

### 3. Interfaz de usuario proporcionada por Linux

#### 3.1 Características de Linux

Al ser Linux un sistema operativo *Unix-like*, comparte con éste múltiples características. Muchas de ellas han sido ya descritas, no obstante, se enuncian a continuación – y a modo de resumen – algunas de las más destacables:

- Se trata de un sistema operativo multitarea, ejecutable en máquinas mono o multiprocesador, pudiendo soportar la ejecución de varios programas concurrentes ya sea con uno o con varios procesadores.
- Existen versiones para diversas plataformas. Si bien, todas ellas presentan un interfaz de comandos común, permitiendo la ejecución de un mismo código fuente sobre máquinas de arquitecturas diversas. Es por tanto, un sistema multiplataforma.
- Linux permite que varios usuarios trabajen al mismo tiempo en la misma máquina, asignándoles incluso distintos recursos, y/o presentándoles la misma máquina de diversas maneras. Linux es un sistema operativo multiusuario.
- Ofrece varios mecanismos de comunicación entre procesos, destacando *pipes*, *sockets*, etc.
- Soporta un gran número de dispositivos y periféricos ampliamente extendidos incluso en el ámbito doméstico o personal, como pueden ser tarjetas gráficas, tarjetas de sonido, tarjetas de red, controladores y/o dispositivos SCSI, etc.
- La gestión de memoria es bajo demanda (*copy - on - demand*) de manera que solamente se carga en memoria aquella información que es necesaria. Además, permite que el código ejecutable sea compartido por gran número de procesos, optimizando así el uso de la memoria.
- Gracias a su sistema de ficheros virtual denominado VFS (*Virtual File System*) se pueden gestionar tanto particiones Linux con el sistema de ficheros *Ext2* , como particiones en otros formatos (MS/DOS, Windows 98/NT, ISO9660...).
- Soporta la pila de protocolos TCP/IP y otros protocolos de red como IPX/SPX, NetBIOS (si bien en algunos casos hace uso de herramientas adicionales, como *Samba*).

### 3.2 Primeros pasos en Linux

En algunos sistemas operativos, como MS-DOS, no es necesario identificarse ante el sistema antes de poder utilizarlo; se supone que en el sistema hay un único usuario y que éste tiene control total sobre el mismo.

Sin embargo, en los sistemas operativos multiusuario como Linux, es posible la existencia de más de una persona trabajando – simultáneamente o no – de manera que cada una de ellas debe poder ser tratada de manera distinta.

Por ello, al comenzar a trabajar con Linux, es necesario que el usuario se identifique. Linux necesita, para reconocer a un usuario, su nombre o identificativo de usuario. Durante el proceso de establecimiento de conexión, el nombre que introduzca el usuario debe ser reconocido por Linux.

En todas las instalaciones Linux existe un usuario especial que tiene control absoluto sobre el sistema. Se trata del superusuario o *root*. De hecho, el resto de usuarios del sistema deberán ser definidos por éste – no existirán hasta que un usuario *root* cree las cuentas de usuario correspondiente – y tendrán los privilegios o permisos que éste les otorgue, en el momento de la creación de la cuenta o más tarde.

#### 3.2.1 EL INGRESO

Como se ha señalado, para poder utilizar un sistema Linux, lo primero que debe hacer un usuario es identificarse en el mismo. Este proceso se conoce como *logging in* (registro de ingreso) y es la manera que tiene Linux de “conocer” si cierto usuario puede o no conectarse al sistema, y para cada uno de los potenciales usuarios, saber a qué están autorizados.

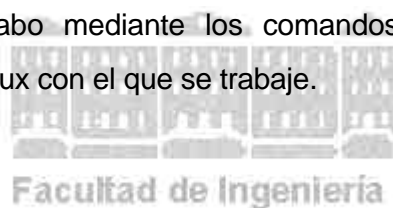
Durante el ingreso en el sistema al usuario se le pide un nombre de cuenta y una contraseña. Una vez comprobado que ambos son correctos, el sistema procede a crear un entorno o máquina virtual para ese usuario y mostrar a continuación un saludo de bienvenida y cierto carácter indicativo de que el sistema está listo para trabajar. A este carácter que da pie a que el usuario teclee alguna orden o comando se le denomina *prompt* del sistema. En Linux, el *prompt* para el usuario *root* es el símbolo “#”, mientras que para el resto de los usuarios es el símbolo “\$”.

### 3.2.2 APAGADO DEL ORDENADOR

En Linux, un ordenador no puede apagarse simplemente haciendo uso del interruptor, o cortando el suministro eléctrico del sistema sin más. Antes de hacerlo, es necesaria la ejecución de un proceso denominado "*shutdown*".

Este proceso cierra ordenadamente el sistema, y es necesario para mantener la integridad de la información del sistema así como para garantizar que los dispositivos del mismo no se vean dañados por un apagado repentino. Este proceso da por finalizados todos los procesos que se estén ejecutando en ese momento en el sistema, sin importar a qué usuario correspondan. Por tanto, no puede ser invocado por cualquiera, únicamente el usuario *root* – u otro con permisos equivalentes – tiene la potestad para ponerlo en marcha.

No obstante, cuando un usuario que carece de estos permisos desea finalizar su trabajo con el sistema, debe terminar su sesión de trabajo – iniciada al llevar a cabo el *login* – notificando al sistema que va a dejar de usar el mismo. Este proceso de desconexión se lleva a cabo mediante los comandos *logout*, *quit* o *exit*, dependiendo del sistema Linux con el que se trabaje.



### 3.3 Estructura de ficheros

Los sistemas de archivos son la base de la organización de todos los datos en un sistema operativo, y por tanto en Linux. En este sistema operativo, todos los programas del sistema, las bibliotecas de código fuente, y los archivos de usuario, se organizan en sistemas de archivos.

Se conoce como archivo la agrupación de datos que pueden ser referenciados mediante un nombre simbólico. Del mismo modo, un directorio no es sino un archivo cuyo contenido es una colección o agrupación de referencias a ficheros y/u otros directorios.

Bajo Linux, los usuarios perciben un sistema de archivos como una estructura en forma de árbol –una estructura jerárquica de directorios y archivos. Al directorio del cual cuelgan el resto de estructuras de datos del sistema se denomina directorio raíz (/). A continuación, se muestra un ejemplo de la estructura típica de directorios Linux:

```

/
  /etc
  /sbin

```

```
/bin
/tmp
/var
/lib
/home
/install
/usr
    /src
    /bin
/proc
/dev
/mnt
    /floppy
    /cdrom
    ...
```



El directorio `/etc` contiene la mayor parte de los datos específicos del sistema, necesarios para arrancar y para que el sistema funcione una vez arrancado. Entre los archivos que contiene, destaca el fichero `passwd` en el que se almacena la información relativa a todos los usuarios con capacidad para conectarse al sistema, incluidos sus permisos o niveles de privilegio. También en este directorio se encuentra el fichero `inittab`, necesario para la ejecución del proceso de inicialización del sistema (proceso `init`).

El directorio `/lib` contiene bibliotecas de funciones necesarias para el compilador C. Es un directorio de gran importancia aunque no se disponga de un compilador de C en el sistema pues contiene todas aquellas bibliotecas que pueden ser invocadas por los programas de aplicación; destacan entre su contenido el conjunto de funciones que utilizan los programas para hacer uso de las distintas llamadas al sistema. El contenido de estas bibliotecas, que puede ser compartido por varios programas de aplicación, se carga en memoria de manera explícita, sólo cuando es utilizado. De esta manera los programas ejecutables pueden ser de menor tamaño, pues no es

necesario repetir en cada uno de los programas código que va a ser utilizado por todos ellos.

