

Comandos Linux:

Para tener informacion relativa a un comando se debe teclear el comando

\$ man ls

Este comando nos dara infomación a cerca de `ls` listar archivos. Lamentablemente, el manual no siempre es claro, ni esta siempre disponible; esto implica aprenderse de memoria las cosas mas importantes. Esto sucede naturalmente con el tiempo y la practica.

Indice de Comandos según su función

Para...	un....	Uso el comando...
Borrar	archivo	rm
Borrar	directorio	rmdir
Crear	archivo (vacío)	touch
Crear	directorio	mkdir
Cambiar	directorio	cd
Cambiar	archivo	sed
Copiar	archivo	cp
Editar	archivo	sed
Encontrar	archivos	find
Encontrar	patron en archivo	grep
Encadenar	archivos	cat
Mover	archivos	mv
Ir a	directorio	cd

Listar	directorio	<u>ls</u>
Listar	procesos	<u>ps</u>
Ordenar	lineas de archivo	<u>sort</u>
Partir lineas	de archivo	<u>cut</u>
Pegar	archivos	<u>cat</u>
Pegar	columnas de archivos	<u>paste</u>
Procesar	archivos	<u>awk</u>
Renombrar	archivo	<u>mv</u>
Reformatear	archivo	<u>awk</u>
Ver	sistema de arch.	<u>df</u>
Ver	un archivo	<u>more</u>
Ver	directorio	<u>ls</u>
Ver	cabeza de archivo	<u>head</u>
Ver	Cola de archivo	<u>tail</u>

Descripcion de los Comandos

awk: Procesamiento de archivos generalizado

El comando awk es un lenguaje de programacion. En otra leccion, daremos mas detalles de este utilisimo comando. Sin embargo, muchas cosas se pueden hacer sin mayores conocimientos de awk.

awk lee lineas de un archivo. Cada linea se parte en **campos**, segun un **separador**, por defecto espacio en blanco. A cada linea se le aplica uno o mas **procedimientos** de awk. Un procedimiento consta de dos partes:

`/patron/{accion}`

El patron es una expresion regular (ver **), igual que grep o sed; la accion es un "programa", que se aplica a los campos de cada linea.

Veamos unos ejemplos:

`awk '{print $1}' arch`

Notese el uso de las comillas, necesarias para evitar problemas con el shell. Imprime el primer campo de cada linea-- es decir, imprime la primera columna de un archivo.

`awk '{print $2, $1}' arch`

Imprime la segunda, seguida por la primera, columna de un archivo

Tambien pueden usarse expresiones aritmeticas:

`awk '{print $1*$1, $2 - 5.0}' arch`

Lo cual imprime el cuadrado de la primera columna, etc.

Seleccionemos las lineas que continen el texto "Hidrogeno"

`awk '/Hidrogeno/{print $1, $2*3.1416}' arch`

Si el programa de awk es muy complicado, puede residir en un archivo:

`awk -f miprog arch`

Esto le aplica el programa "miprog" al archivo awk.

Los seleccionadores BEGIN y END seleccionan procedimientos que se ejecutan respectivamente al principio y fin de procesamiento:

```
awk '{s = s + $1}  
    END{print s}' arch
```

(notese que se puede dar retorno de linea dentro de las comillas.) Este programa suma la primera columna del archivo arch. El programa se basa en la (afortunada) casualidad que awk pone todas las variables numericas a cero inicialmente. Una variable (s en el ejemplo anterior es numerica cuando se usa aritmeticamente.

[Regresar al Indice](#)

cat: Cadenar (o ver) archivos

La funcion "oficial" de cat es de pegar o encadenar archivos. El archivo resultado va a stdout. Cuando hay un solo archivo este aparece por pantalla. Por eso, cat se usa mucho para ver el contenido de un archivo, aunque para eso es mejor more.

Ejemplos:

```
cat a1 a2 a3      # a1, a2, a3 a pantalla  
cat a1 a2 a3 >a4   # a1, a2, a3 a a4  
cat a1            # a1 a pantalla
```

[Regresar al Indice](#)

cd: Cambio de directorio

Con cd cambiamos el directorio donde estamos trabajando.

Ejemplos: Cambio absoluto de directorio:

```
cd /usr/local/bin
```

Cambia al directorio citado,

```
cd subdir
```

Cambia al directorio subdir del directorio actual; si este no existe hay error.

```
cd
```

Cambia al "directorio base" o "*home directory*" designado por el super-usuario; en algunos sistemas esto lo puede cambiar el usuario. El directorio base esta grabado en la variable de shell \$HOME.

[Regresar al Indice](#)

cp: Copia de archivos

El comando cp copia archivos. El ultimo argumento es el destino, los precedentes son el origen.

El comportamiento de cp depende del destino. Si el destino es un subdirectorio, los archivos son copiados a ese subdirectorio; pero si el destino no existe o es un archivo el origen es copiado al destino.

En el caso de multiples origenes, todos los origenes son sucesivamente copiados al destino. Eso quiere decir que en efecto solo el penultimo sobrevive la operacion, lo cual es probablemente no deseable.

Ejemplos:

cp origen destino

copia el archivo origen al archivo destino.

cp origen1 origen2 destino

Si destino es un directorio, origen1 y origen2 son copiados al directorio destino. Si destino es un archivo, el resultado es el mismo que si origen2 fuese copiado a destino.

cp -i origen destino

Copia origen a destino como en el caso anterior, pero en el caso que destino ya existe pregunta antes de hacer la operacion final.

[Regresar al Indice](#)

cut: Cortar archivo por campo o columnas

Se usa para separar de cada linea de un archivo uno o mas campos, o una parte del archivo. Los campos se delimitan, por defecto, por espacio en blanco; pero se puede especificar el contenido.

Ejemplos: Supongamos que el archivo prueba tenga la siguiente apariencia:

```
1 arepas 250
2 tostones 350
3 cachapas 125
```

Entonces

cut -f2 prueba

produce:

arepas

tostones

cachapas

Es decir, separa el segundo campo (-f2), mientras que

cut -c1-6 prueba

separa las primeras seis columnas y por lo tanto produce

1 ar

2 to

3 ca

El delimitador de campos puede cambiar:

cut -d: -f1,2 /etc/passwd

Produce los dos primeros campos de /etc/passwd, los cuales (verifiquelo) estan separados por los dos puntos :.

[Regresar al Indice](#)

Este comando da las características de un sistema de archivos. Su primer uso es de ver que es lo que está montado; su segundo uso es de ver el espacio libre de un sistema de archivos.

