Empezando con GNU

José M. González
Compilado para: Colegio Salesiano Santo Domingo Savio
info@salesianosdosa.com

Palabras Clave: Terminal, UNIX, Ubuntu, linux, shell, command, commands, bash, tutorial, guide, introduction, FDL, FSF, GNU
Clasificación ACM: D.4.0 OPERATING SYSTEMS - General

[1] lo cual es bastante proteccionista y anti liberal.

Mayo, 2012. Madrid, España.

Estructura del Sistema

GNU

Intorducción

GNU es un *sistema operativo* que es *software libre*. *Richard Stallman* comenzó el proyecto GNU en el año 1984 para desarrollar un sistema operativo que sustituyera a UNIX y fundó la Free Software Foundation (FSF) . El logotipo del proyecto es un ñu, dado que su pronunciación es muy parecida. La palabra gnu también suena como *new*, que significa nuevo. Sus siglas —GNU— son un acrónimo de `GNU is Not UNIX`. En tanto que la `G` significa `GNU`, podríamos volver a sustituir y quedaría: `GNU is not Unix is Not Unix`. Y así sucesivamente hasta el infinito donde leeríamos un `GNU` e infinitas veces `is not UNIX`. Por eso, se dice que es un acrónimo recursivo, lo cual se ha demostrado que sólo te hace mucha gracia si eres programador.
Inicialmente el software de UNIX se distribuía a través del código fuente. Los usuarios recibían el programa y lo compilaban —lo hacían ejecutable— para la máquina que lo iba a ejecutar. Los programadores iban aprendiendo unos de otros mirando el trabajo de los demás. Dos hechos cambiaron el escenario de manera radical: la introducción de las contraseñas y, la aparición de las patentes. Las leyes estadounidenses, en especial, prohibían mirar el código que lo

componía: algo así como si nos prohibieran abrir el capó de nuestro coche[1].

La Labor de GNU

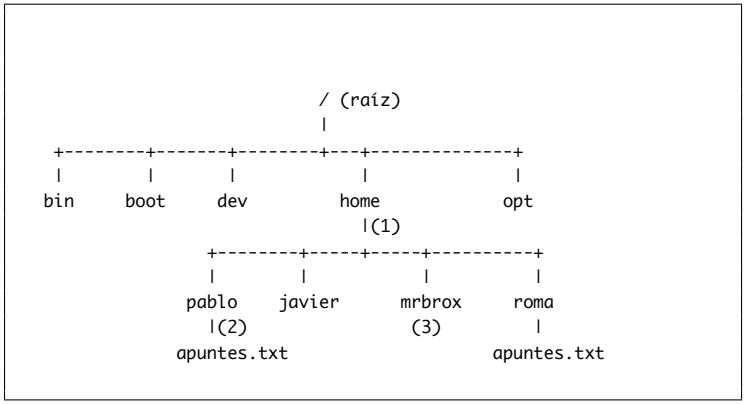
Richard Stallman anunció en ARPANET y USENET en septiembre de 1983 su intención de desarrollar un sistema operativo compatible con UNIX . Esto significa principalmente la capacidad de contar con una linea de comandos con las mismas herramientas — los comandos reciben el nombre de herramientas — con las que contaba su predecesor. Para ello, además se creó un sistema/metodología de desarrollo que incluyen un editor —emacs—, un sistema de compilado con distintos lenguajes de programación, herramientas de depuración —quitar errores a los programas— y, un sistema de compilación e instalación compatible con muchísimos sistemas operativos —autotools—. Dicen las malas lenguas que algunos desarrolladores de sistemas operativos usan estas herramientas para desarrollar los suyos.
EL sistema operativo GNU tiene un núcleo The Hurd que todavía no es estable. De momento se está usando el núcleo de Linux que da lugar al sistema operativo GNU/Linux. Por lo tanto, la extendida denominación Linux a secas es incorrecta. debian , el sistema operativo que ha sido tomado por el archiconocido ubuntu como base de sus desarrollos, está trabajando activamente en el desarrollo de este *kernel*.
Además de todo lo anterior, la Free Software Foundation (FSF) ha sido pródija en la creación de patentes que garanticen los principios del software libre y han peleado en los juzgados contra las violaciones de estas patentes. Estos principios / libertades son las siguientes:

- La libertad de usar el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar el funcionamiento del programa, y adaptarlo a las necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias, con lo que puede ayudar a otros (libertad 2).
- La libertad de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (libertad 3). De igual forma que la libertad 1 el acceso al código fuente es un requisito previo.

Directorios y Permisos

Estructura de Árbol

El sistema de ficheros tiene estructura de árbol, donde desde un punto común — *raíz* / *root* — se puede localizar un fichero concreto a través de una *ruta*.
Directorio Raíz (root): / lo representa.
Esquemáticamente podemos representarlo, asÍ:



El sistema operativo nos provee del comando `tree`, escasamente empleado en la práctica, que representa el árbol de directorios y de `ls`, de uso frecuente, que muestra los ficheros del directorio actual.

```
user@computer:~$ tree
   .
   ├── comandos
   ├── editor_vi
   ├── ejercicios
```

```
   |   └── metro
   |       ├── intersecc
   |       ├── linea.01
   |       ├── linea.02
   |       ├── linea.03
   |       ├── linea.04
   |       ├── linea.05
   |       └── linea.ramal
   ├── fhs
   ├── filtros
   ├── introduccion_shell
   ├── La estructura del sistema de archivos en Linux.pdf
   ├── manuscript
   |   ├── comandos.tex
   |   ├── directorios.tex
   └── shell
user@computer:~$
```

```
user@computer:~/Documentos/$ ls
examenes  informacion  practicas  software
user@computer:~/Documentos/$
```