El directorio `/tmp` se utiliza para almacenamiento temporal. Los programas que lo utilizan, por lo general dejan en el mismo fichero datos que serán necesarios únicamente durante su ejecución, eliminándolos una vez dejan de ser necesarios. No obstante, es importante tener en cuenta que Linux elimina periódicamente y de manera automática el contenido de este directorio, por lo que no debe almacenarse en él ningún tipo de información que se desee que perdure.

El directorio `/usr` se utilizaba en las primeras versiones de UNIX y Linux para almacenar el resto de informaciones. Así, por ejemplo el directorio `/usr/bin` contenía los programas del sistema. Actualmente, los programas del sistema de Linux, y por tanto, los programas correspondientes a los comandos que se pueden ejecutar, se encuentran en el directorio `/sbin` y el directorio `/bin` se usa para el almacenamiento de los programas de usuario. De todas formas, el directorio `/bin` tiene un enlace al directorio `/usr/bin` haciendo posible acceder al contenido del primero desde `/usr/bin` manteniendo así la compatibilidad y filosofía de los sistemas anteriores. El directorio `/usr/src` contiene el código fuente del sistema operativo.

El directorio `/bin` contiene los programas ejecutables. Por tanto, es necesario indicar al sistema operativo que tiene que buscar en el directorio `/bin` el código de los programas de usuario que se ejecutan desde la consola. Además, cada uno de los usuarios reconocidos por el sistema dispone de un espacio en disco en el que almacenar sus archivos. Así, siempre que se añade un usuario al sistema (comando `adduser`), el sistema le asigna un directorio *inicial*. Por convenio, los directorios iniciales de todos los archivos se ubican en el directorio `home`, ubicado bajo el raíz. Así, si se creara un usuario de nombre Gregorio, el sistema crearía y le asignaría un directorio denominado `/home/gregorio`, que almacenará todos los archivos que ese usuario cree.

En `/var` se guardan las distintas tablas que definen la configuración del sistema de manera que si ésta variara bastaría con hacer una copia de seguridad de este directorio por si fuera necesario reestablecerla.

El directorio `/dev` alberga los ficheros especiales asociados a cada uno de los dispositivos en los sistemas *Unix-like*, lo cual les permite trabajar con éstos como si de ficheros se tratara (especialmente de cara a operaciones de lectura / escritura)

dotando al sistema de una mayor uniformidad. El contenido de este directorio se analizará más en detalle al hablar de los ficheros especiales.

El directorio `/mnt` se utiliza para enlazar los sistemas de archivos almacenados en dispositivos de almacenamiento distintos del dispositivo raíz, tal y como se verá más adelante.

### 3.4 Los Sistemas de archivos

La expresión *sistema de archivos* tiene en Linux dos significados distintos. Por un lado, hace referencia al sistema de discos y a los mecanismos que posibilitan el trabajo con varias unidades de disco en conjunto, y por otro al aspecto lógico, es decir, la organización de las informaciones que visualiza y manipula el usuario.

Antes de nada, es importante señalar que en Linux, toda entidad de información, tanto física como lógica, se representa internamente como un archivo en el sistema de archivos. Entre las entidades físicas representadas se incluyen discos, impresoras, terminales, tarjetas de red, etc., mientras que entre las entidades lógicas se encuentran los directorios y archivos de datos, en los que se almacenan documentos, programas, etc.

En todo el sistema de almacenamiento de información, es necesario poder identificar cada una de estas entidades dentro del sistema de archivos representado. Para ello se dispone por un lado de los nombres de archivo y por otro de los nombres de las rutas de acceso, o simplemente, rutas de acceso.

#### 3.4.1 NOMBRES DE ARCHIVO

Un *nombre de archivo* consiste en una secuencia de caracteres (letras, números y algunos símbolos alfanuméricos). Los nombres de archivo no pueden incluir, espacios ni cualquier otro carácter que represente un separador de campo. Es posible el uso de todos los caracteres alfanuméricos que no tengan otros significados en Linux, como es el caso de: `!", "@", "#", "$", "%", "&", "*", "(", ")", "[", "]", "{", "}", "'", "\"", "\\", "/", "|", ";", "<"` y `>`. Por ejemplo, el nombre de archivo `"trabajos.cso"` es válido, pero el nombre `"trabajos cso"` no.

Otra limitación a la hora de designar a un archivo es el número de caracteres. En la mayoría de las versiones iniciales de UNIX, en las que como se ha visto se basa Linux, los nombres de archivo no podían tener más de 14 caracteres. Versiones más recientes de UNIX, como la versión Berkeley (BSD), permiten nombres de hasta 64 caracteres, pero sólo los primeros 14 son significativos, es decir, los archivos deben

quedar identificados unívocamente mediante los primeros 14 caracteres de su nombre dentro de un directorio, garantizando así la compatibilidad con los sistemas anteriores. Sin embargo Linux permite hasta 256 caracteres.

Por otro lado, todo archivo debe almacenarse en algún punto de la estructura jerárquica que conforman los sistemas de archivos. Es decir, todos los archivos se ubicarán dentro de algún directorio, sea el raíz (Directorio “/”) o no.

### 3.4.2 NOMBRES DE RUTA

El nombre de ruta de un archivo especifica no sólo su nombre dentro de un directorio, sino también su ubicación dentro del árbol de directorios del sistema.

Un nombre de ruta puede ser absoluto o relativo. Se dice que un nombre de ruta es absoluto si especifica todos los directorios tomando como punto de partida el directorio raíz, que es necesario recorrer para encontrar el archivo en cuestión. Un nombre de ruta es relativo cuando especifica los directorios que es necesario recorrer a partir del directorio actual.

En sistemas operativos como MS-DOS, el directorio actual suele aparecer representado en el propio *prompt* del sistema (Ej: C:\Windows> indica que el directorio de trabajo actual es el directorio Windows de la unidad “C”). En Linux, para conocer el directorio actual se puede ejecutar el comando `pwd` (*print working directory* – Visualizar Directorio de Trabajo). Otra posibilidad es acceder al contenido de la variable de entorno<sup>11</sup> `$PWD` mediante la orden `echo $PWD`.

Así, para localizar el archivo `fichero.de.gregorio` ubicado en el directorio `/grerorio` a partir de su nombre de ruta absoluto, se especificaría `/home/gregorio/fichero.de.gregorio`, y a partir de ese nombre de ruta relativo a ese directorio, se escribiría `/gregorio/fichero.de.gregorio`, suponiendo que el directorio actual es `/home`.

### 3.4.3 MONTAR Y DESMONTAR SISTEMAS DE ARCHIVOS

Una característica destacable del sistema de archivos en los sistemas UNIX es que éste es único dentro del sistema. Es decir, UNIX, y por tanto Linux, únicamente reconocen un árbol de directorios en el sistema. Este aspecto es radicalmente

---

<sup>11</sup> Las variables de entorno son variables propias del intérprete de comandos de un sistema operativo. Generalmente se dispone de algún comando para acceder a su valor.

diferente al utilizado en los sistemas de Microsoft, en los que se puede acceder a un árbol de directorios y archivos en cada una de las unidades instaladas (las operaciones de montaje y desmontaje son transparentes al usuario, pero se llevan a cabo).

En Linux, para poder acceder a los datos almacenados en un dispositivo de almacenamiento (disco o disquete) es necesario enlazar esa unidad con el sistema o árbol de archivos del sistema o sistema de archivos raíz (todos los sistemas UNIX tienen un sistema de ficheros ubicado en la unidad desde la cual se arranca). A esta operación de relacionar la estructura de directorios contenida en una unidad con el sistema de ficheros accesible desde Linux se le llama “Montaje”. De manera análoga, al acabar de trabajar con un disco es necesario desvincular su árbol de directorios del sistema de archivos del sistema para garantizar que la información se mantenga de manera íntegra y consistente, es decir, hay que realizar una operación de “Desmontaje”.

