

Miembros del grupo de estudios	1
Materiales de estudio	1
Algunos criterios.....	2
.....	2
4.1.- Operadores de redirección	4
Tabla 1.....	4
Script N° 1 (Alejandro)	5
Script N° 2.....	7
archivos de configuración de Ubuntu	8
sort	12
wc	12
uniq.....	12
Desafío N° 1 (María).....	14
Autoevaluación con desafío de María:	15
sort, wc, tee y uniq	16
Ubuntu Forums	19
Tuberías.....	21
Redirección.....	23
Redirección de escritura.....	23
Redirección de lectura	24
Soluciones al desafío N° 1	25
María.....	25
Alejandro	25
Jorge.....	26
Asterix.....	27

Miembros del grupo de estudios

María, Fabricio, Diego, Sergio --> **Argentina**

Jorge, Rodrigo, --> **Chile**

Alejandro --> **Brasil**

Con la reciente incorporación de: Sergio (asterix) --> **Chile**



Materiales de estudio

(Si tienen alguno más para recomendar, lo agregan)

[Apuntes_bash_0.pdf](#) / [Apuntes_bash_1.pdf](#) / [bash.pdf](#) [Comandos.pdf](#)

**** PRIMERA SEMANA "COMODINES" **** **NUEVO!**

[Ejemplos de scripts - Ubuntu Forum](#)

[Un comando cada día - Ubuntu Forum](#)

[Bash scripting de supervivencia](#)

[Taller de programación shell](#)

[Shell Scripting](#)






[Google Docs](#)

***** Redireccionamiento de Entrada-Salida ***** **NUEVO!**

***** Redireccionamiento y tuberías ***** **NUEVO!**

Algunos criterios

- Podemos reutilizar los iconos para señalar algunas cuestiones (copiar y pegar).

	Referencia al material de estudio
	Desafío
	Script
	Atención!
	Mi Reino por un caballo!!!

	Una duda...
	Premio "Rex Tux Scripting"
	Consejo

- Delimitar las intervenciones con barras horizontales.
- No usar comentarios (en la publicación web no aparecen)
- Firmar con el nombre resaltado con color.

María, Fabricio, Diego, Jorge, Rodrigo, Alejandro, Sergio, Sergio (asterix)

- Fuente Courier New para los scripts y comandos. Aplicar sangrado de párrafo (ahora está en Formato-->Estilo de Párrafo).
- Si no visualizan correctamente las fuentes verdana y courier, instalen los paquetes:

```
sudo aptitude install msttcorefonts ttf-mscorefonts-installer
```

- Poner onda con el formato para facilitar la legibilidad del documento y el trabajo de publicación posterior.



Suerte!

Alejandro



4.1.- Operadores de redirección

Tabla 1

¿Alguien se anima a completar la información en esta tabla (resumiendo)?

Operador o comando	Función
cat	Permiten ver uno o más archivos o parte de un archivo Ej: <code>\$ cat script.sh</code> muestra el texto de script.sh - Diego
CTRL+D	Indica el final de un stream Fabricio
<	Operador de redireccionamiento de entrada estándar - Jorval
>	Operador de redirección de salida. Permite cambiar la salida estándar de un comando - Alejandro
2>	Operador de redirección de errores estándar. permite cambiar la salida de errores estándar - Diego
>>	Redireccionamiento de anexión - Jorval
2>>	Permite redireccionar los errores a un fichero y no sobrescribirlo, sino añadir el contenido al final - Diego
/dev/null	Fichero de errores estándar del sistema - Diego
	Operador de canalización. Se usa para enviar salidas de un comando a la entrada de otro - Jorval También conocido como "pipeline" - Alejandro
more	Comando de paginación - Se emplea para leer archivos de texto interactivamente - Jorval Por ej: Si tengo una carpeta con 1000 mp3, hago " <code>\$ ls Música more</code> " y voy a poder leer página por página - Alejandro
less	Comando de paginación - Se emplea para leer archivos de texto interactivamente - Jorval
cut	Permite buscar y/o seleccionar columnas o campos dentro de un archivo - Jorval
sort	Para ordenar líneas de texto a partir de varios criterios - Jorval
&	Control de tareas - Iniciar y ejecutar un programa en segundo plano - Jorval
&&	Permite ejecutar un comando de manera condicional - Jorval
;	Para introducir varios comandos en una sola línea de comandos - Jorval

Alejandro

Muchachos y Muchacha! Hace dos días que estoy pensando cómo se podría vincular esto con lo que hemos hecho hasta ahora y no le estoy encontrando la vuelta. Estoy bloqueado.

Alguien ha hecho alguna prueba, o se la ha ocurrido algo? Una propuesta de script? Un desafío? Será bienvenida cualquier idea!!! A ver si conseguimos arrancar la semana!!! Mientras, sigo pensando...

Muy buena la info de la tabla, gente!!!

Alejandro

Alejandro, entiendo que lo que deseas es hacer un solo documento con los documentos definitivos de cada semana. Si es así, te contaré como lo hago yo, que estoy traduciendo un libro con la ayuda de Google Docs y Google Traductor. Para cada hoja abro un documento nuevo y cuando ya los tengo depurados y he completado un capítulo, aproximadamente unas 20 hojas, abro un documento de office y ahí los pego y los vuelvo a revisar y desde ese documento lo subo a un blog. No sé si te sirve y me di a entender. (Supongo que esta intervención la pasarás a "Comentarios, sugerencias y otras yerbas...")