Las Rutas

Un fichero queda identificado en el disco duro por su *nombre completo*, que es la *ruta* desde el directorio raíz y el nombre del fichero. Dos ejemplos serían:

Nombre completo: /home/pablo/apuntes.txt
Nombre completo: /home/roma/apuntes.txt

Como el nombre completo utiliza toda la ruta desde el raíz, *si empieza por (/), entonces es una* ruta absoluta.
Ruta relativa: Especifica qué directorios hay que atravesar para llegar a un archivo *desde donde yo estoy ahora* — directorio actual de trabajo —.
Para ir de 1 a 2, o de 2 a 3 usamos las siguientes rutas relativas:

(1) ->(2) => pablo/apuntes.txt
(3) ->(2) => ../pablo/apuntes.txt

Para especificar el directorio superior —en el árbol de directorios—, se escribe .. y se lee "punto punto" para evitar confusión con los dos puntos. Cuando queramos especificar el directorio actual usamos un punto (.).
Nótese que las rutas relativas no empiezan por /.

Los inodos

Los ficheros son un conjunto de datos —también puede ser un conjunto de 0 datos— que queda perfectamente identificado por número: el *inodo*.
Sobre el sistema de archivos se crea una tabla, a modo de diccionario, donde se va asociando uno o varios nombres sobre el mismo fichero —en realidad sobre su *inodo*—.
El fichero se mantiene en el sistema de archivos mientras tenga asociado, al menos, un nombre.

Los Permisos

Sobre cada archivo se define un conjunto de permisos de lectura (r), escritura (w) y ejecución (x).

FHS

Introducción

Filesystem Hierarchy Standard: Estándar jerárquico del sistema de ficheros.

Tipos	Estáticos:	Los datos cambian poco.
	Dinámicos:	Los datos cambian a menudo.
	Compartidos:	Contiene ficheros que pueden usar varios ordenadores / usuarios.
	Restringidos:	Depende de este ordenador solamente.

Directorios Estándar

/	Raíz
/bin	(binaries) Ejecutables
/boot	Arranque (Gestor de arranque GRUB2)
/dev	(devices) dispositivos
	serial device attachment
/etc	Configuración de los programas
/home	Directorios de los usuarios
/media	Montados los dispositivos extraibles.
/opt	Donde se lo instalan los paquetes
	opcionales de las aplicaciones
/proc	Ficheros que representan el estado de un proceso.
/root	Home del administrador
/sbin	(system binaries) Ejecutables del administrador.
/var	Donde se ponen los ficheros que pueden
	cambiar los programas.
/usr	Dónde los usuarios instalan los programas.

Terminal

Shell

Cuando iniciamos una sesión usando el terminal podríamos encontrar algo parecido a la siguiente pantalla:

```
user@computer:~/Documentos/code/c/ncurses$ ls -lh
total 32K
-rwxr-xr-x 1 txema txema  288 2010-04-27 01:40 01_chgat.c
-rwxr-xr-x 1 txema txema    0 2010-04-27 01:43 a.out
-rwxr-xr-x 1 txema txema 8,5K 2010-04-27 01:40 chgat
-rwxr-xr-x 1 txema txema 8,7K 2010-04-27 21:55 window
-rwxr-xr-x 1 txema txema 1,3K 2010-04-27 21:55 window.c
user@computer:~/Documentos/code/c/ncurses$
```

De todas las lineas que vemos sólo la última es una linea activa. Se llama: *linea de comandos*, y la maneja un programa que se llama Readline.
La linea de comandos tiene dos partes –separadas por el signo $ en el ejemplo–: el *prompt*, y el *área de escritura*.
En el *prompt* es habitual ver el nombre del usuario, separado por el signo *at* –@–, que se lee "en". –y no "arroba"–. Significa que la persona que está manejando la computadora es –tiene la identidad de– `user` en esta máquina –`computer`–.
El símbolo ~ se suele conocer con el nombre de *tilde* y quiere decir: el directorio *home* del usuario. De esta manera, la cadena completa `~/Documentos/code/c/ncurses` hace referencia al directorio actual de trabajo.
Por último, vemos el signo $ que quiere decir que somos un usuario normal. El *superusuario*, también conocido como *root* tiene el símbolo # cerrando su *prompt*.

EDICIÓN ONLINE

El término *online*, contrariamente a lo que la gente cree, significa en la linea de comandos. Veamos las teclas y comandos para editarla de manera eficiente.

Moverse por las pestañas.

Para abrir y cerrar terminales:

Ctrl Alt t:	Abrir un terminal
Ctrl Mays t:	Abre una nueva pestaña.
Alt <número>:	Cambia a la pestaña <número>.
Ctrl Alt f<[1-7]>:	Cambiar entre terminales.
Tab:	Autocompletar. 1 vez completa el comando. 2 veces muestra todas las posibles opciones.
exit:	Cierra el (pseudo)terminal sale del sistema.
reset:	Reinicia la terminal.
logout:	Cierra sesión en la terminal.