Para montar un sistema de archivos ubicado en una unidad distinta de la que contiene el sistema operativo Linux, en el sistema de archivos raíz es necesario que establezca al menos una partición en la unidad a montar y designar un directorio existente del sistema de ficheros raíz para utilizarlo como punto de montaje, caso de no existir dicho directorio, éste no se creará, y la operación de montaje no se llevará a cabo.

Así, si se desea montar la unidad de CD-ROM de un sistema, identificada en Linux como unidad `/dev/cdrom`, bajo el punto de montaje `/mnt` será necesario que exista un directorio llamado `mnt` en el directorio raíz del sistema de ficheros raíz o el montaje fallará.

Como resultado de la operación de montaje el directorio tomado como “punto de montaje” hará referencia a la unidad que ha sido montada.

#### 3.4.4 TIPOS DE ARCHIVOS BÁSICOS

En Linux existen cuatro tipos de archivos: los básicos – archivos normales y directorios –, los enlaces y los archivos especiales. Además, es posible establecer otros criterios de clasificación dentro de cada uno de ellos. La orden `file` permite reconocer para un archivo determinado si se trata de un archivo ejecutable, o de texto, o de datos, etc.



## Archivos normales

Son aquellos que contienen texto, código fuente en lenguaje C, archivos de órdenes *shell* (programas o guiones para ser interpretados por los *shell* de Linux), programas binarios ejecutables u otros datos de naturaleza diversa.

Entre estos archivos, Linux puede distinguir los que son ejecutables – pueden ejecutarse directamente – y los que no. Para ejecutar un archivo es necesario que éste sea ejecutable y que se encuentre en una ruta de acceso de búsqueda<sup>12</sup>.

## Archivos de directorio

Son archivos que almacenan nombres de archivos y subdirectorios, así como punteros que permiten acceder a los archivos y subdirectorios que contienen. Estos archivos son el único lugar donde Linux almacena nombres de archivos. Cada referencia a un archivo o subdirectorio contenida en un directorio se denomina entrada de directorio y está formada por al menos el nombre del archivo o subdirectorio y un puntero al mismo.

Así, cuando se ejecuta la orden `ls` para listar el contenido de un directorio, lo que se lista es el contenido de un archivo de directorio.

De manera análoga, si se varía el nombre de un archivo (por ejemplo mediante comando `mv`) y siempre que el archivo no varíe de directorio, simplemente se varía la entrada correspondiente en el archivo de directorio que lo contiene. Por otro lado, si se mueve un archivo de un directorio a otro, se traslada la entrada referente al mismo de un archivo de directorio a otro.

### 3.4.5 DIRECTORIOS Y DISCOS FÍSICOS (I-NODOS)

Internamente, Linux representa los archivos mediante *i*-nodos. A cada archivo se asocia un *i*-nodo en el momento de su creación, que contiene toda la información referente al mismo, incluyendo la dirección de los datos en el disco y el tipo de archivo (archivo normal, directorio o archivo especial), y que permite acceder a los datos ubicados en dicho archivo. Todos los *i*-nodos correspondientes a los archivos existentes en una unidad se almacenan en una tabla, llamada tabla de *i*-nodos, que se crea durante el proceso de inicialización del disco.

---

<sup>12</sup> Conjunto de rutas dentro de las que Linux busca archivos ejecutables. Es simplemente una lista de rutas de acceso absolutas, y se configura dentro de la *shell* de Linux.

Todos los sistemas Linux identifican cada i-nodo con un número entero, de manera unívoca. Así, el i-nodo 1 se corresponde siempre con el directorio raíz, que, a partir de sus entradas, permite acceder a todo el árbol de archivos y subdirectorios del sistema.

A continuación, se muestra un ejemplo de contenido del i-nodo del directorio raíz, en el que cada línea indica el número de i-nodo correspondiente al directorio o fichero y el nombre del mismo:

```
1      .
1      ..
45     etc
230    dev
420    home
123    .profile
```

Todos los i-nodos de archivos de directorio contienen una entrada “.”(punto) que hace referencia al propio directorio (en este caso el directorio raíz) y otra entrada “..”, que hace referencia a su directorio padre (en el caso del directorio raíz hace también referencia a sí mismo, pues el directorio raíz no tiene padre). No obstante, si se accede por ejemplo al contenido del archivo de directorio /home se obtiene:

```
420    .
1      ..
643    fred
```

Obsérvese que al haberse tomado ambos ejemplos del mismo sistema, el i-nodo correspondiente al directorio actual (directorio “.”), en este caso, el 420, coincide con el indicado en el i-nodo del directorio raíz para el directorio home, y que el número 1, correspondiente al i-nodo raíz, aparece ahora indicado como directorio padre.

De esta manera se navega en Linux por su sistema de archivos, mediante el encadenamiento hacia arriba y hacia debajo de los archivos de directorio. Si se mueve un archivo a un directorio en otro disco físico, Linux lo detecta al leer la tabla de i-nodos. Para llevar a cabo esta operación, se asigna al archivo un nuevo i-nodo en el nuevo disco antes de suprimirlo el fichero (y su i-nodo correspondiente) de su ubicación original.

Al igual que con la orden `mv`, cuando se suprime un archivo con la orden `rm` en realidad no se toca el archivo, sino que Linux marca ese *i*-nodo como libre y lo devuelve al conjunto de *i*-nodos disponibles. Posteriormente, se borra la entrada del archivo en el directorio.

## Enlaces

Un enlace no es verdaderamente un archivo sino simplemente una entrada de un directorio. En Linux, es probable que un archivo pueda localizarse desde varios directorios, de manera que para referirse a un mismo archivo, además de a través de su entrada y nombre originales suele hacerse desde otro punto del árbol de directorio, incluso con otro nombre diferente, siempre y cuando la nueva entrada o enlace haga referencia al mismo *i*-nodo.

De cualquier manera, el término “enlace” se utiliza en Linux para hacer referencia a entradas de directorio que señalan a *i*-nodos ya existentes. Así, un *i*-nodo puede estar referenciado desde varios directorios (desde varias entradas de directorio, incluso en un mismo directorio).

La tabla de *i*-nodos contiene una relación, por cada *i*-nodo, de todas las entradas que hacen referencia al mismo, de forma que un *i*-nodo estará libre o disponible únicamente cuando se eliminen todas las entradas que hacen referencia a él.

Siempre que una entrada o enlace hace referencia a algún archivo – y por tanto a algún *i*-nodo – ubicado en el mismo dispositivo que el enlace, se habla de enlaces ordinarios. Linux, al igual que la mayoría de versiones modernas de UNIX, tiene otro tipo de enlace llamado enlace simbólico. En este caso, la entrada de directorio conduce a un archivo que no es más que una referencia a otro archivo localizado en otra ubicación dentro del sistema de archivos lógico, permitiendo de esta manera acceder tanto a archivos del mismo disco, como a los ubicados en otro disco, o incluso otra computadora.

Una diferencia importante entre enlaces normales y simbólicos es que los normales tienen todos la misma categoría (es decir, el sistema trata cada enlace como si fuera el archivo original) y no se suprimen los datos hasta que no se suprime el último enlace del archivo. Con los enlaces simbólicos, cuando se suprime el archivo original (el último enlace ordinario), también se suprimen todos los enlaces simbólicos a ese archivo. No obstante, el tratamiento y manejo de archivos es idéntico a través de ambos tipos de enlaces.

