COMANDOS EN LINUX

Ap Solange Mikeliunas
Agosto 2010

Índice de contenido

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.
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

Comandos uso general

Comando pwd

```
sintaxis: pwd
```

Muestra el directorio actual

Comando echo

```
sintaxis: echo [-ne]
```

Escribe los argumentos separados por blancos y terminados en un nueva línea en la salida estándar.

Opciones	Descripción
-n	sin salto de línea
-e	activa la interpretación de caracteres de control.

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 `Crl + 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
N° 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.

```
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 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
```

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

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.

Comando history
Muestra los comandos ingresados en la consola.
`sintaxis: history [nro | c ]`
Ejemplo
```
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:
Muestra las últimas 10 líneas
`fc –l`
Busca en la historia por el `string` y muestra desde la coincidencia hasta el final.
`fc -l string #`

Muestra desde el comando Nro1 hasta el comando Nro2
`fc -l Nro1 Nro2`

Edita desde el comando que coincida con el string
`fc -n string #`

Edita desde el comando Nro1 hasta el comando Nro2.
`fc -n Nro1 Nro2`

Teclas para la búsqueda en la historia

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

`man comando`	Para consultar sobre un comando
`man –a comando`	Para consultar todas las páginas existentes sobre un comando
`man –k [clave]`	Busca la clave en la descripcion de las paginas man, que se encuntra en la base de datos de whatis.
`man n comando`	Para consultar sobre una sección de ayuda, del 1 al 9
`man -K [clave]`	Busca la clave en todas las paginas man.
`man -f comando`	Descripcion del comando.
`man -w comando`	Devuelve la localizacion 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 /dev)
5	File formats and conventions eg. /etc/passwd
6	Games
7	Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
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`

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.

Facilidades del shell:

Metacaracteres

Los metacaracteres son caracteres con significado especial.
En general se utilizan para sustituciones.
Ejemplos:
; $ > ~ * ? [ ] <

Es conveniente NO usarlos en nombres de archivos (usar . y _).
Hay básicamente tres clases de metacaracteres:

1. para construcción de patterns: * ? [ ]
2. para combinación de comandos: ; & | || &&
3. para redireccionamiento: > < >>

Construcción de patterns

Antes de la ejecución de un comando, el shell busca los caracteres * ? [ ] en los parámetros del comando. Si alguno de ellos aparece la palabra que lo contiene es vista como un pattern.
Cada pattern se reemplaza por los nombres de archivos del directorio actual (por orden alfabético) que coincidan con él (pattern matching).
Si ningún archivo del directorio actual coincide con el pattern se deja la palabra original sin modificar.
Los patterns los resuelve el shell, los comandos sólo deben estar preparados para recibir una cantidad variable de nombres como parámetro.

Construcción de patterns: reglas

Carácter	Significado
*	Cualquier string, incluso el nulo
?	Un carácter cualquiera
[...]	Cualquiera de los caracteres dentro de los paréntesis rectos
[c1-c2]	Cualquier carácter entre c1 y c2
[!c1-c2]	Complemento de [c1-c2]
[[:upper:]]	mayúsculas
[[:lower:]]	minúsculas
[0-9]	Digito
{string1,string2}	Coincide con string1 o string2

Comandos para manejo de archivos y directorios

Manipulación de directorios
Comandos relativos a manejo de directorios:

- pwd
  Muestra el directorio actual.
- cd directorio
  Para cambiar el directorio actual.
- mkdir directorio
  Crea directorios.
- rmdir directorio
  Borra directorios vacíos.
- ls directorio
  Lista el contenido de un directorio
- tree
  Muestra la estructura de directorios

Caminos (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/usr1/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/usr1/textos/texto1`
   `../textos/texto1`

Abreviaturas para algunos nombres de caminos:

.	Directorio actual
..	Padre del directorio actual
~	Camino absoluto al home directory
~user	Al home del usuario user

Ejemplo:

```
> 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] [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
```

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
Ejemplo:

```
> cd
> ls
archivos  textos  personal mails
> ls archivos
arch1 arch2 largos
> ls archivos/largos/ejemplos.del.curso
```

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. (ls –d */)
-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).

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 del 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

