Introducción

Los flujos, la redirección y las tuberías son algunas de las herramientas más potentes de la línea de comandos. Linux trata la entrada y salida de comandos como un flujo (datos que podemos manipular). Normalmente la entrada es el teclado y la salida la pantalla, pero se pueden redirigir estos flujos de entrada y salida hacia otros comandos o archivos. La tuberia [|] se utiliza para redirigir la salida de un comando hacia otro. Esto supone una magnífica herramienta de conexión entre comandos, lo cual permite realizar tareas complejas combinando varias herramientas sencillas.

Para empezar a entender el redireccionamiento y tuberías hay que comprender primero los diferentes tipos de flujos de entrada y salida. Todo comando en Linux posee tres canales básicos por donde fluye la información y estos canales son:

- **Entrada Estándar** , en inglés standard input (**stdin**) es el mecanismo por el cual un usuario le indica a los programas la información que estos deben procesar. Esta información suele introducirse por el teclado. Ejemplos:

```
$ cat
$ wc
```

  Al ejecutar estos ejemplos sin argumentos el comando espera a que el usuario inserte información para posteriormente procesarla, dicha información de entrada debe ser proporcionada por la entrada estándar, en este caso por el teclado.

- **Salida Estándar** , en inglés standard output (**stdout**) es el método por el cual el programa puede comunicarse con el usuario. Por defecto la salida estándar es la pantalla donde se ejecutan dichos comandos. Ejemplos:

```
$ ls -l /etc/hosts /etc/passwd
-rw-r--r-- 1 root root 251 feb 23 15:52 /etc/hosts
-rw-r--r-- 1 root root 1635 feb 27 19:05 /etc/passwd
$ head -10 /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
```

  Al terminar de procesar la información dada como argumentos el comando mostrará un mensaje en la pantalla donde se ha ejecutado. Dicha pantalla o ventana será la salida estándar del comando ejecutado.

- **Error Estándar** , en inglés standard error output (**stderr**) es utilizado para mostrar mensajes de error que surjan durante el transcurso de su ejecución. Al igual que **stdout, stderr** será la pantalla donde se procesaron las instrucciones. Ejemplos:

```
$ rm esteArchivoNoExiste
rm: no se puede borrar «esteArchivoNoExiste»: No existe el fichero o el directori
$ rmdir /carpetaInexistente
rmdir: fallo al borrar «/carpetaInexistente»: No existe el fichero o el directorio
```

  De igual forma **stderr** devolverá mensajes que indicarán los errores de los comandos ejecutados.

El flujo de datos para redireccionamiento y tuberías en línea de comandos se inicia de izquierda a derecha.

Redireccionamiento

Para redireccionar la salida estándar de un comando a un archivo se utiliza el símbolo **>** después del cual debe indicarse el nombre del archivo que se creará:

```
$ cat /etc/passwd > copia_passwd
```

Si el archivo existe previamente, se sobrescribirá. Para agregar los valores sin borrar el contenido existente, se utiliza **>>** .

Para redireccionar el contenido de un archivo a la entrada estándar de un comando se utiliza **<** . En este caso, el flujo de datos va de derecha a izquierda. Este tipo de redirección se suele utilizar con comandos como **tr** que no lee archivos directamente.

Normalmente se suele redireccionar la salida estandar pero si se quiere redireccionar la salida de errores podemos utilizar **2>** . Para redireccionar ambos simultáneamente, se usa **&>** .

```
$ ls x* z*
ls: z*: No such file or directory
xaa xab
```

Muestra un error porque no existe ningún archivo que empiece por z y muestra los archivos que empiezan por x

```
$ ls x* z* >stdout.txt 2>stderr.txt
```

Se ejecuta el mismo comando anterior pero redirigiendo la salida de errores al fichero stderr.txt y la salida estándar al fichero *stdout.txt*

```
$ ls w* y*
ls: w*: No such file or directory
yaa yab
```

Se ejecuta **ls** de nuevo en el directorio y muestra un error porque no existe ningún archivo que empiece por w y muestra los archivos que empiezan por y.

```
$ ls w* y* >>stdout.txt 2>>stderr.txt
```

Se ejecuta el mismo comando anterior pero redirigiendo la salida de errores al fichero *stderr.txt* y la salida estándar al fichero *stdout.txt* . Como estos ficheros contenían información se ha utilizado el redireccionador >> para agregar el contenido y no sobrescribirlo.

```
$ cat stdout.txt
xaa
xab
yaa
yab
```

Se muestra el contenido del fichero al que hemos redireccionado la salida estándar. Como se observa contiene la salida estándar de las dos ejecuciones del comando **ls.**

```
$ cat stderr.txt
ls: z*: No such file or directory
ls: w*: No such file or directory
```

Se muestra el contenido del fichero al que ha redireccionado la salida de errores. Como se observa contiene la salida de errores de las dos ejecuciones del comando **ls.**

A continuación se listan los operadores de redirección más utilizados:

> Crea un fichero nuevo con el contenido de la salida estándar. Si el archivo existe se sobrescribirá.