En mi sistema, el comando

```
df
```

produce

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/hda2	19424	18165	1259	94%	/
/dev/hda3	51718	42116	9602	81%	/usr
/dev/hda4	22578	13781	8797	61%	/u
/dev/hda1	32678	20052	12626	61%	/dos

De lo cual se puede ver, por ejemplo, que el sistema raíz (/) está a punto de llenarse. También se puede ver que hay cuatro sistemas de archivos.

La apariencia de la salida de este comando varía entre diferentes versiones de Linux.

find: Encontrar un archivo con ciertas características

El comando `find` se usa para encontrar archivos en el árbol de directorios de Linux. La estructura de directorios puede ser arbitraria. `find` requiere un punto de partida y las características del archivo a encontrarse. Después, `find` revisa ese directorio y todos los directorios subordinados, buscando los archivos que cumplan la condición(es) citada(s).

Lo examinaremos en más detalle en otra lección, (ver **) pero algunas formas comunes son:

```
find . -name perdido -print
```


Esto busca en el directorio actual (.) todos los archivos o directorios de nombre perdido y pone el resultado a pantalla (-print). En algunas versiones modernas de Linux, la opción -print no es necesaria, pero en otras sí (de lo contrario no pasa nada!)

```
find /usr/people -name '*.f' -print
```

Busca, a partir del directorio /usr/people, todos los archivos que terminen en .f. El uso de las comillas es **indispensable** porque de lo contrario, el shell sustituye por el asterisco los nombres de todos los archivos en el directorio de partida.

[Regresar al Índice](#)

grep: Encontrar un patron en una lista de archivos

El nombre grep es criptico: significa "*global regular expression and print*", pero su función es sencilla: encontrar un patron en una lista de archivos (por ejemplo, todos los archivos que contengan la palabra "CALL").

Para verdaderamente usar grep y muchos otros comandos de Linux hay que aprender el sistema de expresar patrones, llamado **expresiones regulares**, lo cual haremos en otra lección (ver **). Por los momentos nos contentaremos con patrones simples. Grep tiene muchas opciones (ver manual) para buscar con o sin mayúsculas, buscar archivos que **no** contienen al patron (especie de anti-grep), etc.

```
grep CALL *.f
```

Encuentra todas las líneas de todos los archivos que contienen la palabra CALL en todos los archivos que terminan en .f. Esto se podría usar como base a una tabla de referencias de un programa fortran.

```
grep juanr /etc/passwd
```

Busca la linea(s) de /etc/passwd que contiene(n) "juanr".

Mucho cuidado con patrones que contienen caracteres especiales del shell, como * y ?. Estos deben ser escapados, o usar comillas.

head: Ver las primeras n lineas de archivos

Se usa para ver las primeras lineas (cabeza) de un archivo. Por defecto, se ven 10 lineas, pero esto se puede cambiar. Por ejemplo,

```
head /etc/passwd
```

Pone en pantalla las primeras 10 lineas de /etc/passwd, mientras que

```
head -2 /etc/passwd
```

pone en pantalla las primeras dos.

[Regresar al Indice](#)

ls: Listar archivos en un directorio

Probablemente el comando mas usado en Linux, ls nos permite ver el contenido de un directorio y opcionalmente sus subdirectorios. Este comando tiene muchas opciones. La forma mas corriente es simplemente

```
ls
```

Que lista en varias columnas los nombres de los archivos en el directorio actual. Otra variante comun es

```
ls -l
```

lo cual da el listado largo (permisos, tama#os, due#o, etc.). Una opcion util es

```
ls -FC
```

que le pone a los ejecutables un asterisco, a los directorios la barra /, y a los archivos comunes nada (pruebe esto en us sistema). ls tambien acepta especificaciones:

```
ls -l *.f
```

Esto da todos los archivos que terminan en .f, con listado largo. Tambien podemos ordenar la lista de varias maneras, por ejemplo por edad:

```
ls -lt *.f
```

Esto nos lista en edad descendiente (mas viejo de ultimo) los archivos en que terminan en .f, con listado largo.

[Regresar al Indice](#)

man: Ver paginas del manual en linea

Para ver las paginas del manual, suponiendo que esten en linea. Ejemplos:

man cp

Nos da la informacion en el manual sobre el comando cp. Para averiguar mas sobre el comando man, pruebe

man man

[Regresar al Indice](#)

mkdir: Crear un directorio

Crea un directorio vacio en el directorio actual, por ejemplo

mkdir xyz

Crea el directorio xyz en el directorio actual.

[Regresar al Indice](#)

more: Ver archivos con control de pantalla

Sirve para examinar un archivo. Es preferible al uso de cat, que se usa para el mismo proposito, porque more permite retroceder, avanzar, o hacer busquedas. Por ejemplo

more xyz

Nos permite ver el archivo xyz en pantalla. Una vez entrado a more, se controla con los siguientes comandos, o mejor dicho "subcomandos:

- espacio-- adelanta una pantalla
- b (back) retrocede una pantalla
- enter avanza una linea
- /patron busca "patron" en el archivo
- n busca la proxima ocurrencia de el patron anterior
- q abandona el programa (salida).

En algunos sistemas de sabor a SysV, este mismo comando se llama pg.

[Regresar al Indice](#)

mv: Mover archivos entre directorio (o renombrar)

El comando mv es similar a [cp](#), excepto que borra el origen. En otras palabras, mueve archivos de un directorio a otro, o de un archivo a otro. En este ultimo caso, como el original desaparece, mv puede a veces tener efectos inesperados. El ultimo argumento de mv indica el destino del movimiento; los primeros son los origenes.

Un uso muy frecuente de mv es de cambiar el nombre a un archivo. Supongamos, por ejemplo, que viejo existe y le queremos cambiar al nommbre nuevo; nos aseguramos primero con ls que el nombre nuevo no existe; luego hacemos

```
mv viejo nuevo
```

con lo cual viejo queda rebautizado a nuevo.

Ahora suponemos que subdir es un directorio. Para mover archivos a este directorio, pudieramos usar

```
mv xyz uvw subdir
```

En este caso, xyz y uvw se mueven al subdirectorio subdir. Pero si subdir fuese un archivo, o no existiese, este comando mueve, esencialmente, el penultimo al ultimo- un comportamiento probablemente inesperado. Se pueden evitar accidentes con la opcion interactiva:

```
mv -i xyz xxx
```

En este caso, mv pregunta antes de mover; esto es bueno en general pero latoso si hay que mover grandes cantidades de archivos.

[Regresar al Indice](#)

rm: Borrar archivos

rm borra archivos y con ciertas opciones, hasta directorios. **Advertencia:** este comando es **irreversible**.

