

SISTEMA OPERATIVO

GNU/LINUX

COMANDOS BÁSICOS

Administración de Infraestructuras
Técnologo en Informática
2012

Ap Solange Mikeliunas

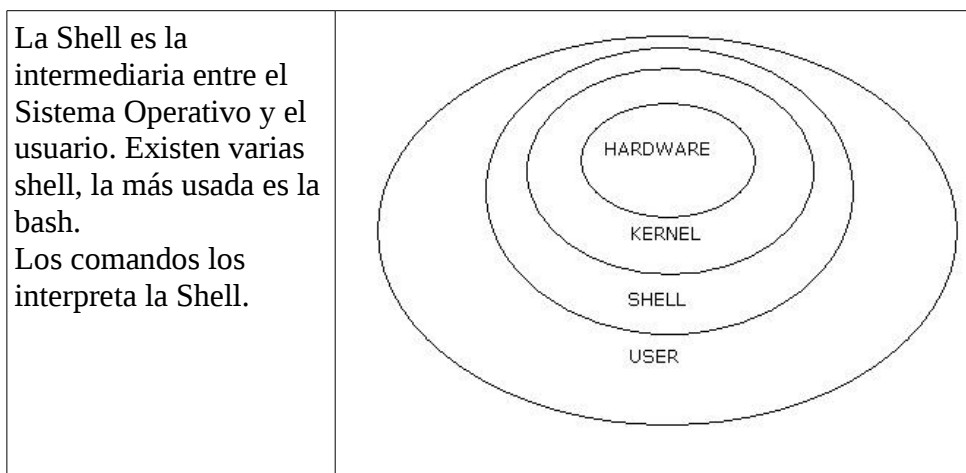
Índice de contenido

SISTEMA OPERATIVO GNU/LINUX.....	5
COMANDOS LINUX.....	5
Ingreso de comandos.....	5
FACILIDADES DE LA SHELL.....	6
Expansión.....	6
Expansión de llaves { }.....	6
Expansión de tilde ~ -.....	6
Sustitución de comando.....	6
Sustitución de proceso.....	7
División de palabras.....	7
Expansión de nombre de camino.....	7
ENTRADA – SALIDA	9
Comando tee.....	10
Exit status.....	11
COMBINACIÓN DE COMANDOS.....	11
Pipes y pipeline.....	11
Órdenes compuestas.....	12
Parentizado (lista).....	12
Parentizado {lista; }.....	12
Comando alias.....	12
Comando unalias.....	13
Secuencias.....	13
ENCOMILLADO.....	14
VARIABLES.....	15
Variables del entorno del sistema.....	15
Comando env.....	15
Variable \$PATH.....	15
Variable \$PS1	15
Variable \$TERM.....	15
Variable \$HOME.....	15
Variable \$HOSTNAME.....	15
Variable \$CDPATH.....	16
Definición de variables.....	16
Comando set.....	16
Comando unset.....	16
Comando export.....	16
Comando declare.....	17
Comando readonly.....	17
Utilización de variables.....	17
COMANDOS USO GENERAL.....	18
Comando pwd.....	18

Comando echo.....	18
Comando clear.....	18
Comandos who, w, who a mi, users, whoami.....	18
Comando tty.....	19
Comando cal.....	19
Comando date.....	20
Comando bc.....	20
Comando uname	20
Comando passwd.....	21
Comando su.....	21
Comando history.....	21
Comando fc.....	21
Teclas para la búsqueda en la historia.....	22
Ayuda en línea.....	22
Comando man.....	22
Comando apropos.....	23
Comando whereis.....	23
Comando whatis.....	23
Comando info.....	24
MANIPULACIÓN DE DIRECTORIOS.....	25
Caminos (path).....	25
Comando mkdir.....	26
Comando rmdir.....	26
Comando ls	27
Comando tree.....	28
MANIPULACIÓN DE ARCHIVOS.....	29
Comando touch.....	29
Comando stat.....	29
Comando file	30
Comando cp.....	30
Comando mv.....	31
Comando rm.....	32
Comando rename.....	33
COMANDOS PARA VER EL CONTENIDO DE UN ARCHIVO.....	34
Comando more.....	34
Comando less.....	34
Comando cat.....	34
Comando tac.....	35
Comando fmt.....	35
Comando pr.....	36
BÚSQUEDA DE ARCHIVOS.....	37
Comando find.....	37
Comando xargs.....	38
Comando locate.....	40
Comando which.....	41
ASIGNACIÓN DE PERMISOS.....	42

Permisos según el tipo de elemento:	43
Comando chmod.....	43
Tabla octal	44
Valor numérico.....	44
Permisos por defecto	46
Comando mkdir.....	46
Comando umask.....	46
Comando chown.....	47
Comando chgrp.....	47
Permisos especiales.....	48
Asignar UID.....	48
Asignar GID.....	48
Asignar Sticky.....	48
Comando stat.....	49
Comando file.....	49
EXPRESIONES REGULARES.....	51
Expresiones básicas:.....	51
Expresiones regulares compuestas:.....	51
Comando grep.....	52
Expresiones regulares extendidas.....	53
Comando egrep.....	53
Comando fgrep.....	54
Comando rgrep.....	54
MANEJO DEL CONTENIDO DE LOS ARCHIVOS (FILTROS).....	55
Comando cut.....	55
Comando tr.....	55
Comando expand.....	56
Comando head.....	57
Comando wc.....	57
Comando tail.....	58
Comando join.....	58
Comando nl.....	59
Comando od.....	60
Comando hexdump.....	61
Comando paste.....	61
Comando sort.....	61
Comando uniq.....	62
Comando split.....	63
Comando md5sum.....	64
Comando unexpand.....	64
EDITORES DE TEXTO.....	65
vi - vim.....	65
Archivo .exrc.....	65

Sistema Operativo GNU/Linux



Comandos Linux

Ingreso de comandos

Se escriben los comandos y se presiona la tecla Enter. Si el comando es válido se ejecuta, en caso contrario el sistema responde con un mensaje de error.

Bash busca los comandos a ejecutar en los directorios indicados en la variable de entorno \$PATH, pero además existen una serie de comandos que no corresponden a archivos del disco duro, sino que son internos a Bash y están siempre cargados en memoria. Ejemplos de estos comandos son: cd, chdir, alias, set o export.

Para obtener una lista completa de estos comandos con su descripción ejecutar:

```
man builtin
```

Los comandos tiene la siguiente sintaxis:

comando opciones argumentos

Teclas para la edición de la línea de comandos:

Opción	Descripción
ctrl + c	finalizar tarea, limpiar línea
ctrl + z	suspender tarea
ctrl + l	limpia la pantalla
ctrl + b	retrocede un espacio (tecla ←)
ctrl + f	Adelante un espacio (tecla →)
ctrl + a	Al principio de la línea (tecla home)
ctrl + e	Al fin de la línea (tecla end)
del	Elimina a la derecha del cursor.
ctrl + k	Elimina desde el cursor al final de la línea
ctrl + d	Elimina de la izquierda del cursor (tecla backspace)
esc del	Elimina palabra a la izquierda del cursor.
esc + d	Elimina desde el cursor al final de la palabra corriente
ctrl + y	Pega la ultima palabra eliminada
ctrl + d	exit de la sesión
tab	Autocompletar

ESC	Autocompletar, se presiona dos veces
-----	--------------------------------------

Facilidades de la shell

Expansión

Expansión de llaves { }

La expansión de llaves es un mecanismo por el cual pueden generarse cadenas arbitrarias. Los patrones a ser expandidos con la expansión de llaves toman la forma de un preámbulo opcional seguido por una serie de cadenas separadas por comas entre un par de llaves, seguido por un post scriptum opcional. El preámbulo sirve de prefijo a cada cadena de entre las llaves, y el post scriptum se añade luego a cada cadena resultante, expandiendo de izquierda a derecha.

Ejemplos

```
mkdir /home/usr1/prev{old,new,dist,bugs}fin
```

El comando mkdir crea directorios, en este ejemplo crea:

```
prevoldfin prevnewfin prevdistfin prevbugsfin
```

Expansión de tilde ~ -

Este tipo de expansión obtiene el valor de un directorio, tanto de las cuentas de usuarios, como de la pila de directorios accedidos. Los formatos válidos de la expansión de tilde son:

Formato	Descripción
~[Usuario]	Directorio personal del usuario indicado.
~	Directorio personal del usuario, es equivalente a digitar cd
-	Directorio anterior. Contenido en la variable: OLDPWD

Sustitución de comando

Esta expansión sustituye el comando ejecutado (incluyendo sus parámetros) por su salida normal. La secuencia **\$ (comando)** ejecuta el comando y permite tomar el valor devuelto por otro comando. La salida se puede almacenar en una variable.

Formatos:

```
$ (comando)
`comando`
```

Bash realiza la expansión ejecutando orden y reemplazando la sustitución de orden con la salida estándar de la orden, quitando los saltos de línea finales. Los saltos de línea empotrados no se borran, pero pueden ser eliminados durante la división de palabras.

La sustitución de orden `$(cat archivo)` puede reemplazarse por lo equivalente que es más rápido `$(< archivo)`.

Ejemplo

```
ls -la /lib/modules/$(uname -r)/*
kill -9 $(ps aux|tr -s " " "\t"|cut -f1,2|grep user2|cut -f2)
touch file$(date +%y-%M-%d)
```

La versión anterior usaba las comillas.

Ejemplo

```
echo "My present directory is `pwd`"
```

Sustitución de proceso.

La sustitución de proceso permite utilizar un archivo especial de tipo cola (FIFO) para intercambiar información entre 2 procesos, uno que escribe en la cola y el otro que lee de ella en orden (el primero en llegar es el primero en salir). La sustitución de procesos es un complemento a la sustitución de comandos.

Formato:

```
<(lista)
```

Ejemplo

```
tail <(sort /etc/passwd) <(sort /etc/group)
```

```
grep "prueba.txt" <(ls -la)
```

es equivalente a:

```
ls -la |grep "prueba.txt"
```

División de palabras

El shell examina los resultados de la expansión de parámetro, sustitución de orden y expansión aritmética que no ocurrieron dentro de comillas dobles para realizar la división de palabras.

El shell trata cada carácter de la variable **IFS** como un delimitador, y divide los resultados de las otras expansiones en palabras separadas por estos caracteres.

IFS normalmente contiene: `<espacio><tab><nueva-línea>`

Expansión de nombre de camino

Si algunas de las palabras obtenidas tras la división anterior contiene algún caracteres especial conocido como comodín (`*`, `?` o `[]`), ésta se trata como un patrón que se sustituye por la lista de nombres de archivos que cumplen dicho patrón, ordenada alfabéticamente. El resto de caracteres del patrón se tratan normalmente.

Patrones

Cualquier carácter que aparezca en un patrón, aparte de los especiales descritos más adelante, concuerda consigo mismo. El carácter NUL no puede estar en un patrón. Los caracteres de patrón especiales deben protegerse si han de concordar literalmente consigo mismos.

Los caracteres de patrón especiales tienen los siguientes significados:

*	Concuerda con cualquier cadena de caracteres, incluida la cadena vacía.
?	Concuerda con un solo carácter cualquiera.
[...]	Concuerda con uno de los caracteres entre corchetes. Un par de caracteres separados por un signo menos denota un rango
[!...]	La concordancia es con cualquier carácter de los que no estén entre los corchetes.

Dentro de [y], se pueden especificar clases de caracteres mediante la sintaxis [:clase:], donde clase es una de las siguientes clases definidas en el estándar POSIX.2:

- alnum
- alpha
- ascii
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

Una clase de caracteres concuerda con cualquier carácter que pertenezca a esa clase.

Entrada – Salida

Toda operación dentro de un proceso tiene una entrada y una salida (I/O) y en algunas ocasiones una salida de error.

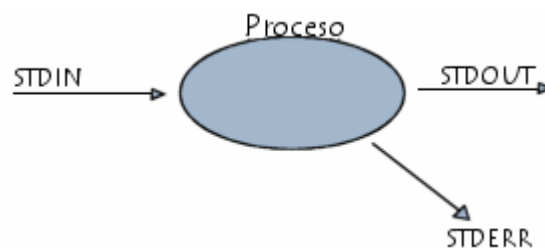
Linux al igual que todos los Unix, permite que la entrada y la salida de los comandos sea redireccionada.

