~/Documents/practica$ ls
fabiloco@miPC:~/Documents/practica$ ls -l ../
total 48
drwx------ 2 fabiloco fabiloco 4096 Oct 29 21:30 10cat
drwx------ 2 fabiloco fabiloco 4096 Oct 28 20:32 11more
drwx------ 2 fabiloco fabiloco 4096 Oct  5 16:45 12less
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 18:45 13uniq
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 21:41 14sort
drwx------ 3 fabiloco fabiloco 4096 Oct 29 22:29 17archivosTar
drwxr-xr-x 3 fabiloco fabiloco 4096 Oct 30 01:22 18archivosZip
-rw-r--r-- 1 fabiloco fabiloco 5581 Oct  5 17:54 Comandos
-rw-r--r-- 1 fabiloco fabiloco  362 Oct 29 18:22 documentoCopia.txt
-rw-r--r-- 2 fabiloco fabiloco   22 Oct 29 16:05 enalceSimbolico.txt
drwxr-xr-x 2 fabiloco fabiloco 4096 Nov  7 21:14 practica
fabiloco@miPC:~/Documents/practica$ ls -l ../ > archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$ ls
archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$
```

Como se puede ver, estoy en el directorio **práctica**, el cual no contiene ningún fichero ni directorio. Hago un **ls -l** un nivel más abajo, en el directorio **Documents** para todos los archivos y ficheros que tengo ahí.

Quiero guardar esa información en un archivo de texto, para saber cuantos archivos tenia ahí el día de hoy, entonces hago el respectivo uso del redireccionamiento usando el mismo comando, seguido del símbolo **>** seguido del nombre del archivo, el cual por cierto no existe, pero como se menciono más arriba, si el archivo no existe, este se crea automáticamente con el nombre que se le haya dado. El comando no imprimirá la consola directamente lo que haya en el archivo, sino que lo guardará en el nuevo fichero **archivosDocuments.txt.**

Luego, se hace otro **ls** para comprobar que el archivo se haya creado correctamente. Ahora, si le hacemos el comando **cat** a dicho archivo, debería mostrarnos el retorno del comando **ls -l ../**

```
fabiloco@miPC:~/Documents/practica$ ls
archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$ cat archivosDocuments.txt
total 48
drwx------ 2 fabiloco fabiloco 4096 Oct 29 21:30 10cat
drwx------ 2 fabiloco fabiloco 4096 Oct 28 20:32 11more
drwx------ 2 fabiloco fabiloco 4096 Oct  5 16:45 12less
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 18:45 13uniq
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 21:41 14sort
drwx------ 3 fabiloco fabiloco 4096 Oct 29 22:29 17archivosTar
drwxr-xr-x 3 fabiloco fabiloco 4096 Oct 30 01:22 18archivosZip
-rw-r--r-- 1 fabiloco fabiloco 5581 Oct  5 17:54 Comandos
-rw-r--r-- 1 fabiloco fabiloco  362 Oct 29 18:22 documentoCopia.txt
-rw-r--r-- 2 fabiloco fabiloco   22 Oct 29 16:05 enalceSimbolico.txt
drwxr-xr-x 2 fabiloco fabiloco 4096 Nov  7 21:14 practica
fabiloco@miPC:~/Documents/practica$
```

Podemos ver que, efectivamente, el contenido del archivo **archivoDocuments.txt** es el retorno del comando **ls -l ../**.

Ahora, para probar el siguiente símbolo, digamos que quiero añadirle la fecha al archivo, porque como ya dije, quiero guardar la información de los archivos que tengo el día de hoy, pero no quiero borrar nada de lo que tenía dicho archivo. Entonces para eso, se hace uso del símbolo **>>**.

```
fabiloco@miPC:~/Documents/practica$ ls
archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$ echo "Registro de los archivos dentro de D
ocuments el 7/11/2020" >> archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$
```

De esta forma, con el comando echo, que lo único que hace es retornar la cadena de texto que le demos como parámetro, le envío la cadena al archivo para que la anexe al final. Ahora, hacemos **cat** al documento para ver si se guardó como queríamos.