Ejemplos:

```
rm xyz *.o
```

Este comando borra el archivo xyz y todos los archivos que terminan en .o. **PELIGRO:** La especificacion *.o, y todas las expresiones que contienen el asterisco, son sumamente peligrosas. Por ejemplo supongamos que por error se deja un espacio en blanco en el ejemplo anterior:

```
rm xyz * .o
```

Esto es desastroso: el asterisco borra todos los archivos, silenciosamente, y despues se queja que "no puedo encontrar .o". La mejor manera de evitar este accidente es con cuidado. Otra posibilidad es de usar la opcion interactiva:

```
rm -i xxx xyz
```

Esto pregunta antes de efectuar la remocion.

Con la opcion recursiva, se puede borrar un directorio y todos los archivos dentro de ese directorio. Esto es equivalente a hacer cd a ese directorio, borrar todos los archivos, subir, y hacer rmdir. Use esta opcion con sumo **cuidado**.

```
rm -r direc
```

Esto borra el directorio direc y todo lo que pueda estar por debajo de direc.

[Regresar al Indice](#)

rmdir: Borrar un directorio

Este comando borra un subdirectorio vacio. Si no esta vacio, rmdir se queja y no efectua la operacion.

```
rmdir direc
```

Borra el directorio direc.

[Regresar al Indice](#)

paste: Unir archivos horizontalmente (por columnas)

paste es lo suficientemente util para incluirlo en esta lista de comandos basicos, porque puede ahorrar mucho trabajo con el editor. Si tenemos dos archivos con columnas, por ejemplo:

```
archivo a      archivo b
xx yy          uu vv
zz 11          ww 22
```

entonces el comando

```
paste a b >c
```

produce un archivo de esta forma:

```
xx yy uu vv
zz 11 ww 22
```

paste tiene varias opciones que controlan la seleccion de columnas. Ver el manual para mas informacion. El comando no esta en algunos sistemas (como Sun OS 4.0).

[Regresar al Indice](#)

pwd: Dar el nombre del directorio actual

Nos recuerda, cuando estamos perdidos, del nombre del directorio actual. pwd nos da el camino completo.

pwd

[Regresar al Indice](#)

ps: Listar procesos en ejecucion

Los principales usos de ps son:

- Ver lo que esta corriendo-- estimar la carga sobre el sistema.
- Ver si uno de nuestros procesos todavia esta vivo (una corrida larga, por ejemplo)
- Encontrar el PID de un proceso para matar.

La sintaxis de ps depende de la variedad de Linux. En sistemas de origen Berkeley, la sintaxis para un listado completo es

ps aux

Mientras en sistemas de variedad SysV, es

ps -ef

La forma corta (sin argumentos):

ps

Nos da los procesos de la actual sesion de login, sin contar los que pudieran seguir corriendo de otras sesiones.

El formato de las salidas tambien depende de la variedad del sistema.

[Regresar al Indice](#)

tail: Ver las ultimas n lineas de un archivos

tail da las ultimas 10 lineas de un archivo, u opcionalmente las ultimas n lineas del archivo. Por ejemplo

```
tail xxx
```

Da las ultimas 10 lineas del archivo xxx, mientras que

```
tail -1 xxx
```

Da la ultima.

Este comando es util para seguir el progreso de un programa de larga corrida.

[Regresar al Indice](#)

touch: Actualizar fecha de archivos

Este comando tiene dos usos: uno, crear un archivo vacio y dos, actualizar la fecha de un archivo-- cambia la fecha a la de "ahora". Esto a su vez se usa frecuentemente con el programa make, objeto de otra sesion.

```
touch xxxx
```

Si xxxx existe, le cambia la fecha; si no crea ese archivo sin contenido alguno (longitud cero bytes).

[Regresar al Indice](#)

sed: Edicion "batch" de un archivo

Este programa tiene su mayor utilidad en **shell scripts**. Lo veremos en detalle en otra leccion. Para comenzar, basta decir que sed es un editor estilo "batch" que aplica unos **subcomandos** a todas las lineas de un archivo. Esto implica que para usar a sed, hay que aprenderse estos subcomandos. Lamentablemente, el manual correspondiente es casi indescifrable. Afortunadamente, los comandos se parecen a los de vi, (ver **). Damos aqui unas aplicaciones simples:

```
sed '500,600pq' archivote
```

las lineas de archivote a entre la numero 500 y la 600 van a pantalla; (esto es print o p) y luego sed termina debido al q (quit). Esto se puede utilizar para seleccionar grupos de lineas de un archivo.

```
sed 's/patron1/patron2/g' arch1 >arch2
```

Cambia todas las ocurrencias de "patron1" en arch1 por "patron2" y por redireccion manda el resultado a arch2. (Normalmente sed dirige sus salidas a stdout. El ejemplo anterior es mas sencillo que usar un editor.

[Regresar al Indice](#)

sort: Ordenamiento de un archivo

La función de sort es de ordenar las líneas de un archivo. Las líneas (registros o *records* en la terminología de sort) se pueden subdividir en campos; el ordenamiento puede ser alfabético o numérico y el orden ascendente o descendiente.

sort es un programa generalizado de ordenamiento con muchas opciones; por lo tanto diferimos su discusión completa hasta otra lección (ver **). Mientras tanto, unos ejemplos:

```
sort arch1 >arch2
```

Ordena alfabéticamente las líneas de arch1. Por defecto, sort envía su salida a stdout; usamos redirección para crear un archivo de salida. El ordenamiento es alfabético por defecto y el orden ascendente (es decir, "a" sale por delante de "z"). El campo es todo el registro.

```
sort -n arch1 >arch2
```

Como el anterior, pero el orden es numérico. Usase cuando hay columnas numéricas. El orden es ascendente (1 sale antes que 2).

```
sed sort -nr arch1 >arch2
```

El anterior, pero en orden descendente. El número mayor sale primero.

```
sort +0 -1 arch1 >arch2
```

En este caso, el archivo está dividido en campos, separados por defecto por blancos. El ordenamiento es solo por primer campo. La manera de especificar los campos significa "final de campo cero hasta principio de campo uno".

Comandos básicos de UNIX

Comando ls

El comando ls permite al usuario visualizar los archivos y subdirectorios que se tienen dentro de un directorio.

Comando cd

El comando cd permite cambiarnos de directorio. Este comando es muy similar al de una PC, con la única diferencia que cuando se desea cambiar hacia un subdirectorio exterior, se debe dejar un espacio entre los (..).

Comando pwd

El comando pwd es útil cuando se tiene un subdirectorio que a su vez tiene otros subdirectorios en su interior. A veces surgen confusiones acerca de en cuál directorio nos encontramos. Este comando despliega el path o la ruta actual.

Comando mkdir

mkdir es usado cuando se desea crear un directorio sobre el directorio presente, siempre y cuando se cuente con los permisos adecuados.