Jorval

No, **Jorge**, la verdad es que me refería a que no se me ocurre cómo vincular el tema de los "operadores de redirección" y los comandos nuevos (cat, less, more, cut y sort), que aparecen en este capítulo, con el tema de los "comodines" que trabajamos la semana pasada. Estaba pensando qué script se podría escribir con lo que sabemos hasta ahora; o qué desafío se podría plantear; y no se me está ocurriendo nada.

Estoy necesitando una ayudita para dar el puntapié inicial a la semana. Alguien tiene alguna idea para proponernos?

Por lo pronto, voy haciendo, de a poco, una breve recopilación de utilidades prácticas para estos operadores que se me van ocurriendo:

- Si quiero listar (no sé para qué) los archivos que tengo en determinados directorios e ir guardándolos en un archivo de texto:



Script N° 1 (Alejandro)

```
#!/bin/bash

# me ubico en mi home:

cd ~

# creo el fichero "recopiladir.txt" con un texto en su interior
(usando ">"):
```

```
echo "contenido de mi carpeta personal" > recopiladir.txt

# voy listando las carpetas que quiero y voy redireccionando la
salida del comando ls al interior del fichero (usando ">>").
Cada una con un texto que explica de qué se trata:

ls >> recopiladir.txt
echo "contenido de mi Escritorio" >> recopiladir.txt
ls Desktop >> recopiladir.txt
echo "contenido de mi carpeta de fotos" >> recopiladir.txt
ls Imagens >> recopiladir.txt

# finalmente, visualizo el contenido del fichero:

cat recopiladir.txt
```

Cuando estaba armando ese script, se me ocurrió probar esto:

```
ls ~ Desktop Imagens Documentos > recopiladir.txt
cat recopiladir.txt
```

Y funcionó!!!

Se puede ir indicando qué carpetas listar en el comando "ls", una al lado de la otra, o mejor dicho, sucesivamente; lo cual haría innecesario el script.



Una pregunta... Cómo me explicarían la diferencia entre el operador de redireccionamiento de salida ">" y el operador de canalización (pipeline) "|". Por qué usarían uno u otro?

Y otra más... Cómo me explicarían la diferencia entre "more" y "less"? Cuándo usarían uno y cuándo el otro?

Alejandro

Muy bueno !!

me gustó este script. con respecto a las preguntas.

1º Pregunta: Creo que usamos ">" porque queremos redireccionar la salida del comando hacia determinado archivo y no evitar la salida del comando como es el caso "|".

La **segunda** no la se

En cuanto a la forma de seguir, sería tratar de dar un ejemplo, de un script, de otro autor y explicar lo que hace. Luego si se puede agregar funcionalidades o readaptarlo. Este es un buen ejemplo de redirecciones y pipes

Aquí va un ejemplo, si no les gusta edito y listo.



Script N° 2

```
#!/bin/bash

# Lista de Paquetes
# se forma el nombre del archivo
DIA=`date +%d`
MES=`date +%m`
AnO=`date +%Y`
time=`date +%R`

ARCHIVO=paquetes.$AnO-$MES-$DIA.$time.lst
# lista y ordena los paquetes instalados
dpkg --get-selections | grep -v deinstall | sort > ~/Documentos/
Respaldo/$ARCHIVO
```

Consejo: cortar y pegar en un editor de texto para ver la estructura en colores (fuente scripts "útiles para la cartera de la dama y el bolsillo del caballero" - ubuntu-ar)

La idea es ir viendo línea por línea e interpretar.



Gran duda. Estoy tratando de trabajar con el comando **less**. Cuando trato de abrir un archivo .odt e incluso uno .txt pero escrito con el procesador de texto de OO me dice "file.txt" may be a binary file. See it anyway? " le pongo yes y me salen caracteres ilegibles. Cuando empleo el comando con un archivo que generé con el editor de texto gedit ahí funciona sin problemas. ¿Estos comandos sólo funcionan con textos generados con "editores de texto"? o bien tengo que hacer algo más para ver mis archivos .odt Gracias.

Jorval

Jorge, creería que los archivos de **OpenOffice** tienen algún tipo de codificación (donde se guarda la información del formato) que hace que el comando no los interprete correctamente. Quizá más adelante vemos como cambiar esa codificación, si es que es posible hacer eso.

En cuanto al **script**, me parece buena idea analizar scripts. Lo que yo diría: no analizar las líneas que contienen elementos nuevos o demasiado complejos y concentrarnos en las que son analizables con las herramientas que tenemos. Por ejemplo, comandos como **grep** y **sed**, variables o bucles y rutinas condicionales, todavía no los hemos visto y son bien complejos.

Nos vendría **apuntar los cañones** a los "operadores de redirección" y a comandos como "less", "more", etc., para descubrir lo más posible de su funcionamiento.

Y hay líneas con "echo" y "rm" que presentan algunos parámetros que nos vendría bien averiguar cómo se comportan, qué hacen y para qué:

```
echo -laae

echo -e

rm -rf

rm -rf /home/*/.local/share/Trash/*/** &> /dev/null

rm -rf /root/.local/share/Trash/*/** &> /dev/null
```

archivos de configuración de Ubuntu

Algo que podemos investigar también, es cuáles son los **archivos de configuración de Ubuntu** y usar esos comandos para leer información útil en ellos y mandarlos a un archivo, por ejemplo (estos los encontré [aquí](#) y [aquí](#), seguro que hay muchos más):

```
/boot/grub/menu.lst

/etc/samba/smb.conf

/etc/X11/xorg.conf

/usr/bin/compiz

/etc/network/interfaces

/etc/resolv.conf

/etc/passwd

/etc/fstab
```

Posibilidades:

- Listar los ficheros que hay en cada uno de esos directorios y mandar la lista a un archivo de texto para poder imprimir y tener una especie de índice.
- Visualizar en la terminal el contenido de esos ficheros (cat).
- Visualizar sólo una parte de esos ficheros cuando son muy extensos o sólo interesa algo en particular.
- Armar un fichero con toda la información configurable del sistema.
- etc.