```
fabiloco@miPC:~/Documents/practica$ ls
archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$ cat archivosDocuments.txt
total 48
drwx------ 2 fabiloco fabiloco 4096 Oct 29 21:30 10cat
drwx------ 2 fabiloco fabiloco 4096 Oct 28 20:32 11more
drwx------ 2 fabiloco fabiloco 4096 Oct  5 16:45 12less
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 18:45 13uniq
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 21:41 14sort
drwx------ 3 fabiloco fabiloco 4096 Oct 29 22:29 17archivosTar
drwxr-xr-x 3 fabiloco fabiloco 4096 Oct 30 01:22 18archivosZip
-rw-r--r-- 1 fabiloco fabiloco 5581 Oct  5 17:54 Comandos
-rw-r--r-- 1 fabiloco fabiloco  362 Oct 29 18:22 documentoCopia.txt
-rw-r--r-- 2 fabiloco fabiloco   22 Oct 29 16:05 enalceSimbolico.txt
drwxr-xr-x 2 fabiloco fabiloco 4096 Nov  7 21:14 practica
Registro de los archivos dentro de Documents el 7/11/2020
fabiloco@miPC:~/Documents/practica$
```

Ahí podemos apreciar que se anexo la cadena de texto al final, sin alterar ni sobrescribir la información que tenía anteriormente el documento.

Ahora, digamos que quiero crear una copia del fichero **archivosDocuments.txt** por seguridad, pero un nivel más abajo. Para eso, podemos usar el comando **tee** seguido del símbolo **<.**

```
fabiloco@miPC:~/Documents/practica$ ls
archivosDocuments.txt
fabiloco@miPC:~/Documents/practica$ tee ../copiaArchivosDocuments.txt < archiv
osDocuments.txt
total 48
drwx------ 2 fabiloco fabiloco 4096 Oct 29 21:30 10cat
drwx------ 2 fabiloco fabiloco 4096 Oct 28 20:32 11more
drwx------ 2 fabiloco fabiloco 4096 Oct  5 16:45 12less
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 18:45 13uniq
drwxr-xr-x 2 fabiloco fabiloco 4096 Oct 29 21:41 14sort
drwx------ 3 fabiloco fabiloco 4096 Oct 29 22:29 17archivosTar
drwxr-xr-x 3 fabiloco fabiloco 4096 Oct 30 01:22 18archivosZip
-rw-r--r-- 1 fabiloco fabiloco 5581 Oct  5 17:54 Comandos
-rw-r--r-- 1 fabiloco fabiloco  362 Oct 29 18:22 documentoCopia.txt
-rw-r--r-- 2 fabiloco fabiloco   22 Oct 29 16:05 enalceSimbolico.txt
drwxr-xr-x 2 fabiloco fabiloco 4096 Nov  7 21:14 practica
Registro de los archivos dentro de Documents el 7/11/2020
fabiloco@miPC:~/Documents/practica$
```

2. Pipelines

Esta es otra herramienta que podemos usar en la terminal de Linux, la cual también es conocida como tubería y es muy parecida de hecho a los antes vistos, pero con diferencias. Con los pipelines podemos concatenar varios comandos, todos en una sola sentencia, en los cuales, la entrada de uno será la salida del comando anterior. Los pipelines se simbolizan con el carácter **"|"**, y para usarlo solo basta poner un comando, seguido del carácter y otro comando, y así ya estaríamos usando los pipelines.

Hay muchas formas de utilizar los pipelines, depende mucho de lo que queramos hacer y como queremos optimizar nuestro trabajo, ya que para eso sirven, para permitirnos hacer varias cosas de forma mucho más ágil y eficiente.

Ahora, veremos varios ejemplos sencillos usando los pipelines:

En el siguiente ejemplo lo que hice fue usar el retorno del comando **ls** dentro del escritorio **Documents/** y usarlo como entrada para que el comando **grep** busque algún archivo que comience con **Com**, que fue el filtro que le di. Básicamente lo que hice con esto fue buscar un archivo con este nombre dentro del directorio **Documents/.**