El terminal 7 tiene arrancadas las X, conociéndose como X al sistema de ventanas.

Manejo básico de ficheros.

Estos son los comandos fundamentales. Para más información ver la sección comandos –pág. 7–:

pwd:	Present Working Directory - Directorio actual de trabajo.
ls:	list - Vemos la lista de ficheros.
cd:	Cambiar al directorio directorio. cd <directorio>
rmdir:	(remove dir) Elimina un directorio vacío. rmdir <archivo>
rm:	(remove) Elimina un archivo. rm <archivo> Elimna un directorio con contenido: rm -rf <archivo>
cp:	Copia un fichero.
mv:	Cambiar el nombre o mover.

Combinaciones de teclas

Extraídos de la biblioteca Readline presentamos los comandos fundamentales de edición:

Ctrl R	Busca en el historial.
Ctrl D	Final del fichero.
Ctrl Z	Pasar un programa a segundo plano (background)
Ctrl C	Interrumpe un programa. (break signal)
Ctrl A	Ir al prinicipio de linea.
Ctrl E	Ir al final de la linea.
Ctrl K	Kill (cortar)
Ctrl Y	Pegar (yank)
Ctrl W	Corta un carácter.
Crtl Mays C	Copia
Ctrl Mays V	Pega

Comandos Útiles

history:	Da los comandos tecleados. history history <num> ó history 10
!<num>:	Ejecuta un comando anterior.
!!:	Ejecuta el último comando.
!<comando>:	Ejecuta el último comando con sus parámetros correspondientes.
<comando> !*:	Coge los argumentos de la última orden.

Autocompletar

Las siguientes teclas permiten autocompletar:

Tab:	1 vez completa el comando. 2 veces muestra todas las posibles opciones.
Esc <letra especial>:	Completa si puede.
Ctrl X <letra especial>:	Muestra todas las posibles opciones.
echo:	Hace eco, expandiendo los metacaracteres.

Cuando no se puedo completar suena un beep.
Las teclas especiales son:

~:	completa un nombre de usuario.
@:	completa un nombre de máquina.
$:	completa una variable de entorno.
!:	completa un nombre de comando o un nombre de fichero o nombres de páginas del manual.

Metacaracteres

*

*	Patrón. Wildcard. Globbing. "Cualquier cosa".
r	cualquier cosa y una r. Cualquier cosa que acabe en r.
r*	Cualquier cosa que empiece por r.
r*	Cualquier cosa que tenga una r.
*.doc	Cualquier fichero con extensión doc.

?

Cualquier carácter.

usuari?.txt

[ ]

Conjunto de caracteres.

usuari[ao].txt	
usuario[23456789].txt	
usuario[2-9].txt	
usuario[13-9].txt	usuario1 y del 3 al 9.
usario[^0-9].txt	El acento circunflejo (*caret*) invierte el conjunto de caracteres.
usario[!0-9].txt	Igual que el *caret*.

\

Secuencia de escape.
Quita el significado de metacaracter.

Espacio en blanco
Separa las palabras.

{ }

Agrupar partes comunes de las palabras. Sacan la parte común de las palabras.

```
ls {abue,fi}lete
```

equivale a:

```
ls abuelete filete
```

Ejemplos:

`ls [A-Z]*`	Palabras que empiezan por mayúsculas.
`ls [^0-9]*`	Palabras que no empiezan por números.
`ls *[aeiouAEIOU]`	Todo lo que acabe en vocal.

Los Comandos

---

cp - Copia dos ficheros.

```
cp [<fichero>]+ (<nuevo_fichero> | <directorio)
```

```
cp f1.txt f2.txt dir # Copia f1.txt y f2.txt en el directorio dir.
```

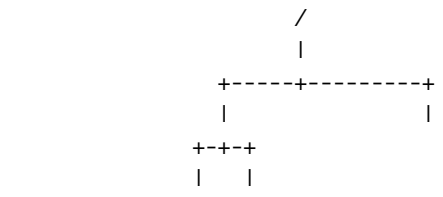
```
cp f1* dir # Copia los ficheros que empiezan por f en el directorio dir.
```

Opciones:

`-r` Copiar los subdirectorios recursivamente.

```
cp -r <directorio> <nuevodirectorio> # Crea la carpeta nuevodirectorio y copia los contenidos de directorio en nuevodirectorio
```

```
cp -r f1* dir # Copia todos los ficheros y directorios que empiezan por f en el directorio dir.
```



---

mv - Cambiar el nombre o mover.

```
mv <nombre_antiguo> <nombre_nuevo>
```

Notas:

El *nombre completo* de un archivo –agenda.txt– incluye su ruta –/home/user/Documentos –. Es decir:

```
/home/user/Documentos/agenda.txt #
```

Cambiar el archivo a la carpeta /home/user es lo mismo que cambiar su nombre completo por /home/user/agenda.txt
También se pueden usar rutas relativas:

```
mv agenda.txt ../agenda.txt # Mueve agenda.txt al directorio superior [a].
```

[a] .. inidca el directorio superior.