Entrada (STDIN): se envían datos a un comando.

Salida (STDOUT): recibe datos de un comando.

Salida error (STDERR): salida de error de un comando

Los comandos siguen el siguiente esquema:



- Si no hay redireccionamiento la entrada y la salida son la entrada estándar y salida estándar respectivamente.
- Si ocurrió un error la salida es la salida estándar por error.
- La entrada estándar usualmente es el teclado.
- La salida estándar usualmente es la ventana actual o la terminal.

El símbolo > permite redireccionar la **salida** a un archivo

El símbolo < permite redireccionar la **entrada**, de modo que el comando tome datos de un archivo

Si el nombre de archivo existe, > sobrescribe.

Si se desea agregar al final de un archivo (append) se utiliza >> (si no existe el archivo, se crea).

Ejemplo:

```
cat > archivo.a.editar
ls /etc >> lista
mailx usu < carta.para.usu
```

Ejemplo Redireccionar el error

```
ls archivo 2>/dev/null
ls archivo 2>>file.error
```

Ejemplo Redireccionar la salida y el error

```
ls archivo 1>/dev/null 2>&1
ls >>file 2>&1
ls >file 2>file.error
```

```
ls      2>>file    >>file2
ls    &>file
ls    &>> file
ls    >& file
```

Comando tee

Lee de la entrada estándar y escribe en la salida estándar o un archivo.
sintaxis: tee -a file

Ejemplo agrega la entrada al final del archivo prueba.

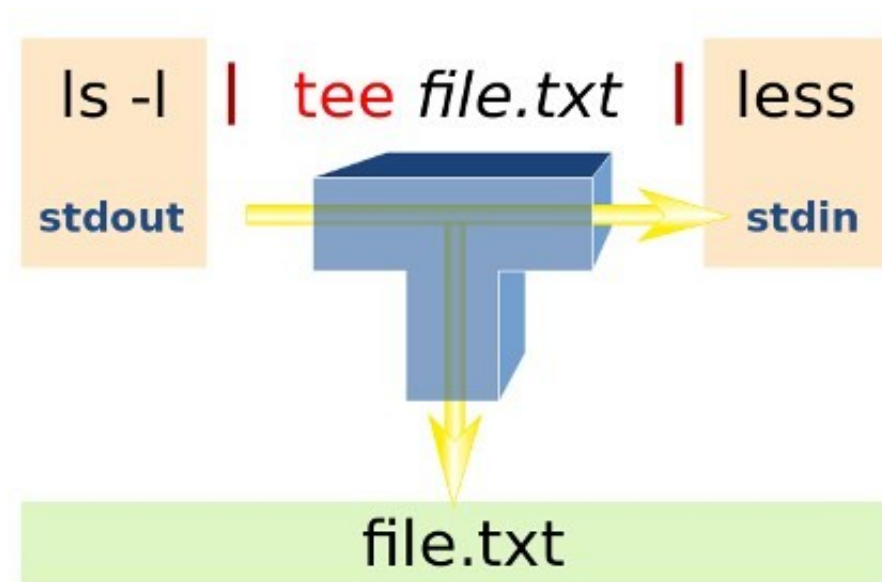
```
cat /etc/passwd |tee -a prueba
```

Sobrescribe

```
echo "Texto"|tee prueba
```

Envía la salida a dos archivos

```
cat /etc/group|tee file1 1>file2
```



Exit status

Todo comando devuelve un exit status luego de su ejecución:

- Si terminó correctamente el exit status es 0
- En caso contrario el exit status es distinto de 0

El comando **echo \$?** muestra el valor del exit status del último comando ejecutado.

Combinación de comandos

Pipes y pipeline

Las **tuberías** (en inglés "*pipes*") son mecanismos de comunicación específicos para todos los sistemas UNIX. Una tubería, simbolizada por una barra vertical (carácter "|"), permite asignar la salida estándar (stdout) de un comando a la entrada estándar (stdin) de otro. Un pipe permite enviar información de un proceso a otro.

Los procesos comunicados se ejecutan al mismo tiempo: en cuanto el primero deja datos en el pipe el segundo los toma.

Los comandos se ejecutan en paralelo, cada uno asociado a un proceso separado. Pasan sus datos a través de un buffer.

Un **pipeline** es la ejecución simultánea de 2 o más comandos simples comunicados por un pipe. El exit status de un pipeline es el exit status del último comando.

Sintaxis: comando | comando | comando ...

Ejemplos

```
ps aux | tr -s " " "\t" | cut -f2-5 | sort | uniq | nl
```

Órdenes compuestas

Parentizado (lista)

Si un comando es escrito entre paréntesis curvos (), el shell invoca a un nuevo shell que ejecuta dicho comando.

De esa forma es posible alterar las precedencias de los operadores.

Ejemplos:

```
> (cd /etc ; ls passwd) ; pwd
```

```
passwd  
/home/user1
```

```
> cd /etc ; ls passwd ; pwd
```

```
passwd  
/etc
```

```
> cat /etc/passwd ; ls -R / | more
```

```
es lo mismo que :
```

```
> cat /etc/passwd ; (ls -R | more)
```

```
pero no es lo mismo que:
```

```
> ( cat /etc/passwd ; ls -R / ) | more
```

```
>(ZZ=hola; echo "Dentro $ZZ" ; date;) ; echo "Fuera $ZZ"
```

```
Dentro hola
```

```
mié jun 20 12:05:11 UYT 2012
```

```
Fuera
```

Como se ve en el ejemplo la variable ZZ fue definida en la subshell solamente.

Parentizado {lista; }

Si un comando es escrito entre llaves { }, el shell se comporta como si hubieran () pero NO invoca a un nuevo shell para ejecutar dicho comando. De este modo es posible juntar la salida de varios comandos.

Ejemplo

```
{ id ; who ;id; who; } | sort
```

```
{ ZZ1=hola; echo "Dentro $ZZ1" ; date;} ; echo "Fuera $ZZ1"
```

```
Dentro hola
```

```
mié jun 20 12:07:11 UYT 2012
```

```
Fuera hola
```

Como se ve en el ejemplo la variable ZZ1 se crea en la misma shell.

Comando alias

Permite asociar la ejecución de un conjunto de comandos.

sintaxis: alias

Muestra todos los alias definidos.

Crear un alias:

Ejemplo

```
alias TL='ls -li;date;who'
```

Ejecución del alias, al digitar: TL, se ejecutan los comandos definidos en secuencia.

Comando unalias

Para desactivar una alias.

sintaxis: unalias nombrealias

Secuencias

Una secuencia es un conjunto de comandos simples separados por:

; && || y opcionalmente terminada por ; &

Ejecución secuencial: se ejecuta comando1 y luego comando2.

```
comando1 ; comando2
```

Se ejecuta comando1 y si la ejecución no es exitosa se ejecuta comando2. Se evalúa el exit status.

```
comando1 || comando2
```

Se ejecuta comando1 y si la ejecución es exitosa se ejecuta comando2. Se evalúa el exit status.

```
comando1 && comando2
```

Se ejecuta comando1, si es exitosa se ejecuta comando2, pero si no lo es se ejecuta comando3

```
comando1 && comando2 || comando3
```

Ejemplos

```
cat archivo &>/dev/null|| echo "El archivo no existe"  
cat archivo &>/dev/null && echo "Fin del archivo"  
cat archivo &>/dev/null && echo OK || echo MAL
```

Ejecución de un comando en foreground:

```
comando
```

En esta modalidad, los comandos son interactivos: se debe esperar al fin de la ejecución de un comando para comenzar la del siguiente.

Ejecución de un comando en background:

```
comando &
```

En este caso el shell devuelve el número de proceso asociado al comando para posibilitar el control sobre él, y devuelve de inmediato el control, dando así la posibilidad de ejecutar otros comandos al mismo tiempo.

Encomillado

- Las comillas “dobles” preservan el contenido de la variables.
- Comillas 'simples' toman el contenido literal.

Hay dos clases de comillas, y su uso lo ilustran los siguientes ejemplos:

```
> echo "mi home es $HOME"
    mi home es /home/usu101
> echo 'mi home es $HOME'
    mi home es $HOME
> cat arch
    hola
```

```
>echo $HOME
    /root
>echo '$HOME'
    $HOME
>echo "'$HOME'"
    '/root'
```

Variables

Variables del entorno del sistema

Comando env

El comando env lista todas las variables del ambiente. (idem printenv)

Para ver una variable en particular se utiliza el comando echo y el nombre de la variable con un signo de \$ (pesos) adelante.

Ejemplo:

echo \$VARIABLE

Variable \$PATH

Se utiliza en la ejecución de los comandos. Al ingresar un comando, el shell busca el programa a ejecutar en la lista de caminos que contiene la variable PATH.

Esta variable contiene una lista de caminos separados por : (dos puntos).

Si se encuentra en el PATH se ingresan directamente.

Sino se encuentra en el PATH un script o archivo ejecutable y se quiere ejecutar se debe ubicar en el directorio del comando y se digita:

./command

Ejemplo

```
> PATH=/usr/bin:/usr/openwin/bin:.  
> export PATH  
> PATH=$PATH:/usr/ucb  
> echo $PATH  
/usr/bin:/usr/openwin/bin:../usr/ucb
```

Variable \$PS1

Define el Prompt del usuario.

Variable \$TERM

Contiene el tipo de terminal.

La base de datos de configuración de terminal se encuentra en:

/etc/termcap	RedHat SuSE
/etc/terminfo/*	Debian

Variable \$HOME

Contiene la dirección del directorio personal del usuario.

Variable \$HOSTNAME

Contiene el nombre del host.

Variable \$CDPATH

Esta variable por defecto está vacía. Contiene directorios que se utilizarán con el comando cd. Al hacer cd se busca en los paht definidos en la variable.

Ejemplo:

```
CDPATH=~/dir:/tmp
cd /etc
cd subdirectorio
pwd
/root/dir/subdirectorio
```

Definición de variables

Comando set

Permite la modificación de variables del shel del usuario, y también lista todas las variables locales y variables del ambiente

sintaxis: set [-o|+o] opción

Ejemplos

set	lista todas las variables
set -o	lista cada opción del shell y su propiedad (on off)
set -o allexport	se activa esta opción cada variable que se defina automáticamente será exportada.
set +o allexport	se desactiva la opción.

Opciones activas:

```
SHELLOPTS=allexport:braceexpand:emacs:hashall:histexpand:history:in
teractive-comments:monitor
```

Comando unset

Desasignar variables asignadas

sintaxis: unset variable

Comando export

Exportar variables del ambiente, o muestra todas las variables que se exportan a otros ambientes

sintaxis: export variable[=value]

sintaxis: export

Comando declare

Agrega la variable a la lista de variables a exportar, otra forma de exportar.

Sintaxis `declare [airx] variable`

Opciones	descripción
-a	vector
-i	entera
-r	readonly
-x	exportar

Ejemplo igual que export

```
declare -x variable[=value]
```

Ejemplo variable numérica, luego podrá realizar operaciones aritméticas con ellas.

```
declare -i a=0 b=10 c=5
```

```
> a=b+c
> echo $a
15
```

Comando readonly

Lista todas read-only variables, o asigna el atributo a una variable, estas variables no se pueden cambiar o unset.

sintaxis: `readonly`

sintaxis: `readonly variable`

Utilización de variables

Ejemplos

```
var=$(date +%a-%b)
echo $var
```

```
ls >file$( date +%a-%b).txt
```

```
var=$(ls b*)
cp $var /directorio
```

Comandos uso general

Comando pwd

sintaxis: pwd

El comando pwd muestra el directorio actual.

Comando echo

sintaxis: echo [-ne]

Escribe los argumentos separados por blancos y terminados en una nueva línea en la salida estándar, de forma predeterminada.