Para averiguar si una entrada de un directorio hace referencia a un archivo en sí o es únicamente un enlace, es posible ejecutar el comando `ls -l`, puesto que en el caso de los enlaces se obtendrá tanto el nombre del archivo reseñado en el enlace como una indicación del archivo enlazado. En el ejemplo siguiente, el apartado “a)” representa un enlace, y el “b)”, un archivo.

```
lrwxrwxrwx 1 root root 4 Oct 17 15:27 Info ->infol
-rwxrwxrwx 1 root root 2 Oct 17 15:24 Archivo
```

## Archivos especiales

En el sistema de archivos de Linux tienen cabida además de los archivos de datos, representaciones de todos los dispositivos físicos del sistema incluyendo discos, terminales e impresoras. Todos los dispositivos tienen un archivo asociado, ubicado generalmente en el directorio `/dev`. Así, por ejemplo, la consola se representa mediante el nombre `/dev/console` y un terminal estándar podría venir representado por `/dev/tty01`. A los archivos empleados en Linux para representar dispositivos físicos se les denomina archivos especiales.

Los terminales e impresoras se denominan dispositivos especiales por caracteres, puesto que manejan – aceptan y/o producen – cadenas de caracteres. Por otro lado, los discos manipulan la información en forma de bloques identificables que constituyen la unidad mínima de transferencia. A los dispositivos que operan de esta manera – discos, cintas, etc. – se les denomina dispositivos especiales por bloques.

En el caso de los dispositivos de bloques, existe además una característica adicional: su capacidad de operar como si se tratase de dispositivos orientados a caracteres, de manera que estos dispositivos deben representarse tanto como dispositivos de bloques como dispositivos de caracteres, dentro del sistema. Afortunadamente, Linux lleva a cabo todas las operaciones necesarias de manera automática cuando se hace referencia a estos dispositivos como a los de caracteres, evitando la intervención del usuario.

Un tercer tipo de dispositivos son las “conducciones con nombre” o *FIFOs* (Memoria intermedia First Input First Output). Se trata de archivos aparentemente normales, que se utilizan principalmente por parte del sistema para que los procesos de usuario puedan enviar información a un proceso de control.

Orden	Descripción de su comportamiento
ls	Mostrar el contenido de un directorio
cd	Cambiar de directorio
mkdir	Crear un directorio
rmdir	Borrar un directorio
pwd	Muestra el directorio de trabajo
cp	Copiar ficheros
mv	Mover ficheros
rm	Borrar ficheros
cat	Mostrar el contenido de un fichero
tail	Mostrar el contenido del final de un fichero
head	Mostrar el contenido del principio de un fichero
mount	Montar un sistema de ficheros
umount	Desmontar un sistema de ficheros
mkfs	Crear un sistema de archivos

Tabla 3: Relación de comandos de manejo de ficheros

El último tipo de archivo especial es el cubo de bits, `/dev/null`. Todo lo que se envíe a este archivo se ignora, siendo de gran utilidad cuando no se desea por ejemplo visualizar la salida de una orden. Por ejemplo, si no se quieren ver los informes de diagnóstico impresos en el dispositivo estándar de error, se pueden poner en el cubo de bits, utilizando la orden siguiente:

```
ls -la > /dev/null
```

### 3.4.6 RESUMEN DE LOS COMANDOS DE MANEJO DE FICHEROS EN LINUX

A lo largo de los puntos anteriores se han ido citando diversos comandos para el manejo de archivos en Linux. En la tabla 3, se presenta una relación de todos ellos; si bien, se recomienda acceder a la información que el propio Linux ofrece de cada uno de ellos, accesible a través del comando `man <nombre de comando>`.

### 3.5 Permisos de los archivos.

En Linux, existen informaciones relativas a las operaciones que pueden llevarse a cabo sobre cada archivo: quién está capacitado para leer, escribir o ejecutar un archivo, el tipo de archivo, y la forma de ejecutarlo. Para averiguar los permisos de un archivo se puede usar el comando, `ls -l`, como se muestra en el siguiente ejemplo:

```
drw----- 2 sglines doc 512 Jan 1 13:34 Maild
-rwx----- 5 sglines doe 1024 Jan 17 08:22 News
-rw----- 1 sglines doe 1268 Dec 7 15:01 biblio
drw----- 2 sglines doe 512 Dec 15 21:28 bin
-rw----- 1 sglines doe 44787 Oct 20 06:59 books
-rw----- 1 sglines doc 23001 Dec 14 22:50 bots.msg
-rw-r----- 1 sglines doc 105990 Dec 27 21:24 ducke.gif
```

La ejecución de este comando, muestra prácticamente toda la información relativa a cada archivo de un directorio que se encuentra almacenada en el directorio correspondiente. El primer bloque de bits, muestra los permisos del archivo. A continuación se detalla el número de enlaces al archivo, la tercera el propietario del mismo – en este caso todos pertenecen al usuario “sglines” –; la cuarta, el grupo de usuarios al que pertenece el archivo, la quinta el número de bytes que ocupa, la sexta la fecha y hora de creación, y la séptima muestra el nombre del archivo.

La columna de permisos se divide en cuatro subcampos:

```
-      rwx  rwx  rwx
```

El primer bit constituye el primer subcampo y define el tipo de archivo: un guión “-” indica que se trata de un archivo normal, una “d” representa un directorio, una “l” un enlace una “b” un dispositivo especial de bloques, y una “c” un dispositivo de caracteres.

El resto de bits representan los permisos de lectura, escritura y ejecución del archivo. Estos permisos pueden definirse en los sistemas UNIX, a tres niveles. El primero de ellos se refiere al propietario del archivo, el segundo al grupo de usuarios al que pertenece el propietario del archivo – en UNIX todos los usuarios pertenecen a un grupo de usuarios – , y el tercero hace referencia al resto de usuarios del sistema. Así, existen tres conjuntos de bits “rwx” que hacen referencia respectivamente a cada uno de los tres niveles indicados.

Cada tríada de bits, representa para el nivel afectado la capacidad de leer el archivo (bit “r” activado), la capacidad de escribir en el mismo (bit “w” activado) y la capacidad de ejecutar su contenido (bit “x” activado).

En cualquier caso, es necesario tener presente que normalmente, un programa en ejecución pertenece a quien lo ha mandado ejecutar. No obstante, cuando el bit `setuid` asociado al archivo en cuestión está activado, el programa en ejecución pertenece al propietario del archivo, independientemente de quién lo haya lanzado. En este caso, el programa en ejecución tendrá los permisos asociados al propietario del archivo. Esto mismo es aplicable al grupo de usuarios, mediante el bit `setgid`.

Por tanto, la activación de estos bits no es trivial. Si el usuario que ejecuta el programa es un usuario “normal” siendo el programa propiedad del usuario `root`, con la activación de alguno de estos bits podría dotar al programa de la capacidad de leer y escribir cualquier archivo en el sistema sin tener en cuenta los permisos del usuario que lo ejecuta. En consecuencia el valor de estos dos bits puede ser modificado exclusivamente por el propietario del fichero o por el usuario `root`.

Valor Octal	Descripción
0001	Permiso de ejecución para el propietario
0002	Permiso de escritura para el propietario
0004	Permiso de lectura para el propietario
0010	Permiso de ejecución para el grupo
0020	Permiso de escritura para el grupo
0040	Permiso de lectura para el grupo
0100	Permiso de ejecución para el resto
0200	Permiso de escritura para el resto
0400	Permiso de lectura para el resto

Tabla 4: Valores octales de los permisos

Los permisos de un archivo pueden modificarse con la orden `chmod`, permite indicar en octal los permisos de un archivo.

Estos valores pueden combinarse en una misma sentencia, de manera que hagan referencia a los distintos niveles de privilegio. Así, por ejemplo, para dar permisos de lectura y escritura al propietario, grupo del propietario y resto de usuarios, se sumarían

los valores octales indicados hasta obtener el total, tal y como se muestra en el ejemplo siguiente:

0002	Permiso de escritura para el propietario
0004	Permiso de lectura para el propietario
0020	Permiso de escritura para el grupo
0040	Permiso de lectura para el grupo
0200	Permiso de escritura para todos los demás
0400	Permiso de lectura para todos los demás

---

0666	Permiso de lectura y escritura para todos los usuarios.
------	---

A continuación, bastaría con ejecutar `chmod 666 fichero` para asignar estos permisos a un archivo de nombre "fichero".

