

REDIRECCIONAMIENTOS Y FILTROS

Redireccionamiento

Stdin: archivo estándar de entrada, su descriptor es 0 y es donde los programas leen su entrada.

Stdout: archivo estándar de salida, su descriptor es el 1 y es donde los programas envían sus resultados.

Stderr: archivo estándar de error, su descriptor es 2 y es donde los programas envían sus salidas de error.

Redireccionamiento de la entrada estándar

Cualquier orden que lea su entrada en **stdin** puede ser avisada para que tome dicha entrada de otro archivo.

Cuando utiliza el símbolo <

```
cat < HOLA.TXT
```

Debido a que se usa el símbolo menor que (<) para separar el comando cat de un archivo, cat lee la salida de HOLA.TXT

Otro ejemplo sería enviar un archivo al correo de un usuario llamado miguel. Será

Mail miguel <vis.c

El archivo vis.c contendrá el argumento del archivo que enviamos al usuario miguel.

Redireccionamiento de salida

El redireccionamiento de salida estándar (**stdout**) significa hacer que la shell cambie lo que está considerado como entrada estándar.

Para redireccionar la salida estándar, usaremos el símbolo >.

Al colocar > tras el comando cat (o tras cualquier utilidad o aplicación que escriba la salida estándar) reorientará su salida al nombre de archivo que siga al símbolo.

Para redirigir la salida cat a un archivo, escriba lo siguiente en el intérprete de comandos (si presiona la tecla [Intro] lo llevará a la siguiente línea en blanco):

```
cat > sneakers.txt
```

Use el redireccionamiento de la salida y crea otro archivo y llámelo home.txt. Con el contenido de los dos archivos creas el archivo friday.txt

```
cat sneakers.txt home.txt > friday.txt
```

Adjuntar a la salida estándar

Puede utilizar el redireccionamiento para añadir información nueva al final de un archivo ya existente. Parecido a cuando ha usado el símbolo >, le indica a su shell que envíe la información a algún otro sitio que no sea el de la salida estándar.

No obstante, cuando usa >>, está *añadiendo* información más que reemplazándola.

Le presentamos un ejemplo práctico para aclarar este concepto. En este ejemplo unamos dos archivos creados anteriormente (sneakers.txt y home.txt) utilizando el símbolo para adjuntar la salida. Queremos añadir la información presente en home.txt a la ya presente en sneakers.txt. Basta con teclear:

```
cat home.txt >> sneakers.txt
```

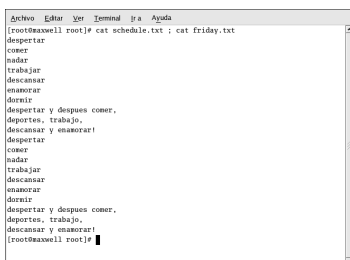
Ahora, verifique el archivo usando el comando cat sneakers.txt. La salida final muestra los contenidos del archivo home.txt al final del archivo:

Añadiendo la salida directamente, hemos ahorrado uno o dos pasos, en vez de crear uno nuevo.

Compare los resultados de los archivos sneakers.txt y friday.txt notará que son iguales. Para efectuar esta comparación teclee:

```
cat sneakers.txt; cat friday.txt
```

Los contenidos de los dos archivos se visualizan en la pantalla — primero sneakers.txt, luego friday.txt.



Redireccionamiento de errores.

La mayoría de las órdenes de Linux producen diagnósticos para ver si algo va mal en su ejecución. Con cualquier orden que genere mensajes de error enviándolos a stderr (por defecto, el Terminal) podemos redireccionar su salida a otro archivo utilizando el operador

2> o 2>>, dependiendo de si lo que queremos crear o añadir datos al archivo, respectivamente.

A modo ejemplo vamos a ejecutar la orden cp sin argumentos

cp

cp: faltan parámetro

Pruebe ‘cp –help’ para mas infomacion.

Vemos como por pantalla, se visualizan los mensajes de error generados por cp. Es lógico el error, puesto que cp necesita como mínimo dos argumentos para poderse ejecutar correctamente. Si queremos que estos mensajes de error no salgan por pantalla podríamos realizarlo de la siguiente forma.

cp >basura

Saldrá el mensaje en el arcivo basura. Ahora si visualizamos basura contendrá los mensajes de error generados por cp.

Si lo único que deseamos es evitarnos mensajes de error pero sin crear archivos, debemos enviar dichos mensajes a un archivo de dispositivo denominado /dev/null. El archivo dev/null es un pozo sin fondo donde podemos enviar toda la información no deseada sin tener que preocuparnos de borrar su contenido.

cp 2>> /dev/null

Tuberías y paginadores

Las tuberías (también conocidas como pipes) relacionan la salida estándar de un comando con la entrada estándar de otro comando.

