Resumen comandos en GNU/Linux

1. Comandos básicos

En las tablas que se presentan a continuación se tiene la lista de comandos mas frecuentes.

Comando/Sintaxis	Descripción	Ejemplos
cat *fichero1 [...ficheroN]*	Concatena y muestra archivo/s.	`cat /etc/passwd` `cat dict1 dict2 dict`
cd *[dir]*	Cambia de directorio.	`cd /tmp`
chmod *permisos fichero*	Cambia los permisos de un archivo.	`chmod +x miscript`
chown *usuario:grupo fichero*	Cambia el dueño un archivo.	`chown nobody miscript`
cp *fichero1 ...ficheroN dir*	Copia archivo/s.	`cp foo foo.backup`
diff [-e] *fichero1 fichero2*	Encuentra diferencias entre archivos.	`diff foo.c newfoo.c`
du [-sabr] *fichero*	Muestra el tamaño de un directorio.	`du -s /home/`
file *fichero*	Muestra el tipo de un archivo.	`file archivo_desconocido`
find *dir test acción*	Encuentra archivos.	`find . -name ".bak" -print`
grep [-cilnv] *expresión archivos*	Busca patrones en archivos.	`grep mike /etc/passwd`
head [-count] *fichero*	Muestra el inicio de un archivo.	`head prog1.c`
mkdir *dir*	Crea un directorio.	`mkdir temp`
mv *fichero1 ...ficheroN dir*	Mueve un archivo(s) a un directorio.	`mv a.out prog1`
mv *fichero1 fichero2*	Renombra un archivo.	`mv .c prog_dir`
less / **more** *fichero(s)*	Visualiza página a página un archivo.	`more muy_largo.c` `less muy_largo.c`
ln [-s] *fichero acceso*	Crea un acceso directo a un archivo.	`ln -s /users/mike/.profile`
ls	Lista el contenido del directorio.	`ls -l /usr/bin`

pwd	Muestra la ruta del directorio actual.	`pwd`
rm *fichero*	Borra un fichero.	`rm foo.c`
rm -r *dir*	Borra un todo un directorio.	`rm -rf prog_dir`
rmdir *dir*	Borra un directorio vacío.	`rmdir prog_dir`
tail [-count] *fichero*	Muestra el final de un archivo.	`tail prog1.c`
vi *fichero*	Edita un archivo.	`vi .profile`

2. Comandos de manipulación de archivos y directorios

Comando/Sintaxis	Descripción	Ejemplos
at [-lr] *hora [fecha]*	Ejecuta un comando mas tarde.	`at 6pm Friday miscript`
cal *[[mes] año]*	Muestra un calendario del mes/año.	`cal 1 2025`
date *[mmddhhmm] [+form]*	Muestra la hora y la fecha.	`date`
echo *string*	Escribe mensaje en la salida estándar.	`echo "Hola mundo"`
finger *usuario*	Muestra información de un usuario en la red.	`finger nn@maquina.aca.com`
id	Número id de un usuario.	`id usuario`
kill [-señal] *PID*	Mata un proceso.	`kill 1234`
man *comando*	Ayuda del comando especificado.	`man gcc` `man -k printer`
passwd	Cambia la contraseña.	`passwd`
ps [-axiu]	Muestra información sobre los procesos que se están ejecutando en el sistema.	`ps –ux`
who / **rwho**	Muestra información de los usuarios conectados al sistema.	`who`

3. Comandos más frecuentes

UNIX/Linux	MS-DOS	Descripción
cat	**type**	Ver contenido de un archivo.
cd, chdir	**cd, chdir**	Cambio el directorio en curso.
chmod	**attrib**	Cambia los atributos.
clear	**cls**	Borra la pantalla.
cp	**copy**	Copia uno o varios archivos.
ls	**dir**	Ver contenido de directorio.
mkdir	**md, mkdir**	Creación de subdirectorio.
more	**more**	Muestra un archivo pantalla por pantalla.
mv	**move**	Mover un archivo o directorio.
rmdir	**rd, rmdir**	Eliminación de subdirectorio.