Comando rmdir

Cuando se desea borrar un subdirectorio se usa el comando rmdir acompañado del nombre del directorio que se desea borrar. Para borrar un subdirectorio, éste se debe encontrar vacío, es decir, no debe contener ningún archivo.

Comando rm

El comando rm sirve para eliminar o borrar archivos. También se puede utilizar para borrar subdirectorios, ya que como se dijo anteriormente, en UNIX no hay distinción entre lo que es un archivo y un subdirectorio. El comando rm acepta el uso de comodines como lo es el (*).

Comando cp

El comando cp permite realizar una copia de un archivo o moverlo de un subdirectorio a otro. También con este comando se puede cambiar de nombre a un archivo.

Comando mv

Otro comando de mucho uso es mv. Con este comando se puede renombrar un archivo.

Comando !!

Al igual que en el sistema operativo DOS, en UNIX existe un comando que nos permite volver a ejecutar el último comando que se tecleó, éste es !!.

Comando ping

Muchas veces al estar trabajando en ambientes de redes, se necesita conocer si una maquina se encuentra encendida o si la red esta funcionando de forma correcta, para eso se cuenta con este comando.

Comando finger

Si se desea saber qué usuarios de una máquina cuya dirección es conocida se encuentran activos, con este comando se puede conocer el login de ese usuario, su verdadero nombre, en qué consola está activo y la dirección de la máquina desde la que se está conectando.

Comando alias

En ciertas ocasiones se suelen utilizar comandos que son difíciles de recordar o que son demasiado extensos, pero en UNIX existe la posibilidad de dar un nombre alternativo a un comando con el fin de que cada vez que se quiera ejecutar, sólo se use el nombre alternativo. Para definir un alias sólo se necesita poner entre "" el nombre que tendrá el nuevo comando, seguido del comando que deseamos sea ejecutado al escribir el nombre alternativo.

Comando unalias

Cada vez que se requiera cambiar o borrar un alias creado dentro de nuestra área de trabajo, se debe utilizar el comando unalias.

Comando more

El comando more funciona de manera muy parecida al type del DOS. Con este comando se puede visualizar el contenido de un archivo de texto dentro de un ambiente UNIX.

Comando passwd

Este comando permite cambiar el password de acceso a una cuenta dentro de un sistema UNIX. Es muy recomendable cambiar el password de manera frecuente e incluir en él números, ya que al cambiar frecuentemente el password se aumenta la seguridad de un sistema.

Comando uuencode / uudecode

Este comando nos sirve para poder codificar un archivo binario (todo archivo que no es de texto es considerado un archivo binario, es decir una imagen, un archivo ejecutable de DOS, etc.) en un archivo de texto, esto es de suma importancia cuando necesitamos mandar ese archivo por correo electrónico.

Comandos Básicos de UNIX

Dentro de este sistema operativo existe una gran variedad de comandos para realizar las operaciones de trabajo. Empezaremos con comandos simples y de utilidad para los nuevos usuarios:

Comando ls

El comando **ls** permite al usuario visualizar los archivos y subdirectorios que se tienen dentro de un directorio.

Ejemplo:

speedy% ls

devices	kadb	mnt	tmp
etc	kernel	net	ufsboot
bin	export	lib	opt
usr	cdrom	home	lost+found
proc	var	dev	hsfsboot

```
mail          sbin          vol
speedy%
```

Como se puede ver del ejemplo anterior, al dar únicamente el comando **ls** sólo se despliega el *nombre de los archivos o subdirectorios*, en este caso, el despliegue se hace de una manera horizontal y no se puede distinguir entre lo que es un directorio de un archivo, tampoco se visualizan los permisos ni el tamaño de los archivos.

Existen más opciones para este comando, estas se enumeran a continuación acompañadas de ejemplos:

- **ls -l**

Con esta opción se listan los archivos con información adicional como lo constituye su tamaño en bytes, los permisos, la última fecha de modificación. En este modo es posible llevar a cabo la diferencia entre un archivo y un subdirectorio.

Ejemplo:

```
speedy% ls -l
-rwsr-xr-x  1 root  other  68440 Apr  6 09:42 archie
drwxr-sr-x   2 root  other   512 Mar 23 13:40 bin
-rwsr-xr-x  1 root  other   65 May 15 11:05 correo
-rwxr-xr-x   1 root  other   61 Jan 31  1994 editor
drwxr-sr-x   3 root  other   512 Sep 26  1994 etc
drwxr-sr-x   3 root  other  1536 Sep 20  1994 fsp
-rwxr-xr-x   1 root  other 114688 Jan 31  1994 gopher
drwxr-sr-x   9 root  other   512 Sep 20  1994 gopher1.12S.d
-rwxr-xr-x   1 root  other  73728 Apr 21  1994 gopherd
-rwxr-xr-x   1 root  other  98304 Jan 31  1994 gunzip
-rwxr-xr-x   1 root  other   66 Jan 31  1994 hoja
```

Como se aprecia en este ejemplo, el primer campo del parrafo anterior describe los **Permisos** que tiene este archivo o subdirectorio, existen básicamente 3 clases de permisos:

r= PERMISO DE LECTURA

w= PERMISO DE ESCRITURA

x= PERMISO DE EJECUCIÓN

La letra **d** que aparece al principio de algunas lineas, se refiere a la designación de directorio.

- **ls -la**

Con esta opción se listan los archivos ocultos en unix, estos archivos se caracterizan por comenzar su nombre con un punto. Cada cuenta dentro de un sistema unix maneja 2 archivos ocultos que son de suma importancia, dependiendo del Shell al que se tenga acceso, estos archivos son: *.cshrc* y *.login*.

Ejemplo:

```
drwx----- 5 alex  coacade  1024 May 13 12:46 .
drwxr-xr-x 23 root  root    512 Apr  6 17:42 ..
-rw-r--r--  1 alex  coacade   699 May  8 18:20 .cshrc
-rw-r--r--  1 alex  coacade    20 Apr  7 10:55 .forward
-rw-r--r--  1 root  other    528 Apr  6 17:43 .login
-rw-r--r--  1 root  coacade  4792 May 15 13:31 .mosaic-global-history
-rw-r--r--  1 root  coacade   128 May 15 13:31 .mosaic-hotlist-default
-rw-r--r--  1 root  coacade    5 May 15 11:30 .mosaicpid
-rw-r--r--  1 alex  coacade  2585 May 17 18:09 .netscape-bookmarks.html
drwx----- 2 alex  coacade  30208 May 17 18:15 .netscape-cache
-rw-----  1 alex  coacade  71574 May 17 18:09 .netscape-history
-rw-r--r--  1 alex  coacade  1614 May 17 09:33 .netscape-preferences
drwxr-xr-x  2 alex  coacade   2560 May  8 18:26 JPG
-rw-r--r--  1 alex  coacade   1293 May 12 10:57 anon-ftp
```

Note que con esta opción además de mostrar los archivos ocultos se muestran también los archivos que solamente se despliegan con la opción **ls -l**

- **ls -l | more**

Esta opción resulta muy útil cuando se tienen muchos archivos, ya que el despliegue es llevado a cabo por páginas, para seguir viendo el contenido es necesario pulsar la barra espaciadora, para ir página por página o con enter para ir línea por línea.