Por ejemplo (no sé si esto tendría alguna utilidad, pero como ejercicio lógico, sirve):

Si hago:

```
ls /boot/grub
```

Obtengo la lista de todos los archivos de ese directorio. Veo que hay ficheros con extensión ".mod" y ".lst". Si quiero ver que hay adentro de los .lst, hago:

Grupo de estudios: Programación Bash Script

Segunda semana: del 15 al 21 de octubre de 2009

```
cat /boot/grub/*.lst
```

Si quiero guardar esa información:

```
cat /boot/grub/*.lst > fichero.txt
```

Si quiero agregar más info al fichero:

```
cat /etc/fstab /etc/passwd >> fichero.txt
```

Alejandro

Coincido con lo de **Alejandro**. El script es algo complejo, pero a groso modo nos podemos dar cuenta de como funciona. Por lo menos sirve como disparador de ideas. Pienso un poco esta nueva consigna y respondo.

Jorge: no tendrá que ver el tipo de fuente que usas en OO que no pueden decodificar los editores de textos mas básicos. Digo como posibilidad habría que chequear eso. yo para los scripts uso "kate" o "nano" en kde o "gdit" en gnome.

Diego

A la pregunta de **Alejandro** relacionada con la diferencia entre **redireccionar y canalizar**. Les copio la dirección de un taller de Unix que explica claramente la diferencia. Vale la pena leerla completa. [Redireccionar - canalizar](#)

Jorval

(Gracias **Jorge**. Muy, pero muy recomendable su lectura. **Alejandro**)

Edité el script, en su lugar coloqué otro muy simple que tiene un buen ejemplo de redirecciones y pipes. Es también de ubuntu-ar y lo que hace es listar todos los paquetes instalados y crear un fichero de respaldo con toda la lista de paquetes.

```
dpkg --get-selections | grep -v deinstall | sort > ~/Documentos/Respaldo/  
$ARCHIVO
```

esta es la línea más importante. las tuberías o pipes nos permiten pasar la salida de un comando a otro.

<comando1> | <comando2>. Con esto, la salida de *comando1* será la entrada de *comando2*.

lo que hace *grep* (*grep*, *egrep*, *fgrep*) - muestran líneas que concuerdan con un patrón. O en otras palabras.

grep es un *parseador* de expresiones regulares, es decir, le damos un patrón y un fichero (o introducimos lo que sea por consola, o lo pasamos con un *pipe*) y de ese texto nos devuelve sólo loque coincide con el patrón. Este es el funcionamiento básico de **grep** pero es muchísimo más potente.

luego tenemos **sort** que me gustaría que alguien me diga que hace. :D

Con respecto a la pregunta de **Alejandro**

echo Muestra una línea de texto.

rm Borra ficheros o directorios.

rm -rf r borra recursivamente arboles de directorios. f no pide confirmación. no escribe mensaje de diagnóstico, no produce estado de salida de error si los ficheros no existen.

Diego



Comencé a estudiar este apartado. Copio mi terminal para plantear algunas dudas:

```
jorval@jorval-laptop:~$ pwd
/home/jorval
jorval@jorval-laptop:~$ cat
Esta es una práctica de "Redirecciones y pipes" para nuestro
taller de Bash-Scripts (No olvidar terminar con CTRL+D)
Esta es una práctica de "Redirecciones y pipes" para nuestro
taller de Bash-Scripts (No olvidar terminar con CTRL+D)
jorval@jorval-laptop:~$
jorval@jorval-laptop:~$ cat <file.txt
Esta es una prueba si es verdad
que puedo escribir desde aquí
a un archivo
jorval@jorval-laptop:~$
jorval@jorval-laptop:~$ cat file.tx
cat: file.tx: No existe el fichero ó directorio
jorval@jorval-laptop:~$ cat ./file.txt
Esta es una prueba si es verdad
que puedo escribir desde aquí
a un archivo
jorval@jorval-laptop:~$
jorval@jorval-laptop:~$
```

- 1.- Verifico donde estoy posesionado: en mi home
- 2.- Practico escribir en el terminal empleando el comando CAT. Funciona perfecto, excepto que al darle al CTRL+D , al final me copia también mi dirección(jorval@jorval-laptop:~\$) pero le doy enter y paso a una nueva línea para introducir comandos
- 3.- Pruebo ver en la terminal lo escrito en mi archivo file.txt y funciona perfecto el redireccionamiento < pero nuevamente al final copia mi dirección completa ¿por qué sucederá esto? Le doy enter y paso a la siguiente línea.
- 4.- Para verificar lo que dice el manual, con el comando CAT también podría leer el archivo sin emplear redireccionamiento. Claro, para eso es CAT, pero al probarlo sucede que si coloco solo file.txt no encuentra el archivo y tengo que colocarle ./file.txt y ahí sí funciona ¿Cual es la diferencia? las dos son direcciones relativas, ha y vuelve a copiar al final mi dirección y tengo que darle enter para pasar a una nueva línea. Gracias.

Jorval

Una aclaración referente a **redirigir la salida de errores por medio de los números**. Copio de un Manual: "Las entradas y salidas estándar poseen números de archivo asignados en el shell. La entrada estándar utiliza el número 0, mientras que la salida estándar emplea el número 1. Otro tipo de salida es error estándar, y su número es 2".