```
Liste los archivos de /var/log que terminan en 1 o en 2.
Liste los archivos de /var/log que comienzan con boot o con
mail.
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 [-augdfp]

Opciones	Descripción
-a	all
-d	directorios
-f	camino total
-u	dueño
-g	grupo
-p	permisos

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 modificacion
-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

> stat xrdb.txt
  File: `xrdb.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

Modificación de la fecha de modificación: 2006/01/02 10:30
>touch -m -t 200601021030 xrdb.txt
>stat xrdb.txt
  File: `xrdb.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
```

Para ver el tipo de archivo se puede usar el comando *file.*
Sintaxis: file Nombre_archivo.

```
 [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] 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	Recursivo: 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.

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 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.
Ejemplo:

```
cp --parents /etc/shadows /home/usr1
```

resultado:

```
/home/usr1/etc/shadow
```

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, sólo se copia el contenido del archivo. Esto puede llevar a inconsistencias.

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 sobreescribir cualquier archivo o subdirectorio.

Sintaxis: mv [-fi] 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, el comportamiento es el mismo que en el caso que sigue.

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 [-fir] 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 `rm -r` es un directorio, se borra su contenido, y recursivamente el de todos los subdirectorios que este contenga
`-v`	Explica lo que esta haciendo

Ejemplos:
```
cd /home/usr1
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.

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/usr3/.??*

Comandos para ver el contenido de un archivo

Comando more

Sintaxis: more [archivo(s)]

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.
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

Sintaxis: less [archivo(s)]

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

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 edicion.
> 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 edicion.

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

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

Comando tac

Como el comando cat lista un archivo pero en orden inverso.

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

```
>head /etc/group |fmt -w 40
root:x:0:root bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon tty:x:5:
disk:x:6:root lp:x:7:daemon,lp mem:x:8:
kmem:x:9:
```

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 condición

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
Name	nombre de archivo
Type	tipo: b archivo especial de bloques d directorio c archivo de caracteres f archivo
size nro	tamaño en bloques del archivo buscado.
atime nro	día del último acceso al archivo
mtime nro	día de la última modificación.
User	del usuario
group	del grupo
newer	archivo Archivos modificados más recientemente que archivo.
print	Muestra en la salida estándar el camino donde se encontró el archivo. Por defecto
exec comando	busca los archivos y exec ejecuta el comando
Mmin	Minutos, mmin + 5 hace mas de 5; mmin –10 hace menos de 10
perm –perm	Busca por un permiso
nouser	Sin usuario propietario
nogroup	Sin grupo propietario
maxdepth	Niveles, en 1 busca en primer nivel

Ejemplos:

```
find /internet -name netscape –print
find -name "file*" -exec ls -l "{}"  ";"
find  /var -user "pepe" -name "d*"
find  /var -user "pepe" -name "d*"
find  /var -perm -2  -type f #busca con permiso de
escritura
find  /var –perm 0755 –type d #por el permiso
find  /var -nouser #sin usuario propietario
find  /var -nouser -nogroup #sin usuario ni grupo
propietario
find  /etc –maxdepth 1 –name "a*"
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
```

Comando locate

Sintaxis locate nombre

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.

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
```

Manejo con la salidas/entrada standard

Redireccionamiento

Toda operación dentro de un proceso tiene una entrada y una salida (I/O) y en algunas ocasiones una salida de error.
Unix permite que la entrada y la salida de los comandos sea redireccionada.
Entrada: se envían datos a un comando.
Salida: recibe datos 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
```

Redireccionar el error
Ejemplo
```
ls  archivo 2>/dev/null
ls  archivo 2>>file.error
```

Redireccionar la salida y el error
Ejemplo
```
ls archivo 1>/dev/null 2>&1
ls  >>file  2>&1
ls >file 2>file.error
ls     2>>file   >>file2
ls  &>file
ls  >& file
```

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.

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
```

Combinación de comandos

Comandos simples:
Son secuencias de palabras separadas por espacios: la primera es el nombre del comando el resto son los parámetros
Ejemplo:

```
ls
who am i
man id
```