Comando cd

El comando **cd** permite cambiarnos de directorio, este comando es muy similar al de una PC con la única diferencia que cuando se desea cambiar hacia un subdirectorio exterior, se debe dejar un espacio entre los ..

Ejemplo:

Supongase que Ud. como usuario se encuentra en el subdirectorio **/export/home/alex/JPG** y desea cambiarse a otro subdirectorio cuya ruta es **/export/home/alex/JPG/paisajes** lo que tiene que teclear es:

```
speedy% cd paisajes [Enter]
```

Una vez que este en la ruta **/export/home/alex/JPG/paisajes** y desee regresar un subdirectorio hacia afuera, lo que deberá teclear será:

```
speedy% cd .. [Enter]
```

Notece como fué necesario dejar un espacio en blanco despues de teclear el comando **cd** y antes de poner los ..

Comando pwd

El comando **pwd** es útil cuando se tiene un subdirectorio que a su vez tiene otros subdirectorios en su interior, a veces surgen confusiones acerca de en cuál directorio nos encontramos, esto también es muy frecuente al estar en sesiones de **Ftp** .

Este comando *despliega el path o la ruta* en que nos encontramos.

Ejemplo:

```
speedy% pwd [Enter]  
/export/home/gaby/programas
```

Comando mkdir

mkdir es usado cuando se desea crear un directorio sobre el directorio presente, siempre y cuando se cuente con los permisos adecuados. Ya que existen subdirectorios dentro de un sistema Unix en los cuales los usuarios comunes no pueden o no tienen los permisos para poder crear algún subdirectorio. Generalmente el usuario tiene la capacidad de crear subdirectorios dentro de su espacio asignado en el servidor, usando este comando.

Ejemplo: Supongamos que un usuario se encuentra dentro de su cuenta en un servidor que maneje Unix, y desea crear un subdirectorio llamado *programas* lo que tiene que teclear es:

```
speedy% mkdir programas [Enter]  
speedy%
```

Con esto el subdirectorio programas ha sido creado sobre el path en donde se encontraba el usuario trabajando.

Comando rmdir

Cuando se desea **borrar un subdirectorio** se usa el comando **rmdir** acompañado del nombre del directorio que se desea borrar, para borrar un subdirectorio este se debe encontrar vacío, es decir, no debe contener ningún archivo.

Ejemplo:

Si el usuario *gaby* se encuentra bajo la ruta */export/home/gaby/antivirus* pero desea borrar el subdirectorio *antivirus*, lo primero que debe checar es si efectivamente no hay archivos presentes en ese subdirectorio, si esto es afirmativo puede proceder a borrar el subdirectorio o bien en caso contrario deberá borrar previamente los archivos de ese subdirectorio.

Para borrar ese subdirectorio es necesario salir un directorio hacia afuera, usando el comando **cd ..** en síntesis lo que deberá teclear es:

```
speedy% pwd [Enter]
/export/home/gaby/antivirus
speedy% ls -l [Enter]
total 0
speedy% cd .. [Enter]
speedy% pwd [Enter]
/export/home/gaby
speedy% rmdir antivirus [Enter]
speedy%
```

Observe como en este ejemplo, lo primero que se verifica es la ruta en la que se encuentra, después se checa si no hay archivos en el subdirectorio que se va a borrar y finalmente se procede a borrar el subdirectorio cambiandose un directorio hacia afuera.

Comando rm

El comando **rm** sirve para eliminar o borrar archivos, también se puede utilizar para borrar subdirectorios, ya que como se dijo anteriormente en unix no hay distinción entre lo que es un archivo y un subdirectorio.

El comando **rm** acepta el uso de comodines como lo es el *

Ejemplo:

Para borrar los archivos que terminen con una extensión .txt por ejemplo, si se tiene el siguiente listado:

```

-rw-r--r-- 1 gaby coacade 26511 May 16 09:24 00_index.txt
-rw-rw-rw- 1 gaby coacade 23831 May 9 09:46 18.cp.html
drwx----- 4 gaby coacade 512 Apr 26 10:45 Mail
drwx----- 2 gaby coacade 1024 Apr 28 11:32 data
-r--r--r-- 1 gaby coacade 130 Feb 27 17:40 datos.txt
-rw-r--r-- 1 gaby coacade 1419 Jan 5 12:03 east.txt
drwxr-xr-x 2 gaby coacade 512 May 17 12:31 gs
-rw-r--r-- 1 gaby coacade 6308 May 16 09:24 hdcodes.txt
-rw-rw-rw- 1 gaby coacade 189 May 17 09:41 hello.c
-rw-r--r-- 1 gaby coacade 2244 Sep 27 1994 install.gcc
drwxr-xr-x 2 gaby coacade 512 Apr 25 10:17 jpg
-rw-r--r-- 1 gaby coacade 23 May 15 11:30 m
-rw----- 1 gaby coacade 55177 May 16 09:07 mbox
-rw-r--r-- 1 gaby coacade 1599738 May 15 10:49 n16e11n.exe

```

lo que se debe teclear es lo siguiente:

```

speedy% rm *.txt [Enter]
rm: remove 00_index.txt (y/n)?y
rm: remove datos.txt (y/n)?y
rm: remove east.txt (y/n)? y
rm: remove hdcodes.txt (y/n)? y

```

Observe cómo el sistema le pide que autorice la eliminación de los archivos, esto es porque en los archivos de inicialización `.cshrc` y `.login` se ha editado previamente el comando **rm** para que cuando se ejecute en realidad del comando que se este ejecutando es: **rm -i**, con esta opción siempre se fuerza al sistema a que pregunte si en realidad se desea borrar ese archivo.

Si Ud. desea que el sistema no pregunte, tendra que editar su archivo `.cshrc` y cambiar la línea del alias definido para el comando **rm**.