Jorval

Y con canalizaciones termino la lección. Les copio mi terminal porque es entretenido:

```
jorval@jorval-laptop:~$ cat gedit1.txt | wc -l >Numerar.txt
jorval@jorval-laptop:~$ find ~ | wc -l >>Numerar.txt
jorval@jorval-laptop:~$ find ~ | sort | uniq -d >Duplicados.txt
jorval@jorval-laptop:~$
```

- 1.- Cuenta las líneas de un archivo y genera un informe.
- 2.- Cuenta el número de archivos que hay en mi home y genera un informe.
- 3.- Genera un informe de los archivos duplicados (que tienen el mismo nombre)

Jorval

Jorge, una observación. Tal como está escrito en tu ejemplo "`cat <file.txt`" no va a funcionar, porque no está redireccionando la salida de "`cat`" (es decir, lo que tú escribes en la terminal) hacia el fichero "`file.txt`". Lo que está haciendo es justamente lo contrario, está enviando el contenido de un fichero "`file.txt`" a la salida estándar de `cat` (a la pantalla). Pero como el fichero no fue creado, no lo encuentra.

Por ejemplo:

```
# Creo un fichero añadiéndole al mismo tiempo un texto

$ cat >file.txt
Hola, estoy probando!    (<-- escribo eso y CTRL+D)

# Leo lo que está dentro del fichero

$ cat <file.txt
Hola, estoy probando!
```

comando 1 Salida estándar > Entrada estándar **comando 2**

comando 1 Entrada estándar < Salida estándar **comando 2**

Encontré esta información que puede ayudar a entender los ejemplos presentados por **Jorge**:

sort

Por defecto ordena las líneas de un archivo, con la opción **-m** mezcla dos archivos ordenados y con la opción **-c** verifica que un archivo esté ordenado. Con **-b** descarta los espacios en blanco al principio. Al ordenar puede considerar las líneas completas, considerarlas números enteros (opción **-n**) o flotantes (opción **-g**), considerarlas meses (opción **-M**) o dividirlos en campos y emplear algunos campos como llaves de ordenamiento (opción **-k** para especificar llaves y **-t** para cambiar separador de campos). En las comparaciones puede ignorar diferencias entre mayúsculas y minúsculas con la opción **-f**, puede ordenar de mayor a menor con la opción **-r** ---por defecto ordena de menor a mayor--- y puede eliminar líneas repetidas con la opción **-u**.

Por ejemplo, para ordenar por líneas el archivo nombres.txt:

```
sort nombres.txt
```

si cada línea del archivo nombres.txt tiene el apellido y después el nombre de una persona separados por espacios, puede ordenarse por nombre con:

```
sort -k 2 nombres.txt
```

wc

Cuenta cantidad de palabras, líneas y caracteres en uno o más archivos. Por defecto presenta los tres datos por cada archivo que reciba y después presenta las sumas. Con la opción **-w** presenta la cuenta de palabras, con **-l** la cuenta de líneas, con **-c** la cuenta de letras y con **-L** la longitud de la línea más larga.

uniq

Descarta todas las líneas sucesivas idénticas, menos una de ENTRADA (o entrada estándar), escribiendo en SALIDA (o en la salida estándar). **-c** precede a las líneas con el número de ocurrencias, **-d** muestra sólo las líneas duplicadas, **-D** muestra todas las líneas duplicadas, método-de-delimitación={none(predeterminado),prepend,separate} La delimitación se hace con líneas en blanco, **-f** no compara los N primeros campos, **-i** ignora diferencias entre mayúsculas y minúsculas, **-s** no compara los N primeros caracteres, **-u** muestra líneas repetidas una sola vez, **-z** termina las líneas con un byte a cero en lugar de una nueva línea, **-w** sólo compara los primeros N caracteres de la línea.

Un campo es cada conjunto de caracteres separados por espacios.
Se pasan por alto los campos y después los caracteres.

Nota: 'uniq' no detecta líneas repetidas a menos que sean adyacentes.
Tal vez prefiera ordenar primero la entrada, o utilizar 'sort -u' sin 'uniq'.

Más información:

```
sort --help
```

```
wc --help
```

```
uniq --help
```

...

La parte en que decís que te copia tu dirección después de finalizar el stream (CTRL+D) no termino de entenderla muy bien. A mí, lo que hace es devolverme el prompt "alejandro@alejandro:~\$" y ahí mismo puedo ejecutar otro comando. Pero no me aparece duplicado como en tu ejemplo.

Alejandro

Pues a mi me pasaba lo mismo que a **Jorge**, es decir al final se me copiaba la dirección. Al parecer el error está en uno de los pasos

```
$cat >texto.txt
```

```
Esta es una prueba (Aquí le di un enter, y solo después del enter le dí el control D)
```

Ahora sí

```
$cat <texto.txt
```

```
Esta es una prueba (y me devuelve luego al prompt)
```

Otra cosa que me di cuenta es que si queremos redireccionar a un archivo con espacios (ej. archivo uno, archivo dos, etc), este debe ir entre comillas o de lo contrario devuelve un error

```
$cat >"archivo uno".txt
```

Una duda que me asaltó es que si se puede hacer un redireccionamiento de entrada y salida, hacia otro lugar del directorio en el que se está. Pues lo intenté, y si dió resultado

Asterix

Alejandro, creo que no es como tú lo entiendes. Tengo en mi home un archivo que se llama file.txt y deseo ver en mi terminal lo que dice ese archivo, entonces la salida de cat la redirijo < hacia la terminal, que es lo mismo que cat file.txt

Práctica para colocar la copia del terminal como lo hace Alejandro:

```
jorval@jorval-laptop:~$ cat file.txt
```

```
Esta es una prueba si es verdad  
que puedo escribir desde aquí
```

```
a un archivo
jorval@jorval-laptop:~$ cat file.txt
Esta es una prueba si es verdad
que puedo escribir desde aquí
a un archivo
jorval@jorval-laptop:~$ cat ./file.txt
Esta es una prueba si es verdad
que puedo escribir desde aquí
a un archivo
jorval@jorval-laptop:~$
jorval@jorval-laptop:~$
```

Ahora sí, había olvidado como hacerlo. En todo caso pueden ver que ahora `cat file.txt` me funciona igual que `cat ./file.txt`. Y continúa apareciendo mi dirección al final y eso que no le di CTRL+D. Bueno son los misterios de Unix, ya aparecerá lo que no hay que hacer.