Opciones	Descripción
-n	sin salto de línea
-e	activa la interpretación de caracteres de control: \n salto de línea \t tabulador

Ejemplos:

```
[root@localhost root]# echo "Salida del comando"
Salida del comando
```

```
[root@localhost root]# echo -n "Salida del comando"
Salida del comando[root@localhost root]#
```

```
[root@localhost root]# echo -e "Salida \n del \t comando"
Salida
  del      comando
[root@localhost root]# echo "El path $PATH"
```

Comando clear

Sintaxis: clear

Limpia la pantalla. Idem que presionar las teclas: `Ctrl + l`

Comandos who, w, who a mi, users, whoami

Estos comandos muestran los usuarios conectados al sistema.

sintaxis: w

sintaxis: who a mi

sintaxis: whoami

sintaxis: users

sintaxis: who [u|q|a|b|d|--login|p|r|t|T]

Ejemplos

Mostrar solo el nombre del usuario

```
whoami
```

root

Cuantos usuarios en el sistema

```
who -q
root solange root root
Nº de usuarios=4
```

Es lo mismo: who a mi y who -m

```
who a mi
root      tty1          Sep  5 09:08
who -m
root      tty1          Sep  5 09:08
```

Información total

```
who -a es equivalente: -b -d --login -p -r -t -T -u
who -a
```

```

                                Sep  5 09:07                16 id=si
term=0 salida=0
      system boot   Sep  5 09:07
      `run-level' 3 Sep  5 09:07                Ultimo=S
                                Sep  5 09:08                738 id=l3
term=0 salida=0
root      - tty1          Sep  5 09:08                .          1160
solange   + tty2          Sep  5 09:08 00:03          1161
root      + tty3          Sep  5 09:08 00:19          1162
root      + tty4          Sep  5 09:32 00:01          1163
LOGIN     tty5          Sep  5 09:08                1164 id=5
LOGIN     tty6          Sep  5 09:08                1165 id=6
```

Nivel del sistema

```
who -r
      `run-level' 3 Sep  5 09:07                Ultimo=S
```

Comando tty

En que consola se encuentra el usuario. Hay que recordar que hay seis terminales de texto mas la interfaz gráfica, de forma determinada.

```
tty
/dev/tty1
```

Comando cal

Muestra el calendario en la salida estándar.

Sintaxis: cal [[mes] año] [-3]

Opciones	descripción
-3	muestra el mes anterior el actual y el siguiente
mes año	el mes y año correspondiente
año	todo el año.

Comando date

Sin argumentos, despliega la fecha en la salida estándar del sistema. El formato de salida se puede especificar precedido por un +. La opción -u es para utilizar la hora universal (Greenwich). El único usuario que puede cambiar la fecha del sistema es root. Basta ingresar date y la nueva fecha.

sintaxis: date [-u] [+formato] [yymmddhhmm[.ss]]

Ejemplos:

```
> date
> date -u
> date +%D
> date +Dia :%d/%m/%y
> date +%H:%M:%t%t%T
```

El siguiente cuadro muestra algunas de opciones disponibles, para ver todas las opciones consulta las páginas del man.

Opción	Descripción
n	Inserta un enter
t	Inserta un carácter
m	Meses del 1 al 12
d	Días del 1 al 31
y	Últimos dos dígitos del año
D	Fecha con formato mm/dd/aa
H	Hora de 00 a 23
M	Minutos de 00 a 59
S	Segundos de 00 a 59
T	Hora con formato HH:MM:SS
j	Día del año de 001 a 366
w	Día de la semana, domingo =0
a	Abreviatura del día de la semana: Sun, Mon, etc.
h	Abreviatura para el mes: Jan, Feb, etc.
r	Hora con formato AM/PM

Comando bc

Calculadora binaria.

Comando uname

Sintaxis: uname [a|s|n|r|v|m|p|i|o]

Muestra la información del sistema operativo.

Opciones	descripción
-a	Muestra toda la información
-s	Nombre del sistema operativo
-n	Nombre del host
-r	Versión del sistema
-v	Fecha de la versión
-m	Tipo de maquina
-p	Tipo de procesador

-i	Tipo de hardware
-o	Sistema operativo

Ejemplo

```
uname -a
Linux acer1.solange.edu.uy 2.6.32-220.7.1.el6.x86_64 #1 SMP Wed Mar
7 00:52:02 GMT 2012 x86_64 x86_64 x86_64 GNU/Linux
```

Comando passwd

Permite cambiar la contraseña del usuario.

Comando su

Ejecuta la shell sustituyendo al usuario logeado.

Siendo un usuario común puede transformarse en el usuario root si conoce la password.

Sintaxis: su - [-c comando]

Opciones	descripción
-c	Ejecuta un comando como root:, ejemplo: su - '-c /sbin/halt'
-	Pasa a ser root. Con su perfil

Comando history

Muestra los comandos ingresados en la consola.

sintaxis: history [nro | c]

Ejemplos:

```
history          #muestra todo el historial
history 10       #muestra las últimas 10
history -c       #limpia el historial
```

Apagar o prender el historial

```
set +o history   #Apaga el historial
set -o history   #Prende el historial
```

Variables del sistema involucradas con el historial

\$HISTFILE	Contiene el nombre del archivo. Normalmente es: ~/.bash_history
\$HISTFILESIZE	Esta variable contiene el tamaño máximo del archivo
\$HISTSIZE	Esta variable contiene el tamaño máximo de comandos

Comando fc

Comando asociado al historial, lista, busca, edita y ejecuta comandos.

sintaxis: fc [-l|-n]

Opciones	Descripción
-l	Lista
-n	edita y ejecuta

Ejemplos:

<code>fc -l -##</code>	Muestra las últimas líneas, o las ##, <code>fc -l -10</code>
<code>fc -l string #</code>	Busca en la historia por el string y muestra desde la coincidencia hasta el final.
<code>fc -l Nro1 Nro2</code>	Muestra desde el comando Nro1 hasta el comando Nro2
<code>fc</code>	Edita el último comando, y al cerrar el editor ejecuta el comando.
<code>fc -n string</code>	Edita desde el comando que coincida con el string
<code>fc -n Nro1 Nro2</code>	Edita desde el comando Nro1 hasta el comando Nro2.

Teclas para la búsqueda en la historia

Opción	Descripción
<code>!!</code>	Ejecuta el último comando
<code>!nro</code>	Ejecuta el comando numero nro
<code>ctrl r</code>	Buscar comando
<code>! -n</code>	Ejecuta el comando ejecutado hace n posiciones anteriores.
<code>! string</code>	Ejecuta el comando que comienza con el string, recientemente ejecutado.
<code>!? string</code>	Ejecuta el comando que contiene el string.
<code>ctrl p</code>	Línea previa (tecla ↑)
<code>ctrl n</code>	Línea siguiente (tecla ↓)
<code>alt <</code>	Ir al principio
<code>alt ></code>	Ir al final
<code>^string1^string2</code>	Ejecuta el comando anterior sustituyendo string1 por string2

Ayuda en línea

Muchos comandos ofrecen una ayuda sintáctica sobre las posibles opciones.

Sintaxis: `comando --help`

Comando man

Manual en línea, el comando `man` permite acceder al manual en línea de Linux.

Este contiene la descripción exhaustiva de todos los comandos y sus opciones.

Sintaxis: `man n [a|k|f|w] comando`

Opción	Descripción
<code>man comando</code>	Para consultar sobre un comando
<code>man -a comando</code>	Para consultar todas las páginas existentes sobre un comando
<code>man -k [clave]</code>	Busca la clave en la descripción de las paginas <code>man</code> , que se encuentra en la base de datos de <code>whatis</code> .
<code>man n comando</code>	Para consultar sobre una sección de ayuda, del 1 al 9
<code>man -f comando</code>	Descripción del comando.
<code>man -w comando</code>	Devuelve la localización de la pagina.

Secciones del man

Sección de man	Descripción
1	Executable programs or shell commands
2	System calls (functions provided by the kernel)
3	Library calls (functions within program libraries)
4	Special files (usually found in <code>/dev</code>)
5	File formats and conventions eg. <code>/etc/passwd</code>
6	Games
7	Miscellaneous (including macro packages and conventions), e.g. <code>man(7)</code>
8	System administration commands (usually only for root)
9	Kernel routines [Non standard]

El orden de búsqueda en las paginas es: 1,8,2,3,4,5,6,7,9

Comando apropos

Este comando cumple la misma función que el comando `man -k`.

Comando whereis

Este comando devuelve la localización de un comando y de su ayuda, si existe. Devuelve mas información que el comando `man -w`

sintaxis: `whereis comando`

Comando whatis

Devuelve la cabecera de las paginas `man` que coinciden con el comando.

Es como el comando: `man -f`

sintaxis: `whatis comando`

La base de datos de "whatis" se crea con el comando `/usr/sbin/makewhatis`

Comando info

Manual en línea, el comando info permite acceder a las páginas info de los comandos, al igual que el comando man brinda documentación y ayuda sobre los comandos del shell.

Sintaxis: `info comando`

Ejercicio

1. Cuales son las paginas man del comando passwd.
2. Obtenga ayuda del comando passwd.
3. Obtenga ayuda del archivo passwd.
4. Comandos para manejo de archivos y directorios

Manipulación de directorios

Comandos relativos a manejo de directorios.

Comando	Descripción
<code>pwd</code>	Muestra el directorio actual.
<code>cd directorio</code>	Para cambiar el directorio actual.
<code>mkdir directorio</code>	Crea directorios.
<code>rmdir directorio</code>	Borra directorios vacíos.
<code>ls directorio-archivos</code>	Lista el contenido de un directorio
<code>tree</code>	Muestra la estructura de directorios
<code>rm directorio-archivos</code>	Elimina directorios no vacíos y archivos.

Camino (path)

Un nombre de camino (path name) identifica un archivo o directorio en forma única dentro de la estructura de archivos.

Contiene las “direcciones” a tomar dentro de la estructura de modo de localizar un determinado archivo o directorio. El separador de “direcciones” es /.

Ejemplo:

```
/home/user1/textos/mi_texto
```

Hay dos clases de nombres de caminos:

1. **absolutos:** describen la ubicación de un archivo o directorio en el contexto de toda la estructura de archivos. Comienzan con /

ejemplo:

```
/home/usr1/textos
```

2. **relativos:** describe la ubicación de un archivo o directorio en relación al directorio actual.

ejemplos:

```
home/user1/textos/texto1
```

```
../textos/texto1
```

Abreviaturas para algunos nombres de caminos:

<code>.</code>	Directorio actual
<code>..</code>	Padre del directorio actual
<code>~</code>	Camino absoluto al home directory
<code>~user</code>	Al home del usuario user

Ejemplo:

Para ver estos ejemplos deben estar creados los usuarios `usr1` y `usr2`

```
> su - usr1
```

```
> cd /tmp
```

```
> cd
```

```
> pwd
```

```
/home/usr1
```

```

> cd textos
textos: bad directory
> mkdir textos
> cd textos
> pwd
/home/usr1/textos
> cd ..
> pwd
/home/usr1
> exit
> cd /tmp
> cd ~usr2
> pwd
/home/usr2

```

Comando mkdir

Crea un directorio, o un conjunto de directorios

Sintaxis: `mkdir [-p|-m|-Z|-v] [directorio/directorio...]`

Ejemplos

```
>mkdir dir1
```

Crea el directorio dir1

```
>mkdir dir2 dir3 dir4
```

Crea los directorios dir2 dir3 dir4

```
>mkdir -p dir/dir5/dir6
```

El modificador `-p` permite crear todo un camino, en este caso crea primero el directorio dir, dentro de este dir5 y dentro de dir5 el dir6.

Ejemplo

```
mkdir -p primero/a/b/{abc,cdf}/otro
```

```

primero/
|-- a
    |-- b
        |-- abc
        |   |-- otro
        |-- cdf
        |   |-- otro

```

```
mkdir -m 755 directorio
```

El modificador `-m` permite asignar permisos al directorio en el momento de su creación.

Comando rmdir

El comando `rmdir` permite eliminar directorios vacíos.