Si lo que se desea es borrar un subdirectorio con todo y los archivos que contiene, se puede usar la opción **rm -r**

Ejemplo:

Supongamos que se desea borrar el subdirectorio *data* con todo y los archivos que contiene, lo que se debe teclear es:

```
-rw-r--r-- 1 gaby coacade 26511 May 16 09:24 00_index.txt
-rw-rw-rw- 1 gaby coacade 23831 May 9 09:46 18.cp.html
drwx----- 4 gaby coacade 512 Apr 26 10:45 Mail
drwx----- 2 gaby coacade 1024 Apr 28 11:32 data
-r--r--r-- 1 gaby coacade 130 Feb 27 17:40 datos.txt
-rw-r--r-- 1 gaby coacade 1419 Jan 5 12:03 east.txt
drwxr-xr-x 2 gaby coacade 512 May 17 12:31 gs
-rw-r--r-- 1 gaby coacade 6308 May 16 09:24 hdcodes.txt
-rw-rw-rw- 1 gaby coacade 189 May 17 09:41 hello.c
-rw-r--r-- 1 gaby coacade 2244 Sep 27 1994 install.gcc
drwxr-xr-x 2 gaby coacade 512 Apr 25 10:17 jpg
-rw-r--r-- 1 gaby coacade 23 May 15 11:30 m
-rw----- 1 gaby coacade 55177 May 16 09:07 mbox
-rw-r--r-- 1 gaby coacade 1599738 May 15 10:49 n16e11n.exe
```

```
speedy% rm -r data
rm: examine files in directory data (y/n)? y [Enter]
rm: remove data/Makefile (y/n)? y
rm: remove data/README (y/n)? y
rm: remove data/crc.c (y/n)? y
rm: remove data/crc.doc (y/n)? y
rm: remove data/crctab.c (y/n)? y
rm: remove data/gz (y/n)? y
rm: remove data/mailer.rz (y/n)? y
rm: remove data/minirb.c (y/n)? y
rm: remove data: (y/n)? y
```

Observe como en la última línea se pide la confirmación de eliminación del subdirectorio.

Comando cp

El comando **cp** permite realizar una copia de un archivo o moverlo de un subdirectorío a otro, también con este comando se puede cambiar de nombre a un archivo.

Ejemplo:

Si se tiene el archivo llamado **reporte.txt** sobre la ruta `/export/home/gaby` y se quiere copiar hacia el subdirectorío `/export/home/gaby/documentos` o que se debe teclear es:

```
speedy% pwd [Enter]
```

```
/export/home/gaby
```

```
speedy% ls -l [Enter]
```

```
drwx----- 4 gaby  coacade   512 Apr 26 10:45 Mail
drwx----- 2 gaby  coacade  1024 Apr 28 11:32 data
-r--r--r-- 1 gaby  coacade   130 Feb 27 17:40 datos.txt
-rw-r--r-- 1 gaby  coacade  1419 Jan  5 12:03 east.txt
drwxr-xr-x 2 gaby  coacade    512 May 17 12:31 documentos
-rw-r--r-- 1 gaby  coacade  6308 May 16 09:24 reporte.txt
-rw-rw-rw- 1 gaby  coacade   189 May 17 09:41 hello.c
```

```
speedy% cp reporte.txt /export/home/gaby/documentos [Enter]
```

```
speedy%
```

También se puede copiar un archivo hacia otro archivo cuyo nombre se especifica.

Comando mv

Otro comando de mucho uso es **mv** con este comando se puede renombrar una archivo, la sintaxis es: **mv [nombre_archivo] [nuevo_nombre]**

Ejemplo:

Si se desea renombrar el archivo resumen.txt al nuevo nombre abstracto.txt lo que se debe teclear es:

```
colmena% mv resumen.txt abstracto.txt [Enter]
```

```
colmena%
```

Comando !!

Al igual que en el sistema operativo DOS, en Unix existe un comando que nos permite volver a ejecutar el último comando que se tecleó este es !!

Ejemplo:

```
speedy% ls -l [Enter]
```

```
-rwsr-xr-x  1 root  other   68440 Apr  6 09:42 archie
drwxr-sr-x  2 root  other    512 Mar 23 13:40 bin
-rwsr-xr-x  1 root  other    65 May 15 11:05 correo
-rwxr-xr-x  1 root  other    61 Jan 31  1994 editor
drwxr-sr-x  3 root  other    512 Sep 26  1994 etc
drwxr-sr-x  3 root  other   1536 Sep 20  1994 fsp
-rwxr-xr-x  1 root  other  114688 Jan 31  1994 gopher
drwxr-sr-x  9 root  other    512 Sep 20  1994 gopher1.12S.d
```

```
speedy% !! [Enter]
```

```
-rwsr-xr-x  1 root  other   68440 Apr  6 09:42 archie
drwxr-sr-x  2 root  other    512 Mar 23 13:40 bin
-rwsr-xr-x  1 root  other    65 May 15 11:05 correo
-rwxr-xr-x  1 root  other    61 Jan 31  1994 editor
drwxr-sr-x  3 root  other    512 Sep 26  1994 etc
drwxr-sr-x  3 root  other   1536 Sep 20  1994 fsp
-rwxr-xr-x  1 root  other  114688 Jan 31  1994 gopher
drwxr-sr-x  9 root  other    512 Sep 20  1994 gopher1.12S.d
```


Como puede observarse el listado es exactamente el mismo que se despliega cuando se ejecutó por primera vez el comando ls -l

Comando ping

Muchas veces al estar trabajando en ambientes de redes, se necesita conocer si una maquina se encuentra encendida o si la red esta funcionando de forma correcta, para eso se cuenta con este comando: ping [nombre_o_dirección_de_la_máquina] si la maquina esta encendida o si la red esta funcionando de manera correcta, se obtendrá una respuesta de la serie de paquetes enviados, y el tiempo que tardo en ser recibido.

Ejemplo:

```
speedy# ping speedy  
speedy is alive
```

Existe una opción que nos da mayor información con respecto a la red como lo son el tamaño de los paquetes que fueron enviados y el tiempo que se tradó en recibirlos.

Ejemplo:

```
speedy# ping -s speedy  
PING speedy: 56 data bytes  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=0. time=2. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=1. time=1. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=2. time=1. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=3. time=1. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=4. time=1. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=5. time=1. ms  
64 bytes from speedy.coacade.uv.mx (148.226.1.1): icmp_seq=6. time=1. ms
```

Comando finger

Si se desea saber que usuarios de una maquina cuya dirección es conocida, se encuentran activos, con este comando se puede conocer el login de ese usuario, su verdadero nombre, en que consola esta activo y la dirección de la máquina desde la que se está conectando.

Ejemplo:

```
dino% finger
```

Login	Name	TTY	Idle	When	Where
jacome	Maria Luisa Jacome M	pts/0		Thu 12:32	148.226.4.167
atejeda	Adalberto Tejeda Mar	pts/1	1:28	Thu 11:11	148.226.1.228
root	0000-Admin(0000)	pts/4	1:36	Wed 13:02	
root	0000-Admin(0000)	pts/2	48d	Wed 13:11	
gaby	Gabriel Lozano Garci	pts/4	1:36	Thu 10:12	toontown

```
dino%
```

Este comando se puede utilizar para saber que usuarios se encuentran dentro de un sistema unix aún fuera del dominio de la red donde se localiza la máquina que se está usando. Para ello se debe colocar la letra @ antes de la dirección de la máquina cuyos usuarios en activos deseamos conocer.