Asterix, gracias, efectivamente cuando redireccionas hay que hacer como dices, darle enter primero y luego CTRL+D. Respecto a lo que te sucede con los nombres de los archivos es porque dejas un espacio, cosa muy desaconsejable, buscaré la explicación científica y la pondré, pero consejo, no dejes espacios en los nombres de los archivos.

Jorval



Desafío N° 1 (María)

Holas.

Encontré estos ejercicios. Los comparto para quien tenga ganas de resolverlos. Esta vez no pongo las soluciones.

Yo recién voy resolviendo la mitad de la lista.

1) Crear el directorio "comandosConsola" y dentro de él, crear varios archivos vacíos con nombres de colores utilizando el comando touch:

```
$ touch verde azul rojo amarillo violeta celeste blanco rosa lila anaranjado marron gris
```

2) Crear el subdirectorio "colores_dir" adentro de "comandosConsola" y mover todos los archivos de colores adentro de "colores_dir"

3) Crear un archivo llamado "listaColores" que contenga el listado del directorio

```
~/comandosConsola/colores_dir
(Uso obligatorio de: ls; ">" )
```

4) Crear un archivo llamado "binarios" que contenga el listado del directorio /bin.
(Uso obligatorio de: ls; ">")

5) Crear un archivo llamado "union" que contenga ambos listados (colores y binarios).
(Uso obligatorio de: cat; ">")

- 6) Ordenar alfabéticamente el listado "union" y guardar el resultado en un archivo "unionOrdenado".
(Uso obligatorio de: sort; "<"; ">")
- 7) Verificar que los datos en "unionOrdenado" sean correctos.
- 8) Crear un archivo llamado "datosv" con los siguientes datos personales dentro: Nombre, apellido y DNI.
(Uso obligatorio de: cat; ">")
- 9) Agregar a "datosv" una línea que indique el directorio actual.
(Uso obligatorio de: pwd; ">>")
- 10) Agregar a "datosv" un listado en formato largo del directorio /etc.
(Uso obligatorio de: ls; ">>")
- 11) Observar (por pantalla) el archivo "datosv" resultante a través del filtro more y verificar que los datos estén correctos.
(Uso obligatorio de: more; "<")

Suerte!

María

Muy bueno, **María!!!** vamos a ver qué podemos hacer...

Jorge, disculpas, había entendido todo al revés...

Sergio: Bienvenido!!!

Alejandro



Autoevaluación con desafío de María:

	1	2	3	4	5	6	7	8	9	10	11
María	X	X	X	X	X		X	X	X	X	
Jorge	X	X	X	X	X	X	X	X	X	X	X
Diego											
Sergio											
Asterix	X	X	X	X	X	X	X	X	X	X	X
Fabrizio											
Alejandro	X	X	X	X	X	X	X	X	X	X	X

María, ¿Te enviamos las soluciones para que vos las publiques una vez que todos las

hayamos resuelto? **Alejandro**

Tengo algunas dudas en la parte 2 y 6, pero las haré públicas una vez que demos por finalizado el ejercicio de María.

Saludos

Asterix

sort, wc, tee y uniq

Les dejo una prueba de los comandos que hice. No sirve para nada, el objetivo era simplemente probar algunas opciones de **sort**, **wc**, **tee** y **uniq**.

Incorporé el uso de **tee** porque su uso me parece una buena práctica para cuando estamos probando cosas nuevas y queremos ver los resultados en pantalla. Claro que en un script podemos no querer que el resultado de un proceso se visualice y en ese caso usaríamos los operadores de redirección comunes.

Usando la metáfora de las tuberías, **tee** es una T en mi tubería, divide el flujo, un chorro de agua va para un lado (un fichero por ejemplo) y el otro chorro de agua va para el otro lado (la pantalla). Se usa combinado con **pipelines** "|".

En el post de **Ubuntu Forum** ("un comando cada día") dice que hay que agregar la opción **-a** para que, si el archivo al que se redirige la salida no existe, lo cree. Pero yo he probado sin la opción y lo crea igual. Así que no sé...

```
# creo un directorio de prueba y lo convierto en el directorio
actual

mkdir pruebas
cd pruebas

# listo el contenido del directorio /bin y lo guardo en un
fichero

ls /bin | tee bin.lst

# compruebo que esté ordenado alfabéticamente de menor a mayor

sort -c bin.lst

# ordeno /bin de mayor a menor y guardo en un fichero

sort -r /bin | tee bin-r.lst

# compruebo que el nuevo fichero no está ordenado de menor a
mayor

sort -c bin-r.lst
```

```
# combino los dos listados

cat bin.lst bin-r.lst | tee binAll.lst

# cuento cuántas palabras, líneas y letras tiene el nuevo
fichero

wc -wlc binAll.lst

# busco líneas sucesivas idénticas

uniq -d binAll.lst

# listo sin líneas duplicadas (queda un listado capicúa ¿en
Chile existe esa palabra? Por ej: 4590954 es un número capicúa,
es simétrico)

uniq binAll.lst

# borro la carpeta pruebas para liberar espacio (rmdir borra
directorios vacíos únicamente, por eso no se podría usar en este
caso)

rm -r ~/pruebas
```

Ahí averigué cómo era la historia de la opción **-a**

```
~$ tee --help
Uso: tee [OPCIÓN]... [FICHERO]...
Copia la entrada estándar a cada FICHERO, y también a salida
estándar.

-a, --append           añade a los FICHEROs dados, no los
sobreescribe
```

Alejandro

20/10/09

Si quieren enviarme las soluciones no hay problema, pero me parece que ya se pueden publicar aquí.