---

mv(2) - Cambiar el nombre o mover.

```
mv [<fichero>]+ <directorio>
```

Notas:

Mueve los ficheros al directorio especificado.

```
mv f1.txt f2.txt dir # Mueve los ficheros –f1.txt, f2.txt– al directorios dir.
```

---

ln - Enlaza. Asigna un nuevo nombre a un inodo.

```
ln [<nombre_viejo>]+ <nombre_nuevo>
```

`ln` crea enlaces (nombres) para asignar a los ficheros. Los ficheros no tienen nombre *per se*, sino quede están identificados por número de *inodo*. Los nombres del fichero se pueden crear a voluntad a través del comando `ln` y borrar con `rm`. Cuando ya no queda ningún nombre apuntando al *inodo*, entonces el fichero es borrado del disco duro.

```
ln f1.txt f2.txt dir  # Crea dos enlaces dir/f1.txt y dir/f2.txt
```

Opciones:

`-s` Crea enlaces simbólicos. Los enlaces simbólicos pueden apuntar a otras particiones o volúmenes. Se recomienda usar rutas absolutas para el archivo fuente.

```
ln -s archivo_fuente nombre_del_link    # Crea nombre_del_link. Editarlo es lo mismo que editar archivo_fuente.
```

---

file - Analiza el contenido de un archivo.

```
file <fichero>
```

```
user@computer:~/Documentos$ file *
comandos:                          UTF-8 Unicode text
editor_vi:                         data
ejercicios:                        directory
fhs:                               UTF-8 Unicode text
filtros:                           UTF-8 Unicode text
introduccion_shell:                UTF-8 Unicode text
sistema de archivos en Linux.pdf: PDF document, version 1.4
manuscript:                        directory
shell:                             UTF-8 Unicode text
user@computer:~/Documentos$
```

---

more - Permite ver un fichero pantalla a pantalla.

```
more <fichero>
```

Notas:

`more` estaba pensando para leer un fichero pantalla a pantalla. Como la memoria era muy cara en el tiempo en el que se diseñó, no cargaba todo el programa a la vez, y por tanto, no es posible ir para atrás. Una vez dentro, la tecla Espacio avanza una pantalla, Intro una linea y, q sale. Las demás teclas son parecidas a las del editor vi –pág 22 –.

---

less - more mejorado.

```
less <fichero>
```

Notas:

Con el abaratamiento de la memoria se pensón en cargar todo el fichero en memoria y permitir la navegación bidireccionalmente (volver para atrás). Fue entonces cuando se desarrolló `less`, mejorando a `more`. Su nombre viene es un juego de palabras con el dicho inglés *less is more than more*.

`less` toma todos los movimientos de editor `vi`. En concreto, Ctrl B retrocede una pantalla.

---

od - (octal dump) - volcado de octectos .

`od <fichero>`

Permite ver lo que está escrito en el disco duro octeto a octeto[2] .

```
user@computer:~/Documentos/nix-dir$ cat en_un_lugar.txt
En un lugar de la mancha
user@computer:~/Documentos/nix-dir$ od en_un_lugar.txt
0000000 067105 072440 020156 072554 060547 020162 062544 066040
0000020 020141 060555 061556 060550 000012
0000031
user@computer:~/Documentos/nix-dir$
user@computer:~/Documentos/nix-dir$
```

Opciones:

`-c` Muestra los caracteres.

`-b` Muestra los bytes.

```
user@computer:~/Documentos/nix-dir$  od -cb en_un_lugar.txt
0000000   E   n       u   n       l   u   g   a   r       d   e       l
    105 156 040 165 156 040 154 165 147 141 162 040 144 145 040 154
0000020   a       m   a   n   c   h   a  \n
        141 040 155 141 156 143 150 141 012
0000031
user@computer:~/Documentos/nix-dir$
```

[2] 8 bits son un byte u octeto.

---

lpr - Imprime un fichero.

`lpr <fichero>`

Opciones:

`-P <destino>` Especifica el nombre de la impresora de destino.

`-#<copias>` Especifica el número de copias.

---

lpq - Cola de impresión (ficheros que faltan por imprimir).

`lpq`

Notas:

Si bien se suele trabajar con la impresora por defecto, es posible especificar el servidor de impresión mediante opciones.

---

lprm - Borra un trabajo de la cola de impresión

`lprm <identificador>`

```
lprm 331  # Elimina el trabajo de impresión con el número 331
```

xargs - Concatena un comando con un filtro.

```
xargs [comando [argumentos]]
```

Ejemplo:

```
find ~ -name *.pdf | xargs rm # Borra todos los pdf's
```

Equivalente a:

```
find ~ -name *.pdf -exec rm  #
```

du - Disk Usage. Nos da la ocupación en disco.

```
du [FICHERO]
```

Opciones:

-a (all) Busca en los subdirectorios.

-s (summary) Resumen

df - Disk free.

```
df [FICHERO]
```

Opciones:

Información por columnas

Nombre del dispositivo.
Espacio disponible
Bloques usados.
Bloques disponibles
Porcentaje de ocupación.
Punto de montaje.

cal - Calendario

```
cal [-3hjy] [-A number] [-B number] [[month] year]
```

Notas:

El calendario mostrado comienza en domingo, muestra fechas del calendario gregoriano — no en todos los países se adoptó el mismo año — y, no sirve para fechas anteriores a 1586.

```
user@computer:~$ cal 12 2020
   Diciembre 2020
do lu ma mi ju vi sá
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

user@computer:~$
```

bc - Calculadora básica

```
bc [<fichero>]
```

Notas:

Realiza sumas, restas, multiplicaciones, divisiones, restos y potencias. Permite, además, variables y bucles. bc, también se puede usar como filtro.

```
user@computer:~$ echo ”2+3” | bc
5

user@computer:~$
```

Podemos cambiar la base de entrada *ibase* y la base de salida *obase* de datos una vez. El operador punto (.) representa el último resultado.

```
user@computer:~$ bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type ‘warranty’.
numero=64
numero*=2
numero
128
.
128
obase=16
.
80
^D
user@computer:~$
```

spell - Comprueba la correción ortográfica de las palabras

```
spell <fichero>
```

Notas:

Cuando la entrade es correcta no muestra nada en la salida.

```
user@computer:~$ echo cinco | spell
cinco
user@computer:~$ echo five | spell
user@computer:~$ echo one chu three four faif | spell
chu
faif
user@computer:~$
```

Opciones:

`-b, --british` Usa el diccionario británico.