>> Añade la salida estándar al contenido de un archivo existente. Si no se especifica el archivo este se crea.

2> Crea un fichero nuevo con el contenido de la salida de errores. Si el archivo existe se sobrescribirá.

2>> Añade la salida de errores al contenido de un archivo existente. Si no se especifica el archivo este se crea.

&> Crea un fichero nuevo con el contenido de la salida estándar y la salida de errores. Si el archivo existe se sobrescribirá.

< Envía el contenido del fichero especificado para su uso en la entrada estándar.

<< Añade más texto a la entrada estandar.

<> Usa un fichero tanto para la entrada estándar como para la salida estándar.

A continuación se listan los operadores de redirección más utilizados:

> Crea un fichero nuevo con el contenido de la salida estándar. Si el archivo existe se sobrescribirá.

>> Añade la salida estándar al contenido de un archivo existente. Si no se especifica el archivo este se crea.

2> Crea un fichero nuevo con el contenido de la salida de errores. Si el archivo existe se sobrescribirá.

2>> Añade la salida de errores al contenido de un archivo existente. Si no se especifica el archivo este se crea.

&> Crea un fichero nuevo con el contenido de la salida estándar y la salida de errores. Si el archivo existe se sobrescribirá.

< Envía el contenido del fichero especificado para su uso en la entrada estándar.

<< Añade más texto a la entrada estandar.

<> Usa un fichero tanto para la entrada estándar como para la salida estándar.

Tubería (pipe)

Es posible enviar la salida de un comando a la entrada de otro comando, utilizando el carácter de tubería [|]. Por ejemplo:

```
$ cat /home/usuario/prueba.txt | grep -i linea
```

Filtra las líneas que contengan la palabra línea en el archivo *prueba.txt.*

También es posible redireccionar la salida simultáneamente tanto para un archivo como para *stdout,* por medio del comando **tee** . Para ello se redirige la salida del comando al comando **tee** suministrando a éste un nombre de archivo para almacenar la salida:

```
$ cat /home/usuario/prueba.txt | tee salida
```

El contenido de */home/usuario/prueba.txt* se mostrará en la pantalla y se copiará en el archivo salida.

Sustitución de Comandos

También es posible utilizar la salida de un comando como argumento para otro, utilizando comillas ejecutivas [`] (en el teclado se ubica en la tecla de apertura de corchetes):

```
$ ls -ld `echo $PATH|tr ':' ' '`
drwxr-xr-x 2 root root   4096 feb 24 13:50 /bin
drwxr-xr-x 2 root root  69632 mar  1 21:39 /usr/bin
drwxr-xr-x 2 root root   4096 feb 24 13:46 /usr/games
drwxrwsr-x 2 root staff  4096 mar  1 21:04 /usr/local/bin
drwxrwsr-x 2 root staff  4096 ago  4  2010 /usr/local/games
```

En este ejemplo se formatea el contenido de la variable PATH con el comando **tr** , para listar con **ls** los directorios contenidos en dicha variable.

El resultado será el mismo con:

```
$ ls -ld $(echo $PATH|tr ':' ' ')
drwxr-xr-x 2 root root   4096 feb 24 13:50 /bin
drwxr-xr-x 2 root root  69632 mar  1 21:39 /usr/bin
drwxr-xr-x 2 root root   4096 feb 24 13:46 /usr/games
drwxrwsr-x 2 root staff  4096 mar  1 21:04 /usr/local/bin
drwxrwsr-x 2 root staff  4096 ago  4  2010 /usr/local/game
```

Parecido a la sustitución de comandos el comando **xargs** desempeña la función de intermediario, pasando los datos que recibe por **stdin** como argumento para un segundo comando.

```
$ ls /home/usuario/Descargas/ | xargs rm -rf
```

Ejecuta "rm -rf" para cada archivo del directorio */home/usuario/Descargas.*