No puedo resolver ni el nro. 6 ni el nro. 11 con los comandos de uso obligatorio que nos piden que usemos. Los resuelvo con otros comandos.

Alejandro:

```
sort -r /bin | tee bin-r.lst    (me tira error, me dice que
/bin es un directorio)
```

María

Tenés razón **María**, a mí también me da ese error. Como probé varias opciones, debo haber colocado una previa, porque juro que comprobé todo. Bueno, pasa en las mejores familias...

Debe ser:

```
sort -r bin.lst | tee bin-r.lst
```

Alejandro



Tengo una intriga con la resolución del ejercicio nro. 6 que puse mas arriba.

6) Ordenar alfabéticamente el listado "union" y guardar el resultado en un archivo "unionOrdenado". (Uso obligatorio de: sort; "<"; ">")

Jorval lo resolvió correctamente:

```
$ sort < union > unionOrdenado
```

¿Alguien entiende esa lógica del "<" y del ">" juntos?

María

María, lo que sucede es que se ordena el archivo "union" y es enviado a dos partes: a un nuevo archivo "unionOrdenado" y a tu terminal para que lo leas.

Jorval

Es exactamente lo mismo que hacer:

```
$ sort union >unionOrdenado
```

sort tiene una entrada estándar (stdin) que es el teclado o el archivo que se le indique como parámetro y una salida estándar (stdout) que es la terminal. De esa forma, si yo escribo:

```
$ sort union
```

toma de stdin (que es el archivo union) y devuelve por stdout (pantalla) los datos ordenados.

Al colocar el operador de redirección, en el primer caso (`sort <union`) sería una redundancia. En el segundo caso (`union >unionOrdenado`) estoy indicando que en lugar de mandar el resultado a `stdout` lo mande a un archivo.

Hagan la prueba de escribir únicamente **sort** en terminal y presionar enter; escriban una lista de palabras, presionando enter después de cada una y CTRL+D al finalizar. Ahí está el comando **sort** actuando en su estado puro.

(Una aplicación: tengo una lista de 40 y tantos alumnos en un papelito. Quiero organizar rápidamente esa lista para armar un registro en OpenOffice. Realizo los pasos antes reseñados, copio y pego. Ya está!)



Alejandro

Ubuntu Forums

Yo había entendido mal, pero eso dio pie a una serie de aclaraciones. Copio lo que surgió en **Ubuntu Forums**, que me parece interesante:

Faktorqm dice:

```
lspci | tee nombre_de_archivo
```

es el equivalente a hacer

```
lspci > nombre_de_archivo
```

```
lspci | tee -a nombre_de_archivo
```

es el equivalente a hacer

```
lspci >> nombre_de_archivo
```

si haces `tee` solo, te guarda la salida de lo que estas haciendo en el nombre de archivo que especifiques, si existe, lo sobrescribe, y si no existe lo crea.

si haces `tee -a`, te AGREGA la salida de lo que estas haciendo en el FINAL del archivo llamado nombre de archivo que especificaste, si existe, lo agrega, y si no existe lo crea.

Es una convención puramente literaria la de "si no existe lo crea", lo vas a

ver en todos los comandos y queda explicito en todas las ayudas que leas, ya sea man o --help.

Guillermolisi dice:

La diferencia funcional mas importante que existe entre el redireccionamiento usando "> o >>" y "|" (tee) es que si necesitamos interconectar (piping) dos procesos (la salida que produce uno para que funcione como datos de entrada del segundo) es este ultimo redireccionador el indicado.

Aqui deberiamos hablar de los tres canales de intercomunicacion que existen en los sistema *nix: Standard input, Standard output & Standard error.

Los redireccionadores "> y >>" modifican el flujo de datos entre un proceso y los tres canales. Tee intercomunica procesos sin modificar el flujo entre canales.

Corrijanme si me equivoque en algo.

Alejandro

```
$ sort < union > unionOrdenado
```

Cito lo de **Jorval**: "**María**, lo que sucede es que se ordena el archivo "union" y es enviado a dos partes: a un nuevo archivo "unionOrdenado" y a tu terminal para que lo leas. "

¿como es eso de que lo envía al terminal para que lo leas?. ¿Debiera aparecer en la pantalla?, si es así a mi no me sucede eso, solo ordena el contenido de los archivos en cuestión

Asterix

Asterix, respecto al ejercicio \$ sort <union>unionOrdenado, efectivamente ahora no me aparece en el terminal, pero juraría que cuando lo hice por primera vez, si me apareció, a lo mejor es porque ahora no hay nada que ordenar ni crear, porque ya está ordenado y creado el archivo, pero...

Jorval

Moví la otra conversación a nuestra página de comentarios. La seguimos allá.

Alejandro

Gracias a lo que expusieron **Alejandro** y **Jorval**, para mi ya quedó claro. Es lo mismo decir:

```
$ more union
```

que decir:

```
$ cat union | more
```

que decir:

```
$ more < union (aquí es redundante)
```

Por esto, todavía no le encuentro utilidad al "<", al menos en estos ejemplos.