`-d, --dictionary=FILE` Especifica qué fichero se usará como diccionario.

Los diccionarios se guardan — en debian — en /usr/share/dict/. Un ejemplo de lo que se obtiene al instalar `spell`:

```
user@computer:~$ ls /usr/share/dict/
american-english  README.select-wordlist  words
british-english   spanish                 words.pre-dictionaries-common

user@computer:~$
```

---

units - Conversor de unidades.

```
units
```

Uso:

```
user@computer:~$ units ”1 euro” ”dollar”
        * 1.4128
        / 0.70781427
user@computer:~$ units
2411 units, 71 prefixes, 33 nonlinear units

You have: 1 euro
You want: dollar
        * 1.4128
        / 0.70781427
You have: 1hp
You want: watt
        * 745.69987
        / 0.0013410221
You have: ^D
```

Compresión

---

pack - Comprime

```
pack [<fichero>]+
```

Notas:
Es el algortimo de compresión más antiguo de los que se usan. Añade la extensión `.z` a los ficheros comprimidos. Para descomprimir se usa `unpack`. Para mandar el resultado de descomprimir a la salida estándar se usa `pcat`.

```
pack f1 f2  # Crea f1.z y f2.z
```

---

compress - Comprime

```
compress
```

Opciones:

`-d` Originalmentes se descomprimía con esta opción. Hoy se usa uncompress.real

```
compres f1 f2  # Crea f1.Z y f2.Z
```

Para mandar el resultado de descomprimir a la salida estándar se usa `zcat`.

---

gzip - Comprime

```
gzip [<fichero>]+
```

Notas:
`gzip` es *copyleft* y se basa en el algoritmo *Lempel-Ziv*

```
gzipf1 f2  # Crea f1.gz y f2.gz
```

Para descomprimir se usa el comando gunzip. Para mandar el resultado de descomprimir a la salida estándar se usa gzcat.

---

bzip2 - Comprime

```
bzip2 [<fichero>]+
```

Notas:

Se basa en el algoritmo *Burrows-Wheeler*, que es el que más comprime. El fichero recibe la extensión .bz2 cuando se comprime. En alguna ocasión puede resultar un poco lento.

```
bzip2 f1 f2 # Crea f1.bz2 y f2.bz2
```

Para descomprimir se usa el comando bunzip2. Para mandar el resultado de descomprimir a la salida estándar se usa bzcat.

---

tar - Utilidad para archivar cintas

```
tar [pathname]*
```

Notas:

Pese a su nombre, es el programa que se usa mayoritariamente para el empaquetado y compresión — aunque tar no comprime directamente— de directorios.

Opciones:

`-A, --catenate, --concatenate` Concatena a un archivo *tar* existente.

`-c, --create` Crea un nuevo archivo.

`-r, --append` Añade más ficheros al final del *tar*.

`-x, --extract, --get` Extrae los ficheros del *tar*.

`-t, --list` Lista los archivos sin descomprimirlos.

`-f, --file ARCHIVE` Usar un fichero en vez de la cinta.

`-v, --verbose` Ser prólijo en las explicaciones.

Ejemplos:

Comprimir:

```
tar cvzf <comprimido.tar.gz> <directorio/> # Comprime el directorio.
```

Descomprimir:

```
tar xvzf <comprimido.tar.gz> # Descomprime el .
```

---

find - Busca ficheros.

```
find <ruta> [<expresión>]
```

Opciones Comunes:

`-name <pattern>` Nombre del fichero.

`-size <n[cwbkMG]>` Con un tamaño de n (bytes, palabras, bloques, kilobytes, megabytes o gigabytes).

`-type <c>` El fichero es del tipo c, donde c puede tomar los siguientes valores:

b	dispositivo de bloque (buffered)
c	dispositivo de caracter (unbuffered)
d	directorio
p	Tubería con nombre - named pipe (FIFO)
f	fichero normal
l	Fichero simbólico
s	socket
D	door (Solaris)

[3] n: exactamente n, +n:mayor que n, -n:menor que n.

Opciones de Búsqueda:

`-d, -depth` Profundidad de la recursión — cantidad de subdirectorios en los que buscar —.

`-amin <n>` Tiempo, en minutos, en el que se accedió al fichero[3]

`-atime` Tiempo en días en el que se ha accedido al fichero.

`-executable` El fichero es ejecutable

`-links <n>` Número de enlaces que tiene el fichero.

`-mmin <n>` Tiempo en minutos desde que fue modificado.

`-mtime n` Tiempo en días desde que fue modificado.

`-newer <fichero>` Que ha sido modificado más recientemente que fichero.