Existen varias opciones disponibles con el comando ls, pero ¿qué pasa si la visualización del contenido de un directorio es demasiado rápida como para verla?

Vamos a ver el contenido del directorio /etc/ con el comando:

```
ls -al /etc
```

¿Cómo podemos visualizar tranquilamente la salida antes de que desaparezca de la pantalla?

Una forma es entubando la salida a una utilidad llamada less, un paginador que permite ver la información por páginas (en la pantalla).

Use la barra vertical (|) para entubar comandos.

```
ls -al /etc | less
```

De esta manera verá el contenido de /etc en una pantalla a la vez. Para acceder a la pantalla siguiente, pulse [Barra espaciadora], para salir, presione [Q]. También puede usar las flechas direccionales para navegar con less.

Concepto de filtro

Cualquier proceso que lea su entrada en la entrada estándar y escriba su salida en la salida estándar se denomina **filtro**.

Filtros standard: sort, more, less, wc, grep, head, tail.

Filtros avanzados: cut, tr, pr, tee, find

Orden sort:

Permite ordenar y fusionar archivos de texto. Las clasificaciones pueden basarse en campos y de caracteres numéricos, y pueden especificarse varias claves de ordenación:

Sintaxis: sort [-dtbn] [+campo] [archivo(s)]

-d: ordena según el criterio de un diccionario. Para ordenación, solo se utilizan las letras, los dígitos y los blancos.

-t sep-campo: especifica que separador de campo es el carácter

-b: Ignora los espacios en blanco precedentes cuando se determina el valor de las claves de clasificación.

-n ordena numéricamente.

Comandos more y less

Estos comandos permiten visualizar un fichero pantalla a pantalla. El número de líneas por pantalla es de 23 líneas de texto y una última línea de mensajes, donde aparecerá la palabra more. Cuando se pulsa la barra espaciadora (el espacio en blanco), se visualizará la siguiente pantalla. Para salir de este comando (terminar la visualización) se pulsa **<ctrl>d** o **q**.

Por ejemplo: *more file*

El comando **less** es muy similar al anterior pero permite el desplazamiento a lo largo del texto empleando las teclas de cursores pudiendo desplazarse hacia arriba o abajo de un fichero.

Orden: wc [-lwc] [archivo(s)]

Cuenta el número de caracteres, palabras y líneas de un archivo.

-c: Cuenta sólo el número de caracteres.

-w: cuenta sólo el número de palabras.

-l: cuenta sólo el número de líneas

Comandos grep

El comando **grep** localiza una palabra, clave o frase en un conjunto de directorios, indicando en cuáles de ellos la ha encontrado. Este comando rastrea fichero por fichero, por turno, imprimiendo aquellas líneas que contienen el conjunto de caracteres buscado. Si el

conjunto de caracteres a buscar está compuesto por dos o más palabras separadas por un espacio, se colocará el conjunto de caracteres entre apóstrofes ('). Su formato es el siguiente:

grep 'conjuntocaracteres' file1 file2 file3 siendo 'conjunto caracteres' la secuencia de caracteres a buscar, y ***file1***, ***file2***, y ***file3*** los ficheros donde se debe buscar. Veamos un nuevo ejemplo:

grep 'TRIANGULARIZACION MATRIZ' matrix.f scaling.f

Este comando buscará ***TRIANGULARIZACION MATRIZ*** entre las líneas de los ficheros ***matrix.f*** y ***scaling.f***. Este comando permite seleccionar, entre todas las líneas de uno o más ficheros, aquellas que contienen un motivo que satisface una expresión regular determinada.

grep [-opción] expresión_regular [referencia...]

Las **opciones** principales son:

c lo único que se hace es escribir el número de las líneas que satisfacen la condición.

i no se distinguen mayúsculas y minúsculas.

l se escriben los nombres de los ficheros que contienen líneas buscadas.

n cada línea es precedida por su número en el fichero.

s no se vuelcan los mensajes que indican que un fichero no se puede abrir.

v se muestran sólo las líneas que no satisfacen el criterio de selección.

Las expresiones regulares

Una expresión regular es un patrón que define un conjunto de cadenas de caracteres. Las expresiones regulares se construyen de forma análoga a las expresiones aritméticas.

Una **expresión regular** es un patrón que nos permite buscar un texto formado por metacaracteres y caracteres ordinarios.

Los **metacaracteres** son ciertos caracteres con un significado específico dentro de una expresión regular. Estos caracteres tienen un significado que va más allá del símbolo que representan y tienen un comportamiento especial en una expresión regular.