### 3.6 Edición de archivos.

Los editores son herramientas que facilitan la creación de nuevos archivos y la modificación de los que ya existen. Pueden ser de tres tipos:

- **Editores de línea:** En ellos la mayoría de los cambios se aplican a una línea o grupo de líneas. Para hacer una operación, se debe indicar primero el número de línea y luego la operación a realizar. Un ejemplo es el editor `ed`.
- **Editores de texto:** El usuario visualiza en pantalla el texto que se edita y puede mover el cursor por él para realizar cambios. Como ejemplo está el estándar `vi`, el `pico` y el `joe`.
- **Formateador de texto:** Los editores anteriores no pueden formatear texto; no proporcionan funciones de ajuste de márgenes o centrado de líneas. Para preparar documentos se utilizan los formateadores. El `LaTeX` es uno de los más populares, en entornos UNIX.

### 3.7 Estructura de un disco de Linux

Bajo UNIX, y por tanto bajo Linux, un disco es un dispositivo de bloques, los cuales se dividen en cuatro regiones.



### 3.7.1 BLOQUE DE ARRANQUE

Almacena el programa de arranque, que es un programa especial, que se activa al poner en marcha el sistema, encargándose de cargar el sistema operativo en memoria.

### 3.7.2 SUPERBLOQUE

Contiene información relativa al disco, es decir, describe el disco. Entre estas informaciones destacan:

- El número total de bloques del disco.
- El número de bloques libres.
- El tamaño de cada bloque en bytes.
- El número de bloques utilizados.

### 3.7.3 LA TABLA DE I-NODOS O BLOQUE DE LISTA DE I-NODOS

Este í nodo, el número 1, guarda la lista de todos los í nodos del sistema. Cada entrada en esta lista es de hecho un í nodo, un espacio de almacenamiento de 64 bytes que permite localizar un archivo, y los bloque(s) que ocupa dentro del disco. En el í nodo se guardan además otras informaciones como son:

- Permisos de acceso del archivo.
- Identificadores del propietario y grupo del propietario del archivo.
- Número de enlaces del archivo.
- Fecha y hora de la última modificación del archivo.
- Fecha y hora del último acceso al archivo.
- Localización de los bloques (en el caso de archivos normales y directorios).
- Identificación del dispositivo representado (en el caso de ficheros especiales).

### 3.7.4 I-NODOS Y DIRECTORIOS

El í nodo número 2 contiene la localización de los bloques que conforman el directorio raíz, que permite el acceso a toda la estructura de directorios del sistema. A partir del

mismo, y como se ha indicado anteriormente, es posible localizar cualquier directorio y/o archivo en el sistema.

### **3.8 Lectura: Introducción al editor “vi”**

El editor “vi” (pronunciado “vi ai” en inglés) es el único editor que se puede encontrar en prácticamente cualquier instalación de Unix. Escrito originalmente en la Universidad de California en Berkeley, prácticamente todas las versiones de UNIX o Linux incluyen alguna versión del mismo.

En una primera toma de contacto, cuesta bastante acostumbrarse a trabajar con él, pero tiene características muy importantes. Si bien se suele recomendar a los usuarios noveles el manejo del editor Emacs – mucho más sencillo – el editor “vi” no puede dejarse de lado, pues seguro que en más de una ocasión, uno deberá enfrentarse a él. Puede ser necesario tener que utilizar este editor mientras se instala algún otro, como el Emacs; tampoco es raro encontrar en UNIX o Linux, herramientas, aplicaciones e incluso juegos que hacen uso de subconjuntos de comandos del vi.



Algunas de las teclas que especifican comandos, pueden precisar de aclaraciones históricas ya que no existen otras motivaciones que justifiquen algunas de sus opciones. No obstante, y además de una lectura detallada sobre el manejo de este editor, se recomienda hacer uso de cuantos tutoriales se disponga para poder al menos adquirir conocimientos relativos a los comandos existentes, no siendo mala idea disponer de una guía de comandos a mano al trabajar con “vi”.

#### **3.8.1 BREVE HISTORIA DE VI**

Los primeros editores de texto que existieron eran editores de línea. Un ejemplo de este tipo de editores es “ed”, un editor potente y eficiente, que además tenía la ventaja además de utilizar muy pocos recursos del sistema. Por otro lado, el editor “vi” ofrecía la posibilidad de ver en pantalla el contenido de los ficheros, además de ofrecer un grupo de comandos bastante amplio (especialmente si se comparaba con el que disponía “ed”). No obstante, el editor “vi” se basa también en otro editor

en línea, el editor “ex”. No en vano, hoy por hoy, “ex” es un modo especial de edición propio del editor “vi”.

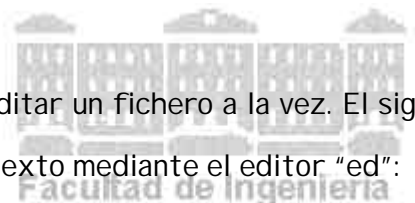
Tanto “ex” como “vi” fueron desarrollados por W. Joy en la Universidad de California en Berkeley. Originalmente se suministraba con UNIX como una utilidad sin soporte, hasta que fue incluido oficialmente en la distribución System V. Su popularidad ha ido pasando por diferentes etapas, si bien últimamente ha tenido que hacer frente a la gran competencia de editores de pantalla más modernos.

### 3.8.2 TUTORIAL DE “ED”

Un objetivo en el diseño del editor “ed” fue que resultara fácil de usar, que no requiriera un gran entrenamiento para empezar a utilizarlo. Así, basta con seguir unas breves instrucciones para poder hacer uso del mismo.

#### Creación de un fichero

El editor “ed” sólo puede editar un fichero a la vez. El siguiente ejemplo permite la creación de un fichero de texto mediante el editor “ed”:



```
$ ed
a
Este es mi primer fichero de texto usando Ed.
Esto es divertido de verdad.
.
w primero.txt
q
$
```

En este ejemplo se muestran varios aspectos del editor. Si se invoca al mismo tecleando “ed” como en el ejemplo, se obtendrá un fichero vacío. El comando a se utiliza para añadir texto al fichero hasta teclear el comando “punto” (“.”). Para salvar el texto añadido al fichero, se utiliza el comando w seguido del nombre del fichero, y finalmente, el comando q permite salir del editor.

Es decir, este editor ofrece dos modos de operación: el "modo comando", donde cada comando está definido por un carácter y el "modo texto", que permite la edición propiamente dicha del contenido del archivo..

### Edición de un fichero existente

Para añadir una línea de texto, por ejemplo al fichero creado en el epígrafe anterior, puede seguirse el siguiente guión:

```
$ ed primero.txt
a
Esta es una nueva linea de texto.
.
w
q
```

El editor siempre recuerda cuál es la línea en curso o actual para cada fichero, de manera que el comando `a` siempre añade el nuevo texto tras la línea actual. No obstante, si se desea insertar texto antes de la línea actual se puede utilizar el comando `i`.

Para añadir una línea de texto al final de un fichero puede seguirse el siguiente ejemplo:

```
$ ed primero.txt
$a
La ultima linea de texto.
.
w
q
```

El modificador de comandos "\$" indica al editor que la adición debe hacerse tras la última línea del archivo. Si se desea añadir la nueva línea tras la 1ª, el modificador sería el "1", de manera que el comando sería "1a". Así, y mediante la combinación de

los números de líneas con los comandos "a" o "i" es posible llevar a cabo inserciones de texto antes o después de las líneas indicadas.