Opciones de Ejecución:

`-ls` Escribe la información que da `ls -l`

`-print` Imprime el nombre de fichero en la salida.

`-exec <comando>` Ejecuta un comando sobre el fichero encontrado. Las llaves ocupan el lugar del nombre del fichero.

```
find / -name p* -exec file '{}' \;  # Muestra el tipo de archivo de todos los ficheros que comienzan con la letra p.
```

Ejemplos:

```
find / -name linea.01 # Busca linea.01 desde el raíz.

find metro -links 2 # Busca en el directorio metro los archivos que tengan dos enlaces.

find metro -links +2 # Busca en el directorio metro los archivos que tengan mas de dos enlaces.

find metro -mtime -7 # Ficheros modificados en los últimos 7 días.

find metro -atime +7 # Accedido hace más de 7 días.

find metro -size +40 # Ficheros que tengan más de 40 bloques. (1kb = 2bloques).

find metro -newer f1 # Ficheros que sean más recientes que f1
```

Operadores:
Los operadores son:

-a	(and) Y
-o	(or) O
!	(not) NO
()	Grupos

Filtros

Los filtros son comandos que pueden, en vez de leer de un fichero de entrada, o de la entrada estándar, leer de la salida de otro proceso — comando —.
Esto se hace a través de tuberías — *pipes* —, cuyo símbolo es `|`.
De esta manera,

```
cat quijote.txt | grep Sancho # Busca Sancho en el quiojote
```

`cat` envía el texto de quijote.txt a la salida, pero esta — la salida — está reconducida por una tubería como entrada de `grep`.
Según la función que realizan, los filtros son:

Cortan

---

head - Muestra las primeras lineas de un fichero.

```
head <fichero>
```

Opciones:

`-<num>` Muestra num lineas.

```
head -3 quijote.txt # Muestra las 3 primeras lineas del quijote.txt.
```

---

tail - Muestra las últimas lineas de un fichero.

```
tail <fichero>
```

Opciones:

Sus opciones son idénticas a las de head. Incluye además:

`-f` Pone el final del fichero y lo deja abierto, de manera que si se incluyese algo más en él quedaría reflejado en el momento.

---

split - Parte la entrade estándar en trozos de 1000 lineas y crea ficheros de nombre xaa, xab,etc.

```
split <fixhero>
```

Opciones:

`-<numero>` Especifica el número de lineas

```
split -200 f.txt trozo  # Parte f.txt en grupos de 200 lineas llamados trozo.aa, etc.
```

```
split -500 - trozo  # Parte la entrada estándar en grupos de 500 lineas llamados trozo.aa, etc.
```

`-C, --line-bytes=SIZE` Parte por *bytes* —caracteres— en vez de por lineas.

```
split -500M f.zip parte   # Parte f.zip en trozos de 500 megabytes llamadas parte.aa, parte.ab, etc.
```

[4] 1 KB es un kibibyte o kilobyte binario

Las unidades disponibles son: K, M, G, T, P, E, Z, Y y KB, MB, GB, TB, PB, EB, ZB, YB. Las primeras son múltiplos de 1024 y las acabadas en B son múltiplos de 1000[4].

---

cut - Corta cada línea

```
cut [<fichero>]
```

Opciones:

`-c, --characters=LIST` Corta por columnas.

```
cut -c1-10,21-  # Selecciona los caracteres de la columna 1 a la 10 y de la 21 en adelante.
```

`-f, --fields=LIST` Selecciona que campos —*fields*— se han de seleccionar.

```
cut -f1,5-7 notas.txt  # Selecciona los campos 1 y del 5 al 7 de notas.txt
```

El delimitador por defecto es la tecla TAB. EL parámetro LIST es idéntico al de la opción `-c`

`-d, --delimiter=DELIM` Especifica el delimitador de campo.

```
cut -d: -f1,3,5 notas.txt  # Muestra los campos 1, 3 y 5 de notas.txt, siendo los dos puntos el separador de campos.
```

Los campos de una hoja de excel o los contactos de un móvil (*csv – comma separated values*) son un conjunto de datos separados por campos.

Unen

---

paste - Concatena linea a linea.

```
paste [<fichero>]+
```

Muestra cada linea de un fichero a continuación de las lineas de otro fichero separadas por un TAB.

```
paste meses.txt semanas.txt  # Aparece el nombre de un mes y el de un día de la semana en cada línea.
```

Opciones:

`-d, --delimiters=LIST`

```
paste -d: semanas.txt meses.txt # Une los ficheros mediante los dos puntos.
```

```
user@computer:~/Documentos/nix-dir$ \
seq 12 | paste -d '-:' - meses.txt semanas.txt
1-Enero:Lunes
2-Febrero:Martes
3-Marzo:Miércoles
4-Abril:Jueves
5-Mayo:Viernes
6-Junio:Sábado
7-Julio:Domingo
8-Agosto:
9-Septiembre:
10-Octubre:
11-Noviembre:
12-Diciembre:

user@computer:~/Documentos/nix-dir$
```

El guión indica que se tome la entrada estándar (los números del 1 al 12 que produce `seq`) como contenido del fichero. El primer delimitador de la lista es el guión y el segundo es el signo =.