Aquí tenéis una lista de metacaracteres que usamos en expresiones regulares:

- **.** Significa cualquier carácter.
- **^** Indica el principio de una línea.
- **\$** Indica el final de una línea.
- ***** Indica cero o más repeticiones del carácter anterior.
- **+** Indica una o más repeticiones del carácter anterior.
- **\<** Indica el comienzo de una palabra.
- **\>** Indica el final de una palabra.
- **** Carácter de escape. Da significado literal a un metacaracter.
- **[]** Uno cualquiera de los caracteres entre los corchetes. Ej: **[A-Z]** (desde A hasta Z).
- **[^]** Cualquier carácter distinto de los que figuran entre corchetes: Ej: **[^A-Z]**.

- { } Nos permiten indicar el número de repeticiones del patrón anterior que deben darse.
- | Nos permite indicar caracteres alternativos: Ej: (^[?&])
- () Nos permiten agrupar patrones. Ej: ([0-9A-F]+:)+

Ojo. En las expresiones regulares se distingue entre mayúsculas y minúsculas.

Nota:

Usar “entrecomillado” (caracteres \ o “” o ‘’) para que los caracteres “<”, “>”, “|”, “*”... pierdan su significado especial para el shell y tengan el significado usado por la orden “grep”.

Ejemplos de texto o “patrones”:

gato Línea contiene “gato” o “gatos” o “maragatos” (es decir, palabra entera o parte de palabra).

^gato Línea empieza por “gato”.

gato\$ Línea termina por “gato”.

^gato\$ Línea empieza y termina por “gato” (es decir, línea sólo contiene “gato”).

[Gg]at[oa] Línea contiene “Gato”, “gato”, “Gata” o “gata”.

ga[^aeiou]o Tercer carácter no es vocal

ga.o Tercer carácter cualquiera

^....\$ Línea empieza y termina por 4 caracteres cualesquiera (es decir, línea sólo contiene 4 caracteres cualesquiera).

\<ga Línea contiene palabra que empieza por “ga”.

to\> Línea contiene palabra que termina en “to”.

Vamos a ver algunos ejemplos de expresiones regulares:

grep '^La' fichero

El comando anterior nos devuelve todas las líneas del fichero que comienzan por **La**.

grep '^ *La' fichero

El comando anterior nos devuelve todas las líneas del fichero que comienzan por cualquier número de espacios seguido de **La**.

grep '^\..*' fichero

El comando anterior nos devuelve todas las líneas del fichero que comienzan por punto y tienen cualquier número de caracteres.

ls -la | grep ' \..*'

El comando anterior nos devuelve la lista de ficheros que comienzan por un espacio seguido de un punto y cualquier número de caracteres, es decir, la lista de ficheros ocultos.

ls -l | grep '^d'

El comando anterior nos devuelve la lista de ficheros que comienzan por **d**, es decir, la lista de directorios.

grep ‘^d’ text líneas que comienzan por d dentro del fichero text

grep ‘[^d]’ text líneas que no comienzan por d en el fichero txt

grep -v ^C file1 > file2 quita las líneas de file1 que comienzan por C y lo copia en file2.

head

Filtro: Mostrar la primera parte de un fichero o de la salida de una orden, por defecto, las 10 primeras líneas.

head [-opciones] [fichero(s)]

orden con S | head [-opciones]

Opciones:

-n n° o -n° Primeras n° líneas.

-c n° Primeros n° caracteres (bytes).

-c n°b Primeros n° bloques (512 bytes).

-c n°k Primeros n° kb (1 kb = 1.024 bytes).

-c n°m Primeros n° mb (1 mb = 1.024 kb= 1.048.576 bytes).

Ejemplos:

Primeras 10 líneas de un fichero:

head /etc/passwd

Primeras 20 líneas de un fichero:

head -n 20 /etc/passwd

Primeros 2.048 kb de un fichero:

head -c 2k ~/datos.txt

Primeras 5 líneas de un listado:

ls -l /etc | head -5

tail

Filtro: Mostrar la última parte de un fichero o de la salida de una orden, por defecto, las 10 últimas líneas. Si antes del “n°” se pone un “+” (por ejemplo,

-n +n°), mostrar desde dicha parte hasta el final.

tail [-opciones] [fichero(s)]

orden con S | tail [-opciones]

Opciones:

-n n° o -n° Últimas n° líneas.

Ejemplos:

tail /etc/passwd (visualiza las 10 últimas líneas del fichero passwd)

tail -n 20 /etc/passwd (visualiza las 20 líneas del fichero passwd)

Orden cut

Extrae campos de una lista de archivos. Campos pueden estar definidos como posiciones de caracteres o relativamente con un separador de campos.