Pipes y pipelines:
Un pipe permite enviar información de un proceso a otro. Conecta la stdout del primero con la stdin del segundo.
Los procesos comunicados se ejecutan al mismo tiempo: en cuanto el primero deja datos en el pipe el segundo los toma.
Un pipeline es la ejecución simultánea de 2 o más comandos simples comunicados por un pipe. Sintácticamente se especifica por medio de un "|":

comando | *comando*| *comando ...*

Los comandos se ejecutan en paralelo, cada uno asociado a un proceso separado. Pasan sus datos a través de un buffer.
El exit status de un pipeline es el exit status del último comando.

Parentizado:
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/usr1
> 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
```

Parentizado, otra forma :
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.

Ejemplos:

```
$ { date ; who ; } | lp
```

En la impresora sale:

```
Tue Mar 28  10:20:20 WST 1997
usr1 ttyp0     Mar 28  10:00
```

Ejemplo

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

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 comado1 y si la ejecución no es exitosa se ejecuta comando2.

```
comando1 || comando2
```

Se ejecuta comando1 y si la ejecución es exitosa se ejecuta comando2.

```
comando1 && comando2
```

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

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

Ejemplos

```
ls -l | more ; date ; who
cat archivo || echo "El archivo no existe"
cat archivo && echo "Fin del archivo"
cat archivo && echo OK || echo MAL
find –name file 1>/dev/null 2>&1 && echo "Existe"|| echo "No"
```

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.

Substitución de comando

La secuencia **$(comando)** ejecuta el comando y permite tomar el valor devuelto por otro comando.

Ejemplo

```
ls -la /lib/modules/$(uname -r)/*
kill -9 $(ps aux|tr -s " " "\t"|cut -f1,2|grep usr2|cut -f2)
```

La versión anterior usaba las comillas.
Ejemplo

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

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

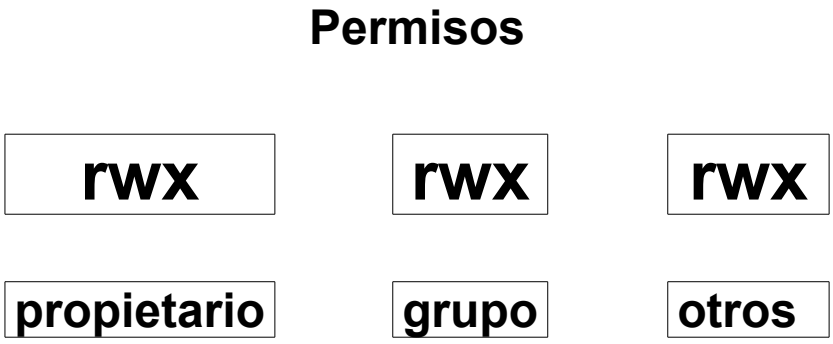
Para desactivar una alias.

sintaxis: unalias nombrealias

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.



r	read, leer
w	write, escribir
x	execute, ejecutar
-	sin permisos

Permisos según el tipo de elemento:

Para un Plain File:
- 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.

Cambio de permisos

Comando chmod

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
>ll httpd.conf
---------- 1 smikeliu smikeliu 31050 mar 3 2005 httpd.conf

Opciones de chmod

```
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

Tabla octal

Valor numérico	*Permisos*
0	---------
1	--------x
2	-------w-
3	-------wx
4	------r—
5	------r-x
6	------rw-
7	------rwx
10	-----x---11
11	-----x—x
22	-----w--w-
33	----wx-wx
55	---r-xr-x
77	---rwxrwx
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-xr--r—
644	rw-r--r—
666	rw-rw-rw
755	rwxr-xr-x
777	rwxrwxrwx

Ejercicio

¿Cuál de los siguientes permisos se representa por el valor numérico 44?

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

Comando umask
Modificar los valores de los permisos por defecto
`sintaxis umask [-pS]`

El comando umask muestra los permisos por defecto, y permite modificarlos, esta variable del sistema se setea en el arranque del sistema.
Ejemplo:
`umask 002`
significa: 775
`umask 022`
significa 755

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.

