Filtros

Contenido

Se da el nombre de *comandos de filtro* o *filtros* a un grupo de comandos que reciben un flujo de datos en su entrada, realizan una transformación y escriben los datos transformados en la salida. Además de los que veremos aquí, los filtros incluyen comandos vistos anteriormente tales como `head, tail, wc, y cut`.

Comandos de filtro

tr

El comando `tr` translitera los caracteres de la entrada: sustituye unos caracteres por otros. La sustitución es caracter a caracter. Entre sus opciones se encuentran `-d` para borrar caracteres; `-c` para sustituir complemento de los caracteres indicados, es decir, que no sean ésos; y `-s` para comprimir caracteres repetidos en uno solo.

Los siguientes comandos crean un archivo con los días de la semana separados por varias líneas en blanco. El ultimo comando suprime las lineas en blanco de más.

```
echo -e "lunes\n\nmartes\n\n\nmiércoles\n\njueves\n" > dias1.txt
echo -e "viernes\nsábado\n\n\ndomingo\n\n\n" >> dias1.txt
cat dias1.txt
cat dias1.txt | tr -s "\n*" > dias.txt
cat dias.txt
```

Se convirtieron los caracteres nueva línea seguidos en una solo, eliminando renglones en blanco.

```
cat dias.txt | tr a-z A-Z
```

convierte todo a mayúsculas.

```
cat dias.txt | tr -d aeiou
```

borra todas las vocales del archivo `dias.txt`.

```
echo "Aa;Bb?:+98"
echo "Aa;Bb?:+98" | tr -c "[a-zA-Z0-9]" "_"
```
transforma todos los caracteres que no sean letras o números en subrayas.

```
echo "Aa;Bb?:+98" | tr -cs "[a-zA-Z0-9]" "[\n*]"
```
transforma todos los caracteres que no sean letras o números en nueva línea, y comprime los caracteres nueva línea repetidos en uno sola; deja cada palabra sola en un renglón.

```
ls -l /usr
ls -l /usr | tr -s " " | cut -f3,4
```
comprime los blancos en la salida para poder cortar campos.

sort

El comando `sort` realiza una ordenación de los datos recibidos. La comparación para la ordenación puede ser por caracteres o por valor numérico. La ordenación por caracteres es la más parecida a la ordenación alfabética; sigue el orden del juego de caracteres según la codificación de caracteres usada en el sistema (en general UTF-8). El siguiente comando muestra la ordenación de caracteres alfabéticos:

```
echo "a b c ñ n m l e i h q d f á é í ó ú u o g j k p r s t v w x y z" | tr " " "\n" | sort | tr "\n" " "
```

En este ordenamiento, las vocales con tilde aparecen luego de las mismas vocales sin tilde; la ñ aparece después de la n. La conversión entre espacio y nueva línea es necesaria, porque `sort` actúa sobre líneas, ordena líneas de texto.

Para obtener una ordenación numérica, es preciso indicar la opción `-n`; en la ordenación alfabética, 101 queda antes de 21, puesto que el caracter 1 está antes del caracter 2. En los siguientes ejemplos puede compararse la ordenación alfabética (sin `-n`) con la ordenación numérica (con `-n`), y con la ordenación numérica inversa (con `-nr`):

```
echo 101 21 2 1 11 | tr " " "\n" | sort | tr "\n" " "; echo
echo 101 21 2 1 11 | tr " " "\n" | sort -n | tr "\n" " "; echo
echo 101 21 2 1 11 | tr " " "\n" | sort -nr | tr "\n" " "; echo
```

Si no se indican campos de ordenación, la comparación se hace sobre toda la línea. Si se indican campos, la comparación se hace considerando la cadena de caracteres iniciada en el primer caracter del primer campo hasta el último caracter del último campo indicado.

```
ls -l /etc | grep ^- | tr -s " " | sort -t" " -k9
```
ordena los archivos (solo archivos, no directorios) del directorios `/etc` por su nombre (campo 9, separador de campos " ", ordenación alfabética). Es necesario comprimir los espacios para la detección correcta de los campos (solo un espacio entre campos).

```
ls -l /etc | grep ^- | tr -s " " | sort -t" " -n -k5
```
ordena los archivos (solo archivos, no directorios) del directorios `/etc` por su tamaño (campo 5, ordenación numérica, separador de campos " ").

Es posible también indicar también posiciones de caracteres en los campos para definir el intervalo de ordenación (desde qué caracter de qué campo hasta qué caracter de qué campo):

`-k` *campo1*`[.`*posición1*`][,`*campo2*`[.posición`*2*`]]`
La secuencia de caracteres para la ordenación empieza en la *posicion1* del *campo1* y termina en la *posición2* del *campo2*.