Comando ls

Despliegue del contenido de un directorio

Sintaxis: `ls [-opciones] [nombre(s) de camino]`

Los caminos pueden corresponder a:

directorios: en ese caso se muestra su contenido

archivos: en ese caso se muestran datos sobre ese archivo

Opciones	Descripción
-a	Muestra archivos ocultos. Éstos comienzan con "."
-A	Como el anterior, pero no muestra "." y ".."
-d	Cuando el argumento para ls es un directorio, muestra el nombre y otros datos del directorio en lugar de su contenido. (<code>ls -d */</code>)
-F	Permite diferenciar los directorios, los archivos ejecutables y los links de los archivos comunes: @ link simbólico * ejecutable / directorio
-l	Formato largo, en orden alfabético por nombre de archivo.
-r	Ordena la salida en forma inversa a la establecida.
-R	Lista los directorios en forma recursiva (en profundidad desde el actual).
-i	Muestra el número de i-nodo en la primer columna.
-t	Ordena la salida por fecha de modificación.
-C	Muestra la fecha de modificación del i-nodo.
-u	Muestra la fecha del último acceso (en lugar de la de modificación).
-g	Idem l pero no muestra el propietario.
-h	Formato humano
-I	Excluye.: <code>ls -I "t*"</code>
-Z	Contexto selinux

Ejemplo:

```
> ls -l
total 2
-rwxr--r--  1 usr1 class 2048 Oct 24 11:10 prueba
-rwxr--r--  1 usr2 class   48 Oct 26 10:05 ejecut
```

La información corresponde (de izq. a der.) a:

- Tipo de archivo
- Permisos para el dueño, el grupo del dueño y el resto
- Contador de links
- Dueño
- Grupo dueño
- Tamaño (bytes)
- Fecha y hora de la última modificación
- Nombre del archivo

Códigos para los distintos tipos de archivos

- - Archivo común
- d Directorio
- b Archivo especial de bloques
- c Archivo especial de caracteres
- l Link simbólico
- p Named pipe o stream, utilizados para comunicación entre procesos
- s Archivo asociado a un socket

Ejemplos:

```
> cd /etc
>ls pass*
passwd passwd- passwd.lock passwd.OLD

> ls -d rc*/
rc0.d/ rc1.d/ rc2.d/ rc3.d/ rc4.d/ rc5.d/ rc6.d/ rc7.d/ rc.d/

> ls -d rc[0-2]*/
rc0.d/ rc1.d/ rc2.d/

>ls -d rc[!0-2]*/
rc3.d/ rc4.d/ rc5.d/ rc6.d/ rc7.d/ rc.d/

>ls /etc/rc?.d/ -d
/etc/rc0.d/ /etc/rc1.d/ /etc/rc2.d/ /etc/rc3.d/ /etc/rc4.d/
/etc/rc5.d/ /etc/rc6.d/

>ls o*i
odbc.ini odbcinst.ini

>ls /etc/{cups,samba}
/etc/cups:
certs client.conf interfaces mime.convs ppd
printers.conf pstoraster.convs
classes.conf cupsd.conf lpoptions mime.types ppds.dat

/etc/samba:
lmhosts.res secrets.tdb smb.conf
```

Ejercicio

1. Liste los archivos de /var/log que terminan en 1 o en 2.
2. Liste los archivos de /var/log que comienzan con boot o con mail.
3. Liste solamente los directorios contenidos en /var/log

Comando tree

El comando despliega la estructura del árbol de directorios, opcionalmente los archivos y sus permisos.

Sintaxis: `tree [-augdfpi]`

Opciones	Descripción
-a	Incluso los archivos ocultos
-d	directorios
-f	camino total
-u	dueño
-g	grupo
-p	Permisos
-i	Sin líneas, se utiliza junto con -f

Manipulación de archivos

Comando touch

Crea un archivo vacío, también permite modificar la fecha de acceso y modificación.

Sintaxis: `touch [a|m] [-r archivo] [-t fecha] archivo[s]`

Opciones	detalle
-a	Cambia la fecha de acceso del archivo
-m	Cambia la fecha de modificación
-r archivo	Toma la fecha del archivo como referencia
-t time	Valor de la fecha en decimal. Formato: aaaaMMddHHmm.ss

Ejemplo

```
touch /tmp/archivo
touch /tmp/file-{one,two}-{a,b}
ls -l /tmp
total 0
-rw-rw-r-- 1 jack jack 0 Apr 8 10:33 file-one-a
-rw-rw-r-- 1 jack jack 0 Apr 8 10:33 file-one-b
-rw-rw-r-- 1 jack jack 0 Apr 8 10:33 file-two-a
-rw-rw-r-- 1 jack jack 0 Apr 8 10:33 file-two-b
-rw-rw-r-- 1 jack jack 0 Apr 8 10:29 archivo
```

Comando stat

Devuelve información del archivo.

Sintaxis: `stat archivo`

Ejemplo:

```
stat xrbdb.txt
  File: `xrbdb.txt'
  Size: 417          Blocks: 8          IO Block: 4096   Regular
File
```

```
Device: 305h/773d      Inode: 37634      Links: 1
Access: (0644/-rw-r--r--)Uid: (0/ root) Gid:( 0/ root)
Access: 2006-04-13 13:56:00.000000000 +0200
Modify: 2006-02-28 22:07:26.000000000 +0100
Change: 2006-03-03 11:11:08.000000000 +0100
```

Tiempo:

Access: cuando se accedió al archivo por ejemplo con el comando cat.

Modify: cuando se modifico su contenido.

Change: cuando se modifico su contenido o sus permisos, su información.

Modificación de la fecha de modificación: 2012/01/02 10:30

```
>touch -m -t 201201021030 xrbdb.txt
```

```
>stat xrbdb.txt
```

```
File: `xrbdb.txt'
Size: 417          Blocks: 8          IO Block: 4096   Regular
File
Device: 305h/773d      Inode: 37634      Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2006-04-13 13:56:00.000000000 +0200
Modify: 2006-01-02 10:30:00.000000000 +0100
Change: 2006-04-13 14:05:47.000000000 +0200
```

Comando file

Muestra información del tipo de archivo.

Sintaxis: file Nombre_archivo.

Ejemplo

```
[root@rh4 etc]# file /etc/passwd
/etc/passwd: ASCII text
```

```
[root@rh4 etc]# file /etc/rc.d/rc
/etc/rc.d/rc: Bourne-Again shell script text executable
```

```
[root@rh4 etc]# file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, dynamically linked (uses shared libs),
stripped
```

Comando cp

Este comando permite copiar archivos y/o directorios.

Sintaxis: cp `-[i|r|R|p|--parents|a|d|x|v]` origen destino

Opciones	Descripción
-i	Interactivo: pide confirmación de la copia cuando el archivo destino existe.
-p	Preserve: No cambia ni permisos ni fecha de modificación

-r, -R	Rekursivo: si alguno de los archivos origen es un directorio, copia (recursivamente) su contenido. El destino debe ser un directorio.
--parents	Copia el archivo creando toda la estructura de directorios
-a	Es como -dpR
-d	Copia los enlaces simbólicos como tales, no los archivos a los que apunta.
-x	Se salta subdirectorios que están en sistemas de archivos diferentes al que empezó la copia.
-f	Fuerza, sobrescribe.
-n	No sobrescribe
-s	Crea un enlace simbólico: <code>cp -s ejer.txt sejer</code>
-l	Crea un hard link
-v	Verbose. Muestra el nombre de cada archivo antes de copiarlo.

Ejemplos

Sintaxis: `cp archivo1 archivo2`

Copia el contenido de archivo1 en archivo2. Se modifica la fecha de modificación y el dueño del archivo.

Sintaxis: `cp -[rR] directorio1 directorio2`

- Si directorio2 no existe:
Crea directorio2 y copia recursivamente el contenido de directorio1 en directorio2.
- Si directorio2 ya existe:
Crea debajo de él un directorio1 donde realiza la copia.

Sintaxis: `cp archivo(s) directorio`

Copia los archivos a directorio, que debe existir.

Sintaxis: `cp --parents /dir/file /dir2/dir3`

Copia el archivo file en /dir2/dir3/dir, creando toda la estructura de directorios que se especifique.

Por ejemplo:

`cp --parents /etc/hosts /home/user1`

resultado:

`/home/user1/etc/hosts`

- Por defecto, el archivo creado en la copia tiene como dueño a quien hace la copia, y como fecha de modificación la de la copia.
- No es posible copiar un archivo sobre sí mismo.
- En el caso de los links simbólicos, no se copia el link, se copia el contenido del archivo al cual apunta.

Comando mv

El comando mv permite mover archivos o directorios, o cambiarles el nombre.

Sintaxis: `mv [-fi] origen destino`

Opciones	Descripción
-f	Fuerza: suprime cualquier mensaje de advertencia y realiza el movimiento suprimiendo cualquier tipo de restricción (siempre que los permisos lo permitan).
-i	Interactivo: Pregunta antes de sobrescribir cualquier archivo o subdirectorio.

Sintaxis: `mv -f archivo1 archivo2`

Renombra archivo1 a archivo2. Borra archivo2 si existía (si los permisos lo permiten).

Sintaxis: `mv directorio1 directorio2`

Si directorio2 no existe, entonces renombra directorio1 a directorio2. Si existe, mueve directorio1 dentro de directorio2.

Sintaxis: `mv archivo(s) directorio`

Mueve los archivos o directorios al directorio destino especificado.

- No es posible mover un archivo o directorio sobre sí mismo.

Comando rm

Borrar archivos o directorios.

Borra uno o más archivos. Como borra la entrada del directorio, si se borra el último link a un archivo, el contenido de éste se pierde de forma definitiva.

¡Linux no tiene undelete!

Para poder borrar un archivo es necesario tener permiso de escritura sobre el directorio en el que éste se encuentra.

Sintaxis: `rm [-f|i|r|v] archivo(s)`

Opciones	Descripción
-f	Fuerza: borra sin hacer caso a los permisos (siempre que se tengan las potestades adecuadas sobre el archivo o directorio a borrar).
-i	Interactivo: Pide confirmación antes de borrar.
-r	Recursivo: si el argumento de <code>rm -r</code> es un directorio, se borra su contenido, y recursivamente el de todos los subdirectorios que este contenga
-v	Verbose. Explica lo que esta haciendo

Ejemplos:

```
cd /home/user1
```

```
rm -rf *
```

Elimina todos los archivos y directorios.

```
rm -rf .??* *
```


Elimina todo, incluso los ocultos.

Comando rename

Cambiar el nombre a un conjunto de archivos que tiene un patrón común.

Sintaxis: `rename a b c`

Opciones	descripción
A	que cambiar
B	cambiar por
C	donde

Ejemplo

Modificar los archivos para que no sean ocultos:

```
rename "." "" /home/user3/.??*
```

Comandos para ver el contenido de un archivo

Comando more

Permite desplegar en pantalla el contenido de uno o más archivos.

El despliegue se organiza de a pantallas, mostrando en la última línea el porcentaje ya desplegado.

Se debe tener permiso "r" (lectura) sobre el archivo.

Se utiliza en archivos cortos.

Sintaxis: `more [archivo(s)]`

Ejemplos:

```
cd /etc/xinetd.d
```

```
more telnet
```

```
more *
```

Comandos que permiten controlar el scroll:

espacio	scroll hasta la próxima pantalla
enter	scroll de una línea
b	retrocede una pantalla
f	avanza una pantalla
h	help
q	quit
/string	búsqueda hacia adelante de string

Comando less

Idem que more pero permite el retroceso, se utiliza para desplegar archivos largos.

Sintaxis: `less [archivo(s)]`

Comando cat

Concatena archivos y los muestra en la salida estándar, también permite la creación desde la entrada estándar de un nuevo archivo.

Sintaxis: `cat [opciones] [archivos]`

Opciones	Descripción
-A	muestra todos los caracteres equivalente a -vET
-b	numera las líneas que no están en blanco
-e	equivalente a -vE
-E	muestra el final de la línea con \$
-n	numera todas las líneas
-s	salta varias líneas en blanco
-t	equivalente a -vT
-T	muestra los tabuladores como ^I
-v	muestra caracteres no imprimibles

Ejemplos:

```
cat -n /etc/passwd
```

```
cat -A /etc/hosts /etc/profile
```

Ingresar los siguientes comandos para crear archivos desde la entrada estándar:

```
[root@localhost bor]# cat >file
Primera linea creando el archivo con cat
segunda linea
Termino con ctrl d

[root@localhost bor]# cat >>file
Se ingresan lineas al final del
archivo ya existente.

[root@localhost bor]# cat <<FIN >>file
> otra forma de ingresar, al digitar la palabra FIN
> en una linea se termina la edición.
> FIN

[root@localhost bor]#cat file

[root@localhost bor]# cat <file
Primera linea creando el archivo con cat
segunda linea
Termino con ctrl d
Se ingresan lineas al final del
archivo ya existente.
otra forma de ingresar, al digitar la palabra FIN
en una linea se termina la edición.

[root@localhost bor]# cat <file >otrofile

[root@localhost bor]# cat otrofile
```

Comando tac

Como el comando cat lista un archivo pero en orden inverso. No son válidas las mismas opciones.