Archivos	***Directorios***
022 ----w--w-	022 ----w--w-
644-rw-r—r-	755 dwxr-xr-x

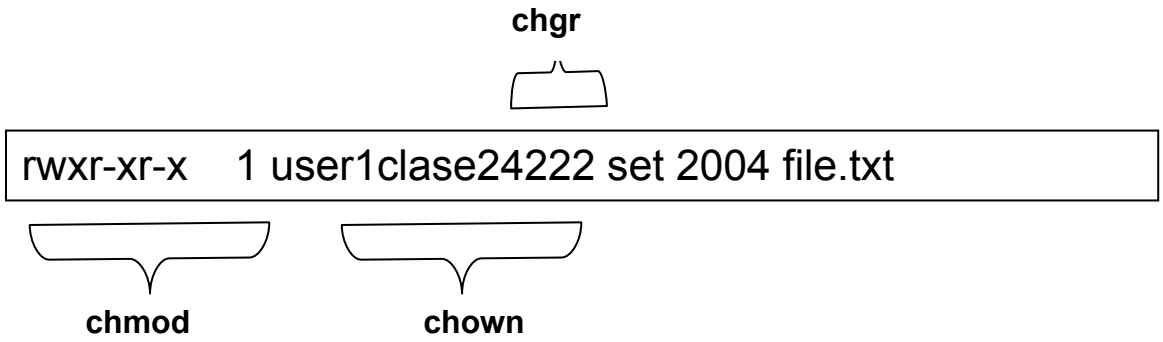
Comando chown
Cambiar el usuario o grupo propietario de un archivo o directorio.
```
sintaxis  chown usuario:grupo archivo
sintaxis  chown usuario.grupo archivo
```
Ejemplos:
Asignar el usuario y grupo
`chown usr1:usr1 file`

Modificar solo el grupo
`chown :usr1 file`

Modificar solo el usuario
`chown usr1: file`

Comando chgr
Modificar únicamente el grupo propietario
`sintaxis chgr grupo file`

A quien afecta cada comando



Permisos especiales

Asignar UID

Al activar este permiso el archivo 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. Ejemplo cambiar la contraseña de un usuario.

Asignar GID

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.

Asignar Sticky

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

valores		
4	2	1
s	s	T
SUID	SGID	sticky

Información de un archivo

Comando stat

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.

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
```

Ejercicio

¿Cuales serán los permisos del archivo ejecutable “portable” 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

Variables

Variables del entorno del sistema

Comando env
El comando env lista todas la variables del ambiente.

Variable $PATH
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` se ubica 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
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 el 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:hi
story:interactive-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*]

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
```

Encomillado

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

```
     > echo "mi home es $HOME"
          mi home es /home/usr1
     >echo 'mi home es $HOME'
          mi home es $HOME
     > echo "el contenido de arch es  `cat arch`"
          el contenido de arch es contenido de archecho
$HOME
     >echo $HOME
          /root
     >echo '$HOME'
          $HOME
     >echo "'$HOME'"
          '/root'
```

Las comillas “dobles” preservan el contenido de la variables.
Comillas 'simples' toman el contenido literal.
Las comillas `tilde` son el equivalente a $(comando) ejecutan el comando.

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,bqe,be,bccce,bqqe
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
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 pude repetirse desde 2 hasta cuatro veces.
[0-9]\{4\}	Es un dígito y se repite cuatro veces
[0-9]\{4,\}	Es un dígito y se repite cuatro veces o mas
[0-9]\{4\}[ ][.]	Es un dígito y se repite cuatro veces, luego un espacio en blanco, luego un punto.
[[:upper:]]\ {3\}	Son exactamente tres letras mayúsculas.
A[[:lower:]]\ {3\}	Comienza con la letra A, luego siguen exactamente tres letras minúsculas.
\<palabra\>	Exactamente una palabra

Comando grep

Sintaxis: grep [-chinsvlRwxABC] [expresión] [archivo(s)]

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 ‘‘ o “”.