¿Alguien encontró algún ejemplo del uso del "<" que no sea redundante?

Maria

Se me ocurre un ejemplo:

Armo un fichero que en su interior tiene una lista de directorios que me interesan. Ahora listo todos esos directorios haciendo:

```
ls <fichero
```

En ese caso redirijo la entrada estándar (stdin) del comando "ls" -que normalmente sería el directorio actual o un nombre de directorio- al contenido de un fichero.

Pero no, lo probé y no funciona... así que no sé... lo seguiré pensando.

Alejandro

Estimados estudiantes, buscando para María un ejemplo de la utilidad de < , con san Google encontré esta interesante página con la que estimo terminaremos graduados en "Redirecciones y pipes". Espero les sirva [clic aquí](#)

Jorval

Copio algunos conceptos interesantes del material propuesto por **Jorge**:

Tuberias

Podríamos representar cada programa como una **caja negra** que tiene una entrada y una salida que se pueden unir entre ellos.

```
Debido a que la entrada por defecto es el teclado y la salida  
por defecto  
es terminal, graficaremos cuando sea necesario ambos.
```

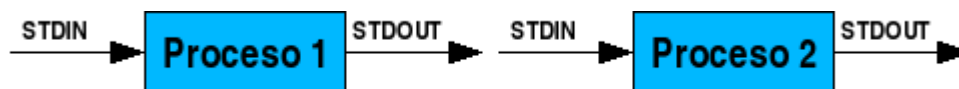
El ejemplo que utilizamos se encuentra esquematizado en la figura



Siendo la entrada estándar el teclado y la salida estándar el **terminal** o por simplicidad la **pantalla**.

Vamos a suponer un caso ficticio donde necesitamos todas las definiciones de cada palabra en un texto. Primero las ordenamos alfabéticamente, luego utilizamos un comando ficticio llamado **diccionario** que toma palabras de la entrada estándar y las reescribe junto a su significado en la salida estándar.

Su esquema se ve en la figura:



En este caso nombramos por separado las entradas y salidas estándares de los dos programas, pero la **unión** entre ambos programas se puede considerar como un sólo **tubo**. En ese tubo, el flujo está en un estado intermedio, donde está ordenado, pero no tiene las definiciones de diccionario. En la línea de comandos esto se escribe de la siguiente manera:

```
$ sort | diccionario
```

Donde el caracter " | " representa la conexión entre la salida estándar de un programa y la entrada estándar de otro. Con este fuerte y simple concepto se pueden concatenar gran cantidad de programas como si fuera una línea de producción en serie para generar resultados complejos. Para mejorar nuestro ejemplo sacaremos las palabras repetidas, antes de mostrarlas con definiciones. Suponiendo que exista un programa llamado **sacar-repetidas**, la línea de comando sería:

```
$ sort | sacar-repetidas | diccionario
```

Simple, utilizando herramientas sencillas logramos algo un poco más complicado. El inconveniente que tenemos en este ejemplo es que hay que escribir aquello a procesar. Normalmente queremos utilizar archivos como entrada de nuestros datos. Es necesario *un comando que envíe a la salida estándar un archivo*, así se procesa como la entrada estándar del **sort** y continúa el proceso normalmente. Este comando es **cat**. La sintaxis es simple:

```
cat nombre-de-archivo
```

Quedando nuestro ejemplo:

```
$ cat archivo.txt | sort | sacar-repetidas | diccionario
```

... esto crea un glosario de las palabras que se encuentren en **archivo.txt**. La combinación de comandos es incalculable y brinda posibilidades enormes.

Redirección

En repetidas ocasiones en la vida de un sistema es mejor tener todo en archivos, ya sea para guardar algún historial o para automatizar ciertas funciones dentro de scripts. Para almacenar o sacar información de archivos y vincularlas con entradas o salidas estándares se utilizan las **Redirecciones**. La redirección se expresa con los símbolos "Mayor" > y "Menor" <. Y se pueden utilizar en forma simple o doble.

Utilizando el comando **cat** se puede hacer una copia de **arch1.txt** a **arch2.txt** utilizando redirección.

```
$ cat arch1.txt > arch2.txt
```

O se puede redireccionar un archivo para visualizarlo con el comando **less**.

```
$ less < arch1.txt
```

Redirección de escritura

Para escribir un archivo se utiliza >. Hay que tener mucho cuidado de no borrar un archivo sobrescribiéndolo. Cuando se utilizan redirecciones, debido a su utilidad en los scripts, "**no se realizan confirmaciones**". Si el archivo a escribir ya existe desde antes, el redireccionador > lo sobrescribe con flujo de texto nuevo. En cambio el operador >> realiza un **agregado** de texto en el flujo existente.

No hay nada mejor que un ejemplo clarificador:

```
$ escribe-en-salida-estandar > archivo.txt
```

El (falso) comando **escribe-en-salida-estándar** justamente hace eso, escribe unas cuantas cosas en salida estándar. Puede ser un comando **ls**, un comando **cal** (calendario) o cualquier comando antes visto, así como también una combinación de comandos por tuberías.

En este punto, el contenido de **archivo.txt** es lo mismo que saldría en pantalla. Si ejecutamos otro comando redireccionado a **archivo.txt** (nuevamente), éste pierde su contenido y el resultado de la operación pasa a estar en el mismo.

Cuando se necesita tener una lista de acontecimientos, no se quiere que un acontecimiento nuevo borre a todos los anteriores. Para lograr esto **agregamos** en vez de sobrescribir.

```
$ echo Este es el acontecimiento Nro. 1 > bitacora.log
$ echo Este es el segundo acontecimiento >> bitacora.log
```

Va a escribir dos líneas en el archivo **bitacora.log** sin eliminar nada.