---

join - Combina varios ficheros en un campo común.

```
join <fichero1> <fichero2>
```

Notas:

`join` presupone que los campos están ordenados alfabéticamente, lo cual se puede conseguir con el comando `sort`.

```
$ cat menu
carne cordero
carne ternera
fruta naranja
$
```

```
$  cat dieta
carne antonio
carne juan
fruta antonio
$
```

```
$ join menu dieta
carne cordero antonio
carne cordero juan
carne ternera antonio
carne ternera juan
fruta naranja antonio
$
```

`join` dispone de muchas más opciones que se pueden consultar en el manual. Para más información: `man join`

---

cat - (concatena) Ver ficheros pequeños.

`cat [<fichero>]+`

Notas:

Cuando no hay ficheros lista la entrada estándar. También se puede poner un guión (-) como nombre de fichero. Para introducir la marca de final de fichero desde la consola, úsese Ctrl D

```
cat f1.txt f2.txt  # Muestra el contenido de f1.tx y f2.txt
```

Cambian

---

tr - Traduce o borra caracteres

`tr SET1 [SET2]`

Uso:

```
tr aeiou aaaaa  # Cambia las vocales minúsuclas por aes.

tr aeiou a  # Cambia las vocales minúsuclas por aes.

tr A-Z _  # Cambia las mayúsculas por subrayados.
```

Opciones:

`-c, -C, --complement` Usa el conjunto complementario del SET1

`-d, --delete` Borra, en vez de traducir los caracteres del SET1

`-s, --squeeze-repeats` Quita los elementos repetidos

```
tr -s aeiou a  # Quita las aes repetidas.
```

Notas:

`tr` sólo puede actuar como filtro y no admite ficheros.

---

sed - Editor no interactivo.

`sed <guión> [<fichero>]`

Reorganizan

---

sort - Ordena

`sort <fichero>`

Opciones:

`-f, --ignore-case` No tiene en cuenta mayúsculas y minúsculas para ordenar.

`-n, --numeric-sort` Toma los caracteres como cifras, de manera que el número 3 sea menor que el 17, es decir no compara el 3 sólo contra el 1.

`-r, --reverse` Ordena de mayou a menor.

---

uniq - Elimina las lineas repetidas.

```
uniq [<entrada> [<salida>]]
```

Opciones:

`-c, --count` Pone el número de repeticiones.

`-d, --repeated` Sólo pone las líneas repetidas

`-i, --ignore-case` No hay diferencias entre mayúsculas y minúsculas.

`-u, --unique` Imprime sólo las líneas no repetidas.

Analizan

---

grep - Busca una palabra en un fichero.

```
grep <palabra> [<fichero>]
```

Notas:

Si no se especifica el fichero, busca en la entrada estándar.

```
user@computer:~/Documentos/nix-dir$ grep Gallo quijote.txt
Yo, Juan Gallo de Andrada, escribano de Cámara del Rey nuestro
Juan Gallo de Andrada.
cada plana y firmado al fin dél de Juan Gallo de Andrada, nuestro
user@computer:~/Documentos/nix-dir$
```

Opciones:

`-R, -r, --recursive` Busca Recursivamente en todos los subdirectorios

```
grep -R todo *  # Busca todas las tareas pendientes en un proyecto de programación.
```

`-v, --invert-match` Invierte la búsqueda. Busca todas las lineas que no contengan.

`-i, --ignore-case` No tiene en cuenta si son mayúsculas o minúsculas (el caso).

---

wc - (word count) Cuenta lineas, palabras y caracteres de un fichero.

```
wc <fichero>
```

Opciones:

`-l` Cuenta lineas

`-w` Cuenta palabras

`-c` Cuenta bytes (caracteres)

```
user@computer:~/Documentos//nix-dir$ wc quijote.txt
  37862  384258 2198903 quijote.txt
user@computer:~/Documentos//nix-dir$
```

Comparan

---

comm - Compara conjuntos.

```
comm <fichero1> <fichero2>
```

Notas:

Compara dos ficheros linea a lina siempre que estén ordenados.

Opciones:

`-1` Suprime la columna 1 — lineas únicas del fichero 1 —.

`-2` Suprime la columna 2 — lineas únicas del fichero 2 —.

`-3` Suprime la columna 3 — lineas comunes —.

---

cmp - Compara dos ficheros cualquiera

```
cmp <fichero1> <fichero2>
```

Notas:
Si los ficheros son iguales no escribe nada. Si no lo son, señala la posición del primer octeto diferente.

---

diff - Compara dos ficheros de texto

```
diff <fichero1> <fichero2>
```

Notas:
El símbolo < indica el primer fiechero, > el segundo.

Opciones:

`-e --ed` Produce un *script* de ed

`-c -C NUM --context[=NUM]` Incluye, además de las diferencias, el contexto — normalmente 3 líneas —.

Decoran

---

patch - Aplica un `diff` a un fichero.

```
patch [<original> [<parche>]]
```

Notas:
Usualmente:

```
patch -pnum <patchfile # Sintaxis habitual
```

`patch` es capaz de regenerar un fichero `f1` a partir de otro fichero `f2` y la differencia obtenida mediante `diff -c`

```
patch doc11 doc1a2 # Modifica doc11 — lo cambia — con la información de doc1a2
```

`patch` nos pregunta si queremos guardar una copia del original antes de cambiarlo.

---

nl - Numera las líneas de un fichero.

```
nl <fichero>
```

Opciones:

`-b, --body-numbering=STYLE` Establece el estilo de numeración.
Los estilos disponibles son:

a	(all) Todas las lineas.
t	Líneas no vacías.
n	Ninguna línea
p<expresión>	las líneas que contengan la expresión.