Ejemplos:

`-k1,3`	campos 1 y 3, desde el caracter 1 del campo 1 hasta el último caracter del campo 3
`-k3.3,3.5`	desde el caracter 3 del campo 3 hasta el caracter 5 del mismo campo 3
`-k3.3,5`	desde el caracter 3 del campo 3 hasta el último caracter del campo 5
`-k3.3,5.6`	desde el caracter 3 del campo 3 hasta el caracter 6 del campo 5
`-k3,5.6`	desde el primer caracter del campo 3 hasta el caracter 6 del campo 5

```
ls -l /bin | tr -s " " | sort -n -k5
```
ordena por tamaño: campo 5, en orden numérico; no se indica separador, toma el espacio como separador.

```
ls -l /bin | tr -s " " | sort -nr -k5
```
ordena por tamaño: campo 5, en orden numérico descendente, el mayor primero.

```
ls -l /bin | tr -s " " | sort -k9
```
ordena alfabéticamente por nombre de archivo, campo 9.

`find`

El comando `find` explora una rama de directorios buscando archivos que cumplan determinados criterios. Permite criterios de búsqueda tales como:

- el nombre contiene cierta cadena de caracteres o aparea con algún patrón.
- son enlaces a ciertos archivos.
- fueron usados por última vez en un cierto período de tiempo.
- tienen un tamaño comprendido dentro de cierto intervalo.
- son de cierto tipo (regular, directorio, enlace simbólico, etc.).
- pertenecen a cierto usuario o grupo.
- tienen ciertos permisos de acceso.
- contienen texto que aparea con cierto patrón.

Una vez ubicados los archivos, `find` puede realizar diversas acciones sobre ellos:

- ver o editar.
- guardar sus nombres en otro archivo.
- eliminar o cambiar de nombre los archivos.
- cambiar sus permisos de acceso.
- ejecutar otras acciones sobre ellos.

```
find /var -name *.log -print
```
busca en el directorio `/var` los archivos terminados en `.log`, imprime sus nombres en la salida.

```
find /var/log -size +100k -print
```
busca archivos mayores de 100KB. En los argumentos numéricos, +N es mayor que N, -N es menor que N, N es exactamente igual a N.

```
find /var/log -size +100k -exec ls -l {} \; | less
```
ebusca archivos mayores de 100KB, ejecuta sobre ellos el comando `ls -l`. Los símbolos `{}` indican los archivos encontrados; el símbolo `\;` indica el fin del comando ejecutado, es el punto y coma escapado con la barra inversa.

```
find /var/spool/mail -atime +30 -print
```
busca archivos no accedidos hace más de 30 días. La opción -atime se refiere a tiempo transcurrido desde última lectura, -mtime desde última modificación de estado o permisos, -ctime de contenido.

```
touch /tmp/vacio1 /tmp/vacio2
mkdir /tmp/vaciodir; touch /tmp/vaciodir/vacio3
find /tmp -empty -exec ls -l {} \;
```
busca archivos vacíos y los lista.
```
find /tmp -empty -exec ls -l {} \; 2>/dev/null
```
igual, redirecciona error estándar para no mostrar mensajes de error por directorios inaccesibles.
```
find /tmp -name vacio* -exec rm -r {} \; 2>/dev/null
```
busca archivos y directorios cuyo nombre empieza con "vacio" y los borra.

```
find /home -nouser -ls
```
busca archivos en los cuales en lugar del nombre de usuario dueño aparece un número (UID). Esta situación se da cuando la cuenta de usuario ha sido borrada pero han permanecido los archivos creados por ese usuario.

grep

El comando `grep` (Global Regular Expression and Print) permite buscar las líneas que contienen una cadena de caracteres especificada mediante una expresión regular. Lee la entrada estándar o una lista de archivos y muestra en la salida sólo aquellas líneas que contienen la expresión indicada. Su sintaxis es:
```
grep [opciones] patrón archivos
```
donde el patrón a buscar es una expresión regular.

Crear un archivo con los días de la semana, uno por línea; llamarle `dias.txt`:
```
echo -e "lunes\nmartes\nmiércoles\njueves" > dias.txt
echo -e "viernes\nsábado\ndomingo" >> dias.txt
cat dias.txt
```

```
grep martes dias.txt
grep nes dias.txt
grep es dias.txt
grep m[a-z]*es dias.txt
grep m[[:alpha:]]*es dias.txt
grep [lt]es dias.txt
```
muestran las líneas del archivo `dias.txt` que contienen las cadenas indicadas como expresiones regulares.

```
ls -l /usr | grep '^d'
```
lista sólo los subdirectorios del directorio `/usr` (la línea empieza con "d").

```
ls -l / | grep '.......rw'
```
lista sólo los archivos que otros pueden leer y escribir en el directorio principal.

```
grep '^[^:]*:x:10[0-9]:' /etc/passwd
```
busca usuarios con UID entre 100 y 109: caracteres al principio de línea que no sean ":", luego ":x:", luego "10" seguido de los caracteres 0 a 9.