Comando fmt

Formatea cada párrafo de un archivo o de la entrada estándar, establece un ancho máximo de 75 caracteres por defecto.

Sintaxis: `fmt -[wsu] file`

Opciones	Descripción
-w	cantidad de caracteres
-S	divide las líneas largas para que entren en el ancho especificado
-u	un solo blanco de separación.

Ejemplo

```
cat /etc/hosts |fmt -w 40
```

Comando pr

Prepara un archivo para imprimir.

Sintaxis: `pr -[w|l] archivo`

Opciones	Descripción
-w	ancho de pagina
-l	largo de pagina

Ejemplo

```
pr /etc/hosts -l 20 -w 75
```

Búsqueda de archivos

Comando find

Búsqueda de archivos en la estructura de directorios

Sintaxis: `find camino opción`

Busca en forma recursiva desde camino hacia abajo en la estructura de archivos, para buscar los archivos que cumplan con la condición especificada. Si no se especifica camino busca en el directorio actual.

Opción	Descripción
<code>name</code>	nombre de archivo
<code>type</code>	tipo: b archivo especial de bloques d directorio c archivo de caracteres f archivo l link simbólico s socket
<code>size</code> <code>n[cwbkMG]</code>	tamaño del archivo buscado, b' for 512-byte blocks (this is the default) c' for bytes w' for two-byte words k' for Kilobytes (units of 1024 bytes) M' for Megabytes (units of 1048576 bytes) G' for Gigabytes (units of 1073741824 bytes)
<code>atime nro</code>	día del último acceso al archivo
<code>mtime nro</code>	día de la última modificación.
<code>user</code>	del usuario
<code>group</code>	del grupo
<code>newer</code>	archivo Archivos modificados más recientemente que archivo.
<code>print</code>	Muestra en la salida estándar el camino donde se encontró el archivo. Por defecto.
<code>exec comando</code>	busca los archivos y exec ejecuta el comando
<code>mmin</code>	Minutos, mmin + 5 hace mas de 5; mmin -10 hace menos de 10
<code>perm -perm</code>	Busca por un permiso
<code>nouser</code>	Sin usuario propietario
<code>nogroup</code>	Sin grupo propietario
<code>maxdepth</code>	Niveles, en 1 busca en primer nivel
<code>fprint arc</code>	Almacena la salida en el archivo

Ejemplos:

```
find /etc -name host -print
```

```

find /etc -name "host*" -exec ls -l "{}" ";"
find /var/spool/ /tmp/ /home/ -user solange
find /var/spool/ /tmp/ /home/ -user solange -type d -name "r*"
find /var -group "pepe" -name "d*"
find /var -perm -2 -type f #busca con permiso de escritura
find /var -perm 0755 -type d #por el permiso
find /home/solange/movie/ /tmp -perm +a+x
find /var -nouser #sin usuario propietario
find /var -nouser -nogroup #sin usuario ni grupo propietario
find /etc -maxdepth 1 -name "a*" -type f
find /etc -mmin -5 #hace menos de 5 minutos.
find /var -mmin +60 -mmin -180 #mas de 1 hora y menos de 3
find /var/log -mtime -1 #modificados hace menos de 24 horas
find . -type f -exec file '{}' \;
find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f

```

Enlaces simbólicos:

La opción -P por defecto no busca en los directorios apuntados por enlaces simbólicos.
 find -L -name "nombre*" # busca en los directorios apuntados por enlaces simbólicos.

Comando xargs

En el momento que empezamos a tratar con directorios con una gran cantidad de archivos, en el empezamos a ver como realmente funcionan las cosas y la gran utilidad de xargs.

Suponiendo que tenemos en un directorio muchos archivos y queremos borrarlos, normalmente ejecutaríamos:

```
rm -rf *
```

Transforma la expresión "*" en una lista de todos los archivos que cumplen tal expresión.

Evidentemente, al ser la lista tan larga no cabe en el buffer reservado al comando a ejecutar (todo tiene un límite).

Gracias a xargs podemos ejecutar el comando de la siguiente manera:

```
ls | xargs rm -fr
```

Esto lanzaría un proceso rm para cada archivo del directorio. Tenemos otras opciones disponibles que nos pueden ser útiles:

La opción -t nos permite ver el comando antes de ejecutarlo:

```

ls | xargs -t echo
echo copia.txt link-no link-unexp link-unexp-2 lns-no lns-salida
sejer sejer-d skel
copia.txt link-no link-unexp link-unexp-2 lns-no lns-salida sejer
sejer-d skel

```

En la salida nos muestra el comando echo y la lista repetida de los archivos.

Utiliza como salida el comando echo.

Para ver solamente los argumentos:

```

ls | xargs
copia.txt link-no link-unexp link-unexp-2 lns-no lns-salida sejer
sejer-d skel

```

La opción -n cantidad de argumentos a la vez que se pasan

```
ls |xargs -n3  
copia.txt link-no link-unexp  
link-unexp-2 lns-no lns-salida  
sejer sejer-d skel
```

La opción -I Permite definir donde se van a definir los parámetros, por ejemplo:

```
ls |xargs -I ARG echo INICIO ARG FIN  
INICIO copia.txt FIN  
INICIO link-no FIN  
INICIO link-unexp FIN  
INICIO link-unexp-2 FIN  
INICIO lns-no FIN  
INICIO lns-salida FIN  
INICIO sejer FIN  
INICIO sejer-d FIN  
INICIO skel FIN
```

La opción -P: Permite definir el número de procesos paralelos a lanzar al mismo tiempo. Mediante esta opción podemos usar xargs como un gestor de threads. Por ejemplo, si lanzamos el siguiente comando: en un archivo de texto escribimos nuestra lista con las url para descargar. Guardamos y desde terminal:

```
cat url.txt | xargs -n 1 -P 10 wget
```

Ejemplos

Busca archivos en el directorio /tmp y eliminarlos, Procesamiento de nombres de archivo de tal manera que los nombres de archivo o directorio con espacios o saltos de línea son manejados correctamente.

```
find /tmp -name core -type f -print | xargs /bin/rm -f
```

La opción -0 deshabilita el final de la cadena de fin de fichero, que se trata como cualquier otro argumento. Es útil cuando los argumentos pueden contener espacio en blanco, comillas o barras invertidas. La opción de find de GNU -print0 produce una entrada apropiada para este modo de operación.

```
find /path -type f -print0 | xargs -0 rm
```

Diferencia entre print y print0

```
find -print  
.  
./link-unexp  
./lns-no  
./sejer  
./skel  
./skel/.profile  
./skel/examples.desktop  
./skel/.bash_logout  
./skel/.bashrc  
./copia.txt  
./sejer-d
```

```
./link-unexp-2
./lns-salida
./link-no
```

```
solange@server:~/movie/dir$ find -print0
../link-unexp./lns-
no./sejer./skel./skel/.profile./skel/examples.desktop./skel/.bash_l
ogout./skel/.bashrc./copia.txt./sejer-d./link-unexp-2./lns-
salida./link-nosolange@server:~/movie/dir$
```

```
find -print0 |xargs -0
. ./link-unexp ./lns-no ./sejer ./skel ./skel/.profile
./skel/examples.desktop ./skel/.bash_logout ./skel/.bashrc
./copia.txt ./sejer-d ./link-unexp-2 ./lns-salida ./link-no
```

La opción -d elimina un carácter al final

```
ls -F
copia.txt    link-unexp    lns-no       sejer        skel/
link-no      link-unexp-2  lns-salida@  sejer-d@
```

```
ls -F |xargs -d /
copia.txt
link-no
link-unexp
link-unexp-2
lns-no
lns-salida@
sejer
sejer-d@
skel
```

Comando locate

Buscar en todo el sistema de archivos el nombre especificado, y devuelve la lista de todas las veces que aparece el nombre especificado en el árbol de directorio. La base de datos se crea o actualiza con el comando updatedb.

La base de datos se encuentra en:

/var/lib/mlocate/mlocate.db

Sintaxis locate nombre

Para buscar todos los archivos cuyo nombre coincide con un patrón.

locate NAME

Para buscar un archivo llamado NAME (no *NAME*):

```
locate -b '\NAME'
```

De forma predeterminada la búsqueda que realiza es *NAME* para buscar exactamente utilizamos el modificador -b y la contrabarra.

Para buscar solo los 4 primeros:

```
locate -l 4 -b '\tree'
```


Comando which

Localiza un comando en el path

Ejemplos

```
>which more  
/bin/more
```

```
>which -a ls  
alias ls='ls --color=tty'  
/bin/ls
```

```
>which --skip-alias ls  
/bin/ls
```

Asignación de Permisos

Generalidades.

Los permisos en el sistema de archivos determinan quién puede acceder a los archivos y directorios dependiendo del tipo de acceso que tengan. Los primeros 10 caracteres de un listado ls -l de cualquier entidad se parecen a lo siguiente:

`-rwxrwxrwx`

El primer carácter se identifica con el tipo de entidad: - para un archivo estándar, d para un directorio, b para un grupo de recursos (tales como una unidad de cinta), c para un carácter del recurso, l para un link, o p para una tubería (pipe). El resto de los nueve caracteres se dividen en 3 grupos.

Cuando un usuario intenta acceder a un archivo, el primer control confirma si el es el propietario del archivo. Si lo es, se le aplica el primer tipo de permisos. Si no lo es, el segundo control confirma si es un miembro del grupo propietario del archivo. Si es un miembro del grupo, se le aplica el tipo intermedio de permisos. Si no es propietario del archivo, y no es miembro del grupo propietario, se le aplica el tercer tipo de permisos.

Permisos



r	read, leer
w	write, escribir
x	execute, ejecutar

-	sin permisos
---	--------------

Permisos según el tipo de elemento:

Para un Archivo:

Read: El archivo puede ser desplegado o copiado.

Write: Es posible modificar el contenido del archivo.

Execute: El archivo puede ser ejecutado (shell scripts o archivos ejecutables).

Para un directorio:

Read: Es posible listar el contenido del directorio con el comando ls.

Write: Es posible agregar o borrar archivos dentro de él (incluso si no se tiene el permiso de escritura específico sobre el archivo individual).

Execute: Control de acceso para el directorio.

Comando chmod

El comando chmod permite modificar los permisos de un archivo o directorio.

Sintaxis: `chmod [u|g|o|a] [+|=] [rwx] file`

Modificación de permisos, modo simbólico:

- u usuario
- g grupo
- o otros
- a todos
- + agrega
- - quita
- = setea

Ejemplos:

Quitar el permiso de lectura al grupo:

```
chmod g-r mi.archivo
```

Quitar permiso de lectura a others:

```
chmod o-r mi.archivo
```

Agregar permiso de ejecución al dueño, y permiso de lectura para el grupo y others:

```
chmod u+x,go+r mi.archivo
```

Setear permiso de lectura y escritura a todos

```
chmod a=rw mi.archivo
```

Modificación de permisos modo absoluto (o modo octal)

Sintaxis: `chmod modo_octal archivo`

modo_octal = valor octal que determina permisos de acceso

Valores:

R	4
W	2
X	1

Ejemplo modo absoluto:

```
chmod 644 mi.archivo
r w - r - - r - -
chmod 751 mi.archivo
r w x r - x - - x
chmod 775 mi.archivo
r w x r w x r - x
```

Setear sin ningun permiso.

```
chmod 0 httpd.conf
```

Opciones de chmod

sintaxis : chmod [-fR] modo archivos

Opciones	Descripción
-f	forzar
-R	Recursivo: cuando el argumento es un subdirectorio, se modifican los permisos del directorio, de todos los archivos de dicho directorio y se continúa el cambio recursivamente hacia abajo en la estructura

Recursivo: cuando el argumento es un subdirectorio, se modifican los permisos del directorio, de todos los archivos de dicho directorio y se continúa el cambio recursivamente hacia abajo en la estructura.