Ejemplo: Si queremos combinar el ejemplo de las tuberías con lo aprendido recientemente podríamos escribir:

```
$ cat archivo.txt | sort | sacar-repetidas | diccionario >>
glosario.txt
```

Redirección de lectura

Para la lectura es el símbolo menor < y se utiliza de la siguiente manera:

```
$ comando-que-acepta-stdin < archivo-de-entrada.txt
```

Como por ejemplo:

```
$ mail usuario1@micolegio.edu.ar usuario2@micolegio.edu.ar <
correo.txt
```

Dónde **correo.txt** podría ser un archivo que se genere automáticamente... así como su contenido.

Otra facilidad para redireccionar entrada estándar es <<, que después de un comando, permite ingresar, por teclado, un texto que se constituirá en la entrada estándar.

A continuación de << debe ponerse una palabra, que indicará fin de entrada (en nuestro ejemplo, **chau**). La entrada estándar constará de las líneas que se digiten a continuación hasta la primera que contenga sólo la palabra que indicaba fin de entrada. Por ejemplo:

```
$ sort <<chau
> Perú
> Argentina
> Brasil
> chau
Argentina
Brasil
Perú
```

ordenará las palabras dadas (excepto **chau** que indica el fin de la entrada). Así, << es equivalente a editar un archivo y después redireccionarlo a la entrada estándar de un programa.

Alejandro

Soluciones al desafío N° 1

María



Dejo las respuestas de los 11 ejercicios.

```
1) ~$ mkdir comandosConsola
   ~$ cd comandosConsola
   ~/comandosConsola$ touch verde azul rojo amarillo violeta
   celeste blanco rosa lila anaranjado marron gris

2) ~/comandosConsola$ mv verde azul rojo amarillo violeta
   celeste blanco rosa lila anaranjado marron gris colores_dir

3) ~$ ls colores_dir > listaColores

4) ~/comandosConsola$ ls /bin > binarios

5) ~/comandosConsola$ cat binarios listaColores >> union

6) ~/comandosConsola$ sort < union > unionOrdenado

8) ~/comandosConsola$ cat > datosv
   Nombre Apellido DNI
   Pepe   Perez    123

9) ~/comandosConsola$ pwd >> datosv

10) ~/comandosConsola$ ls /etc >> datosv

11) ~/comandosConsola$ more < datosv
```

María



Alejandro

Ahí van las soluciones que se me ocurrieron (entre paréntesis, alternativas, pero que no respetan la consigna):

```
1) $ mkdir comandoConsola
   touch verde azul rojo amarillo violeta celeste blanco rosa lila
   anaranjado marron gris
```

Grupo de estudios: Programación Bash Script

Segunda semana: del 15 al 21 de octubre de 2009

```
2) $ mkdir colores_dir
   $ mv * colores_dir
   $ cd colores_dir

3) $ ls >cat >listaColores           ($ ls > listaColores)

4) $ ls /bin >cat >binarios          ($ ls /bin > binarios)

5) $ cat listaColores binarios >union

6) $ sort union >cat >unionOrdenado ($ sort union >
unionOrdenado)

7) $ cat unionOrdenado

8) $ cat >datosv

9) $ pwd >cat >>datosv              ($ pwd >>datosv)

10) $ ls -l /etc >cat >>datosv      ($ ls -l /etc >>datosv)

11) $ cat datosv | more
```

Alejandro



Jorge

Copio los mios, se los mandé ayer a **María**, pero parece que no los recibió. Esta fue mi solución:

```
1) jorval@jorval-laptop:~$ mkdir comandosConsola
   jorval@jorval-laptop:~$ cd comandosConsola
   jorval@jorval-laptop:~/comandosConsola$ touch verde azul rojo
   amarillo violeta celeste blanco rosa lila anaranjado marron gris

2) jorval@jorval-laptop:~/comandosConsola$ mv verde azul rojo
   amarillo violeta celeste blanco rosa lila anaranjado marron gris
   colores_dir

3) jorval@jorval-laptop:~$ ls colores_dir >listaColores

4) jorval@jorval-laptop:~/comandosConsola$ ls /bin >binarios

5) jorval@jorval-laptop:~/comandosConsola$ cat binarios
   listaColores >> union

6) jorval@jorval-laptop:~/comandosConsola$ sort
   <union>unionOrdenado
```

- 7) Verificado
- 8) jorval@jorval-laptop:~/comandosConsola\$ cat >datosv
- 9) jorval@jorval-laptop:~/comandosConsola\$ pwd >>datosv
- 10) jorval@jorval-laptop:~/comandosConsola\$ ls /etc >>datosv
- 11) jorval@jorval-laptop:~/comandosConsola\$ more <datosv

Jorval



Asterix

Aquí van mis soluciones:

- 1) \$ mkdir comandoConsola
\$ cd comandoConsola
\$ touch verde azul rojo amarillo violeta celeste blanco rosa
lila anaranjado marron gris
- 2) \$ mkdir colores_dir
\$ mv * colores_dir (dentro del directorio)
- 3) \$ ls /home/asterix/comandoConsola/colores_dir/ >listacolores
- 4) \$ ls /bin/ >binarios
- 5) \$ cat binarios listacolores >union
- 6) \$ sort union >unionordenado
- 7) \$ cat unionordenado
- 8) \$ cat >datosv
- 9) \$ pwd cat >>datosv
- 10) \$ ls /etc/ >>datosv
- 11) \$ cat datosv | more

Asterix