Ninguna expresión regular de `grep` aparea con un caracter nueva línea; las expresiones se aplican individualmente a cada línea.

Entre las opciones de `grep` se cuentan `-i` para evitar distinguir mayúsculas de minúsculas, `-n` para mostrar el número de línea y `-v` para buscar líneas que no contengan la expresión regular indicada.

El comando `grep` acepta distintos tipos de expresiones regulares (ER):

- cadenas fijas de caracteres separadas por nueva línea.
- ER básicas (BRE, Basic Regular Expressions).
- ER extendidas (ERE, Extended Regular Expresions).
- ER compatibles con Perl (PCRE, Perl Compatible Regular Expressions).

El tipo de expresión regular a interpretar se indica a través de opciones al comando:

`-E --extended-regex`	ERE (ER extendida)
`-F --fixed-strings`	cadenas fijas de caracteres separadas por nueva línea
`-G --basic-regexp`	BRE (ER básica)
`-P --perl-gregexp`	PCRE (RE compatible con Perl)

Las características de las expresiones regulares básicas y extendidas pueden verse en el capítulo Expresiones Regulares de este mismo curso.

Las expresiones a buscar pueden leerse desde un archivo externo:

```
  echo -e "tes\njue" > buscar.grep
  grep -f buscar.grep dias.txt
```

equivale a

```
     grep "tes\|jue" dias.txt    # ER básica
     grep -E "tes|jue" dias.txt  # ER extendida
```

En las expresiones regulares básicas el caracter `|` debe escaparse como `\|` para ser interpretado en su significado especial de alternativa.

uniq

El comando `uniq` excluye renglones consecutivos repetidos. La opción `-c` escribe el número de ocurrencias al comienzo del renglón.

El siguiente ejemplo muestra las 5 palabras más frecuentes en el conjunto de archivos:

```
  cat historia.txt | tr -sc [:alpha:]ááéíóúñ '\012' | sort | uniq
-c | sort -nr | head -20
```

Este comando realiza las siguientes acciones:

`cat`	lista el archivo de texto `historia.txt`, disponible en la página de este curso.
`tr`	convierte los caracteres no alfabéticos, incluidas vocales con tilde y la ñ, en caracteres nueva línea. Esto deja una palabra por línea.
`sort`	ordena alfabéticamente las líneas (palabras).
`uniq`	excluye renglones repetidos y cuenta ocurrencias.
`sort -nr`	ordena numéricamente en orden inverso, i.e. mayores primero.
`head`	selecciona los primeros 20 renglones, las palabras más usadas.

dd

El comando `dd` es un convertidor de datos: copia un archivo realizando diversos tipos de conversión. Aquí la palabra "archivo" designa el concepto amplio de Unix/Linux; puede tratarse de un archivo en disco, la entrada o salida estándar, un flujo de entrada o salida desde o hacia un dispositivo. Puede convertir diversos tipos de codificación de caracteres, cambiar tamaño de bloques, transferir datos entre archivos, particiones, discos y volúmenes, respaldar áreas de disco, realizar diversas conversiones de caracteres, generar datos aleatorios, convertir formatos (por ejemplo crear imágenes ISO para CDs). Se usa para variedad de tareas de conversión y

mantenimiento en sistemas operativos, así como para realizar transferencias entre sistemas operativos distintos o datos en bruto sin formato.

```
dd if=historia.txt conv=ucase,notrunc
```

convierte a mayúsculas el texto del archivo `historia.txt`.

El siguiente ejemplo muestra el uso de `dd` para grabar en un pendrive la imagen ISO de una versión de Linux descargada de Internet:

```
dd bs=1M if=linuxmint-18.2-cinnamon-64bit.iso of=/dev/sdb
```

graba la imagen ISO descargada en un dispositivo (e.g. un pendrive) insertado en un puerto USB y reconocido por el sistema como el dispositivo `/dev/sdb`.

Referencias, lecturas complementarias

- Páginas man: `dd`, `find`, `grep`, `sort`, `tr`, `uniq`.
- Wikipedia: `dd`.
- Shotts, William. *The Linux Command Line*. Third Internet Edition, 2016. Chapter 20, Text Processing. Disponible online: https://razaoinfo.dl.sourceforge.net/project/linuxcommand/TLCL/16.07/TLCL-16.07.pdf

---