Tabla octal

Valor numérico	Permisos
0	--- --- ---
1	--- --- -X
2	--- --- -W-
3	--- --- -WX
4	--- --- r--
5	--- --- r-X
6	--- --- rW-
7	--- --- rWX
10	--- -X ---
11	--- -X --X
22	---- -W- -W-
33	--- -WX -WX
55	--- r-X r-X
77	--- rWX rWX
100	--X --- ---
101	--X --- --X
111	--X --X --X
222	-W--W--W-

311	-WX --X --X
322	-WX -W- -W-
400	r-- --- ---
444	r-- r-- r--
511	r-X --X --X
544	r-X r-- r--
644	rW- r-- r--
666	rW- rW- rW-
755	rWX r-X r-X
777	rWX rWX rWX

Ejercicio

¿Cuál de los siguientes permisos se representa por el valor numérico 044?

- a. - - - - - r w -
- b. - - - r w - - -
- c. - - - r - - r - -
- d. r - - r - - - -

Permisos por defecto

archivos: 644
directorios: 755

Comando mkdir

El comando mkdir ya visto anteriormente permite crear un directorio asignándole permisos diferentes a los definidos por la mascara.

Ejemplo

```
>mkdir -m 700 directorio
> ll
total 4
drwx----- 2 root root 4096 abr 11 22:50 directorio
```

Comando umask

Cuando se crea un directorio o un archivo, el sistema asigna permisos de forma predeterminada según el valor de umask. Este valor se asigna en el momento que el usuario ingresa al sistema. El comando umask muestra o modifica estos valores.

```
sintaxis umask [-pS]
```

Ejemplo:

```
umask 002
significa: 775
```

```
umask 022
significa 755
```

Forma de calcular los permisos efectivos:

Para directorios:

```
777
- 022
755
```

Para archivos:

```
666
- 022
644
```

Con un umask de 022, los permisos asignados a los nuevos archivos serán 644 (rw-r--r--) y a los directorios 755 (rwxr-xr-x).

Cálculo de los valores de las nuevas entidades después de sustraer el valor de umask.

<i>Archivos</i>	<i>Directorios</i>
022	022
644 rw- r --r-	755 dwxr-xr-x

Comando chown

Cuando un usuario crea un archivo o directorio a este se le asigna la propiedad del usuario y su grupo principal.

El comando chown permite cambiar el usuario o grupo propietario de un archivo o directorio.

sintaxis `chown usuario:grupo archivo/directorio`

Ejemplos:

Asignar el usuario y grupo

`chown usr1:usr1 file`

Modificar solo el grupo

`chown :user file`

Modificar solo el usuario

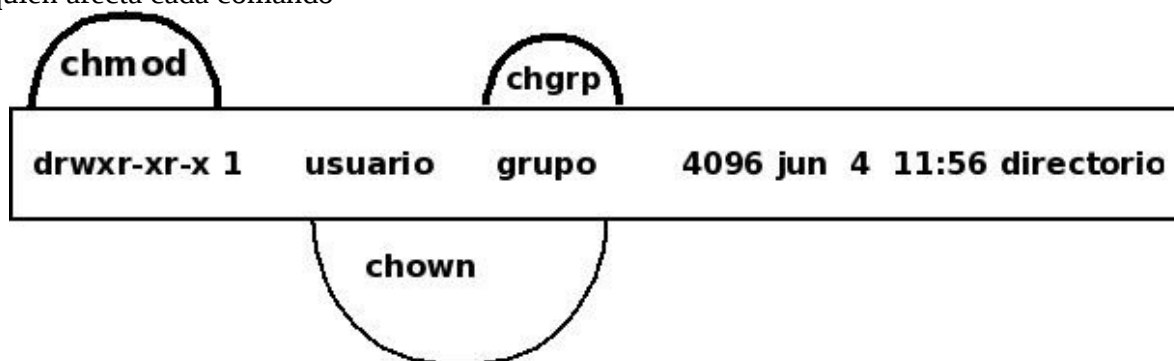
`chown usr1: file`

Comando chgrp

Modificar únicamente el grupo propietario

sintaxis: `chgrp grupo file`

A quien afecta cada comando



Permisos especiales

Asignar UID

Al activar este permiso el archivo ejecutable pasa a estar disponible para todos los usuarios como si fueran sus propietarios. Utilizado para programas.

Normalmente, un programa en ejecución pertenece a quien lo esté ejecutando. Si está activado el UID el programa pertenece al propietario del archivo. Esto quiere decir que el programa en ejecución tiene todos los permisos del propietario del archivo. Si el usuario común ejecuta el programa y el programa es propiedad del root, el programa tiene automáticamente permiso para leer y escribir cualquier archivo del sistema, sin tener en cuenta los permisos del usuario común. Por ejemplo cambiar la contraseña de un usuario.

Ejemplo:

```
-rwsr-xr-x 1 root root 30768 feb 22 09:48 /usr/bin/passwd
```

Permisos: 4755

Asignar GID

Este se aplica en directorios, al activar este permisos en un directorio los archivos y directorios que se creen en el heredan su GID, es decir que pertenecerán al mismo grupo que el directorio padre.

Esta propiedad es útil para crear directorios de trabajo compartido por ejemplo de un sector de la empresa. La forma de trabajar con este permiso es crear previamente un grupo, asignarle los usuarios, y luego asignarle al grupo la pertenencia del directorio.

Por ejemplo:

```
drwxrws---. 2 root usuarios 4096 jun 13 13:30 DirGrupo
```

Permisos: 2770

Asignar Sticky

Este se aplica en directorios, permite que todos los usuarios pueden ver y grabar archivos o directorios. Pero solo podrán eliminarlos si son los propietarios de dichos objetos, o es el usuario root.

Para que este permiso esta vigente, todos los demás permisos primarios deben estar activos. Se utiliza para tener un directorio compartido para todo el mundo. Por ejemplo /tmp

Por ejemplo:

```
drwxrwxrwt. 18 root root 4096 jun 13 12:09 tmp
```

Permisos: 1777

valores		
4	2	1
s	s	t
SUID	SGID	sticky

Comando stat

Información de un archivo.

El comando devuelve mas información del archivo, como ser los permisos en octal, las fechas de: acceso (access), modificacion (modify) y cambio (change).

Sintaxis: stat file

Comando file

Realiza algunos chequeos sobre el archivo para determinar su tipo. Por ejemplo los ejecutables se marcan con un número mágico.

sintaxis file [-il] archivo(s)

Opción	Descripción
-i	Información mas explicita.
-L	Cuando el archivo corresponde a un link simbólico, se testea por el contenido del archivo, no por el link propiamente dicho.

Cuando el archivo corresponde a un link simbólico, se testea por el contenido del archivo, no por el link propiamente dicho.

Ejemplos:

```
file /usr/bin/vim
```

```
/usr/bin/vim: ELF 32-bit LSB executable, Intel 80386, version 1  
(SYSV), for GNU/Linux 2.2.5, dynamically linked (uses shared libs),  
stripped
```

```
file /etc/passwd
```

```
/etc/passwd: ASCII text
```

```
file -i /etc/passwd
```

```
/etc/passwd: text/plain; charset=us-ascii
```

```
cd / ; file $(ls | head -10)
```

Ejercicio

¿Cuales serán los permisos del archivo cuando se utilice chmod con el valor numérico 1777?

e. r w s r w x r w x

f. r w x r w s r w x

g. r w x r w x r w t

h. r w x r w x t w T

Expresiones regulares

Metacaracteres: \ ^ \$. [] | () * + ?

Son utilizadas para buscar expresiones en textos.

Expresiones básicas:

Los caracteres cualquiera (menos los metacaracteres) coinciden con sí mismos.

\c	Coincide con un símbolo especial
\t	Coincide con un tabulador
*	Coincide con un * (no es metacaracter)
^	Coincide con el principio. de un string
\$	Coincide con el final de un string
.	Coincide con cualquier carácter
[...]	Coincide con un conjunto de caracteres
[abc]	Coincide con cualquiera de los caracteres a, b o c.
[[:upper:]]	Coincide con cualquiera de la A a la Z.
[[:lower:]]	Coincide con cualquiera de la a la z.
[0-9].	Coincide con cualquier carácter que sea dígito
[^0-9].	Coincide con cualquier carácter que no sea dígito
C*	El carácter C se repita de cero o mas veces

Ejemplos:

Expresión	Resultado
ab*c	abbbc, ac
b[cq]*e	bce, bqce, be, bccce, bqce
A[A-Z]T	ANT, (no con AnT)
al.o	Algo
h[ae]ll	hall, hell

Expresiones regulares compuestas:

[] \ { } () |

Repetición:

c\{n,m\}	El carácter c se puede repetir desde n hasta m veces
c\{n,\}	El carácter c se puede repetir desde n en adelante
c\{n\}	El carácter se repite exactamente n veces
C?	El caracter C se repite cero o una vez
C+	El caracter C se repite una o mas veces
Ejemplos	
[0-9]\{2,4\}	Es un dígito y puede repetirse desde 2 hasta cuatro veces.
[0-9]\{4\}	Es un dígito y se repite cuatro veces

<code>[0-9]\{4,\}</code>	Es un dígito y se repite cuatro veces o mas
<code>[0-9]\{4\}[][.]</code>	Es un dígito y se repite cuatro veces, luego un espacio en blanco, luego un punto.
<code>[[:upper:]]\{3\}</code>	Son exactamente tres letras mayúsculas.
<code>A[[:lower:]]\{3\}</code>	Comienza con la letra A, luego siguen exactamente tres letras minúsculas.
<code>\<palabra\></code>	Exactamente una palabra

Comando grep

Busca en los archivos las líneas que concuerdan con la expresión regular dada y las despliega en la salida estándar.

Si se pasa más de un archivo, el nombre del archivo aparece delante de cada línea.

[expresión] es una expresión regular, y debe ir entre comillas “expresión”.

Sintaxis: `grep [-chinsvLRwxABC] [expresión] [archivo(s)]`

Opciones	Descripción
-v	Despliega las líneas que no concuerdan con la expresión.
-c	Sólo despliega la cantidad de líneas que concuerdan con la expresión.
-l	Sólo despliega el nombre del archivo al que pertenecen las líneas que concuerdan con la expresión.
-h	Suprime el nombre de archivo en el despliegue.
-n	Numera las líneas que contienen la expresión.
-i	No diferencia entre mayúsculas y minúsculas.
-R	Recursivo, entra en directorios
-w	Busca por palabra
-x	Busca por línea
-A#	Muestra la coincidencia mas # líneas siguientes
-B#	Muestra la coincidencia mas # líneas anteriores
-C#	Muestra la coincidencia mas # anteriores y siguientes

Ejemplos

Obtener las líneas del archivo que contienen la palabra user

```
grep user /etc/group
```

Obtener las líneas del archivo que comienzan con la palabra user.

```
grep "^user" /etc/group
```

Listado de los archivos que contienen user en todos los archivos del directorio actual.

```
cd /etc
grep -l user *
```

Obtener los procesos de user1, comienza con letras y termina en uno.

```
ps aux|grep "^[a-z][a-z]*1"
```

De cualquier usuario que termine en dígito

```
ps aux|grep "^[a-z][a-z]*[0-9]"
```

Todos los usuarios que no termina en dígito

```
ps aux|grep -v '^[a-z][a-z]*[0-9][ ][ ]*'
ps aux|grep '^[a-z][a-z]*^[0-9][ ][ ]*'
```

Obtener los números de proceso de un dígito

```
ps aux|grep '^[a-z][a-z]*[ ][ ][0-9][.][0-9].*'
```

Obtener todos los procesos ssh

```
ps aux|grep ssh|grep -v grep
```

Buscar todos los archivos que contienen la línea disable = yes

```
grep "disable.*=.*yes" /etc/xinetd.d
```

Obtener los procesos de uno a tres dígitos

```
ps -aux|grep '^[a-z][a-z]*[ ][ ][0-9]\{1,3\}[ ][ ][0-9][.][0-9].*'
```

Buscar en el archivo /etc/passwd tres números consecutivos.

```
grep "[0-9][0-9][0-9]" /etc/passwd
```

```
grep "[0-9]\{3\}" /etc/passwd
```

Expresiones regulares extendidas.