Por otro lado, el comando "p", muestra el contenido de lo que "ed" considera la línea actual. La utilidad de este comando puede ampliarse, al permitir también variar la línea actual. Así, si se desea por ejemplo convertir en línea actual a la número 2, y de paso, observar su contenido, se puede proceder de la siguiente forma:

```
$ ed primero.txt
2p
q
```

En cualquier caso, el comando cat permite visualizar por pantalla el contenido de un archivo, de manera inmediata. Este comando equivale al comando type de MS-DOS.

### 3.8.3 NÚMEROS DE LÍNEA EN DETALLE

Una vez que se sabe cómo mostrar el contenido de la línea actual, y que existen modificadores que permiten variar el número de línea sobre los que operan los comandos, debe prestarse atención a otros modificadores que hacen referencia a valores que pueden variar durante la edición del fichero. Así, si el modificador "\$", hace referencia a la última línea de texto, el comando \$p permitirá visualizar el contenido de dicha última línea. El modificador "." Hace referencia a la línea actual, de manera que se podría ver su contenido con el comando ".p".

También es posible trabajar con agrupaciones de líneas. Así, para mostrar el contenido de un fichero desde la línea 1 hasta la línea 2 (ambas inclusive) se hará uso del comando 1,2p donde el primer número indica la línea de comienzo y el segundo a la final, que se convertirá en la línea actual. Por otro lado, para mostrar el contenido del fichero desde el comienzo hasta la línea actual, se escribirá "1, .p". Análogamente, para mostrar el contenido del fichero desde la línea actual hasta el final, se tecleará ". , \$p".

### Borrado de líneas de texto.

Para el borrado de líneas de texto se puede hacer uso del comando "d", de manera que por ejemplo, "1,2d" provocaría el borrado de las dos primeras líneas del fichero, mientras que "1,\$d" el borrado del fichero completo (desde la primera línea hasta la última).

#### 3.8.4 TUTORIAL DE VI

A continuación, se presenta un tutorial pensado para personas que jamás han manejado el editor "vi", y en el que se muestran algunos de sus comandos más básicos. Se trata de 10 comandos, habitualmente suficientes para realizar la mayoría de operaciones de edición sobre un fichero, y que dan la posibilidad al usuario que lo desee de profundizar en el manejo de este editor.

### Ejecución del editor vi

Para ejecutar este editor basta con teclear su nombre "vi" seguido – en su caso – del nombre del fichero a crear o modificar. Inmediatamente se mostrará una pantalla con una columna de tildes (~<sup>13</sup>) en el lado izquierdo.

En este momento, el editor se encuentra en modo comando, de manera que cualquier secuencia de caracteres que se teclee se interpretará como un comando, en vez de como texto a escribir / añadir al fichero. Para poder comenzar la inserción de texto es necesario teclear alguno de los comandos de entrada. Los dos más básicos son los siguientes:

- i      insertar texto a la izquierda del cursor.
- a      añadir texto a la derecha del cursor.

Evidentemente, al acceder a un fichero vacío (cuando se está creando el fichero) da igual hacer uso de uno o de otro. A modo de ejercicio, puede crearse un fichero

---

<sup>13</sup> <ALT + 126>

de texto que contenga el siguiente poema de Augustus DeMorgan, incluido en el texto *The Unix Programming Environment* por B.W. Kernighan y R. Pike:

```
Las pulgas grandes tienen pequeñas pulgas
sobre sus espaldas para que les muerdan.

Y las pulgas pequeñas tienen pulgas mas pequeñas
y asi hasta el infinito.

Y las pulgas grandes, a su vez,
tienen pulgas mas grandes sobre las que estar;

Mientras que estas
de nuevo tienen otras mas grandes aun,
y mas grandes aun, y asi.

<Esc>
```

Hasta la pulsación de la tecla "Esc", se sigue en modo texto. Es decir, es la pulsación de esta tecla la que permite el paso de modo texto a modo comando.

### Comandos de movimiento del cursor

- h      mueve el cursor un espacio a la izquierda.
- j      mueve el cursor un espacio abajo.
- k      mueve el cursor un espacio arriba.
- l      mueve el cursor un espacio a la derecha.

Estos comandos pueden utilizarse de manera repetida, simplemente manteniendo la tecla pulsada. Si en un momento dado se intenta un movimiento imposible, como mover el cursor hacia arriba estando en la primera línea, la pantalla parpadeará momentáneamente o el terminal emitirá algún sonido.

### Borrado de texto

- x      borra el carácter que hay en el cursor.
- dd    borra la línea donde está el cursor.

A modo de ejercicio, puede moverse el cursor a la primera línea y ubicarlo bajo la "n". A continuación podrá observarse cómo con la pulsación del comando "x" desaparece la letra "n". Bastará no obstante con hacer uso del comando "i" para volver al modo de inserción y teclear la letra "n" para que quede como estaba. borrada. Para finalizar, pulse <Esc>.

### **Fin de la edición de un fichero**

Para finalizar la edición de un fichero, hay dos opciones. La primera, accesible mediante el comando ":w" provocará que se escriba en disco (que se salve) el nuevo contenido del fichero, mientras que la segunda de ellas ":q" simplemente permitirá salir el editor. Evidentemente para hacer uso de ambas opciones habrá que estar en modo comando.

Además, estos dos comandos se pueden combinar, de forma que si se desea salvar el contenido del fichero y salir, bastará con teclear, en modo comando, ":wq". Para lograr un efecto similar se puede teclear "ZZ", en vez de ":wq", si bien hay que destacar que esta combinación de teclas no es un sinónimo exacto de ":wq" puesto que únicamente salva el fichero si éste ha sufrido algún cambio durante la edición, y ":wq" lo salva siempre, aunque no haya habido modificaciones). La utilizan los programadores que, tras compilar un programa y lanzarlo a ejecución, detectan algún error. Posteriormente, deben entrar al editor, llevar a cabo algún cambio en el código y volver a salir para ejecutar el programa de nuevo.

### **3.8.5 TUTORIAL AVANZADO DE VI**

La ventaja y la potencia de este editor radica en saber utilizarlo eficientemente conociendo sólo unos pocos comandos. Una vez vistos los 10 más habituales, a continuación se exponen algunas de las facetas más poderosas de vi, que abarcan desde la copia de texto hasta la definición de macros. Se recomienda practicar los comandos que se detallan con algún texto de ejemplo, cuyo contenido pueda "perderse" o no sea trascendental.



## Comandos de movimiento

En algunas implementaciones es posible hacer uso de las teclas de cursor para lograr el mismo efecto, pero son los comandos "h", "j", "k" y "l" los que funcionarán en todos los casos. Otros comandos comunes de movimiento son:

w	mover al principio de la siguiente palabra.
e	mover al final de la siguiente palabra.
E	mover al final de la siguiente palabra antes de un espacio.
b	mover al principio de la palabra anterior.
0	mover al principio de la línea actual.
\$	mover al final de la línea.
<CR>	mover al principio de la siguiente línea.
-	mover al principio de la línea anterior.
G	mover al principio de la última línea del fichero.
1G	mover al principio del fichero.
nG	mover a la línea n.
<Ctrl> G	mostrar el número de línea actual, y el % de fichero ubicado entre el principio de éste y el cursor.
%	ir al paréntesis correspondiente, estando sobre un paréntesis o sobre texto comprendido entre paréntesis.
H	mover a la línea superior en pantalla.
M	mover a la línea de en medio de la pantalla, o a la línea de en medio del fichero si el tamaño de éste es inferior a una pantalla.
L	mover al final de la pantalla, o a la última línea del fichero si ésta se está mostrando actualmente por pantalla.
n_	mover el cursor a la columna n.

Aunque también existen comandos que permiten controlar el desplazamiento, este texto se desplaza automáticamente cuando el cursor alcanza la parte superior o inferior de la pantalla.

