Redirecciones

1. Fundamentos

Las redirecciones son una de las más importantes posibilidades proporcionadas por el shell. Por redirección se entiende la posibilidad de redireccionar la visualización de la pantalla hacia un archivo, una impresora o cualquier otro periférico, los mensajes de errores hacia otro archivo, de sustituir la introducción vía teclado por el contenido de un archivo.

Cualquier flujo de datos en entrada o salida de comando pasa por un canal. Como sucede con el agua, es posible desviar el curso de los datos hacia otro destino o desde otra fuente.

Linux utiliza canales de entradas/salidas para leer y escribir sus datos. Por defecto, el canal de entrada es el teclado, y el canal de salida, la pantalla. Los errores, direccionados por defecto a la pantalla, son tratados como un canal especial.

Es posible redireccionar estos canales hacia archivos o flujo de texto de manera transparente para los comandos Linux.

2. De salida

Se puede utilizar el carácter **>** para redireccionar la salida estándar (la que va normalmente en la pantalla). Luego se indica el nombre del archivo donde se colocarán los resultados de salida.

```
$ ls -l > resultado.txt
$ cat resultado.txt
total 1
-rw-r--r--  1 Administ ssh_user    0 Jul  4 12:04 PEPITO
-rw-r--r--  1 Administ ssh_user    0 Jul 25 15:13 resultado.txt
-rw-r--r--  1 Administ ssh_user  171 Jul 25 15:13 test.txt
```

Si no existe, se creará el archivo. Si existe, se sobreescribirá su contenido, incluso si el comando tecleado no es correcto. **El shell empieza primero por crear el archivo y luego ejecuta el comando**.

Es un aspecto importante de las redirecciones: se interpretan las redirecciones de derecha a izquierda, y se instalan las redirecciones ANTES de la ejecución de los comandos. Hay que crear el archivo antes de poder escribir en él. De ahí que, incluso si el comando es falso, se crea o se «chafa» el archivo...

Para añadir datos a continuación del archivo, o sea, sin sobreescribirlos, se utiliza la doble redirección >>. Se añade el resultado del comando al final del archivo.

```
$ cat resultado.txt
total 1
-rw-r--r--  1 Administ ssh_user     0 Jul  4 12:04 PEPITO
-rw-r--r--  1 Administ ssh_user     0 Jul 25 15:13 resultado.txt
-rw-r--r--  1 Administ ssh_user   171 Jul 25 15:13 test.txt
mar 23 mar 2021 22:03:29 UTC
```

3. En entrada

Los comandos que esperan datos o parámetros desde el teclado pueden también recibirlos desde un archivo usando el carácter <. Un ejemplo con el comando **wc** (*word count*), que permite contar el número de líneas, de palabras y de caracteres de un archivo:

```
$ wc < resultado.txt
    5   35 239
```

4. Documento en línea

La redirección << es algo particular. Permite la utilización de los documentos en línea. Encontrará a veces el término Herescript o Here Document. Esto permite la inserción de un texto hasta un punto dado y el envío de su resultado a un comando o un filtro. Se autorizan las redirecciones clásicas. Después del <<, ha de indicar una cadena que define el final de la inserción; por ejemplo, aquí 'end'.

```
$ tr "[a-z]" "[A-Z]" << end
> hola amigos
> esto es un ejemplo
> de herescript
> end
HOLA AMIGOS
ESTO ES UN EJEMPLO
DE HERESCRIPT
```

Existe una variante, llamada Here String, que usa la redirección <<<. A diferencia de <<, esta no requiere delimitar el final.

```
$ cat <<< toto
toto
```

Lo más interesante es el uso aunado con una substitución de comandos (en este mismo capítulo, sección Programación shell):

```
$ tr "[a-z]" "[A-Z]" <<< $(cat archtest)
HOLA AMIGOS ESTO ES UN EJEMPLO DE HERESCRIPT
```

5. Los canales estándares

Se puede considerar un canal como un archivo que posee su propio descriptor por defecto, y en el cual se puede leer o escribir.