El comando grep buscará que la expresión coincida con A o con B.

```
grep -E '(A | B)'
```

Coincide con "ALFA" o con "BETA" o con "GAMA"

```
grep -E '(AFA|BETA|GAMA)'
```

Ejemplos

Buscar el nombre "root" o "user" en el archivo

```
grep -E '(root|user)' /etc/group
```

Obtener los procesos ssh o login

```
ps aux|grep -E '(ssh|login)' |grep -v grep
```

Procesos del usuario con el pid 2100 al 2199

```
ps aux |grep -E "^[a-z]+[ ][ ]+21[0-9]{2}"
```

Sin la opción -E hay que escapar con \

```
ps aux |grep "^[a-z]\\+[ ][ ]+21[0-9]\\{2\\}"
```

```
cat unexp.txt |grep "(hola|ntp)"
```

```
cat unexp.txt |grep -E "(hola|ntp)"
```

Obtener las líneas que comienzan con la letra a o con la u

```
cat passwd|grep -E ' (^a|^u) '
```

Comando egrep

A pesar del origen del nombre (extendido), egrep en realidad tiene menos funcionalidades, ya que está diseñado para ser compatible con el egrep tradicional. Una mejor manera de hacer un largo "grep" es usar grep -E que utiliza la sintaxis de expresiones regulares extendidas, sin pérdida de

funcionalidad.

egrep es el equivalente grep -E

Comando fgrep

fgrep es el equivalente grep -F

Comando rgrep

rgrep es el equivalente grep -r

Equivalencias

grep	grep -E	Disponible en egrep?
a\+	a+	yes
a\?	a?	yes
expression1\ expression2	expresion1 expresion2	yes
\(expresión\)	(expresión)	yes
\{m,n\}	{m,n}	no
\{,n\}	{,n}	no
\{m, }	{m, }	no
\{m}	{m}	no

Manejo del contenido de los archivos (Filtros)

Los filtros realizan operaciones sencillas sobre archivos. La potencia reside en su combinación.

Comando cut

Se utiliza para seleccionar columnas (opción -c) o campos (opción -f) de un archivo.

Sintaxis: `cut -c lista [archivo(s)]`

Sintaxis: `cut -f lista [-d char] [-s] [archivo(s)]`

Opción	Descripción
-c lista	se refiere a una selección de columnas. lista es una lista creciente de enteros separados por comas, y es posible utilizar el - para indicar rangos.
-f lista	se refiere a una selección de campos. lista es una lista de campos.
-d	El delimitador de campo por defecto es el tabulador, y éste se puede especificar con la opción
-s	Las líneas que no contienen el delimitador se devuelven como tales , a menos que se utilice la opción.

lista	Descripción
N	El carácter número
N-	a partir del número de carácter hasta el final
N-M	un rango
-M	desde el principio hasta el número

Ejemplos

```
> cat archivo
Esto es una prueba
del comando cut
> cut -c1,3 archivo
Et
dl
> cut -d " " -f2 archivo
es
comando
> cat arch
Hola Chau
Prueba1  Prueba2  (separados por tab)
> cat arch | cut -f2
Chau
Prueba2
> cut -f1,3 -d: /etc/passwd
> cut -f1-3 -d: /etc/passwd
```

Comando tr

Copia de la entrada estándar a la salida estándar con sustitución o borrado de caracteres

seleccionados. Los caracteres de la entrada encontrados en string1 son mapeados con el correspondiente carácter del string2.

Sintaxis: `tr [-dsc][string1 [string2]]`

Opciones	Descripción
-d	Borra todas las ocurrencias de string1 de la entrada.
-s	Cuando el mismo carácter de string1 se repite varias veces consecutivamente, lo sustituye por una sola ocurrencia del mismo.
-c	Hace que se traduzcan todos los caracteres que no se encuentren especificados en el primer parámetro por el segundo parámetro.

Ejemplo

```
head /etc/group |tr [a-z] [A-Z] #convierte a mayúsculas
head /etc/group |tr -s ":" "\t" #cambia : por tabulador
ll |tail +2|tr -d " " #elimina todos los espacios
```

```
echo "--HOLA-----MUNDO "|tr -s "-"
-HOLA-MUNDO
echo "--HOLA-----MUNDO "|tr -s "-" ":"
:HOLA:MUNDO
echo "--HOLA----__---MUNDO "|tr -d "\_"
HOLAMUNDO
echo "Welcome aagg to Linux dd" | tr -d agd
"Welcome to Linux "
echo "Hola amigo mio" | tr 'oi' 'ap'
Hala ampga mpa
```

Crear un archivo con palabras y .;()"

```
cat ejer.txt |tr -s " " "\n" |tr -d ".;,( )\""
```

Todas las palabras en mayúsculas

```
cat ejer.txt |tr -s " " "\n" |tr -d ".;,( )\""|tr [a-z] [A-Z]
```

Todo lo que no sea una letra se cambia por el guion.

```
cat /etc/passwd |tr -c "[a-z]" "-"
```

Comando expand

Convierte tabulaciones por espacios en blanco en un archivo, o la entrada estándar.

Opción	Descripción
-i, --initial	convierte al inicio de línea
-t,	cantidad de espacios por defecto 8
--tabs=NUMBER	

Ejemplos:


```
[root]# echo -e "\tHOLA\tMUNDO"|cat -A
^IHOLA^IMUNDO$
[root]# echo -e "\tHOLA\tMUNDO"| expand |cat -A
HOLA      MUNDO$
[root]# echo -e "\tHOLA\tMUNDO"|expand -i |cat -A
HOLA^IMUNDO$
[root]# echo -e "\tHOLA\tMUNDO"|expand -it1 |cat -A
HOLA^IMUNDO$
[root]# echo -e "\tHOLA\tMUNDO"|expand -t1 |cat -A
HOLA MUNDO$
```

Comando head

Retorna la primera n líneas del archivo especificado. Por defecto retorna las 10 primeras

Sintaxis: head [-c|n|q|v] [archivo]

Opción	Descripción
-c nro	Despliega la cantidad de caracteres especificado por nro
-n nro	Despliega las primeras líneas especificadas por nro
-q	No despliega la cabecera
-v	Muestra la cabecera

En todos los casos si no se especifica el archivo, se asume la entrada estándar.

Ejemplos

```
head /etc/group
head -n 5 /etc/passwd /etc/group
```

Comando wc

Cuenta la cantidad de caracteres, palabras o líneas de un archivo.

sintaxis: wc [-cwlL] file

Opciones	Descripción
-C	cuenta caracteres
-w	cuenta palabras
-l	cuenta líneas
-L	El tamaño de la línea mas larga

Ejemplo

```
>wc -l /etc/hosts
  11 /etc/hosts
>wc /etc/hosts
  11      37      342 /etc/hosts
>wc << EOF
> Hello
> There are
> Four lines
> I think
> EOF
4 7 35
```

Ejercicio

1. Obtener cuantos archivos tiene el directorio /etc/samba
2. Guardar el resultado en una variable.
3. Cual es la línea mas larga del archivo /etc/passwd

Comando tail

Muestra las últimas líneas o caracteres de un archivo.

Sintaxis: `tail [-nro|+nro] [l|c|q|v|n] [archivo]`

Sintaxis: `tail -f [archivo]`

Opciones	Descripción
-nro	Retorna las últimas número líneas del archivo especificado. Por defecto asume 10
-n +nro	Muestra desde la línea número hasta el final del archivo.
-l	Líneas: es la opción por defecto.
-c nro	Muestra los últimos caracteres dados por nro
-f	deja abierto el archivo mostrando las modificaciones del archivo, Ejemplo: ver <code>tail -f /var/log/messages</code>
-q	No muestra la cabecera
-v	Muestra la cabecera
-n nro	Es equivalente a -nro

Ejemplo

```
>tail -n 1 /etc/passwd /etc/group
```

```
==> /etc/passwd <==
```

```
user2:x:503:10000:::/bin/bash
```

```
==> /etc/group <==
```

```
user2:x:10000:
```

Ejercicio

1. Muestre a partir de la línea 15 al final de los archivos /etc/passwd y /etc/group sin la cabecera.
2. Muestre las ultimas 5 líneas del archivo /etc/group con la cabecera.

Comando join

El comando join trabaja con dos archivos, realiza la fusión en columnas, en base a un campo en común.

Sintaxis: `join [aivt] [-File1Campo] [-File2Campo] file1 file2`

Opciones	descripción
-a[1 2]	además muestra las líneas no coincidentes, del primer o segundo archivo
-v[1 2]	solo muestra las líneas no coincidentes, del primer o segundo archivo
-t	delimitador, por defecto es el espacio.

-File1Campo	número de campo del primer archivo
-File2Campo	número de campo del segundo archivo.

Ejemplo, cree los archivos filedatos y fileprecio.

```
>more filedatos
```

```
100 Shoes
```

```
200 Laces
```

```
300 Socks
```

```
>more fileprecio
```

```
100 $40.00
```

```
200 $1.00
```

```
300 $2.00
```

```
> join -11 -21 filedatos fileprecio
```

```
100 Shoes $40.00
```

```
200 Laces $1.00
```

```
300 Socks $2.00
```

Ejemplos

```
>join -t: -14 -23 <(head -5 /etc/passwd) <(head -5 /etc/group)
```

```
0:root:x:0:root:/root:/bin/bash:root:x:root
```

```
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:root,bin,daemon
```

```
2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:root,bin,daemon
```

```
4:adm:x:3:adm:/var/adm:/sbin/nologin:adm:x:root,adm,daemon
```

```
>join -a1 -t: -14 -23 <(head -5 /etc/passwd) <(head -5 /etc/group)
```

```
0:root:x:0:root:/root:/bin/bash:root:x:root
```

```
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:root,bin,daemon
```

```
2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:root,bin,daemon
```

```
4:adm:x:3:adm:/var/adm:/sbin/nologin:adm:x:root,adm,daemon
```

```
7:lp:x:4:lp:/var/spool/lpd:/sbin/nologin:
```

```
>join -v1 -t: -14 -23 <(head -5 /etc/passwd) <(head -5 /etc/group)
```

```
7:lp:x:4:lp:/var/spool/lpd:/sbin/nologin:
```

Comando nl

Numera las líneas de un archivo, por defecto las que no están en blanco.

Sintáxis: nl [-b[a|n|t|p] [-n[ln|rn|rz] [-i] [-s string] file

Opciones	Descripción
-b	A quienes numera:
-ba	Numera a todas, incluso las que están en blanco.
-bn	Ninguna, inserta línea en blanco.
-bt	Las que no están en blanco, es la opción por defecto
-	Numera las líneas que contienen el string .
bpstring	
-n	formato de la numeración

-nln	Alinea a la izquierda
-nrn	Alinea a la derecha, por defecto
-nrz	Alinea a la derecha, justificado
-in	Incremento, por defecto n es uno.
-s string	Con esta opción se agrega el string a la salida numerada del archivo.

Ejemplo:

```
nl -s :usuarios: /etc/passwd
nl -bproot /etc/passwd
```

Comando od

Muestra el contenido de un archivo o de la entrada estándar, en octal y otros formatos. Por defecto trabaja en octal.

sintaxis: `od [-A|j|N|s|t|w] archivo`

Opciones	detalle
-A base	Tipo de numeración base: d decimal o octal x hexadecimal n ninguno
-j BYTES	Saltea en la entrada la cantidad de BYTES
-N BYTES	El máximo de BYTES para mostrar
-s [N]	Cantidad de bytes a mostrar, por defecto son tres.
-t TYPE	Especifica como se mostrará la salida TYPE: a caracteres c ASCII d decimal f punto flotante o octal x hexadecimal
-w BYTES	Cantidad de bytes por línea

```
>echo Hello World|od
0000000 062510 066154 020157 067527 066162 005144
0000014
> echo Hello World| od -t c
0000000 H e l l o W o r l d \n
0000014
> echo Hello World|od -t dlc
0000000 72 101 108 108 111 32 87 111 114 108 100 10
H e l l o W o r l d \n
0000014
> echo Hello World|od -t dlcx1
0000000 72 101 108 108 111 32 87 111 114 108 100 10
```

```
H e l l o W o r l d \n
48 65 6c 6c 6f 20 57 6f 72 6c 64 0a
0000014
```

Comando hexdump

Muestra el contenido de un archivo o de la entrada estándar en hexadecimal por defecto, o en otros formatos.