---

pr - Prepara un fichero para imprimir

```
pr [<fichero>]+
```

Notas:

Preparar los ficheros para imprimir implica ponerles una cabecera, numerarlos, ajustar el acnho y el largo de página, así como el número de columnas.

`-m, --merge` Pone todos los ficheros en paralelo.

```
pr f1 f2 # Pone en columnas f1 y f2
```

Editando Ficheros

La Guerra de los Editores

Existen dos grandes editores de texto: emacs y vi[5]. Se conoce, en la cultura Hacker como guerra de editores, la preferencia por uno u otro. Se trata de una materia, donde casi todo el mundo toma opinión, aunque a efectos prácticos no trasciende su importancia más allá del ámbito personal.
Los dos editores forman parte de la historia de la computación y es conveniente saber manejarlos aunque sea mínimamente.

VI

Modos de vi

vi tiene:

- modo comando (Esc).
- modo inserción (i).
- modo visual (v).
- modo bloque visual (Ctrl V).

[5] Aunque nos referimos a vi abreviadamente, siempre hacemos referencia al vim.

Modo Inserción

i:	Inserta donde está el cursor.
I:	Inserta al principio de la linea.
a:	Inserta a la derecha del cursor.
A:	Inserta al final de la linea.
o:	Inserta en la linea de abajo.
O:	Inserta en la linea de arriba.

s:	Substituye
r:	reemplaza
R:	Reemplaza
~:	Cambia el caso.

Ficheros

:w	Graba el archivo con el nombre que tiene.
:w <nombre>	Graba el archivo con <nombre> y sigue editando el archivo anterior.
:r <fichero>	Lee un fichero y lo incluye donde está el cursor
:q	Salir
:q!	Forzar salir (salir sin guardar)
:wq	Grabar y salir

Movimientos

h:	izquierda
j:	baja
k:	sube
l:	derecha

0:	Principio de linea
|:	Principio de la linea
^:	Primera posición no blanca de la linea.
$:	Final de linea

w:	Avanza una palabra
e:	Va al final de la palabra
b:	Retrocede una palabra

(:	Lleva al comienzo de la frase.
):	Lleva al final de la frase.
{:	Lleva al comienzo del párrafo.
}:	Lleva al final del párrafo.
[[:	Lleva al comienzo de sección.
]]:	Lleva al final de la sección.
%:	Lleva hasta el elemento complementario (de un paréntesis a su pareja), etc.

<num>G:	Va a la linea <num>
gg:	Va a la primera linea del fichero.
Ctrl F:	Avanza una pantalla.
Ctrl B:	Retrocede una pantalla.
Ctrl D:	Avanza media pantalla.
Ctrl U:	Retrocede media pantalla.

-:	Lleva al comienzo de la linea anterior.
+:	LLeva a la linea siguiente.
H:	Lleva el cursor a la primera linea de pantalla.
3H:	Lleva el cursor a la tercera linea de la pantalla.
L:	Lleva el cursor a la ultima linea de pantalla.
2L:	Lleva el cursor a la penúltima linea de la pantalla.
M:	Lleva el cursor al medio de la pantalla.

Búsquedas

En la misma línea

f<carácter>:	Busca la siguiente aparación del carácter.
F<carácter>:	Busca la siguiente aparación del carácter, pero hacia atrás.
t<carácter>:	Busca y se queda uno antes.
T<carácter>:	Busca hacia atrás y se queda uno antes.
;:	Repite la búsqueda.
.:	Repite la búsqueda cambiando de sentido.

En todo el fichero

/<regex>	Busca adelante la expresión regular.
?<regex>	Busca hacia atrás la expresión regular.
n	Repite la última búsqueda en la misma dirección.
N	Repite la última búsqueda en dirección contraria.

Apariencia

:set nu	Muestra el número de linea.
:set nonu	Oculta el número de linea.

Observaciones

El . repite el último comando.
x p invierte dos caracteres.
Se pueden combinar movimientos con d, y, i y !. De manera que d2j borra la linea actual y dos hacia abajo. Otro ejemplo sería 80i=Esc, que introduce 80 signos = en una línea.

Cuando se repite la tecla, en vez de suministrar un movimiento el efecto suele ser afectar a toda la línea. dd borra / corta una línea entera, yy la copia.
El uso de !<movimiento><comando> le pasa el texto seleccionado al sistema operativo y lo sustituye por la respuesta. Se puede ordenar un párrafo alfabéticamente haciendo !}sort o calcular el resultado de una operación matemática escrita en una línea con !$bc .

Macros

Las macros graban las acciones del usuario y se pueden reproducir a posteriori.

q	Empieza / termina de grabar.
@	Reproduce la macro.

Para comenzar a grabar una macro se pulsa la tecla q y luego otra que será el nombre de la macro: q<nobre> . Para terminar se pulsa q otra vez.
Para reproducir 3 veces la macro pulsaríamos 3 @ <nombre>

Búferes

Cada vez copiamos o cortamos el texto va a un búfer o registro — en la terminología de vi —. Los búferes estan numerados de 0 a 9. También podemos usar búferes con nombre de la a a la z y con las comillas usamos los búferes numerados: "ap pega lo que hay en el búfer a. Para ver el contenido de los registros hay que escribir :reg