:

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
```

Ejemplo búsqueda de texto:
Listado de los archivos que contienen `user` en todos los archivos del directorio actual.
```
cd /etc
grep -l user  *
```

Ejemplos
Obtener los procesos de usr1, comienza con letras y termina en uno.

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

De cualquier usuario que termine en digito
```
ps aux|grep ^[a-z][a-z]*[0-9]
```

Todos los usuarios que no termina en digito
```
ps aux|grep -v  '^[a-z][a-z]*[!0-9][ ]*[0-9]*'
```

Obtener los números de proceso de un digito
```
ps -aux|grep '^[a-z][a-z]*[ ]*[0-9][ ][ ][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 digitos
```
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.

```
grep –E ´(A | B)’
```

`grep –E ´(A | B)’`	Grep buscará que la expresión coincida con A o con B.
`grep –E ´(AFA|BETA|GAMA)’`	Coincide con “ALFA” o con “BETA” o con “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
```

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

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

Manejo del contenido de los archivos (Filtros)

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

Comando cut

Sintaxis: cut -clista [archivo(s)]
Sintaxis: cut -flista [-d char] [-s] [archivo(s)]

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

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`	carácter número, a partir del primero
`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

Sintaxis: tr [-ds][string1 [string2]]

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.

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.

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
```

Comando expand

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

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

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

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

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

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.

Comando wc

sintaxis: wc [-cwlL] file

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

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

Obtener cuantos archivos tiene el directorio /etc/samba

Guardar el resultado en una variable.

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
+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 tail -f /var/log/messages
-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 <==
usr2:x:503:10000:::/bin/bash

==> /etc/group <==
usr2:x:10000:
```

Ejercicio

Muestre a partir de la línea 15 al final de los archivos /etc/passwd y /etc/group sin la cabecera.
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:

```
>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 opcion por defecto
-bp**string**	Numera las líneas que contienen el **string**.
-n	formato de la numeración
-nln	Alinea a la izquierda
-nrn	Alinea a la derecha, por defecto
-nrz	Alinea a la derecha, justificado

-i**n**	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 d1c
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 d1cx1
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 estandar 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 linea que no cumple:
sort: -:3: fuera de secuencia: 127.0.0.1
```

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

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).

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
-C bytes	escribe un máximo de BYTES bytes sin cortar líneas
-l rno.	pone nro. de líneas en cada fichero de salida, por defecto asume 1000 líneas.

```
> 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
```

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$
```

Procesos

Generalidades.
Un proceso es un programa en ejecución.
Un programa es una entidad inanimada, el procesador trabaja sobre una instancia de éste, y esa entidad "viva" se denomina proceso (H. Deitel, Sistemas Operativos).
En general en el sistema habrán procesos de usuarios y del sistema ejecutándose en forma concurrente.
Cada proceso se identifica por un número: su PID.

Comando ps
El comando ps muestra información de los procesos activos.

Sintaxis. ps [-opciones]

Sin opciones despliega información acerca de los procesos de esa shell.
-e Imprime información de todos los procesos del sistema.
-f Listado completo.

La información desplegada es:

```
UID
PID
PPID (padre)
C (usado por el scheduller)
STIME (tiempo de arranque)
TTY
TIME
COMD
```

Consultar el resto de los modificadores en el man.

Comando kill

El comando kill envía una señal a un cierto proceso. El uso más común es para terminar un proceso corriendo en el sistema.

Sintaxis kill [-señal] PID(s)

La señal asumida por defecto es 15.
A veces los procesos se encuentran bloqueados esperando la ocurrencia de cierto evento (por ejemplo la disponibilidad de cierto recurso), en esos casos kill -9 puede ser más adecuado.

Editor de texto Vi

vi - vim
El vi 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 buffrer pueden hacerse permanentes o pueden desecharse.
Como invocarlo:

```
> vi archivo
> vi +nn archivo #para comenzar la edición en la línea nn.
```

> `vi +/string archivo` #se posiciona donde localiza string
> `vi +"set number" archivo` #activa la numeración de líneas

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

```
set number
```

Cada vez que se ejecute el `vi` la numeración estará activada.

Este editor tiene varias formas de trabajar:

- modo comando, o normal
- modo inserción
- modo visual
- modo de inserción de comandos

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