<Ctrl> f	desplaza una pantalla hacia delante.
<Ctrl> b	desplaza una pantalla hacia atrás.
<Ctrl> d	desplaza media pantalla hacia abajo.
<Ctrl> u	desplaza media pantalla hacia arriba.

Por otro lado, algunos de los comandos anteriores usan un modificador de comandos en forma de un número que precede al comando, para indicar que éste se repita ese número de veces. Así, para mover el cursor ocho posiciones a la izquierda, se indicará "8l", y para introducir cierto número de espacios delante de un texto bastaría con introducir el número de repeticiones (n) y el comando "i" seguido por el espacio (el carácter a insertar).

En cualquier caso, hay que tener en cuenta que en el caso de comandos que manipulan o tratan líneas, la utilización de un modificador hace referencia al número de línea. Así, el comando "1G" provocaría un movimiento del cursor a la primera línea.

De cualquier manera, el editor vi dispone de gran cantidad de comandos susceptibles de ser utilizados para mover el cursor o posicionarse dentro de un texto. Incluso es factible posicionar el cursor en determinada línea desde el *prompt* del sistema (al ser invocado). Tecleando por ejemplo "*vi +10 mi\_fichero.tex*", se abrirá un fichero de nombre "*mi\_fichero.tex*" y el cursor estará posicionado 10 líneas más abajo del comienzo del mismo.

### **Modificación del texto**

El editor vi ofrece un gran número de comandos que permiten variar el contenido de un fichero. A continuación, se expone la manera de añadir, eliminar o bien modificar el texto ya existente, para que el lector complete sus conocimientos y pueda hacer uso de las funcionalidades más habituales en la edición de textos.

Durante la escritura de un fichero de texto, es posible la introducción de varias líneas, simplemente haciendo uso de la tecla <Intro>. Además, si es necesaria la corrección de algún error tipográfico durante la introducción de texto, es posible hacer uso de la tecla de <Retroceso>. No obstante, en este último caso, las distintas implementaciones del editor vi pueden presentar comportamientos diferentes. De hecho, algunas simplemente mueven el cursor hacia atrás, de manera que el texto no se elimina, mientras que otras sí lo eliminan. Sea como fuere, el comportamiento por defecto del editor vi es el siguiente: desplazar el cursor hacia atrás mediante la tecla de <Retroceso> y en el momento de pulsar <Esc> para volver al modo comando se eliminan los caracteres de la misma línea situados a la derecha del cursor.

En lo que a inserción de caracteres se refiere, se muestra a continuación una relación de algunos de los comandos más útiles:

- a      Añadir texto a partir de la posición actual del cursor.
- A      Añadir al final de la línea.
- i      Insertar texto a la izquierda del cursor.
- I      Insertar a la izda. del 1º carácter no espacio en la línea actual.
- o      Abrir una nueva línea y añadir texto debajo de la línea actual.
- O      Abrir una nueva línea y añadir texto encima de la línea actual.

Además, este editor también incluye varios comandos que permiten el borrado de texto. Algunos de estos comandos son:

- x      Borrar el carácter que está debajo del cursor.
- dw    Borrar desde la posición actual al final de la palabra.
- dd    Borrar la línea actual.
- D      Borrar desde la posición actual hasta el final de la línea.

Su funcionalidad puede verse incrementada gracias al uso de modificadores, si bien el comando "D" ignora los modificadores que se le apliquen.

- `nx`     Borrar n caracteres desde el que está bajo el cursor.
- `ndd`   Borrar n líneas.
- `dnw`   Borrar n palabras (igual que `ndw`).
- `dG`    Borrar desde la posición actual hasta el final del fichero.
- `d1G`   Borrar desde la posición actual hasta el principio del fichero.
- `d$`    Borrar desde la posición actual hasta el final de línea. (Igual que "D").
- `dn$`   Borrar desde la línea actual al final de la enésima línea.

La utilidad de lista de comandos anterior se hace patente cuando se aplica en combinación con los comandos de movimiento de cursor.

El editor vi ofrece también la posibilidad de deshacer los últimos cambios llevados a cabo sobre el texto. Así, los comandos siguientes recuperan el texto existente antes de hacer cambios.

- `u`     Deshacer el último comando.
- `U`     Deshacer todos los cambios hechos en la línea actual.
- `:e!`    Editar otra vez. Recupera el fichero desde la última vez que se salvó.

Otra opción de este editor es la posibilidad de llevar a cabo cambios sobre el texto sin necesidad de borrar el texto viejo, reemplazándolo por el nuevo, como se detalla:

- `rc`    Reemplazar con la letra "c" el carácter bajo el cursor.
- `R`     Sobreescribir el texto con el nuevo texto.
- `cw`    Cambiar el texto de la palabra en curso. Comienza en la posición actual dentro de la palabra, y termina al final de la misma.
- `c$`    Cambiar el texto desde la posición actual hasta el final de la línea.
- `cnw`   Cambiar las siguientes n palabras. (Igual que `ncw`).
- `cn$`   Hacer cambios hasta el final de la enésima línea.

- c Hacer cambios hasta el final de la línea actual.
- cc Hacer cambios en la línea actual.

### Copia y movimiento de bloques de texto

Además de combinar cierto número de comandos para mover bloque de texto, el editor vi proporciona la posibilidad de hacer uso de *buffers* en combinación con utilidades que facilitan dicho movimiento.

La copia de texto se realiza en tres pasos principales:

1. Copiar del texto a un buffer, con o sin nombre.
2. Mover el cursor al lugar de destino.
3. Pegar el texto en el fichero de trabajo o *buffer de edición*.

Para copiar texto a un *buffer* sin nombre, se hará uso del comando "y" en cualquiera de las variantes siguientes:

- yy Mover una copia de la línea actual al buffer sin nombre (igual que y).
- nyy Mover las siguientes n líneas al buffer sin nombre (igual que ny).
- yw Mover una palabra al buffer sin nombre.
- ynw Mover n palabras al buffer sin nombre (igual que nyw).
- y\$ Mover el texto desde la posición actual hasta el final de la línea, al buffer sin nombre.
- "ayy Mover la línea actual al buffer de nombre "a" (igual que "ay).
- "byw Mover la palabra actual al buffer de nombre "b".
- "byw Añadir la palabra actual al contenido del buffer "b".
- "by3w Mover las siguientes 3 palabras al buffer "b".

Para pegar el contenido de un buffer dentro del fichero, puede utilizarse el comando p, como se detalla en las siguientes modalidades:

- p Pegar del buffer sin nombre a la DERECHA del cursor.

- P      Pegar del buffer sin nombre a la IZQUIERDA del cursor.
- nP    Pegar "n" copias del buffer sin nombre a la IZQUIERDA del cursor.
- "aP   Pegar del buffer de nombre "a" a la DERECHA del cursor.
- "b3P   Pegar 3 copias de buffer con nombre "b" a la IZQUIERDA del cursor.

Por otro lado, los pasos necesarios para mover un bloque de texto son:

1. Borrar el texto para ponerlo en un buffer con o sin nombre.
2. Mover el cursor a la posición de destino.
3. Pegar el buffer.

El proceso es el mismo que el de copia, variando el primer paso, ya que en este caso se borra el texto original.

- "add   Borrar la línea y ponerla en el buffer con nombre "a".
- "a4dd   Borrar cuatro líneas y ponerlas en el buffer con nombre "a".
- dw    Borrar una palabra y ponerla en el buffer sin nombre.

De forma transparente, el comando dd provoca el borrado de una línea pasándola al buffer sin nombre (si no se especifica lo contrario), de manera que pueda pegarse después en otro punto del fichero.