Ejemplo:

```
speedy# finger @condor.dgsca.unam.mx
```

```
[condor.dgsca.unam.mx]
```

Login	Name	TTY	Idle	When	Where
gopher	Servicio de Gopher	p4		Thu 13:27	ayaic.cecafi.una
silvia	Silvia Beltran S. CS	p5	1:22	Thu 10:41	sisop8.dgsca.una
info	Informacion de la UN	p6		Thu 12:47	148.206.32.40
paco	Jose Francisco Becer	p8	27	Thu 11:45	voyager
info	Informacion de la UN	pa	8	Thu 13:15	copitl.uam.mx
diane	RANDY.Diane Fumiko M	pb	1:21	Thu 11:01	132.248.74.8
antonio	Ing. Antonio Enrique	pc	21	Thu 12:02	puma
gwaldega	WALDEGG CASANOVA GUI	q3	2	Thu 13:21	gw-trouter.noc.u
ekrotza	KROTZ HEBERLE ESTEBA	q8		Thu 13:26	soc.uady.mx

Comando alias

En ciertas ocasiones se suele utilizar comandos que son difíciles de recordar o que son demasiado extensos, pues en Unix existe la posibilidad de Dar un nombre alternativo a un comando con el fin de que cada vez que se quiera ejecutar ese comando, solo se pone el nombre alternativo.

Para definir un alias solo se necesita poner entre "" el nombre que tendrá el nuevo comando, seguido del comando que deseamos sea ejecutado al escribir el nombre alternativo.

Ejemplo:

Si se desea renombrar el comando `ls -l` con el nombre de *dir* lo que da por resultado una equivalencia de los comandos Unix y DOS, solo se tendría que escribir:

```
speedy% alias "dir" ls -l [Enter]
```

```
speedy%
```

Con esto cada vez que se ejecute el comando *dir* en realidad lo que se está ejecutando es el comando `ls -l`.

Es importante aclarar el hecho de que la definición de alias es un proceso que funciona mientras se tenga activa una consola, si la definición desea hacerse de manera definitiva, se deberá editar el archivo `.cshrc`

Comando unalias

Cada vez que se requiera cambia o borrar un alias creado dentro de nuestra area de trabajo, se debe utilizar el comando `unalias`.

Ejemplo: Supongamos que erroneamente pusimos el alias `dis` en lugar de `dir` hay que borrar el alias `dis` y poner el correcto

```
speedy%alias dis "ls -l" [enter]
```

```
speedy%  
speedy%alias [enter]  
dis    ls -l  
ping   (ping -s)  
speedy%
```

Note: con alias sin parámetros, nos da los alias que estan establecidos

```
speedy%unalias dis  
speedy  
speedy% alias  
ping   (ping -s)  
speedy%  
speedy%alias dir "ls -l"  
speedy%alias  
dir    ls -l  
ping   (ping -s)
```

este último paso ya es el correcto.

Comando more

El comando more funciona de manera muy parecida al *type* del DOS, con este comando se puede visualizar el contenido de un archivo de texto, dentro de un ambiente unix.

Si el archivo es demasiado largo y no se alcanza a desplegar en una sola pantalla, en la parte inferior de la pantalla se despliega el mensaje *--More--(x%)* donde la x significa el % del texto total que ha sido desplegado.

Ejemplo:

```
speedy# more ARPANET.txt
```

Mourning the Passing of the ARPANET

Past

On June 1, 1990, the nation's first computer network experiment, ARPANET, was officially removed from service. Although the removal was not totally transparent to network users, there is no need to mourn. The Defense Advanced Research Projects Agency (DARPA) did a great service over the last twenty years to the networking community in developing and providing a research facility that emerged into an operational infrastructure. Many spin-offs were created, for example,

ESnet DOE Energy Scientists Network

MILNET DOD Military Network

NSFnet National Science Foundation Network

NSN NASA Science Network

Present

Today, we live in a world where networks are widely used. The Los Alamos Open Network is tied directly or indirectly to these international networks, commonly called the Internet. The Internet is based on individually registered and operated networks communicating through cooperating gateways. Los Alamos is connected directly to MILNET and ESnet and to NSFnet through NCAR and New Mexico Technet. Most other connections are via gateways maintained by MILNET, NSFnet, or ESnet.

--More--(38%)

Para continuar el despliegue solo es necesario pulsar la barra espaciadora, para poder ir página por página, o enter, para ir renglón por renglón.

Comando passwd

Este comando permite cambiar el Password de acceso a una cuenta dentro de un sistema Unix, es muy recomendable cambiar el password de manera frecuente e incluir en él números, ya que al cambiar frecuentemente el password se aumenta la seguridad de un sistema.

El supervisor de un sistema es la persona adecuada para cambiar el password de una cuenta si es que el usuario no recuerda cual era.

Para que un usuario puede cambiar por si mismo su password solo deberá teclear:

```
speedy% passwd [Enter]
```

```
passwd: Changing password for [Login_Del_Usuario]
```

```
New password:
```

```
Re-enter new password:
```

```
speedy%
```

Observe como cuando escribe su password la maquina no lo muestra en pantalla, con el objeto de mantenerlo confidencial.

Comando uuencode / uudecode

Este comando nos sirve par poder codificar un archivo binario (todo archivo que no es de texto es considerado un archivo binario, es decir una imagen, un archivo ejecutable de DOS, etc) en un archivo de texto, esto es de suma importancia cuando necesitamos mandar ese archivo por [correo electrónico](#)(se vera posteriormente). así pues se debe utilizar de la siguiente manera:

Para codificar:

```

toontown% uuencode foto.gif > foto.txt
begin 600 foto.txt
M_C_X 02D9)1@ ! 0$ 8P!C #_VP!# ! +# X,"A .#0X2$1 3&"@:&!86
M&#$(C)1TH.C,]/#DS.#= 2%Q.0$1713<X4&U15U]B9VAG DUQ>7!D>%QE9V/_
MVP!# 1$2$A@5&"\:&B]C0CA"8V-C8V-C8V-C8V-C8V-C8V-C8V-C8V-C
M8V-C8V-C8V-C8V-C8V-C8V-C8V-C8V/_P 1" $O ?$# 2( A$! Q$!_0
M'P 04! 0$! 0$ $" P0%!@<("0H+_0 M1 @$# P($ P4%
M! 0 %] 0(# 01!1(A,4$&$U%A!R)Q%#*!D:$((T*QP152T? D,V)R@@D*
M%A<8&1HE)B7J#A(6&AXB)BI*3E)66EYB9FJ*CI*6FIZBIJK*SM+6VM[BYNL+#Q,7&
M>M6882#STJ*QI&%RMY)( '6GBWW1D]P*OQ1!3NZU:$0"Y9!6;F7R& B;FQ3
MGC SMJT8/WS <8-+)$57[O-7S$\\AGD<TY0,V::#D4N:@8&DI3TI"*8"@<9%+B@8
MHR,4@#O2XI>* 10 8HQS1D4H([4AB@4N!2 BER,YI#0](\U:AM@W6H(B,BM"
MW(XK&;-8I$D=J%'2IA".E2)C'6IT4&N=R9K9#(K=0,U8$8%/2/BG$ 4G<"$
MI3=N#Q5A0">*DL>E-78,K*WJ*ECQFG&("E"A33;?4":/K4X-0**=D@52$R5
MFJK<'(-/+\\5$_(J9 C*N(Q-498.>E;+H#5:1!6?,T6D8KP?C4#P#' K8>,8
MJL8S6D:@6N9?D?2BM#RA[T5K[47(CG*.)]2@XS[^U=QYX4E+24#%HHH% #E
MH4#/-)2 F@ QU2+\O7I46>:E1CCVI,! #9ZBF":I9 -M1$8H0"H0">]+D
MA,TRG#[N*8A[*0!&<'%6&RZ<=*ACCW,/2K: )W4WWA3(V:&< PRE>

```

donde foto.gif es el nombre del archivo binario (en este caso una imagen) que se desea codificar a texto, foto.txt es el nombre con el que va a quedar salvada la imagen y el signo de mayor que indica al comando que se salve la imagen en foto.txt.

Para decodificar:

```
speedy%uudecode foto.txt
```

Para la decodificación, solo se pone el nombre de la etiqueta utilizada para la creación, si no se sabe, se puede obtener de la primer linea del archivo, donde dice **Begin 600 nombre**, este nombre es la etiqueta.

Se designa la Red UNIX a la red que interconecta las máquinas con sistema operativo UNIX. Es un sistema operativo con más de un centenar de comandos diseñados para realizar una amplia gama de funciones. Está diseñado en forma tal que es posible adaptarlo a los requerimientos particulares de sus usuarios en una gran cantidad de maneras. Este sistema operativo ha sido instalado en una gran cantidad de plataformas de hardware distinto.

A continuación se darán a conocer los comandos básicos de UNIX:

- **awk [-Fc] [modelo{acción} [archivos]]:** Transforma archivos ejecutando ciertas acciones cada vez que encuentra un modelo. Lenguaje de programación.

- **cat arch [arch2...]:** Concatena y escribe en pantalla el contenido de uno o mas archivos de texto arch [arch2...].

- **cd:** cambia el directorio actual de trabajo.

- **chmod modo archivo:** Cambia el modo de uso (protección) de un archivo.

- **chmod 755:** otorga el permiso de acceso al archivo deseado. Por ejemplo **chmod 755 archivo.html**, donde archivo.html es un archivo con etiqueta de hipertexto.

- **chgrp archivo:** Cambia el grupo asignado a archivo.

- **chown propietario archivo:** Cambia el propietario asignado a archivo.

- **cmp archivo1 archivo2:** Compara los dos archivos señalados.

- **cp arch1 arch2:** Copia el archivo llamado arch1 a otro llamado arch2.

- **dd...:** Convierte el formato y copia un archivo.
- **diff archivo1 archivo2:** Compara los archivos señalados línea a línea.
- **ed arch:** Llama al editor de texto ed, para trabajar con el archivo de texto arch.
- **ex arch:** Llama al editor de texto ex, para trabajar con el archivo de texto arch.
- **file arch:** Determina el tipo (texto, programa, etc) del o los archivos indicados.
- **find directorio condicion:** Busca los archivos que satisfacen la condicion señalada a partir del directorio indicado.
- **ll:** muestra los archivos del directorio actual indicando aquellos que tienen o no permiso de acceso.
- **logout (^D):** termina una sesión en UNIX.
- **ls:** muestra una lista de archivos en el directorio actual. **ls -l** proporciona información mas detallada, incluyendo el tamaño del archivo, el dueño y la fecha en que se creó. **ls -F** muestra directorios (se identifican con /), archivos ejecutables (se identifican con *), y enlaces (se identifican con @).
- **ln arch1 arch2:** Establece arch2 como un sinónimo para el arch1.
- **login:** Comando para entrar en cuenta con el sistema operativo UNIX.
- **lpr archivo:** Ordena la impresión del archivo de texto indicado.
- **man:** despliega la documentación en linea de UNIX acerca de un comando. Por ejemplo **man pwd**, da la información respecto a pwd.
- **mdir:** Revisa los archivos o directorios que contiene el disco en la unidad de disco "a".
- **mkdir:** crea un nuevo directorio sin ningun archivo en él. Por ejemplo: **mkdir nuevo.txt**, crea el directorio nuevo.txt

- **more arch:** Escribe en pantalla el contenido del archivo de texto arch, página por página.
- **mv:** renombra archivos. Por ejemplo **mv nombre1 nombre2**, renombra un archivo de nombre1 a un archivo de nombre2.
- **quota -v:** da a conocer la cuota actual en la cuenta personal, es decir, los kilobytes disponibles en la memoria de dicha cuenta.
- **pwd:** muestra el directorio de trabajo actual.
- **passwd:** Asocia a una cuenta una palabra clave sin la cual no se podrá acceder a ella en lo sucesivo.
- **rm:** borra uno o más archivos. Por ejemplo: **rm cota1 cota2**, borra dos archivos llamados cota1 y cota2. Con **rm -i** se pide confirmación antes de borrar cada archivo.
- **rmdir** borra el directorio deseado, dicho directorio debe estar vacío, es decir, no debe tener ningún archivo en él.
- **users:** permite saber que usuarios están trabajando en ese momento en el sistema, y la máquina que están utilizando.
- **sort archs:** Ordena el o los archivos indicados.
- **su:** permite cambiarse de una cuenta a otra, sin necesidad de terminar la sesión para realizar dicho cambio.
- **tail n archivo:** Escribe las últimas n líneas de archivo.
- **tar...:** Manejador de archivos en cintas magnéticas.
- **tee arch:** Copia la entrada estándar en la salida estándar y en el archivo arch.
- **tr cad1 cad2:** Filtro que traduce los caracteres en cad1 a los de cad2.
- **vi arch:** Llama al editor visual, para trabajar con el archivo de texto arch.

• **wc arch:** Cuenta los caracteres, palabras y líneas del archivo de texto arch.