```
fabiloco@miPC:~/Documents$ ls
10cat   13uniq         18archivosZip              documentoCopia.txt
11more  14sort         Comandos                   enalceSimbolico.txt
12less  17archivosTar  copiaArchivosDocuments.txt  practica
fabiloco@miPC:~/Documents$ ls | grep Com
Comandos
fabiloco@miPC:~/Documents$
```

Este uso puede ser mucho más práctico para un directorio que tenga muchísimos más archivos, pero la intención se entiende. Al final, podemos ver que nos retorna la palabra Comandos, que es, efectivamente, un fichero dentro del directorio.

Siguiente ejemplo, ahora tenemos este archivo **lista.txt**

```
fabiloco@miPC:~/Documents$ ls
10cat   14sort         copiaArchivosDocuments.txt  practica
11more  17archivosTar  documentoCopia.txt
12less  18archivosZip  enalceSimbolico.txt
13uniq  Comandos       lista.txt
fabiloco@miPC:~/Documents$ cat lista.txt
Perro
Zorro
Elephante
Alce
Ardilla
Gato
Caiman
Gallo

fabiloco@miPC:~/Documents$
```

Si queremos buscar en el archivo los elementos con la letra a, con los pipelines podemos hacer los siguiente:

```
fabiloco@miPC:~/Documents$ cat lista.txt | grep -i a
Elephante
Alce
Ardilla
Gato
Caiman
Gallo
fabiloco@miPC:~/Documents$
```

Usamos la opción **-i** para que el comando ignore las mayúsculas o minúsculas.

Ahora, si queremos al mismo tiempo ordenar el archivo de forma alfabética, podemos hacer lo siguiente.

```
fabiloco@miPC:~/Documents$ cat lista.txt | grep -i a | sort
Alce
Ardilla
Caiman
Elephante
Gallo
Gato
fabiloco@miPC:~/Documents$
```

Y de esta forma, imprimimos el contenido de **lista.txt** con el comando **cat,** pero lo filtramos para que solo nos mostrara aquellas líneas que tienen la letra **a** y además de eso, organizamos la salida de forma alfabética con el comando **sort,** todo eso hecho en una sola sentencia.

Si además de eso, queremos guardar esta salida en un nuevo archivo, y al mismo tiempo imprimirlo, podemos usar el comando **tee** que guarda la salida de cualquier comando o secuencia de comandos en un fichero que le digamos, y si no existe, directamente lo creara.

```
fabiloco@miPC:~/Documents$ ls
10cat   14sort          copiaArchivosDocuments.txt  practica
11more  17archivosTar   documentoCopia.txt
12less  18archivosZip   enalceSimbolico.txt
13uniq  Comandos        lista.txt
fabiloco@miPC:~/Documents$ cat lista.txt | grep -i a | sort | tee listaOrdenad
a.txt
Alce
Ardilla
Caiman
Elephante
Gallo
Gato
fabiloco@miPC:~/Documents$
```

Acá podemos ver que añadimos todos los comandos anteriores y además añadimos el comando **tee** para que guardara la salida final después de haber sido ordenado y filtrado, dentro del archivo **listaOrdenada.txt.**

```
fabiloco@miPC:~/Documents$ ls
10cat   14sort          copiaArchivosDocuments.txt  lista.txt
11more  17archivosTar   documentoCopia.txt          practica
12less  18archivosZip   enalceSimbolico.txt
13uniq  Comandos        listaOrdenada.txt
fabiloco@miPC:~/Documents$ cat listaOrdenada.txt
Alce
Ardilla
Caiman
Elephante
Gallo
Gato
fabiloco@miPC:~/Documents$
```

Vemos que tenemos toda esa salida guardada en un archivo.

Recordemos que todo lo anterior lo hicimos en una sola y única sentencia de comando, por lo tanto, podemos comprobar que las tuberías son una herramienta sumamente poderosa. Con el manejo de las tuberías podremos mejorar bastante nuestra eficiencia dentro de cualquier sistema Linux. Las posibilidades con los pipelines son infinitas.