### Búsqueda de caracteres

El editor vi dispone de varios comandos de búsqueda, siendo posible buscar caracteres independientes o incluso de expresiones completas dentro de la línea actual. Entre los buscadores de caracteres, destacan el "f" y el "t":

- f c    Encontrar el siguiente carácter "c", quedando a la derecha.
- F c    Encontrar el siguiente carácter "c", quedando a la izquierda.
- t c    Mover a la DERECHA del carácter anterior al siguiente "c".
- T c    Mover a la IZQUIERDA del carácter que sigue al "c" anterior

; Repetir el último comando "f", "F", "t", "T".

, Igual que (;) pero cambiando la dirección del comando original.

En aquellos casos en que la búsqueda resulte infructuosa, el editor avisará mediante la emisión de un pitido o la generación de algún tipo de señal, como un parpadeo de pantalla. Del mismo modo, la búsqueda de cadenas de texto se puede realizar con los comandos siguientes:

/tira Buscar hacia la derecha y abajo la siguiente instancia de "tira".

?tira Buscar hacia la izquierda y arriba la siguiente instancia de "tira".

n Repetir el último comando "/" o "?".

N Repetir el último comando "/" o "?" en la dirección opuesta.

### Caracteres especiales

Pueden utilizarse dentro de una expresión (por ejemplo en una búsqueda) y son los siguientes:

. Vale por cualquier carácter excepto el carácter de nueva línea.

" Tecla de "escape" para cualquier carácter especial.

\* Vale por 0 o más instancias del carácter anterior.

[ ] Buscar exactamente uno de los caracteres incluidos entre los corchetes.

^ El carácter que sigue a "^" debe estar al principio de la línea.

\$ El carácter que precede a "\$" se busca al final de la línea.

[ ^ ] Buscar cualquier cosa que no se encuentre después de ^ dentro de los corchetes.

[ - ] Buscar un rango de caracteres.

## Comandos de búsqueda y sustitución

Además de los comandos de búsqueda descritos, el editor "vi" hace uso del modo comando del editor "ex" para llevar a cabo búsquedas y sustituciones. De hecho, todos los comandos del editor vi que comienzan por ":" son propios del editor ex.

A la hora de utilizar comandos de búsqueda y sustitución es posible hacer uso de expresiones regulares<sup>14</sup> sobre rangos de líneas donde sustituir la cadena de caracteres que se busca, lo que hace que resulten extremadamente potentes. Además, el usuario puede especificar que el editor solicite confirmación antes de hacer una sustitución.

Por otro lado, es posible hacer uso de comandos de sustitución que incluyan algunos caracteres especiales, de manera que por ejemplo, ":1,5s/ayuda/&ndo/g" sustituye la cadena "ayuda" por la cadena "ayudando" en las 5 primeras líneas del fichero al que se aplique, y ":%s/ /&&/g" duplica el número de espacios entre palabras.

El comando general es :<prim>,<ult>s/<busca>/<sust>/g. Algunos ejemplos de su utilización son:

- :1,\$s/los/Los/g    Buscar en todo el fichero y sustituir la cadena "los" por la cadena "Los".
- :. ,5s/^.\*//g    Borrar el contenido desde la línea actual hasta la quinta.
- :%s/los/Los/gc    Sustituir la cadena "los" por la cadena "Los" solicitando confirmación antes de llevar a cabo la operación.
- :%s/^....//g    Borra los primeros cuatro caracteres de cada línea.

Hay que destacar que la no inclusión de la directiva g causará que el cambio se aplique únicamente a la primera instancia que se encuentre en cada línea.

---

<sup>14</sup> Construcciones construidas en base a una sintaxis.



### 3.9 Extracción de archivos de una máquina Linux

#### 3.9.1 COPIA DE FICHEROS A DISQUETE

Exige tener acceso físico a la máquina Linux en que residen, y por tanto a su disquetera. Los ficheros elaborados bajo Linux se pueden copiar a un disquete y ser trasladados a un ordenador con algún sistema operativo de la familia MS Windows para acceder a ellos. No obstante, la máquina Linux debe cumplir los siguientes requisitos:

Linux debe estar preparado para dar soporte al sistema de ficheros de *DOS*. Esta opción deberá haber sido seleccionada en el momento de llevar a cabo la instalación de Linux.

La disquetera del ordenador debe estar configurada para soportar el montaje de sistemas de ficheros *dos*. Como las opciones de montaje relativas a sistemas de ficheros se almacenan en el fichero `/etc/fstab`, en la línea correspondiente a la disquetera (`/dev/fd0`) deberá estar especificado que ésta soporta el sistema de ficheros *dos* en vez del sistema de ficheros *ext2* (Linux Native).

Si el sistema cumple estos requisitos, sólo se podrán copiar ficheros a la disquetera si se tienen privilegios para montar sistemas de ficheros en `/dev/fd0`. Si es así, se introduce un disquete previamente formateado bajo MS-DOS o MS Windows, o bien un disquete sin formatear (es posible darle formato mediante el comando `mkfs`). Una vez que se dispone de un disquete con formato en la disquetera, se monta mediante el comando `mount` y se copian archivos a la misma con el comando `cp`. Cuando la copia ha finalizado, y antes de extraer el disquete es necesario desmontar el sistema de ficheros de `/dev/fd0`, usando el comando `umount`.

#### 3.9.2 ACCESO VÍA FTP

Si la máquina que dispone de Linux no es físicamente accesible, es decir, se trabaja con ella desde otro ordenador, por ejemplo por medio de *telnet*, no se podrá utilizar la disquetera, y por tanto no será posible copiar directamente los archivos a la misma.

En este caso, es posible utilizar el protocolo FTP<sup>15</sup> para transferir ficheros desde la máquina remota. Las transferencias de ficheros vía FTP se hacen entre dos programas denominados “cliente FTP” y “servidor FTP”. Estos dos programas deben

---

<sup>15</sup> File Transfer Protocol

estar instalados cada uno en una de las máquinas que van a intervenir en la transferencia de ficheros (suponiendo que ambas máquinas pueden conectarse la una con la otra a través de alguna red).

En el ordenador en el que se encuentra instalado el sistema operativo Linux, el administrador del sistema será el encargado de instalar y configurar el servidor FTP de manera que atienda las solicitudes de transferencia de archivos. El otro ordenador precisará de un navegador web (MS Internet Explorer, Netscape Communicator, etc.) que incorpora un cliente de FTP, o bien de un programa específico que desempeñe este cometido, como por ejemplo el que se instala junto con Microsoft Windows NT, `c:\winnt\system32\ftp.exe`. No obstante, en este último caso puede ser necesario el conocimiento de los comandos necesarios para la transferencia de ficheros y el recorrido de la estructura de directorios de la máquina remota, propios del protocolo FTP.

Por otro lado, además de disponer del cliente y servidor FTP, es necesaria una cuenta de acceso a la máquina remota vía FTP, que generalmente coincidirá con la cuenta que se utiliza para el establecimiento de la sesión en Linux.

Una vez llevada a cabo la transferencia, bastará generalmente con “cerrar” el cliente FTP, si bien debe tenerse cuidado de finalizar la conexión con la máquina remota de manera correcta, aunque esta tarea suele llevarse a cabo de manera automática. En cualquier caso, cerrado o no el cliente FTP, los ficheros ya serán accesibles dentro de la máquina local.

### 3.9.3 IMPRESIÓN EN LINUX

Otra posibilidad de cara a la obtención de copias de los archivos elaborados bajo Linux, radica en la obtención de copias impresas. Para ello, basta con tener instalada y configurada una impresora en el sistema Linux.

Para poder instalar una impresora en un ordenador con Linux, ésta tiene que ser admitida por el sistema, es decir, deben localizarse los *drivers* necesarios para hacer funcionar la citada impresora en Linux. Es importante destacar el hecho de que no todos los dispositivos compatibles con sistemas MS-DOS o MS-Windows son compatibles con Linux.

También es necesario instalar el programa `lpd`, que se encarga de la gestión de la impresora. La instalación y configuración de esta aplicación es responsabilidad del administrador del sistema Linux.