Sintaxis: cut -c pos-car lista-archivos

Cut -f campos -d sep-campo -s lista-archivos

-cpos-car: posición de los caracteres que van a separar. Puede ser una lista separada por comas, un rango separado por guiones o una combinación de ambos (por ejemplo, son válidos 1,4,5 1-4 y 1-4, 5-10,25)

-fcampos: los campos que se van a separar. Los campos se distinguen por un carácter separador. Si se repite el separador varias veces seguidas, no se tratan como un separador.

Capos utiliza la misma sintaxis que pos-car.

-dsep-campo: especifica el separador de campo. Un carácter de tabulación es el valor predeterminado. sep-campo puede ser cualquier carácter.
-s : suprime la línea si no contiene carácter sep-campo.

Ejemplo: cut -f1,5 -d: /etc/passwd

Este comando extrae el identificador y los nombres de usuario desde el archivo etc/passwd. Corta desde la fila 1 al la 5 y distingue el separador de campo :

Las opciones -c y -f son mutuamente exclusivas. (-c para cortar columnas y -f para cortar campos)

Orden tr

Traduce o asigna caracteres de un archivo desde una forma a otra. El empleo de la orden tr es como convertidor de letras mayúsculas a minúsculas, y viceversa.

Sintaxis: tr [-dsc] cadena1 cadena2

-c: sustituye caracteres que encuentra en la cadena1 por los que encuentra en la cadena2.
-d: suprime caracteres especificados
-s: Filtra los caracteres repetidos

Ejemplo:

Cat fich

Este es un archivo de texto

QUE CONTIENE LETRAS MAYUSCULAS Y minúsculas

tr [A-Z] [a-z] < fich (convierte todos los caracteres de la a la z en minúsculas)
este es un archivo de texto
que contiene letras mayúsculas y minúsculas

tr [a-z] [A-Z] < fich
ESTE ES UN ARCHIVO DE TEXTO
QUE CONTIENE LETRAS MAYUSCULAS Y MINUSCULAS

tr [A-Z] x < fich
xeste es un archivo de texto
xxx xxxxxxxxx xxxxxx xxxxxxxxxx x minúsculas

tr -d [A-Z] < fich
ste es un archivo de texto
minúsculas.

Comando tee

A veces interesa que la salida de un comando, además de redirigirse a un determinado fichero, se bifurque también hacia la Terminal, con objeto de observar inmediatamente el resultado.

Sintaxis:

tee [-a] archivo(s)

Esto se consigue con el operador *tee*, que podría emplearse de la siguiente forma:

ls / tee file

la salida de *ls* se bifurca hacia la Terminal y hacia *file*.

la opción *-a*: la salida de este comando se añadiera al final del fichero

ls / tee -a file

ejemplo:

ls -l | tee fichero | wc

orden find

La orden find nos localiza un archivo en un directorio, pero es algo más potente, es de los comandos más potentes y difíciles de utilizar.

Consta de tres partes:

- donde buscar
- que buscar
- que hacer cuando lo encuentre.

Si conoce el nombre del archivo pero se ignora el directorio donde está :

Find **-name** nombre_del_archivo **-print**

La sintaxis de find es find [opciones]

-name archivo: archivo puede utilizarse un comodín o el nombre de un archivo o caracteres especiales

-links n: selecciona n o más enlaces.

-size n : compara todos los archivos de n bloques de tamaño (bloques de 512 bytes)

-atime n: selecciona cualquier archivo que se haya accedido en n días.

-print: imprime el nombre y la localización de los archivos seleccionados.

Ejemplo: find **-size 100 atime 2**

Encuentra los archivos de mínimo 100 bloques que le último acceso hay sido 2 días.

Las opciones de find pueden agruparse y combinarse para limitar los criterios de búsqueda.

() los paréntesis pueden ser utilizados para agrupar selecciones. Debido a que son específicos del shell deben ser precedidos \< \)

-o OR sustituye al AND

! operador Not: niega la expresión

Ejemplos:

```
find .-name carta -print
```

busca en el directorio actual el archivo carta y lo muestra por pantalla.

```
Find .-name "carta*" -print
```

Busca todos los archivos que empiecen por carta y siga cualquier carácter.

```
find . ! \(-name "carta*" -o -name "*juan" \) -print
```

Busca los archivos que no comiencen por carta ni terminen en juan.

-exec comando el comando se ejecuta por cada archivo comparado. Se utiliza la notación {} para indicar donde debe aparecer el nombre del archivo en el comando ejecutado. El comando debe determinarse por una barra inclinada inversa seguida de punto y coma por ejemplo ls -d {}; este ejemplo el comando ls ejecuta con el argumento -d y cada archivo se pasa a ls en lugar donde se encuentra originariamente.

Ejemplo:

```
Find .-name "carta*" -exec ls -l {};
```

Se visualiza todos los archivos del directorio inicial con el nombre carta y se listan en su localización original.