Sintaxis: `hexdump [b|c|C|d|o|v|x] archivo`

Opciones	descripción
-b	octal, un byte
-c	caracteres
-C	Hexadecimal y caracteres
-d	decimal dos bytes
-o	octal, dos bytes
-x	hexadecimal

Ejemplo

```
>echo hola mundo |hexdump -C
00000000 68 6f 6c 61 20 6d 75 6e 64 6f 0a |hola mundo.|
0000000b
```

Comando paste

Produce la salida de varios archivos en columnas, una columna por archivo.

Sino se especifica un delimitador se asume tab.

Sintaxis: `paste [-d] file1 file2`

Ejemplos

```
paste -d- <(head -5 /etc/passwd) <(head -5 /etc/group)
```

Comando sort

Este comando ordena o fusiona archivos.

Sintaxis: `sort [-cmufnrbdk] [-o archivo] [archivo(s)]`

Opción	Descripción
-c	Comprueba si el archivo está ordenado.
-u	Elimina las líneas duplicadas.
-f	No diferencia entre mayúsculas y minúsculas.
-n	Ordena los campos como si la clave fuera numérica.
-r	Invierte el orden de la clasificación.

-b	Ignora espacios en blanco y tabuladores al principio. de la línea.
-d	Orden de diccionario.
-o	Almacena la salida en el archivo especificado.
-k	Por campo, k#

Un campo es una cadena no vacía, sin blancos, separada de otras cadenas por espacios en blanco.

Ejemplos

```
head /etc/group|sort
ll |sort -nk5      #ordena por tamaño ascendente
ll |sort -nk5 -r   #ordena por tamaño descendente
sort -nk 5 <(ll /tmp) <(ll /boot) #ordena los dos directorios
sort <(head -3 /etc/passwd) <(head -3 /etc/group) -o salida
cat salida |sort -c #no devuelve nada si esta ordenado
cat /etc/hosts |sort -c #sino esta ordenado devuelve la primera
línea que no cumple:
sort: -:3: fuera de secuencia: 127.0.0.1
```

Comando uniq

Lee de entrada (por defecto se asume la entrada estándar) comparando las líneas consecutivas. Las líneas consecutivas repetidas se sustituyen por una sola al colocarlos en salida (que por defecto es la salida estándar).

Sintaxis: `uniq [-ducwi] [entrada [salida]]`

Opciones	Descripción
-d	solo muestra las repetidas
-u	solo las líneas únicas
-c	cuantas veces aparece
-w	especifica cuantos caracteres se van a comparar
-i	ignora mayúsculas minúsculas

Ejemplo

Ordenación y eliminación de líneas repetidas

```
cat <(head -3 /etc/group) <(head -3 /etc/group) |sort|uniq
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
root:x:0:root
```

Muestra cuantas veces aparece cada línea.

```
cat <(head -3 /etc/group) <(head -1 /etc/group) |sort|uniq -c
  1 bin:x:1:root,bin,daemon
  1 daemon:x:2:root,bin,daemon
  2 root:x:0:root
```

Muestra solo las líneas únicas:

```
cat <(head -3 /etc/group) <(head -5 /etc/group) |sort|uniq -u
adm:x:4:root,adm,daemon
```

```
sys:x:3:root,bin,adm
```

Compara solo el primer carácter:

```
head /etc/group |sort |uniq -w1 -c |nl
1      1 adm:x:4:root,adm,daemon
2      1 bin:x:1:root,bin,daemon
3      2 daemon:x:2:root,bin,daemon
4      1 kmem:x:9:
5      1 lp:x:7:daemon,lp
6      1 mem:x:8:
7      1 root:x:0:root
8      1 sys:x:3:root,bin,adm
9      1 tty:x:5:
```

Comando split

Forma varios archivos a partir de uno. Partiéndolo según un tamaño dado, no se modifica el original.

Sintaxis: `split -[bcla] [archivo] [prefijo]`

Opciones	descripción
prefijo	El prefijo por defecto es `x'.
-a	utiliza sufijos de longitud N (por omisión 2)
-b BYTES	escribe BYTES bytes en cada fichero de salida BYTES puede tener un factor indicado con el sufijo: b para 512, k para 1K, m para 1Meg, G gigabyte
-C bytes	escribe un máximo de BYTES bytes sin cortar líneas
-l rno.	pone rno. de líneas en cada fichero de salida, por defecto asume 1000 líneas.

Ejemplos

```
> split -b 3000 sm56
> ll
total 20
-rwxr-xr-x  1 root    root          6637 jun 28  2004 sm56
-rw-r--r--  1 root    root          3000 sep  5  14:07 xaa
-rw-r--r--  1 root    root          3000 sep  5  14:07 xab
-rw-r--r--  1 root    root           637 sep  5  14:07 xac
>split -b 1k sm56setup -a3 r
>ll
-rw-r--r--  1 root    root      1024 abr 12  19:05 raaa
-rw-r--r--  1 root    root       831 abr 12  19:05 raab
-rwxr-xr-x  1 root    root      1855 abr 12  19:01 sm56setup

>split -a 3 -b 1G movie.iso W
>ls -lh
-rw-r--r--  1 solange solange 2,5G 2011-09-18 12:32 movie.iso
-rw-r--r--  1 solange solange 1,0G 2011-09-23 10:46 Waaa
-rw-r--r--  1 solange solange 1,0G 2011-09-23 10:47 Waab
```

```
-rw-r--r--  1 solange solange 491M 2011-09-23 10:47 Waac
```

Un archivo particionado se arma nuevamente utilizando el comando cat.
cat raab >>raaa

Comando md5sum

Comprobación de la integridad de los archivos con el comando md5sum:
md5sum raaa sm56setup
96802e303c1bcdccc6aed576d9880ea6 raaa
96802e303c1bcdccc6aed576d9880ea6 sm56setup

Puede utilizar cat xa* >file si tiene muchos archivos.

Comando unexpand

Convierte espacios en blanco en tabuladores. Inverso al comando expand.

Opciones	Descripción
-a	convierte todos los blancos en un solo tabulador, es lo mismo que el comando tr -s " " "\t"
-t	para especificar la cantidad

Ejemplo

```
>echo "hola      mundo  linux"|cat -A  
hola      mundo  linux$
```

```
>echo "hola      mundo  linux"|unexpand -a  
hola      mundo  linux
```

```
>echo "hola      mundo  linux"|unexpand -a|cat -A  
hola^I mundo^I linux$
```


Editores de texto

vi - vim

El vim es un editor interactivo usado para editar archivos de texto. Utiliza la pantalla. Todas las modificaciones se hacen a través de un buffer.

Los cambios en el buffer pueden hacerse permanentes o pueden desecharse.

Como invocarlo:

- > vim archivo
- > vim +nn archivo #para comenzar la edición en la línea nn.
- > vim +/string archivo #se posiciona donde localiza string
- > vim +set number archivo #activa la numeración de líneas

Archivo .exrc

Se puede generar el archivo ~/.exrc para personalizar el comportamiento del vim, por ejemplo puede contener:

```
set number
```

Cada vez que se ejecute el vim la numeración estará activada.

Invocar los diferentes modos:

ESC	activa el modo normal
i	activa modo inserción
ESC:	activa modo inserción de comandos
v, V	modo visual

Terminan la edición

ESC :wq!	termina y guarda los cambios
ESC :q!	Termina sin guardar los cambios
ZZ	termina y guarda los cambios
ESC :x!	termina y guarda los cambios
ESC :e!	Volver a la ultima version guardada
ESC:w file	Guardar con otro nombre el archivo

Modo inserción

i	antes del cursor
a	después del cursor
A	al fin de la línea
O	línea siguiente del cursos
O	línea anterior del cursor

Ejecutar un comando del bash en el editor






ESC: ! cmd

Se muestra la salida del comando en el editor.

ESC: r !cmd

Se ejecuta el comando y la salida se inserta en la posición del cursor.

Referencia:

Teclas			Función
h	or		Cursor left
l	or		Cursor right.
k	or		Cursor up.
j	or		Cursor down.
	B		Cursor left one word.
	w		Cursor right one word.
	{		Cursor up one paragraph.
	}		Cursor down one paragraph.
	^		Cursor to line start.
	\$		Cursor to line end.
	gg		Cursor to first line.
	G		Cursor to last line.
			Get out of current mode.
	i		Start insert mode.
	O		Insert a blank line below the current line and then start insert mode.
	o		Insert a blank line above the current line and then start insert mode.
	a		Append (start insert mode after the current character).
	R		Replace (start insert mode with overwrite).
	:wq		Save (write) and quit.
	:q		Quit.
	:q!		Quit forced (without checking whether a save is required).
	x		Delete (delete under cursor and copy to register).
	X		Backspace (delete left of cursor and copy to register).
	dd		Delete line (and copy to register).
	:j!		Join line (remove newline at end of current line).
	Ctrl-J		Same.
	u		Undo.
	Ctrl-R		Redo.
	de		Delete to word end (and copy to register).
	db		Delete to word start (and copy to register).
	d\$		Delete to line end (and copy to register).
	d^		Delete to line beginning (and copy to register).
	dd		Delete current line (and copy to register).
	2dd		Delete two lines (and copy to register).
	5dd		Delete five lines (and copy to register).
	p		Paste clipboard (insert register).
	Ctrl-G		Show cursor position.
	5G		Cursor to line five.
	16G		Cursor to line sixteen.
	G		Cursor to last line.
	/search-string		Search forwards for <i>search-string</i> .

<code>?search-string</code>	Search backwards for <i>search-string</i> .
<code>:-1,\$s/search-string /replace-string /gc</code>	Search and replace with confirmation starting at current line.
<code>:\$s/search-string /replace-string /gc</code>	Search and replace with confirmation starting at line below cursor.
<code>:\$s/\<search-string \>/replace-string /gc</code>	Search and replace whole words.
<code>:8,22s/search-string /replace-string /g</code>	Search and replace in lines 8 through 22 without confirmation.
<code>:%s/search-string /replace-string /g</code>	Search and replace whole file without confirmation.
<code>:w filename</code>	Save to file <i>filename</i> .
<code>:5,20w filename</code>	Save lines 5 through 20 to file <i>filename</i> (use Ctrl-G to get line numbers if needed).
<code>:5,\$w! filename</code>	Force save lines 5 through to last line to file <i>filename</i> .
<code>:r filename</code>	Insert file <i>filename</i> .
<code>V</code>	Visual mode (start highlighting).
<code>Y</code>	Copy highlighted text to register.
<code>D</code>	Delete highlighted text (and copy to register).
<code>P</code>	Paste clipboard (insert register).
Press v, then move cursor down a few lines, then,	Search and replace within highlighted text.
<code>:s/search-string /replace-string /g</code>	
<code>:help</code>	Reference manual
<code>:new</code>	Open new blank window.
<code>:split filename</code>	Open new window with <i>filename</i> .
<code>:q</code>	Close current window.
<code>:qa</code>	Close all windows.
<code>Ctrl-W j</code>	Move cursor to window below.
<code>Ctrl-W k</code>	Move cursor to window above.
<code>Ctrl-W -</code>	Make window smaller.
<code>Ctrl-W +</code>	Make window larger.

Ejercicio vim

- 1) Copiar el archivo /etc/group a /tmp/file.txt. Editar el archivo /tmp/file.txt con el vim.
 - a) Buscar la palabra root.
 - b) Apagar el resultado con el comando: nohlsearch
 - c) Sustituir todas las apariciones de la palabra root por pepe
 - d) Terminar la edición sin guardar los cambios.
- 2) Abrir con el vim activando la numeración de las líneas, el archivo anterior.
 - a) Copiar las cinco primeras líneas al final.
 - b) Eliminar de la línea 7 a la 10.
 - c) Salir guardando los cambios.
- 3) Generar el archivo del perfil personal del vim, para que automáticamente numere las líneas.
- 4) Editar el archivo /tmp/file.txt
 - a) Eliminar las ultimas 10 líneas
 - b) Guardar el archivo con el nombre de copia.txt
 - c) Salir del vim.
- 5) Realizar la comparación de los archivos con el comando paste.