- El canal de entrada estándar se llama **stdin** y lleva el descriptor 0.
- El canal de salida estándar se llama **stdout** y lleva el descriptor 1.
- El canal de error estándar se llama **stderr** y lleva el descriptor 2. Se puede redireccionar el canal de error hacia otro archivo.

```
$ rmdir directorio2
rmdir: `directorio2': No such file or directory
$ rmdir directorio2 2>error.log
$ cat error.log
rmdir: `directorio2': No such file or directory
```

Puede redireccionar los dos canales de salida a un único archivo poniéndolos en relación. Para ello, se utiliza el **>&**. También es importante saber en qué sentido el shell interpreta las redirecciones. El shell busca primero los caracteres **<**, **>**, **>>** al final de la línea, ya que las redirecciones suelen estar al final de comando. Por lo tanto, si quiere agrupar los dos canales de salida y de error en un mismo archivo, hay que proceder como a continuación.

```
$ ls -l > resultado.txt 2>&1
```

Se redirecciona la salida 2 hacia la salida 1; por lo tanto, los mensajes de error pasarán por la salida estándar. Luego se redirecciona el resultado de la salida estándar del comando **ls** hacia el archivo resultado.txt. Este archivo contendrá, por lo tanto, a la vez la salida estándar y la salida de error. Existe otra sintaxis con resultados idénticos:

```
$ ls -l &>resultado.txt
```

Puede utilizar los dos tipos de redirección a la vez:

```
$ wc < resultado.txt > cuenta.txt
$ cat cuenta.txt
   4    29   203
```

Para escribir en el canal de error con un `echo`, haga como sigue: redirija la salida estándar

al canal de error.

```
$ echo “error” >&2
```

6. Apertura de canales

Existen tres canales estándares y se numeran de 0 a 2. Así, 0< equivale a < y 1> a >. El comando **exec** permite abrir otros siete canales numerados de 3 a 9. Por lo tanto, hay diez canales en total.

Puede e incluso debe considerar, en el marco de los procesos, sacar algunos resultados por el canal 3, otros por el 4, y así sucesivamente. Los canales están abiertos para la entrada y la salida.

```
$ exec 3>dump.log
$ ls -l >&3
$ cat dump.log
total 3952
-rw-r--r--  1 seb users  167212 oct  9 09:27 battlestar_1280.jpg
drwxr-xr-x  2 seb users    4096 mar  4 08:51 bin
drwxr-xr-x  8 seb users    4096 mar  4 08:45 cxoffice
drwx------  2 seb users    4096 mar 10 12:29 Desktop
drwx------ 13 seb users    4096 mar  6 11:49 Documents
-rw-r--r--  1 seb users       0 mar 11 11:34 dump.log
-rw-r--r--  1 seb users 3785296 dic 12 15:15 e3555_EeePC4G.pdf
drwxr-xr-x  3 seb users    4096 mar 10 11:16 Games
drwxr-xr-x  5 seb users    4096 mar 10 11:16 karchiver-3.4.2.b4
-rw-r--r--  1 seb users     358 mar 11 08:51 lista
-rw-r--r--  1 seb users     608 mar 11 09:14 tmpgrp
-rw-r--r--  1 seb users    1555 mar 11 09:15 tmppwd
```

Todo lo que se escribirá en el canal 3 se escribirá en el archivo dump.log. Luego se puede cerrar el canal relacionándolo con un seudocanal (canal de cierre -).

```
$ exec 3>&-
```

7. Filtro: definición

Un **filtro** (o un comando filtro) es un programa que sabe escribir y leer datos por los canales estándares de entrada y salida. Modifica o trata si es preciso el contenido. **wc** es un filtro. Le presentamos algunos: **more** (muestra los datos página por página), **sort** (ordena los datos), **grep** (criterios de búsqueda).

8. Pipelines/tuberías

Las redirecciones de entrada/salida como las que acaba de ver permiten redireccionar los resultados hacia un archivo. Luego se puede reinyectar en un filtro para extraerle otros resultados. Esto obliga a teclear dos líneas: una para la redirección hacia un archivo, otra para redireccionar este archivo hacia el filtro. Las **tuberías** o **pipes** permiten redirigir el canal de salida de un comando hacia el canal de entrada de otro. El carácter que lo permite | está accesible con la combinación [AltGr] **1** de los teclados españoles, o [Alt][Shift] l (letra L) de los teclados Mac.

```
$ ls -l > resultado.txt
$ wc < resultado.txt
```

se convierte en

```
$ ls -l | wc
```

Es posible colocar varios | en una misma línea.

```
$ ls -l | wc | wc
```

```
      1      3     24
```

El primer comando no tiene por qué ser un filtro. No es el caso más habitual. Lo importante es que se debe facilitar un resultado. Ídem para el último comando, que puede ser por ejemplo un comando de edición o impresión. Finalmente, el último comando puede ser objeto de una redirección de salida.

```
$ ls -l | wc > resultado.txt
```