4. Comandos en background o segundo plano

Linux, como cualquier sistema Unix, puede ejecutar varias tareas al mismo tiempo. En sistemas monoprocesador, se asigna un determinado tiempo a cada tarea de manera que, al usuario, le parece que se ejecutan al mismo tiempo. **Para ejecutar un programa en background o segundo plano, basta con poner el signo ampersand (&)** al término de la línea de comandos. Por ejemplo, si se quisiera copiar el directorio /usr/src/linux al directorio /tmp.

```
#cp -r /usr/src/linux /tmp &
```

```
#
#
[Done] cp -r /usr/src/linux /tmp
#
```

Si se hubiese ejecutado el programa y no se hubiese puesto el ampersand, se podría pasarlo a background de la siguiente manera:

1. Se **suspende la ejecución** del programa, pulsando Ctrl+Z.
2. Se ejecutamos la siguiente orden: **bg**.

5. El intérprete de comandos o shell

El intérprete de comandos es el **programa que recibe lo que se escribe en el terminal** y lo convierte en instrucciones para el sistema operativo. En otras palabras, el objetivo de cualquier intérprete de comandos es **ejecutar los programas que el usuario teclea en el prompt** del mismo. El prompt es una indicación que muestra el intérprete para anunciar que espera una orden del usuario. Cuando el usuario escribe una orden, el intérprete ejecuta dicha orden.

En dicha orden, puede haber programas internos o externos: los programas internos son aquellos que vienen incorporados en el propio intérprete, mientras que los externos son programas separados. En el mundo Linux/Unix existen tres grandes familias de shells. Éstas se diferencian entre sí básicamente en la sintaxis de sus comandos y en la interacción con el usuario.

Sintaxis de los comandos

Los comandos tienen la siguiente sintaxis:

```
# programa arg1 arg2 ... argn
```

Se observa que, en la línea de comandos, **se introduce el programa seguido de uno o varios argumentos**. Así, el intérprete ejecutará el programa con las opciones que se hayan escrito. Cuando se quiere que el comando sea de varias líneas, se separa cada línea con el carácter barra invertida (\). Además, cuando se quiere ejecutar varios comandos en la misma línea, los separa con punto y coma (;). Por ejemplo:

```
# make modules ; make modules_install
```

En los comandos también se pueden utilizar los comodines o metacaracteres. Este ejemplo listará los archivos con extensión .c.

```
ls *.c
```

El signo de interrogación (?) es equivalente a un único carácter. El ejemplo lista el archivo curso.tex completando el último carácter y aquellos cuya extensión sea te y cualquier carácter.

```
ls curso.te?
```

Un conjunto de caracteres entre corchetes es equivalente a cualquier carácter del conjunto.

```
ls curso.t[aeiou]x
```

Variables de entorno

Una variable de entorno es un nombre asociado a una cadena de caracteres. Dependiendo de la variable, su utilidad puede ser distinta. Algunas son útiles para no tener que escribir muchas opciones al ejecutar un programa, otras las utiliza el propio shell. La siguiente tabla muestra la lista de variables más usuales.

Variable	Descripción
DISPLAY	Donde aparece la salida de X-Windows.
HOME	Directorio personal.
HOSTNAME	Nombre de la máquina.
MAIL	Archivo de correo.
PATH	Lista de directorios donde buscar los programas.
PS1	Prompt.
SHELL	Intérprete de comandos por defecto.
TERM	Tipo de terminal.
USER	Nombre del usuario.

La forma de definir una variable de entorno cambia con el intérprete de comandos, se muestra **tcsh** y **bash** siendo los dos más populares en el ámbito Linux.

bash:
```
export VARIABLE=Valor
```
tcsh:
```
setenv VARIABLE Valor
```

bash:
```
export DISPLAY=localhost:0.0
```
tcsh:
```
setenv DISPLAY localhost:0.0
```

Alias

Un alias es un **nombre alternativo para un comando**. Así, en lugar de escribir el comando tal cual, escribiríamos el alias de dicho comando. Un alias se puede definir por varios motivos, por ejemplo: dar nombres familiares a comandos comunes o dar nombres cortos a comandos largos.

```
alias md='mkdir'
```

Crearía un alias para el comando mkdir, similar al de DOS.

```
alias tbz2='tar -cv --use-compress-program=bzip2 -f'
```

Crearía un alias para el comando tar para que use el compresor bzip2 en lugar de gzip.

6. Reutilización de comandos

El shell almacena una historia de los comandos que el usuario ha escrito. Por medio de esta historia es posible volver a ejecutar una orden que ya se ha escrito anteriormente sin tener que escribirla de nuevo. El comando **history** muestra la secuencia de comandos, con un número a su izquierda. Con este número es posible llamar de nuevo el comando utilizando el carácter admiración (!). Por ejemplo history retorna:

```
1 history
2 ls
3 cd public_html
4 ls
5 rm *.bak
6 history
```

Para ejecutar nuevamente el comando rm *.bak solo es necesario escribir !5. También se puede pedir el **último rm ejecutado escribiendo !rm**. El último comando **se repite con doble admiración "!!"**. Es posible también editar el último comando utilizando el carácter "^".

Además, los nuevos shells permiten navegar por el historial y editar los comandos usando las flechas del teclado.